How to Add Application Content to the Warehouse Management Monitor

Applicable Releases:
SAP Extended Warehouse Management 7.02 and higher
### Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Example Text</em></td>
<td>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</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><em>Example text</em></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><code>&lt;Example text&gt;</code></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>⚠️</td>
<td>Important</td>
</tr>
<tr>
<td>📝</td>
<td>Note</td>
</tr>
<tr>
<td>🔧</td>
<td>Recommendation or Tip</td>
</tr>
<tr>
<td>📦</td>
<td>Example</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction .............................................................................................................................. 1  
   1.1 The Proof of the Pudding is in the Eating - Intention of How-To Guide ......................... 1

2. Prerequisites ............................................................................................................................ 1

3. Overview .................................................................................................................................. 2
   3.1 Providing Content for Warehouse Management Monitors .............................................. 3

4. Enhancement of Existing Monitor Node Content ...................................................................... 4
   4.1 Prerequisite Steps ................................................................................................................ 4
      4.1.1 Create Own Monitor by Copying SAP Template ......................................................... 4
      4.1.2 Create Function Group for Customer Function Modules ............................................. 6
   4.2 Example: Display Product Description for Outbound Delivery Order Items ................... 7
      4.2.1 Solution Proposal ........................................................................................................... 7
      4.2.2 Required Steps ............................................................................................................... 7
      4.2.3 Identification of Node Data ........................................................................................... 8
      4.2.4 Copy Node to Customer Node ..................................................................................... 10
      4.2.5 Copy Node Profile ....................................................................................................... 11
      4.2.6 Copy and Enhance DDIC Structures .......................................................................... 12
      4.2.7 Create Wrapper Function Module .............................................................................. 13
      4.2.8 Copy and Adjust Screen (Form Layout) .................................................................... 16
      4.2.9 Change Customizing Settings for Monitor Node ......................................................... 17
      4.2.10 Testing ......................................................................................................................... 19
   4.3 Advanced Example: Extend Select Options for Deliveries and Display Results ............. 21
      4.3.1 Solution Proposal ........................................................................................................... 21
      4.3.2 Required Steps ............................................................................................................... 21
      4.3.3 Identification of Node Data ........................................................................................... 22
      4.3.4 Copy Node to Customer Node ..................................................................................... 22
      4.3.5 Copy Node Profile ....................................................................................................... 22
      4.3.6 Copy and Enhance DDIC Structures .......................................................................... 23
      4.3.7 Create Function Module .............................................................................................. 24
      4.3.8 Copy and Adjust Screen (Form Layout) .................................................................... 25
      4.3.9 Change Customizing Settings for Monitor Node ......................................................... 26
      4.3.10 Testing ......................................................................................................................... 27

5. Add New Methods to Monitor ................................................................................................. 29
   5.1 Example: New Method for Unassigning Outbound Delivery Order Items from Wave 29
      5.1.1 Solution Proposal ........................................................................................................... 29
      5.1.2 Required Steps ............................................................................................................... 30
      5.1.3 Prerequisites for Function Modules used in Object Class Methods ........................... 30
      5.1.4 Sample Function Module ZHTG_WAVEMON_UNASSIGN ....................................... 31
      5.1.5 Define New Custom Object Class ZHTGOI ................................................................. 33
      5.1.6 Maintain Object Class Methods for Custom Object Class ZHTGOI ..................... 34
      5.1.7 Define Method Presentation of Custom Object Class Method .................................... 36
      5.1.8 Exchange Object Class of Monitor Node against Custom Object Class .................. 37
      5.1.9 Testing ......................................................................................................................... 37

6. Add New Application Content .................................................................................................. 38
   6.1 Customize Own Monitors ................................................................................................... 38
   6.2 Example: Create Custom Monitor for New Custom Nodes .............................................. 39
6.2.1 Required Steps ................................................................. 39
6.2.2 Define Custom Monitor ZHTG_S ........................................ 39
6.2.3 Create New Category ZHTG_C ......................................... 40
6.2.4 Define New Object Class ZHTGID ...................................... 41
6.2.5 Define New Object Class Methods ................................. 42
6.2.6 Define Methods Presentation .......................................... 43
6.2.7 Define Node Profiles for Monitor Tree Nodes .................. 44
6.2.8 Define Monitor Tree Nodes ............................................. 45
6.2.9 Define Node Hierarchy for Monitor ZHTG_S .................... 46
6.2.10 Testing New Custom Monitor ZHTG_S ............................ 48

6.3 Function Modules for New Application Content .................. 49
6.3.1 Selection Screens .......................................................... 49
6.3.2 Interface ........................................................................ 50
6.3.3 Mandatory Processing Steps .......................................... 51

7. Maintain Hotspots for Navigation ....................................... 57
7.1 Example: Hotspot for Displaying Outbound Delivery Order Items in Enhanced Monitor Node .......................... 57
7.1.1 Define Hotspot .............................................................. 58
7.1.2 Map Service Keys Fields ............................................... 59
7.2 Example: Maintain Hotspot Navigation to Inbound Delivery .......................................................... 60
7.2.1 Define Hotspot .............................................................. 60
7.2.2 Map Service Keys Fields ............................................... 60

8. Appendix .............................................................................. 62
8.1 Sample Coding .................................................................... 62
8.1.1 Master Program SAPLZHTG (Function Pool) .................... 62
8.1.2 Include LZHTGTOP ....................................................... 62
8.1.3 Include LZHTGP01 ....................................................... 64
8.1.4 Include LZHTGP02 ....................................................... 66
8.1.5 Function Module ZHTG_WHRHEAD_MON_XT ............. 85
8.1.6 Text Symbols and Selection Texts of Function Group ZHTG .......................................................... 95
8.1.7 Function Module ZHTG_WAVEMON_UNASSIGN_XT– Extended Check for Typing of IT_DATA .......................................................... 96
1. Introduction

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:

<table>
<thead>
<tr>
<th>Enhancement Option</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy SAP standard monitor node and enhance content of copied node</td>
<td>4</td>
</tr>
<tr>
<td>Maintain hotspots for easier navigation</td>
<td>7</td>
</tr>
<tr>
<td>Add additional methods to monitor nodes</td>
<td>5</td>
</tr>
<tr>
<td>Add new application content by creating and adding new monitor nodes</td>
<td>6</td>
</tr>
</tbody>
</table>

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
- Authorisations
  - 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
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 contain 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
  - Visualisation 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 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."

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Customize Monitor Tree**
- **Define Nodes**
- **Define Monitors**
- **Define Object Class Methods**
- **Define Navigation**

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: Customizing Monitor Tree

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: Load SAP Standard Monitor Node Hierarchy
Note

The copying function does not create copies of each node that is represented the in 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):

![Change View "Define Methods Presentation": Overview](image)

Figure: Copy Object Class Method Presentation

Change the values in columns Warehouse and Monitor accordingly:

![Change View "Define Methods Presentation": Overview of Selected Rows](image)

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

![Object Navigator](image)

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:

![Image of Warehouse Management Monitor](image)

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

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

![Image of Warehouse Management Monitor](image)

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*:

```
<table>
<thead>
<tr>
<th>Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>▼</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>▼</td>
</tr>
<tr>
<td>Warehouse Management Monitor</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>▼</td>
</tr>
<tr>
<td>Custom Monitor Tree</td>
</tr>
<tr>
<td>Define Nodes</td>
</tr>
</tbody>
</table>
```

**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:
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:

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:
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*.

Mark the node *Outbound Delivery Order Item* 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:

![Image: Remove Original Node from Customer Monitor](image)

**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*:

![Image: Rename Customer Node](image)

**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*:

![Image: Customizing Activity for Changing Node Content like Node Profiles](image)

**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.
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:

<table>
<thead>
<tr>
<th>Component</th>
<th>Typing Method</th>
<th>Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>.INCLUDE</td>
<td>Types</td>
<td>/SCWM/S_WIP_WHRITEM_OUT</td>
</tr>
<tr>
<td>MAKTX</td>
<td>Types</td>
<td>/SCWM/DE_UI_MAKTX</td>
</tr>
</tbody>
</table>

**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.
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.

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: 

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Default</th>
<th>Opt.</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_DATA_PARENT</td>
<td>TYPE</td>
<td>/SCWM/TT_WIP_WH RHEAD_OUT</td>
<td></td>
<td>X</td>
<td>Table consisting of (parent) entries selected for drill down</td>
</tr>
<tr>
<td>IV_CATEGORY</td>
<td>TYPE</td>
<td>/SCWM/DE_CATEG ORY</td>
<td></td>
<td></td>
<td>Category</td>
</tr>
<tr>
<td>IV_LGNUM</td>
<td>TYPE</td>
<td>/SCWM/LGNUM</td>
<td></td>
<td></td>
<td>Warehouse Number/Warehouse Complex</td>
</tr>
<tr>
<td>IV_MODE</td>
<td>TYPE</td>
<td>/SCWM/DE_MON_F M_MODE</td>
<td>‘1’</td>
<td></td>
<td>Execution mode for function modules in WM monitor</td>
</tr>
<tr>
<td>IV_VARIANT</td>
<td>TYPE</td>
<td>VARIANT</td>
<td></td>
<td>X</td>
<td>ABAP report variant</td>
</tr>
</tbody>
</table>

**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:*

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ET_DATA</td>
<td>TYPE</td>
<td>ZHTG_TT_WIP_WHRITEM_OUT</td>
<td>Outbound Delivery Item in WIP Monitor with product description</td>
</tr>
<tr>
<td>EV_RETURNCODE</td>
<td>TYPE</td>
<td>XFELD</td>
<td>‘X’: User cancelled selection</td>
</tr>
<tr>
<td>EV_VARIANT</td>
<td>TYPE</td>
<td>VARIANT</td>
<td>selected ABAP report variant</td>
</tr>
</tbody>
</table>

- Tab strip *Changing:*

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Opt.</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT_FIELDCAT</td>
<td>TYPE</td>
<td>LVC_T_FCAT</td>
<td>X</td>
<td>Field Catalog for List Viewer Control</td>
</tr>
<tr>
<td>CT_RANGE</td>
<td>TYPE</td>
<td>RSDS_TRANGE</td>
<td>X</td>
<td>Table consisting of selection options of previous nodes</td>
</tr>
</tbody>
</table>

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

```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

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,

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>. 
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:

<table>
<thead>
<tr>
<th>El. Type</th>
<th>Name</th>
<th>Text</th>
<th>From Dict.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Field</td>
<td>ZHTG_S_WIP_WRITEMEMON_OUT-MAKTX</td>
<td>Product_Descr._</td>
<td>X</td>
</tr>
<tr>
<td>I/O Field</td>
<td>ZHTG_S_WIP_WRITEMEMON_OUT-MAKTX</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Save and activate your changes.

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](image)

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:
<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Table Type</td>
<td>ZHTG_TT_WIP_WHRITEM_OUT</td>
</tr>
<tr>
<td>List Function Module</td>
<td>ZHTG_WHRITEM_MON_OUT_XT</td>
</tr>
<tr>
<td>Form Structure</td>
<td>ZHTG_S_WIP_WHRITEM_OUT</td>
</tr>
<tr>
<td>Dynpro Program</td>
<td>SAPLZHTG</td>
</tr>
<tr>
<td>Dynpro No.</td>
<td>0210</td>
</tr>
<tr>
<td>Text</td>
<td>Outbound Delivery Ord. Item XT</td>
</tr>
<tr>
<td>Presentation Text</td>
<td>ODO Item XT</td>
</tr>
</tbody>
</table>

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.
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.
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](image)

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”](image)

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

![Figure: Product Short Description Data (Form View)](image)

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](image)

The outbound delivery order item data is displayed as desired.
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.

![Warehouse Management Monitor SAP - Warehouse Number B3TB](image)

**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.
  o 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.
  o 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.
  o Add a label and an output field for the ERP document number.
• Change the Customizing settings for the monitor node:
  o Exchange the function module, DDIC structures, screen program, and screen number in the copied node profile
  o Update the settings for the selected monitor node
• Test changes
  o 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 ZN000000004) and the assigned profile. In our example, the assigned node profile ID is P00000085.

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 P00000085 to the new profile ZHTGID. Save your changes.
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:

<table>
<thead>
<tr>
<th>Component</th>
<th>Typing Method</th>
<th>Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>.INCLUDE</td>
<td>Types</td>
<td>/SCWM/S_WIP_WHRHEAD</td>
</tr>
<tr>
<td>ERP_DOCNO</td>
<td>Types</td>
<td>/SCWM/SP_DOCNO_ERP</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Default</th>
<th>Opt.</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV_LGNUM</td>
<td>TYPE</td>
<td>/SCWM/LGNUM</td>
<td></td>
<td></td>
<td>Warehouse Number/Warehouse Complex</td>
</tr>
<tr>
<td>IV_MODE</td>
<td>TYPE</td>
<td>/SCWM/DE_MON_F M_MODE</td>
<td>‘1’</td>
<td></td>
<td>Execution mode for function modules in WM monitor</td>
</tr>
<tr>
<td>IV_VARIANT</td>
<td>TYPE</td>
<td>VARIANT</td>
<td></td>
<td>X</td>
<td>ABAP report variant</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ET_DATA</td>
<td>TYPE</td>
<td>ZHTG_TT_WIP_WHRHEAD</td>
<td>Delivery Header in WIP Monitor XT</td>
</tr>
<tr>
<td>EV_RETURNCODE</td>
<td>TYPE</td>
<td>XFELD</td>
<td>‘X’: User cancelled selection</td>
</tr>
<tr>
<td>EV_VARIANT</td>
<td>TYPE</td>
<td>VARIANT</td>
<td>selected ABAP report variant</td>
</tr>
</tbody>
</table>

- Tab strip Changing:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Opt.</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT_FIELDCAT</td>
<td>TYPE</td>
<td>LVC_T_FCAT</td>
<td>X</td>
<td>Field Catalog for List Viewer Control</td>
</tr>
<tr>
<td>CT_TAB_RANGE</td>
<td>TYPE</td>
<td>RSDS_TRANGE</td>
<td>X</td>
<td>Table consisting of selection options of previous nodes</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th>El. Type</th>
<th>Name</th>
<th>Text</th>
<th>From Dict.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text Field</td>
<td>ZHTG_S_WIP_WHRHEAD-ERP_DOCNO</td>
<td>ERP Document</td>
<td>X</td>
</tr>
<tr>
<td>I/O Field</td>
<td>ZHTG_S_WIP_WHRHEAD-ERP_DOCNO</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Save and activate your changes.

![Form View Screen containing Label and Field for ERP Document](image)

**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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>List Table Type</td>
<td>ZHTG_TT_WIP_WHRHEAD</td>
</tr>
<tr>
<td>List Function Module</td>
<td>ZHTG_WHRHEAD_MON_XT</td>
</tr>
<tr>
<td>Form Structure</td>
<td>ZHTG_S_WIP_WHRHEAD</td>
</tr>
<tr>
<td>Dynpro Program</td>
<td>SAPLZHTG</td>
</tr>
<tr>
<td>Dynpro No.</td>
<td>0110</td>
</tr>
<tr>
<td>Text</td>
<td>Inbound Delivery XT</td>
</tr>
<tr>
<td>Presentation Text</td>
<td>Inb. Deliv. XT</td>
</tr>
</tbody>
</table>

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:

![Node Data](image)

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.
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.

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

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.
The outbound delivery order item data is displayed as desired:

![Image: Product Short Description Data (List View)](image)

**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.

![Image: Method for Assigning Marked Outbound Delivery Order Items to Waves](image)

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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Associated Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV_LGNUM</td>
<td>Imp.</td>
<td>/SCWM/LGNUM</td>
<td>Warehouse number/complex</td>
</tr>
<tr>
<td>IT_DATA</td>
<td>Imp.</td>
<td>STANDARD TABLE</td>
<td>Table consisting of selection of select entries in the monitor</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Typing</th>
<th>Associated Type</th>
<th>Default</th>
<th>Opt.</th>
<th>Short Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV_LGNUM</td>
<td>TYPE</td>
<td>/SCWM/LGNUM</td>
<td>Warehouse number/complex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IT_DATA</td>
<td>TYPE</td>
<td>STANDARD TABLE</td>
<td>Table containing selected ODO item entries</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• Tab strip Source code:
  o Copy the following lines of code:

```
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 balloghdl,
  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.
*"__________________________________________________________________________
```
* Check that entries have been selected from result list

```plaintext
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 *

```plaintext
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

```plaintext
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 = wmeqc_stwave_initial OR
    t-status = wmeqc_stwave_hold ).
```

```plaintext
lv_lines_w_wave = lines( lt_wave_unass ).
```

```plaintext
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 *

```plaintext
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.
```

```plaintext
IF NOT lt_bapiret[] IS INITIAL.
```
### Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN

* Handle messages that occurred during function processing

```abap
lo_log->add_log( it_prot = lt_bapiret ).
ls_log-extnumber = 1.
ls_log-object = wmec_apl_object_wme.
ls_log-subobject = wmec_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

```abap
CALL FUNCTION 'BAL_DSP_PROFILE_POPUP_GET'
    IMPORTING
    e_s_display_profile = ls_display_profile.

ls_display_profile-use_grid = 'X'.
```

**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.

![Customizing Activity for Creating Monitor Nodes](image)

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

![Custom Object Class for New Object Class Method](image)

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

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>Text</th>
<th>Presentation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGOI</td>
<td>Outbound Delivery Ord. Item XT</td>
<td>ODO Item XT</td>
</tr>
</tbody>
</table>

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.

![Customizing Activity for Creating Object Class Methods](image)

Execute the Customizing activity Define Object Class Methods. On the following screen double-click on the folder Define Methods. Press the button New Entries:
Maintain new entries as described in the table below:

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>Method</th>
<th>Method Function Module</th>
<th>MethodIcon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGOI</td>
<td>Z00001</td>
<td>ZHTG_WAVEMON_UNASSIGN</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AutoRefresh</th>
<th>Text</th>
<th>Presentation Text</th>
<th>Pushb. Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Unassign from Wave XT</td>
<td>Unassign Wav XT</td>
<td>UnassWav</td>
</tr>
</tbody>
</table>

**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:

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

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 \textit{WHRITO} with \textit{ZHTGOI} and press \texttt{ENTER}. Save your changes. Return to the overview screen. Scroll down the method list and check your object class methods.

<table>
<thead>
<tr>
<th>Obj.Class</th>
<th>Method</th>
<th>Method Function Module</th>
<th>MethodIcon</th>
<th>AutoRefresh</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGOI</td>
<td>M00002</td>
<td>/SCWM/WHR_OUT_TFREE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTGOI</td>
<td>M00003</td>
<td>/SCWM/WHRITEM_OUT_QUANT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTGOI</td>
<td>M00004</td>
<td>/SCWM/WHRITEM_OUT_QUANT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTGOI</td>
<td>M00005</td>
<td>/SCWM/WHRITEM_OUT_QUANT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTGOI</td>
<td>M00006</td>
<td>/SCWM/WHRITEM_OUT_BIN_DET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTGOI</td>
<td>Z00001</td>
<td>ZHTG_WAVEMON_UNASSIGN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: Available Methods for Object Class ZHTGOI

5.1.7 Define Method Presentation of Custom Object Class Method

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

Execute the Customizing activity \textit{Define Object Class Methods}. On the following screen double-click the folder \textit{Define Methods Presentation}. Press the button \texttt{New Entries}.

Figure: Definition of Method Presentations for Object Class ZHTGOI

Maintain the new entries as described in the table below:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>1</td>
<td>M00002</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>2</td>
<td>Z00001</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>3</td>
<td>M00003</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>4</td>
<td>M00004</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>5</td>
<td>M00005</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG</td>
<td>ZHTGOI</td>
<td>6</td>
<td>M00006</td>
<td>Accumulated methods menu button</td>
</tr>
</tbody>
</table>

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 \texttt{SQNCE} field.
The field \textit{METHOD\_PRES} is used to determine how the method is to be displayed. All possible values are domain values (domain \texttt{/SCWM/D0\_METHOD\_PRES}).

<table>
<thead>
<tr>
<th>Fix. Val</th>
<th>Short Descript.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal pushbutton with icon</td>
</tr>
<tr>
<td>1</td>
<td>Normal pushbutton with text</td>
</tr>
<tr>
<td>2</td>
<td>Normal pushbutton with icon and text</td>
</tr>
<tr>
<td>3</td>
<td>Accumulated methods menu button</td>
</tr>
</tbody>
</table>

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

![Accumulated methods menu button](image)

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 \textit{Extended Warehouse Management} $\rightarrow$ \textit{Monitoring} $\rightarrow$ \textit{Warehouse Management Monitor}. Execute the Customizing activity \textit{Define Nodes}. On the following screen double-click the folder \textit{Define Node Profile}. Scroll down to the custom node profile ZHTGODOI. Exchange the object class WHRITO with the custom object class ZHTG0I:

![Assignment of Custom Object Class of Custom Node Profile](image)

5.1.9 Testing

Start the custom warehouse management monitor ZHTG in transaction \texttt{/SCWM/MON}. Open the monitor tree branch \textit{Outbound} $\rightarrow$ \textit{Documents} $\rightarrow$ \textit{Outbound Delivery Order}. Double-click on the custom node \textit{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 \textit{More Methods}. Choose the method \textit{Unassign from Wave XT}. 
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*.

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:
The following fields can be maintained:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wareh.</td>
<td>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 ****</td>
</tr>
<tr>
<td>Monitor</td>
<td>Name of your monitor variant</td>
</tr>
<tr>
<td>StdMonitor</td>
<td>Name of standard monitor that is merged with monitor</td>
</tr>
<tr>
<td>+</td>
<td>If checkbox is ticked, monitor tree uses short texts. Otherwise long texts are used and displayed in monitor tree</td>
</tr>
<tr>
<td>Text</td>
<td>Description for your monitor variant</td>
</tr>
</tbody>
</table>

**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>
<thead>
<tr>
<th>Table Name</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/TMONITOR</td>
<td>LGNUM</td>
<td>/SCWM/LGNUM</td>
<td>Warehouse Number/Warehouse Complex</td>
</tr>
<tr>
<td>/SCWM/TMONITOR</td>
<td>MONITOR</td>
<td>/SCWM/DE_MONITOR</td>
<td>Monitor</td>
</tr>
<tr>
<td>/SCWM/TMONITOR</td>
<td>STAND_MONITOR</td>
<td>/SCWM/DE_STAND_MONITOR</td>
<td>Standard Monitor</td>
</tr>
<tr>
<td>/SCWM/TMONITOR</td>
<td>SHORT_TEXT</td>
<td>/SCWM/DE_MONTREE_SHORT_TEXT</td>
<td>Display short text in monitor tree</td>
</tr>
<tr>
<td>/SCWM/TMONITORT</td>
<td>TEXT</td>
<td>/SCWM/DE_TEXT</td>
<td>Text</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>Category</th>
<th>Text</th>
<th>Presentation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_C</td>
<td>Category for simple Monitor</td>
<td>HTG Simple Mon</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table Name</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/CATEGORY</td>
<td>CATEGORY</td>
<td>/SCWM/DECATEGORY</td>
<td>Category</td>
</tr>
<tr>
<td>/SCWM/CATEGORYT</td>
<td>TEXT</td>
<td>/SCWM/DE_TEXT</td>
<td>Text</td>
</tr>
<tr>
<td>/SCWM/CATEGORYT</td>
<td>PTEXT</td>
<td>/SCWM/DE_PTEXT</td>
<td>Presentation text</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>Text</th>
<th>Presentation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGID</td>
<td>Inbound Deliveries (ObjCl)</td>
<td>InbDlvObjCl</td>
</tr>
</tbody>
</table>

### 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>
<thead>
<tr>
<th>Table Name</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/TOBCLS</td>
<td>OBCLS</td>
<td>/SCWM/DE_OBCLS</td>
<td>Object Class</td>
</tr>
<tr>
<td>/SCWM/TOBCLST</td>
<td>TEXT</td>
<td>/SCWM/DE_TEXT</td>
<td>Text</td>
</tr>
<tr>
<td>/SCWM/TOBCLST</td>
<td>PTEXT</td>
<td>/SCWM/DE_PTEXT</td>
<td>Presentation Text</td>
</tr>
</tbody>
</table>

### 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](image)

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:

<table>
<thead>
<tr>
<th>Obj. Class</th>
<th>Method</th>
<th>Method Function Module</th>
<th>MethodIcon</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGID</td>
<td>ZIDM01</td>
<td>/SCWM/INSP_MON</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AutoRefresh</th>
<th>Text</th>
<th>Presentation Text</th>
<th>Pushb. Text</th>
</tr>
</thead>
</table>
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:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGID</td>
<td>1</td>
<td>ZIDM01</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>1</td>
<td>M00002</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>2</td>
<td>Z00001</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>3</td>
<td>M00003</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>4</td>
<td>M00004</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>5</td>
<td>M00005</td>
<td>Accumulated methods menu button</td>
</tr>
<tr>
<td>****</td>
<td>ZHTG_S</td>
<td>ZHTGOI</td>
<td>6</td>
<td>M00006</td>
<td>Accumulated methods menu button</td>
</tr>
</tbody>
</table>

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):

<table>
<thead>
<tr>
<th>Fix. Val</th>
<th>Short Descript.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal pushbutton with icon</td>
</tr>
<tr>
<td>1</td>
<td>Normal pushbutton with text</td>
</tr>
<tr>
<td>2</td>
<td>Normal pushbutton with icon and text</td>
</tr>
<tr>
<td>3</td>
<td>Accumulated methods menu button</td>
</tr>
</tbody>
</table>

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:
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:

<table>
<thead>
<tr>
<th>Node Prof.</th>
<th>Obj. Class</th>
<th>List Table Type</th>
<th>List Funct. Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTGID</td>
<td>ZHTGID</td>
<td>ZHTG_TT_WIP_WHRHEAD</td>
<td>ZHTG_WHRHEAD_MON_XT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Form Funct. Module</th>
<th>Form Structure</th>
<th>Dynpro Program</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZHTG_S_WIP_WHRHEAD</td>
<td>SAPLZHTG</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynpro No.</th>
<th>Text</th>
<th>Presentation Text</th>
<th>Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110</td>
<td>Inbound Delivery XT</td>
<td>Inb. Deliv. XT</td>
<td></td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table Name</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/TNODE_PRF</td>
<td>NODE_PRF</td>
<td>/SCWM/DE_NODE_PRF</td>
<td>ID of node profile</td>
</tr>
<tr>
<td>/SCWM/TNODE_PRF</td>
<td>OBCLS</td>
<td>/SCWM/DE_OBCLS</td>
<td>Object class of the node profile</td>
</tr>
<tr>
<td>/SCWM/TNODE_PRF</td>
<td>LIST_FM</td>
<td>/SCWM/DE_LIST_FM</td>
<td>Name of the function module that gets the data for the ALV List view</td>
</tr>
</tbody>
</table>
### 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.

#### New Entries: Overview of Added Entries

![New Entries: Overview of Added Entries](image)

**Figure: Create Custom Nodes**

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

<table>
<thead>
<tr>
<th>Node</th>
<th>Category</th>
<th>Node Prof.</th>
<th>Variant</th>
<th>Layout</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_S_CAT</td>
<td>ZHTG_C</td>
<td>ZHTGID</td>
<td></td>
<td></td>
<td>Inbound Delivery (ZHTG_S)</td>
</tr>
<tr>
<td>ZHTG_S_ID</td>
<td>ZHTGID</td>
<td></td>
<td></td>
<td></td>
<td>Inbound Delivery (ZHTG_S)</td>
</tr>
</tbody>
</table>

**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.
### 6.2.8.1 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>
<thead>
<tr>
<th>Table Name</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/TNODE</td>
<td>NODE</td>
<td>/SCWM/DE_NODE</td>
<td>Node</td>
</tr>
<tr>
<td>/SCWM/TNODE</td>
<td>CATEGORY</td>
<td>/SCWM/DE_CATEGORY</td>
<td>Category</td>
</tr>
<tr>
<td>/SCWM/TNODE</td>
<td>NODE_PRF</td>
<td>/SCWM/DE_NODE_PRF</td>
<td>Node Profile</td>
</tr>
<tr>
<td>/SCWM/TNODE</td>
<td>VARID</td>
<td>VARIANT</td>
<td>Report variant</td>
</tr>
<tr>
<td>/SCWM/TNODE</td>
<td>LAYOUT</td>
<td>SLIS_VARI</td>
<td>ALV Layout variant</td>
</tr>
<tr>
<td>/SCWM/TNODET</td>
<td>TEXT</td>
<td>/SCWM/DE_TEXT</td>
<td>Text</td>
</tr>
<tr>
<td>/SCWM/TNODET</td>
<td>PTEXT</td>
<td>/SCWM/DE_PTEXT</td>
<td>Presentation Text</td>
</tr>
</tbody>
</table>

### 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**:

![Node Pool](image)

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.

**Notes:**
- 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:

<table>
<thead>
<tr>
<th>HigherNode</th>
<th>Lower Node</th>
<th>Sequence</th>
<th>Hide Node</th>
<th>Hide PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>ZHTG_S_CAT</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTG_S_CAT</td>
<td>ZHTG_S_ID</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZHTG_S_CAT</td>
<td>ZHTG_S_ODI</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

⚠️ 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:

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

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:
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.

```
* 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

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

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

AT SELECTION-SCREEN.
* react on erase button
  CASE  sscrfields-uccom.
    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:

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

The following steps must be implemented in each function module:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Associated Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT_FIELDCAT <em>(optional)</em></td>
<td>Chang.</td>
<td>LVC_T_FCAT</td>
<td>ALV field catalog</td>
</tr>
</tbody>
</table>

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.

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

An example is provided for each step:

1. If only a selection variant of the function module is to be chosen, check that the importing parameter `IV_MODE` is set to 3. If `iv_mode = 3`, call the function module `RS_VARIANT_CATALOG`.

2. Clear dynpro elements.

3. Map select-options and parameters to database tables and fields.

4. Fill selection screen according to variant if a variant is provided.

5. If selection criteria of a previous node were passed fill dynpro accordingly.

6. If drill down is used and parent data is passed move keys of parent data to selection screen.

7. Check whether the selection screen should be displayed.

8. Optionally offer free dynamic selections.


10. Finally the selection can take place.

11. Potentially refine selected data.
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	   "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	   "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
      variantObsolete   = 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
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.

<table>
<thead>
<tr>
<th>Step</th>
<th>Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>*******************************************************************</td>
</tr>
<tr>
<td></td>
<td>* STEP 1: Check whether only selection variant of function module</td>
</tr>
<tr>
<td></td>
<td>* should be chosen</td>
</tr>
<tr>
<td></td>
<td>*******************************************************************</td>
</tr>
<tr>
<td></td>
<td>IF iv_mode = lc_mode_sel_var.</td>
</tr>
<tr>
<td></td>
<td>CALL FUNCTION 'RS_VARIANT_CATALOG'</td>
</tr>
<tr>
<td></td>
<td>EXPORTING</td>
</tr>
<tr>
<td></td>
<td>report = sy-repid</td>
</tr>
<tr>
<td></td>
<td>* NEW_TITLE = ''</td>
</tr>
<tr>
<td></td>
<td>dynnr = lc_dynnr_0100</td>
</tr>
<tr>
<td></td>
<td>* POP_UP = ''</td>
</tr>
<tr>
<td></td>
<td>IMPORTING</td>
</tr>
<tr>
<td></td>
<td>sel_variant = ev_variant</td>
</tr>
<tr>
<td></td>
<td>EXCEPTIONS</td>
</tr>
<tr>
<td></td>
<td>no_report = 1</td>
</tr>
<tr>
<td></td>
<td>report_not-existent = 2</td>
</tr>
<tr>
<td></td>
<td>report_not_supplied = 3</td>
</tr>
<tr>
<td></td>
<td>no_variants = 4</td>
</tr>
<tr>
<td></td>
<td>no_variant_selected = 5</td>
</tr>
<tr>
<td></td>
<td>variant_not_existent = 6</td>
</tr>
<tr>
<td></td>
<td>OTHERS = 7.</td>
</tr>
<tr>
<td></td>
<td>IF sy-subrc &lt;&gt; 0.</td>
</tr>
<tr>
<td></td>
<td>MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno</td>
</tr>
<tr>
<td></td>
<td>WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.</td>
</tr>
<tr>
<td></td>
<td>ENDIF.</td>
</tr>
<tr>
<td></td>
<td>RETURN.</td>
</tr>
<tr>
<td></td>
<td>ENDIF.</td>
</tr>
<tr>
<td></td>
<td>*******************************************************************</td>
</tr>
<tr>
<td></td>
<td>.....</td>
</tr>
<tr>
<td>2.</td>
<td>*******************************************************************</td>
</tr>
<tr>
<td></td>
<td>* STEP 2: Clear dynpro screen elements</td>
</tr>
<tr>
<td></td>
<td>* *******************************************************************</td>
</tr>
<tr>
<td></td>
<td>lv_repid = sy-repid.</td>
</tr>
<tr>
<td></td>
<td>CALL FUNCTION '/SCWM/DYNPRO_ELEMENTS_CLEAR'</td>
</tr>
<tr>
<td></td>
<td>EXPORTING</td>
</tr>
<tr>
<td></td>
<td>lv_repid = lv_repid.</td>
</tr>
<tr>
<td>3.</td>
<td>*******************************************************************</td>
</tr>
<tr>
<td>Step</td>
<td>Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 3.   | **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.
|      |     ENDF.
|      |     ENDF.
|      | `************************************************************************` |
| 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.
|      |     ENDF.
|      | `************************************************************************` |
| 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 7

**Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT**

```plaintext
* 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

```plaintext
* 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.
*************************************************************************
```

### Step 9

```plaintext
* 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.
*************************************************************************
```

### Step 10

```plaintext
* 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 ).
*************************************************************************
```
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.

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:
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_WRITEM_OUT defines the technical properties of the column Document.

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

**Note**

The hotspot DOCNO_H for the service DLV and list structure /SCWM/S_WIP_WRITEM_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:

<table>
<thead>
<tr>
<th>List Structure</th>
<th>Hotspot</th>
<th>Service</th>
<th>Presentation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_S_WIP_WRITEM_OUT</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td></td>
</tr>
</tbody>
</table>

**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.
7.1.1 Involved Customizing Tables and Maintenance Views

Hotspot data is stored in the following tables:
- /SCWM/THOTSPOT Hotspots
- /SCWM/THOTSPOT Hotspots Presentation Texts

Corresponding maintenance view:
- /SCWM/V_HOTSPOT Hotspots

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

<table>
<thead>
<tr>
<th>Database Table</th>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/THOTSPOT</td>
<td>LIST_ST</td>
<td>/SCWM/DE_LIST_ST</td>
<td>Object Structure for List View</td>
</tr>
<tr>
<td>/SCWM/THOTSPOT</td>
<td>HOTSPOT</td>
<td>/SCWM/DE_HOTSPOT</td>
<td>Hotspot Field</td>
</tr>
<tr>
<td>/SCWM/THOTSPOT</td>
<td>SERVICE</td>
<td>/SCWM/DE_SERVICE</td>
<td>Service</td>
</tr>
<tr>
<td>/SCWM/THOTSPOTT</td>
<td>PTEXT</td>
<td>/SCWM/DE_PTEXT</td>
<td>Presentation Text</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>List Structure</th>
<th>Hotspot</th>
<th>Service</th>
<th>Key Field</th>
<th>List View Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_S_WIP_WHRITEM_OUT</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td>DOCCAT</td>
<td>DOCCAT</td>
</tr>
<tr>
<td>ZHTG_S_WIP_WHRITEM_OUT</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td>DOCNO</td>
<td>DOCNO_H</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Data Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST_ST</td>
<td>/SCWM/DE_LIST_ST</td>
<td>Object Structure for List View</td>
</tr>
<tr>
<td>HOTSPOT</td>
<td>/SCWM/DE_HOTSPOT</td>
<td>Hotspot Field Name</td>
</tr>
<tr>
<td>SERVICE</td>
<td>/SCWM/DE_SERVICE</td>
<td>Service to be used (for example, WO for warehouse orders)</td>
</tr>
<tr>
<td>KEYFIELD</td>
<td>/SCWM/DE_KEYFIELD</td>
<td>Key Field Name the service uses to identify the object</td>
</tr>
<tr>
<td>LIST_FIELD</td>
<td>/SCWM/DE_LIST_FIELD</td>
<td>Name of Field in Output List (List View Structure Field)</td>
</tr>
<tr>
<td>CONST_VAL</td>
<td>/SCWM/DE_MON_CONST_Val</td>
<td>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</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>List Structure</th>
<th>Hotspot</th>
<th>Service</th>
<th>Presentation Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_S_WIP_WHRHEAD</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td></td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>List Structure</th>
<th>Hotspot</th>
<th>Service</th>
<th>Key Field</th>
<th>List View Field</th>
<th>Constant Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZHTG_S_WIP_WHRHEAD</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td>DOCCAT</td>
<td></td>
<td>PDI</td>
</tr>
<tr>
<td>ZHTG_S_WIP_WHRHEAD</td>
<td>DOCNO_H</td>
<td>DLV</td>
<td>DOCNO</td>
<td>DOCNO_H</td>
<td></td>
</tr>
</tbody>
</table>
8. Appendix

8.1 Sample Coding

8.1.1 Master Program SAPLZHTG (Function Pool)

Source Coding of Master Program SAPLZHTG

```plaintext
* 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

```plaintext
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,
   ssctfields.
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

CONSTANTS:
go_stm TYPE REF TO /scdl/cl_stm.  "#EC NEEDED

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 b1110.

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 b1110.

SELECT-OPTIONS:
  so_huno FOR /scwm/s_wip_q_whr_inbound-huno.

SELECTION-SCREEN:
  BEGIN OF BLOCK b1111.

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_sfprr FOR /scwm/s_wip_q_whr_inbound-partyno_sfprr_h,
  so_stprr FOR /scwm/s_wip_q_whr_inbound-partyno_stprrf_h,
  so_sflp FOR /scwm/s_wip_q_whr_inbound-locationno_sflp_h,
  so_dwal 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_dgrl FOR /scwm/s_wip_q_whr_inbound-status_value_dgr_i,
  so_deul FOR /scwm/s_wip_q_whr_inbound-status_value_deu_i,
  so_dptl FOR /scwm/s_wip_q_whr_inbound-status_value_dpt_i,
  so_dwhn FOR /scwm/s_wip_q_whr_inbound-status_value_dwn_h.

* dynamic status (item)
8.1.3 Include LZHTGP01

Source Coding of Sample Function Module LZHTGP01

*\$-----------------------------------------------*
*  Include  \_LZHTGP01

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 LINE,
  COMMENT 1(31) gv_txttdt FOR FIELD p_dlvdfr.
PARAMETERS:
  p_dlvdfr TYPE /scwm/s_wip_whrhead-dlv_date,
  p_dltvfr TYPE /scwm/s_wip_whrhead-dlv_time.
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_docid FOR /scwm/s_wip_q_whr_inbound-docid_h NO-DISPLAY.
SELECTION-SCREEN:
  END OF LINE,
  END OF BLOCK b1102.
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_docid FOR /scwm/s_wip_q_whr_inbound-docid_h NO-DISPLAY.
SELECTION-SCREEN:
  END OF BLOCK b1111,
  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.
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_include_data TYPE /scwm/dlv_query.incl.str.prд,
  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_field.cat TYPE /scwm/dlv_field.cat,
  select.huref
    CHANGING ct_aggr TYPE /scwm/tt_wip_whr.aggr,
  select.tu
    CHANGING
      ct_docid.tunum TYPE /scwm/tt_wip_whr.aggr,
  set.status.data
    IMPORTING
      it_status TYPE /scdl/dl_status.tab
    CHANGING
      cs_wrhhead 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

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

PRIVATE SECTION.

ENDCLASS.
```

8.1.4 Include LZHTGP02

Source Coding of Sample Function Module LZHTGP02

```plaintext
*---------------------------------------------------------------*
* 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
    ls_dattim_from = ls_date_time_from
    ls_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.
```
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,
    lt_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.
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_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.
ENDMETHOD.
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.
  ENDLDO.

*  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 ).
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
    lo_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.
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.
Finally specify the datasource (database, archive)

IF p_db IS INITIAL.
  CLEAR ls_select.
ENDIF.

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.
  END_CASE.
ENDIF.

IF ls_select IS NOT INITIAL.
  APPEND ls_select TO ct_selection.
ENDIF.

ENDMETHOD.

"fill_selection_table

METHOD map_whrhead_to_sel_opt.

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

<table>
<thead>
<tr>
<th>MOVE:</th>
<th>TO ls_mapping-tablename,</th>
</tr>
</thead>
<tbody>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_DOCTY'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'DOCTYPE_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_TCDREF'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'REFDOCNO_TCD_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_BOL'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'REFDOCNO_BOL_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_PPO'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'REFDOCNO_PPO_I'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_MANUAL'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'MANUAL_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_TU'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'TRANSMEANS_ID_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_HUNO'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'HUNO'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_DOOR'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'/SCWM/DOOR_I'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_ASN'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'REFDOCNO_ASN_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>TO ls_mapping-tablename,</td>
</tr>
<tr>
<td>/SCWM/S_WIP_Q_WHR_INBOUND'</td>
<td>TO ls_mapping-selname,</td>
</tr>
<tr>
<td>'S_PRO'</td>
<td>TO ls_mapping-fieldname,</td>
</tr>
<tr>
<td>'/SCWM/REFDOCNO_PRO_H'</td>
<td>TO ls_mapping-is_key.</td>
</tr>
<tr>
<td>abap_false</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping</td>
<td>TO ct_mapping.</td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_SFLO'</td>
</tr>
<tr>
<td></td>
<td>'LOCATIONNO_SFLO_H'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_STPRTF'</td>
</tr>
<tr>
<td></td>
<td>'PARTYNO_STPRTF_H'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_CARR'</td>
</tr>
<tr>
<td></td>
<td>'PARTYNO_CARR_H'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_DTRH'</td>
</tr>
<tr>
<td></td>
<td>'STATUS_VALUE_DTR_H'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_DGRI'</td>
</tr>
<tr>
<td></td>
<td>'STATUS_VALUE_DGR_I'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_DPTI'</td>
</tr>
<tr>
<td></td>
<td>'STATUS_VALUE_DPT_I'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'P_DBOI'</td>
</tr>
<tr>
<td></td>
<td>'STATUS_VALUE_DBO_I'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_TU_EX'</td>
</tr>
<tr>
<td></td>
<td>'TU'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>MOVE:</td>
<td>'/SCWM/S_WIP_Q_WHR_INBOUND'</td>
</tr>
<tr>
<td></td>
<td>'S_DEUI'</td>
</tr>
<tr>
<td></td>
<td>'STATUS_VALUE_DEU_I'</td>
</tr>
<tr>
<td></td>
<td>abap_false</td>
</tr>
<tr>
<td>APPEND</td>
<td>ls_mapping</td>
</tr>
<tr>
<td>CLEAR</td>
<td>ls_mapping.</td>
</tr>
<tr>
<td>Source Coding of Sample Function Module LZHTGP02</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'S_DUNI' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'STATUS_VALUE_DUN_I' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_false TO ls_mapping-is_key.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'S_DWAI' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'STATUS_VALUE_DWA_I' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_false TO ls_mapping-is_key.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'S_DWNH' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'STATUS_VALUE_DWN_H' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_false TO ls_mapping-is_key.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'S_DLVDT' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'TSTFR_TDELIVERY_PLAN_H' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_true TO ls_mapping-is_timestamp,</td>
<td></td>
</tr>
<tr>
<td>'P_DLVDFR' TO ls_mapping-p_date_from,</td>
<td></td>
</tr>
<tr>
<td>'P_DLVDFR' TO ls_mapping-p_time_from,</td>
<td></td>
</tr>
<tr>
<td>'P_DLVTO' TO ls_mapping-p_date_to,</td>
<td></td>
</tr>
<tr>
<td>'P_DLVTTO' TO ls_mapping-p_time_to,</td>
<td></td>
</tr>
<tr>
<td>'DLV_DATE' TO ls_mapping-date_field,</td>
<td></td>
</tr>
<tr>
<td>'DLV_TIME' TO ls_mapping-time_field.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'S_TUDT' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'TSTFR_TU' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_true TO ls_mapping-is_timestamp,</td>
<td></td>
</tr>
<tr>
<td>'P_TUDFR' TO ls_mapping-p_date_from,</td>
<td></td>
</tr>
<tr>
<td>'P_TUTFR' TO ls_mapping-p_time_from,</td>
<td></td>
</tr>
<tr>
<td>'P_TUDTO' TO ls_mapping-p_date_to,</td>
<td></td>
</tr>
<tr>
<td>'P_TUTTO' TO ls_mapping-p_time_to,</td>
<td></td>
</tr>
<tr>
<td>space TO ls_mapping-date_field,</td>
<td></td>
</tr>
<tr>
<td>space TO ls_mapping-time_field.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'PO_DSITY' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'STATUS_TYPE_ITEM' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_false TO ls_mapping-is_key.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_WIP_Q_WHR_INBOUND' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'SO_DSIVA' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'STATUS_VALUE_ITEM' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
<tr>
<td>abap_false TO ls_mapping-is_key.</td>
<td></td>
</tr>
<tr>
<td>APPEND ls_mapping TO ct_mapping.</td>
<td></td>
</tr>
<tr>
<td>CLEAR ls_mapping.</td>
<td></td>
</tr>
<tr>
<td><strong>MOVE:</strong> '/SCWM/S_SP_Q_HEAD' TO ls_mapping-tablename,</td>
<td></td>
</tr>
<tr>
<td>'SO_ERPID' TO ls_mapping-selname,</td>
<td></td>
</tr>
<tr>
<td>'REFDOCN0 ERP I' TO ls_mapping-fieldname,</td>
<td></td>
</tr>
</tbody>
</table>

************************************************************************|
**Add additional selection criteria for ERP documents** |
************************************************************************|

"CLEAR ls_mapping."
**Source Coding of Sample Function Module LZHTGP02**

```abap
abap_false
TO ls_mapping-is_key.
APPEND ls_mapping
TO ct_mapping.
************************************************************************
ENDMETHOD.
```

```abap
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.
```

```abap
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.  

```

```
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.

```

```
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(ls_status)
ENDCASE.
```

```
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.
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
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
         ls_status_stm = ls_status_stm
      IMPORTING
         es_status_text = ls_status_text ).
   cs_whrhead-status_plan_putaway_dist
      = ls_status_text-status_value_text.
   ENDCASE.

ENDLOOP.
ENDMETHOD.

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_erpid 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_stpqrt LIKE LINE OF so_stpqrt,
   ls_tcdref LIKE LINE OF so_tcdref,
   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_logfename_c=>sc_doctype_h.
     APPEND ls_select TO ct_selection.
  ENDDO.

* Transport unit
  LOOP AT so_tu INTO ls_tu.
     CLEAR ls_select.
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_tcdref 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.
ENDDOUBLE.

* 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.
ENDDOUBLE.

* 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.
ENDDOUBLE.

* 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.
ENDDOUBLE.

*************
* 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.
ENDDOUBLE.

************************************************************************
ENDMETHOD.

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

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_inpparam TYPE /scwm/s_sp_qry_logfname_inp,
  ls_logfname_key TYPE /scdl/dl_logfname_key_str,
  ls_logfname_map TYPE /scdl/dl_logfname_map_str,
  ls_select TYPE /scwm/dlv_selection_str,
### Source Coding of Sample Function Module LZHTGP02

```plaintext
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.
  ENDDO.

* 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.
  ENDDO.

* 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.
  ENDDO.

* 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.
  ENDDO.

* 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.
  ENDDO.

* 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.
  ENDDO.

* 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 i = 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 i = 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.
```
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_stm = is_status_stm
      iv_language = sy-langu
    IMPORTING
      es_status_text = es_status_text ).
  CATCH /scdl/cx_stm.
    "#EC NO_HANDLER
  ENDTY.

ENDMETHOD.

ENDCLASS.

8.1.5  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_WRHHEAD_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_timestdate TYPE /scwm/s_tstmp_date_time,
  ls_transport TYPE /scdl/dl_transport_str,
  ls_whrhead  TYPE zhtg_s_wip_whrhead.

* Enhanced structure for EWM Monitor including field for ERP document
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
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
      lt_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_dlivto
IMPORTING et_timestamp = so_dlvd[]).

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

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
c_t_selection = lt_selection ).

Adjust selection table for wildcard searches -----------------
/scwm/cl_dlv_ui_services=>modify_wildcard_selections(
CHANGING
c_t_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
```

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_docid2.
so_docid-sign = 'I'.
so_docid-option = 'EQ'.
so_docid2-sign = 'I'.
so_docid2-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_docid2-low = 'ls_head-docid.
  APPEND so_docid2 TO so_docid2.
  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_docid2.
    lv_counter = 0.
  ENDF.
ENDIF.
ENDLOOP.

* Select data for remaining DOCIDs
IF NOT so_docid2 IS INITIAL.
  go_hlp_id=>select_huref(
    CHANGING
t_aggr = lt_aggr ).
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_goodsreceipt
      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.
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_unload_point = 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 ----------------------------------
lv_aggr2-/scwm/door = lv_door.
lv_aggr2-/scwm/unloading_point = lv_unload_point.
lv_aggr2-gr_date_plan = ls_timedate-date.
lv_aggr2-gr_time_plan = ls_timedate-time.
lv_aggr2-no_item = lv_no_item.
lv_aggr2-no_product = lv_no_product.
APPEND lv_aggr2 TO lt_aggr2.

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

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

* Set SAPEXIT data
  ls_whrhead-/scwm/priop = ls_head-sapexit-/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.
  ENDLLOOP.

* 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.
  ENDLLOOP.

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

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
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:

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

Tabstrip Selection Text:

<table>
<thead>
<tr>
<th>Name</th>
<th>Typ</th>
<th>Dictionary ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PO_DSITY</td>
<td>Status Type</td>
<td>X</td>
</tr>
<tr>
<td>P_ARCH</td>
<td>Archive</td>
<td>X</td>
</tr>
<tr>
<td>P_BOTH</td>
<td>Database and Archive</td>
<td>X</td>
</tr>
<tr>
<td>P_DB</td>
<td>Database</td>
<td>X</td>
</tr>
<tr>
<td>P_DBOI</td>
<td>Locked (Overall, Item)</td>
<td></td>
</tr>
<tr>
<td>P_DLVDFR</td>
<td>Planned Delivery Date (From)</td>
<td></td>
</tr>
<tr>
<td>P_DLVDTO</td>
<td>Planned Delivery Date (To)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Typ</td>
<td>Dictionary ref.</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>P_DLVTFR</td>
<td>Planned Delivery Time (From)</td>
<td></td>
</tr>
<tr>
<td>P_DLVTTO</td>
<td>Planned Delivery Time (To)</td>
<td></td>
</tr>
<tr>
<td>P_TUDFR</td>
<td>Selection Date TU (From)</td>
<td></td>
</tr>
<tr>
<td>P_TUDTO</td>
<td>Selection Date TU (To)</td>
<td></td>
</tr>
<tr>
<td>P_TUTFR</td>
<td>Selection Time TU (From)</td>
<td></td>
</tr>
<tr>
<td>P_TUTTO</td>
<td>Selection Time TU (To)</td>
<td></td>
</tr>
<tr>
<td>SOASN</td>
<td>ASN</td>
<td>X</td>
</tr>
<tr>
<td>SO_BOL</td>
<td>Bill of Lading</td>
<td>X</td>
</tr>
<tr>
<td>SO_CARR</td>
<td>Carrier</td>
<td>X</td>
</tr>
<tr>
<td>SO_DEUI</td>
<td>Putaway Stat. (Plan, Item)</td>
<td></td>
</tr>
<tr>
<td>SO_DGRI</td>
<td>Goods Receipt Status (Item)</td>
<td></td>
</tr>
<tr>
<td>SO_DOCNO</td>
<td>Inbound Delivery</td>
<td>X</td>
</tr>
<tr>
<td>SO_DOCTY</td>
<td>Document Type</td>
<td>X</td>
</tr>
<tr>
<td>SO_DOOR</td>
<td>Warehouse Door</td>
<td>X</td>
</tr>
<tr>
<td>SO_DPTI</td>
<td>Putaway Status (Item)</td>
<td></td>
</tr>
<tr>
<td>SO_DSIVA</td>
<td>Status Value</td>
<td>X</td>
</tr>
<tr>
<td>SO_DTRH</td>
<td>Transit Status</td>
<td>X</td>
</tr>
<tr>
<td>SO_DUNI</td>
<td>Unload (Item)</td>
<td></td>
</tr>
<tr>
<td>SO_DWAI</td>
<td>Whse Activity Status (Item)</td>
<td></td>
</tr>
<tr>
<td>SO_DWNH</td>
<td>Transit Procedure</td>
<td>X</td>
</tr>
<tr>
<td>SO_ERPID</td>
<td>ERP Document</td>
<td>X</td>
</tr>
<tr>
<td>SO_HUNO</td>
<td>Handling Unit</td>
<td>X</td>
</tr>
<tr>
<td>SO_MANU</td>
<td>Generated Manually</td>
<td>X</td>
</tr>
<tr>
<td>SO_PRO</td>
<td>PRO Number</td>
<td>X</td>
</tr>
<tr>
<td>SO_SFLO</td>
<td>Ship-From Location</td>
<td>X</td>
</tr>
<tr>
<td>SO_SFPRT</td>
<td>Ship-from</td>
<td>X</td>
</tr>
<tr>
<td>SO_STPRF</td>
<td>Final Ship-To Party</td>
<td>X</td>
</tr>
<tr>
<td>SO_TCDRF</td>
<td>TCD Process Number</td>
<td>X</td>
</tr>
<tr>
<td>SO_TU</td>
<td>Vehicle</td>
<td>X</td>
</tr>
<tr>
<td>SO_TU_EX</td>
<td>Transportation Unit</td>
<td></td>
</tr>
</tbody>
</table>

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
```
```
```
* Determine wave items of waves in status 'initial' (I) or 'hold' (H) that belong to selected outbound delivery order items

```abap
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 ).
```

```abap
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.
```

```abap
*******************************************************************************
* 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.
```

```abap
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
      ls_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.