

How to Develop Programs for SAP Mobile RF



Applies to:

SAP R3 4.6c and above. For more information, visit the [ABAP homepage](#).

Summary

This article will help you how to develop programs for SAP Mobile RF

Author: Ankur Parab

Company: Capgemini Consulting India Pvt. Ltd

Created on: 20 June 2009

Author Bio

Ankur Parab has 5 years of experience in SAP ABAP and an overall experience of 5.5 years in IT.

Table of Contents

Summary.....	3
Introduction	3
Example Scenario	3
Prerequisites	3
Concepts.....	4
RF Screens	4
Development.....	7
Creation of the executable program:	7
Creation of the Screens	9
Creation of Function Modules to Call the Screens	13
Programming of the Screen	15
Subroutines to Call the Screen and Check the Data Coming from the Screen	17
Assign Transaction Code to the Program	18
Assign Transaction Code to the RF Menu	19
Related Content.....	20
Disclaimer and Liability Notice.....	21

Summary

This article will help you how to develop programs for SAP Mobile RF

Introduction

In warehouse management there is always a requirement for faster processes and up to date information.

For this warehouse operators make use of RF devices to capture the various data such as storage unit number, the material number; the quantity etc.

Typical RF devices are handheld terminals, barcode scanners and truck mounted terminals.

These RF devices basically have small screens and therefore the information sent to them is limited and just up to the need. Also the interaction needed must be minimum so that the processed can be made faster.

Basically in any requirement for SAP Mobile RF there will be 2 scenarios:-

- Making changes in standard SAP transaction using user exits.
- Making custom development.

This article will basically concentrate on how to develop custom programs.

Example Scenario

Consider an example scenario as follows:

You have an inbound delivery with handling units. The goods receipt for the same should be done.

For this the GR transfer order should be created; it should be confirmed and a put away transfer order should be created for the same.

All these 3 processes should happen with a single transaction without any break.

The standard RF transaction for put away by TO is LM03. The standard RF transaction for Goods receipt by delivery is LM76.

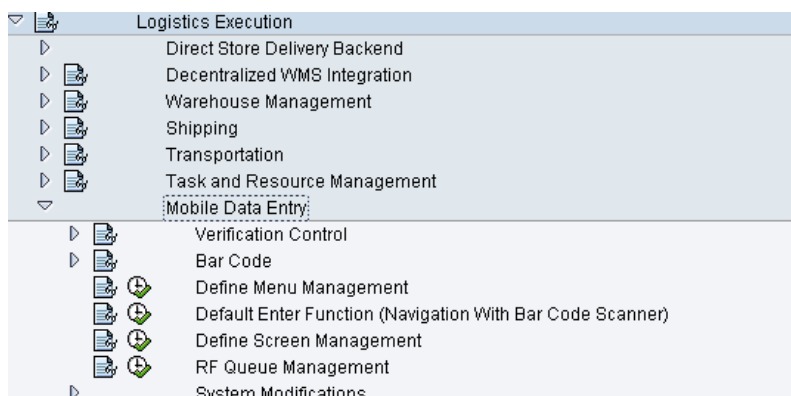
Thus in order to archive our solution we will have to merge together the processes of several RF transactions and create a single RF transaction.

Therefore we will have to develop a custom RF transaction which would merge all the 3 processes.

Prerequisites

Prior to develop any RF program please go through the documentation at the following IMG details in SPRO.

SPRO--> Logistics Execution--> Mobile Data Entry



Concepts

There are 2 parts involved in developing custom RF transactions.

- Development
- Customization

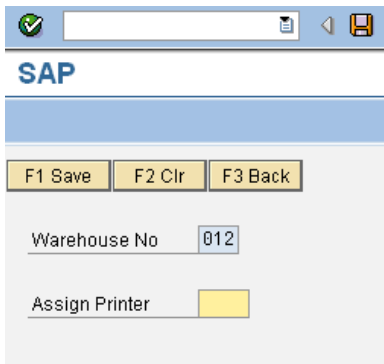
RF Screens

There are 2 size of screen which would be required to be developed.

8X40 and 16X20. The RF devices used will have screens of either of the 2 sizes so you will have to design both in order to serve for any type of RF device.

The examples for the screens are as follows:-

This is the 8X40 format

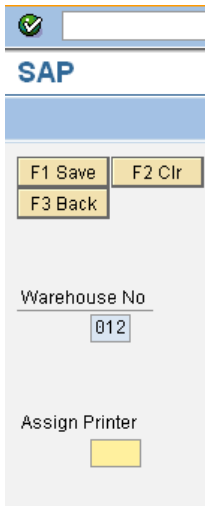


Technically the screen will be as follow:-

The screen size is of 8 rows and 40 columns

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40				
1	F	1	S	a	v	e					F	2	C	l	r					F	3	B	a	c	k																			
2																																												
3		W	a	r	e	h	o	u	s	e	N	°								1																								
4																																												
5		A	s	s	i	g	n	P	r	i	n	t	e	r						2																								
6																																												
7																																												
8																																												

This is the 16X20 format.



Technically the screen will be as follows:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		F	1	S	a	v	e					F	2	C	l	r				
2		F	3	B	a	c	k													
3																				
4																				
5		W	a	r	e	h	o	u	s	e	N	°								
6																				
7									1											
8																				
9																				
10		A	s	s	i	g	n		P	r	i	n	t	e	r					
11																				
12									2											
13																				
14																				
15																				
16																				

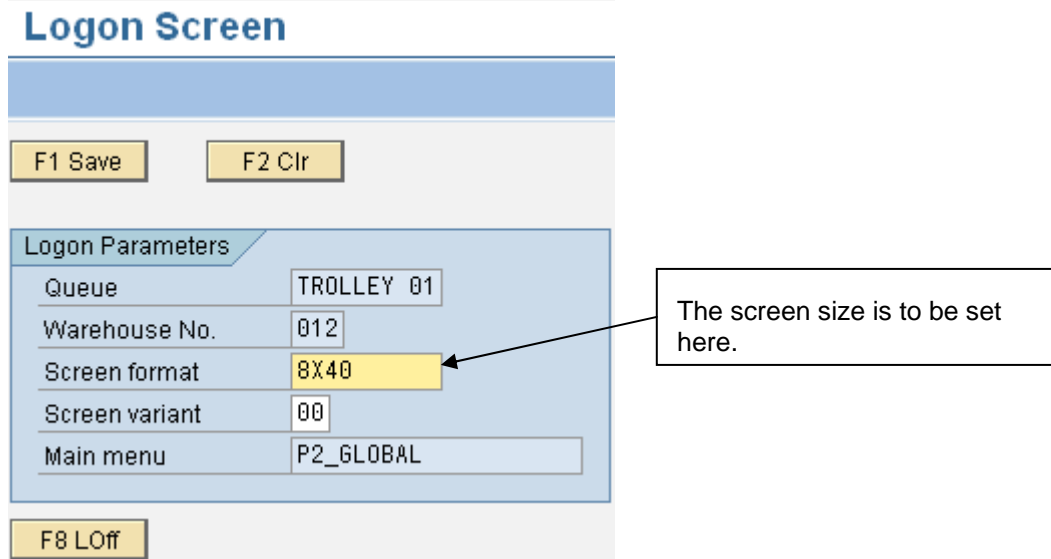
The screen size is of 16 rows and 20 columns

The choice of the screen depends upon the device which will be used. Both the screens will have to be designed.

Depending upon the RF device used the screens should be called.

The selection of which screen to be shown will be a customizing setting which is to be done in transaction **LM00**.

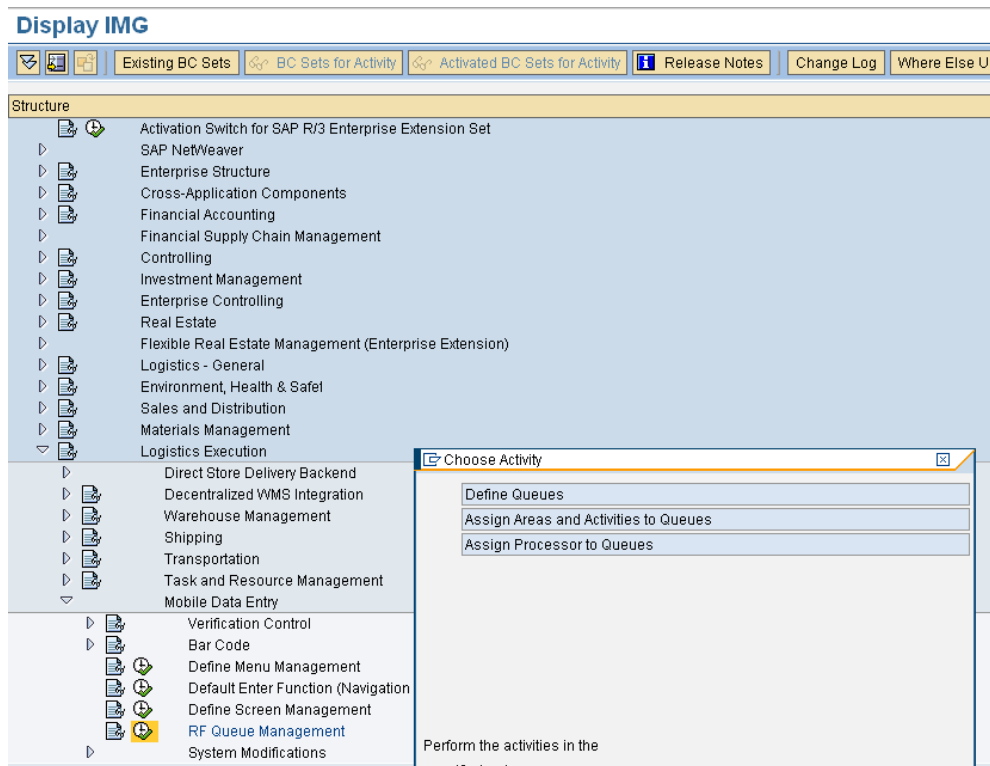
Transaction LM00



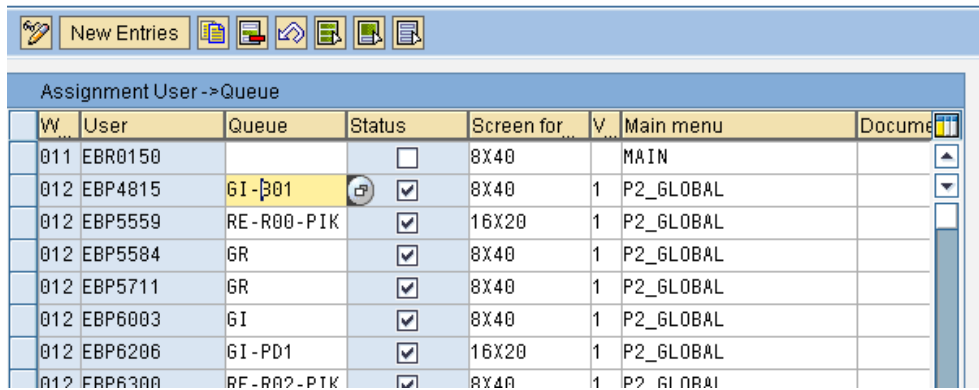
Once the screen format is set; it will remain the same throughout and will only change if it changed once again via LM00.

Alternately you can also specify the screen size for the user via the following IMG path.

Logistics Execution-->Mobile Data Entry-->RF Queue Management-->Assign Processor to Queues



Change View "Assignment User ->Queue": Overview



W	User	Queue	Status	Screen for	V	Main menu	Document
011	EBR0150		<input type="checkbox"/>	8X40		MAIN	
012	EBP4815	GI-301	<input checked="" type="checkbox"/>	8X40	1	P2_GLOBAL	
012	EBP5559	RE-R00-PIK	<input checked="" type="checkbox"/>	16X20	1	P2_GLOBAL	
012	EBP5584	GR	<input checked="" type="checkbox"/>	8X40	1	P2_GLOBAL	
012	EBP5711	GR	<input checked="" type="checkbox"/>	8X40	1	P2_GLOBAL	
012	EBP6003	GI	<input checked="" type="checkbox"/>	8X40	1	P2_GLOBAL	
012	EBP6206	GI-PD1	<input checked="" type="checkbox"/>	16X20	1	P2_GLOBAL	
012	EBP6300	RE-R00-PIK	<input checked="" type="checkbox"/>	8X40	1	P2_GLOBAL	

Here you can assign users to the particular queue along with the specific screen sizes.

Development

An RF program basically involves calling to various screens and processing them.

The development should proceed in the following steps:-

- Creation of the executable program.
- Creation of the screens.
- Creation of the function modules to access the screens.
- Screen Programming.
- Subroutines to call the screen and check the data coming from the screen.
- Assigning Transaction code to the program.
- Assigning the transaction code to the RF Menu.

Creation of the executable program:

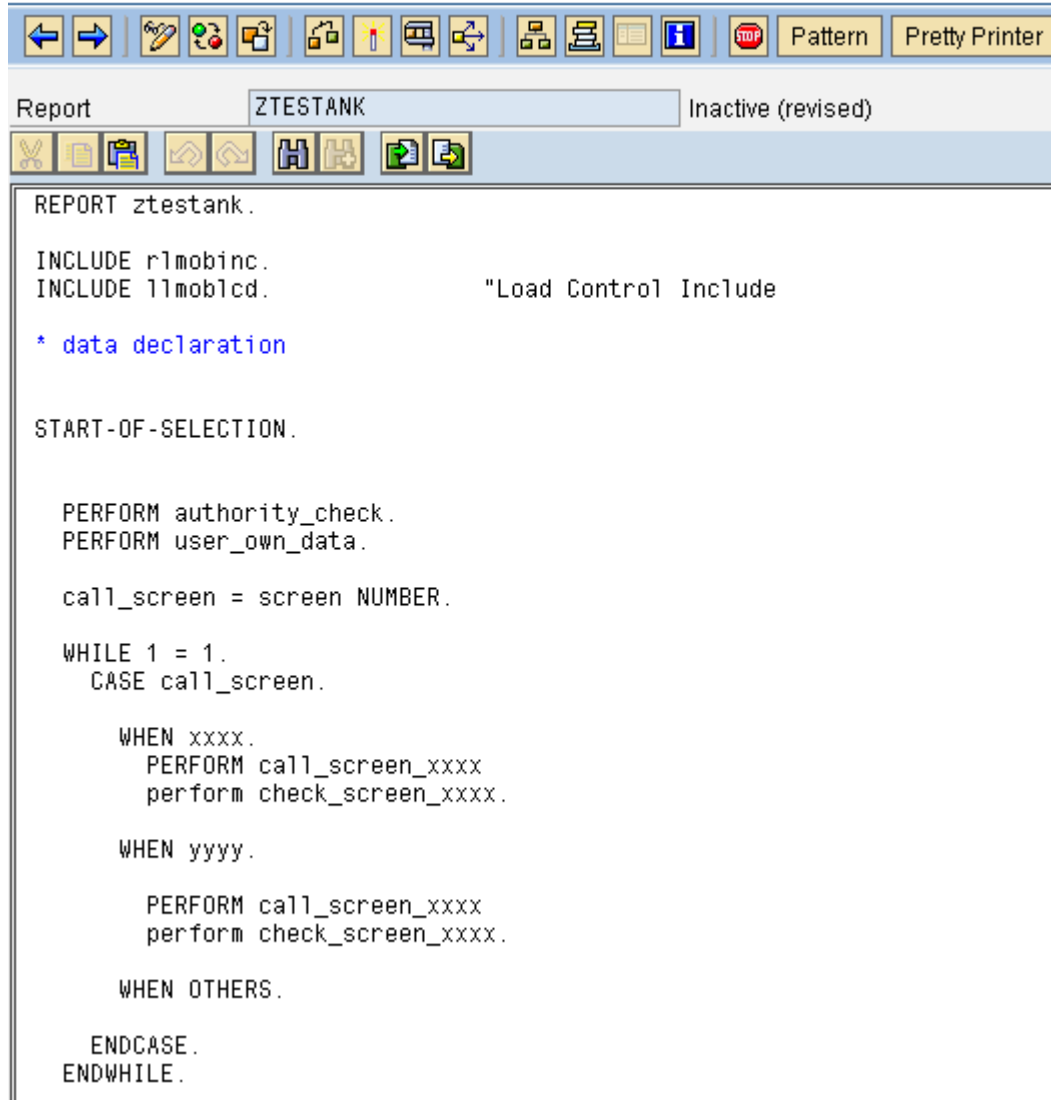
The basic program structure will be as follows:

There will be a single executable program which will be attached to a transaction.

In this program there would be a recursive calls to various screens and processing of the same.

The executable program will have a structure like this:

ABAP Editor: Change Report ZTESTANK



```

REPORT ztestank.

INCLUDE rlmobinc.
INCLUDE llmoblcd.          "Load Control Include

* data declaration

START-OF-SELECTION.

PERFORM authority_check.
PERFORM user_own_data.

call_screen = screen NUMBER.

WHILE 1 = 1.
  CASE call_screen.

    WHEN xxxx.
      PERFORM call_screen_xxxx
      perform check_screen_xxxx.

    WHEN yyyy.

      PERFORM call_screen_xxxx
      perform check_screen_xxxx.

    WHEN OTHERS.

  ENDCASE.
ENDWHILE.

```

The sequence of steps to be followed is as follows:

- Include the standard includes RLMOBINC and LLMOBLCD. These are needed so as to access the various global data which will be needed in the different function modules which will be used to do the processing such as goods movement, transfer order creation, transfer order confirmation etc.
- Check the authority of the user for the particular RF transaction using FM AUTHORITY_CHECK_TCODE as follows

```

call function 'AUTHORITY_CHECK_TCODE'
  exporting
    tcode = sy-tcode
  exceptions
    ok      = 0
    not_ok = 2
    others = 3.
if sy-subrc ne 0.
  message s172(00) with sy-tcode.
endif.

```


- Get the user own data. This is needed to know which format the user is going to use so that the screens can be called accordingly. This data also helps to know about the warehouse and queue to which the user is attached. Get the user's data using the FM 'L_USER_DATA_GET'. This can be done by making a call to subroutine USER_OWN_DATA in the standard include RLMOBOWN.
- As you can see the program basically consists of a recursive calls to different screens and processing them accordingly.
- Assign the first screen of the particular transaction to the variable CALL_SCREEN and then do a recursive call to the different screens within the WHILE-ENDWHILE loop. The screen number assigned to the variable call_screen will always be the logical screen.
- The subroutines CALL_SCREEN_XXXX will basically contain a Function Module which will call a particular RF screen. This Function module will have the importing parameters as the data which is to be passed to the RF screen and the exporting parameters will be basically the data which would be returned from the RF screen after the user has entered the same.
- The subroutines CHECK_SCREEN_XXXX will basically contain the processing logic which is to be done after user has entered some data on the RF screens and pressed any buttons.
- This kind of programming structure is followed so as to put minimum processing load on the RF screens.
- The screens would be just used as a means to capture data from the user. The actual processing would happen in the calling executable program in the subroutines CHECK_SCREEN_XXXX so as to fasten up the processes.

Creation of the Screens

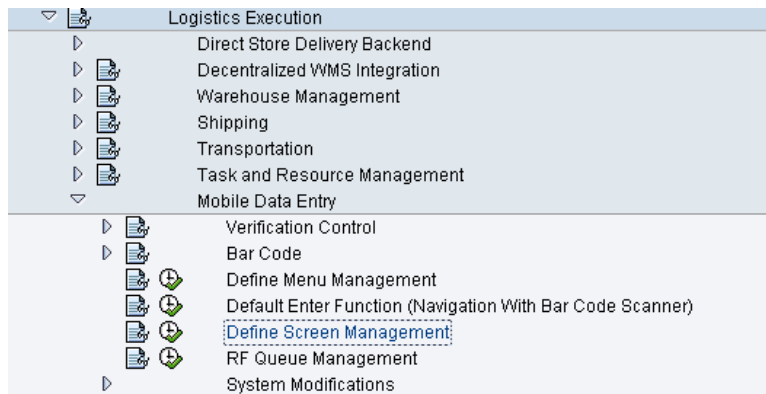
There will be 2 types of screen for any RF development:

Logical Screen and Actual Screen

This concept of logical screen and actual screen combined with the screen format helps us to attend to the needs of different RF devices.

The linkage of logical to actual screen is customized at the following IMG path.

Logistics Execution-->Mobile Data Entry-->Define Screen Management



Display View "Screen Management": Overview

Screen Management						
Mod. pool	ScrnFormat	V	Log	LogicalScr	Act.screen	
SAPLLMOB	16X20 Narrow	1	0	0769	Destination warehouse / bin location	2769
SAPLLMOB	16X20 Narrow	1	0	0777	Queue & warehouse modification	2777
SAPLLMOB	16X20 Narrow	1	0	0800	Load shipment	2800
SAPLLMOB	16X20 Narrow	1	0	0801	Load delivery	2801
SAPLLMOB	16X20 Narrow	1	0	0802	System-guided	2802
SAPLLMOB	16X20 Narrow	1	0	0803	Unload shipment	2803
SAPLLMOB	16X20 Narrow	1	0	0804	Unload delivery	2804
SAPLLMOB	16X20 Narrow	1	0	0805	Load overview	2805
SAPLLMOB	16X20 Narrow	1	0	0806	Details	2806
SAPLLMOB	16X20 Narrow	1	0	0807	Details	2807
SAPLLMOB	16X20 Narrow	1	0	0888	Dynamic menu	2888
SAPLLMOB	16X20 Narrow	1	0	0889	logon / menu screens	2889
SAPLLMOB	16X20 Narrow	1	0	0998	Warning message	2998
SAPLLMOB	16X20 Narrow	1	0	0999	Error message screen	2999
SAPLLMOB	16X20 Narrow	1	1	0302	Destination infos - Multiple materials	9303
SAPLLMOB	8X40 Large	for	0	0100	Select by storage unit	0100
SAPLLMOB	8X40 Large	for	0	0101	Collected	0101
SAPLLMOB	8X40 Large	for	0	0102	Select by TO	0102
SAPLLMOB	8X40 Large	for	0	0104	Select by delivery	0104
SAPLLMOB	8X40 Large	for	0	0105	Identify by MS Area	0105
SAPLLMOB	8X40 Large	for	0	0106	Identify by Shipment	0106
SAPLLMOB	8X40 Large	for	0	0107	Identify by Others	0107
SAPLLMOB	8X40 Large	for	0	0108	Identify by Group	0108
SAPLLMOB	8X40 Large	for	0	0151	Storage unit count	0151
SAPLLMOB	8X40 Large	for	0	0152	Storage bin count	0152
SAPLLMOB	8X40 Large	for	0	0153	Storage bin count details	0153
SAPLLMOB	8X40 Large	for	0	0170	Move handling unit	0170

Change View "Screen Management": Overview of Selected Set

Screen Management						
Mod. pool	ScrnFormat	V	Log	LogicalScr	Act.screen	
SAPLLMOB	16X20 Narrow	1	0	0104	Select by delivery	2104
SAPLLMOB	8X40 Large	for	0	0104	Select by delivery	0104

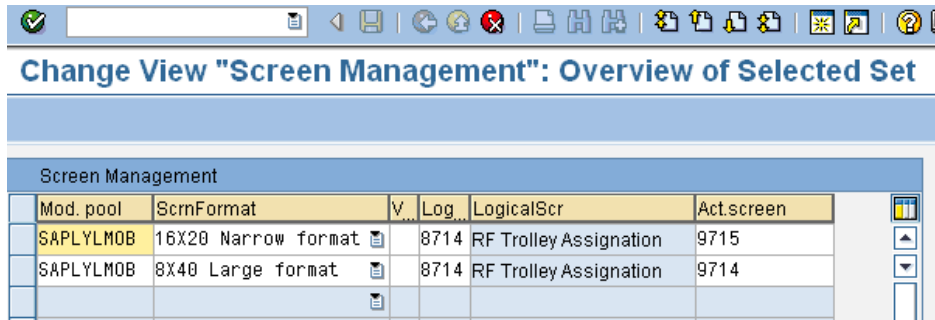
As see above there are 2 actual screens for 'Select by Delivery'. Screen no 2104 and Screen no 0104.

Screen 2104 is of size 16X20 and Screen 0104 is of size 8X40. They both are linked together by the common logical screen 0104.

Both the logical screen as well as the actual screen should physically exist for the particular module pool program.

Incase of custom RF developments we can keep screen numbers 8XXX for logical screen and screen number 9XXX for actual screens. Within the size formats the screen number can be odd or even.

For example:



The screenshot shows the SAP Screen Management tool interface. The title bar reads "Change View 'Screen Management': Overview of Selected Set". Below the title bar is a table with the following data:

Mod. pool	ScrnFormat	V...	Log...	LogicalScr	Act.screen
SAPLYLMOB	16X20 Narrow format		8714	RF Trolley Assignment	9715
SAPLYLMOB	8X40 Large format		8714	RF Trolley Assignment	9714

Here we have screen 8714 as a common logical screen for 'Trolley Assignment' screen and screen 9715 as 16X20 screen and screen 9714 as 8X40 screen. This is just a recommendation for easy understanding and better readability.

Physically all the 3 screens should be created. The size of the logical screen can be any of the two.

Please note that the logical screen will never be called.

As explained earlier there will be 2 screen formats which need to be developed for every screen.

The points to be noted while developing screens are as follow:-

- The overall layout for all the screens should be same.
- The placement of buttons should be same and the function keys associated with the buttons should be always the same so as to maintain consistency with the devices.
- For example the buttons SAVE, NEXT, CLEAR and BACK should be always on the top as in the RF device.
- The function codes associated with them should always be the same i.e F1 (Save), F4 (Next), F2 (Clear) and F3 (Back).
- This is because these buttons are the frequently accessed buttons.
- The function specific buttons such as F5 (Det), F6 (Diff) should be on the lower side of the screen.
- The buttons should always be used from the standard dictionary structure RLMOB.
- The GUI status can be copied from the screens of program SAPLLMOB.
- All the screens developed for your custom RF programs should belong to a single function group.
- This helps to maintain consistency.

Test Screen SAPLLMOB 2302

The screenshot displays the SAP Mobile RF interface for screen SAPLLMOB 2302. At the top, there is a header bar. Below it, a grid of function buttons is shown: F1 Save, F2 Clr, F3 Back, and F4 Nxt. Below the buttons is a section titled 'Destination info.' containing several input fields, including one labeled 'Dest.Bin'. The next section is 'Material information', which includes a 'Material' field with a '^' button, a '0' field, a 'v' button, a 'Batch' field, and two 's' fields. At the bottom, another grid of function buttons is shown: F5 Det, F6 Diff, F7 NBin, and Enter.

Frequently used general push buttons

Function Specific

Creation of Function Modules to Call the Screens

The subroutine CALL_SCREEN_XXXX will basically contain a function module which will call the particular actual screen from the corresponding logical screen.

```

*&-----*
*&      Form  y_call_screen_8709
*&-----*
* Quantity Check Screen
*-----*
* --> p1      text
* <-- p2      text
*-----*
form y_call_screen_8709 .

  call function 'Y_CALL_SCREEN_8709'
    exporting
      y_i_ltap          = y_wa_ltap
      y_i_matnr        = y_v_mat
      y_i_exidv        = y_v_exidv
      y_i_qty          = y_v_vemng1
      y_i_flag         = y_v_quan_flag
    importing
      y_o_screen_fcode = screen_fcode
      y_o_yqty         = y_v_act_qty
      y_o_mat          = y_v_mat
      y_o_flag         = y_v_quan_flag
    exceptions
      fail_in_calling_screen = 1
      fail_in_physical_screen_number = 2
      others                 = 3.
  case sy-subrc.
    when '01'.
*   Failed to call screen
      message_number = '109'.
      perform y_error_message.
      leave to transaction sy-tcode.
    when '02'.
*   Failed to determine screen number
      message_number = '190'.
      perform y_error_message.
      leave to transaction sy-tcode.
  endcase.
endform.          " y_call_screen_8709
*&

```

This function module will have different exporting and importing parameters as per the functionality needed on the screen. These function modules will serve as communicators between the screens and the executable programs. The data which is to be displayed on the screen will be passed to the importing parameters of the function modules. The actions performed on the screen and the relevant data entered on the screen will be returned back to the executable program from the screen via the exporting parameters.

Consider the function module details as follows:

Function module `Y_CALL_SCREEN_8709` Active

Attributes Import Export Changing Tables Exceptions Source code

```

*
*FIND THE PHYSICAL SCREEN FOR "Printer Assignment"
call function 'Y_DETERMINE_SCREEN_NUMBER'
  exporting
    lscrn      = '8709'
  importing
    pscrn      = pscrn
  exceptions
    screen_not_found = 1
    others        = 2.

if sy-subrc ne 0.
*.....Failed to determine screen number.....*
  message e190 raising fail_in_physical_screen_number.
endif.

lmap = y_i_lmap.
vekp-exidv = y_i_exidv.

if y_i_flag = 'X'.
  clear : rlmob-cqty.
else.
  perform y_f_shift_number using y_i_qty changing rlmob-cqty.
endif.

rlmob-cmatnr = y_i_matnr.
call screen pscrn.

clear y_o_flag.

set locale language sy-langu.
translate rlmob-cmatnr to upper case.

y_o_mat      = rlmob-cmatnr.
y_o_qty      = rlmob-cqty.
y_o_screen_fcode = screen_fcode.

```

Annotations:

- Get the physical screen from the logical screen
- Data from FM passed to the screen fields
- Call the physical screen
- Data from screen passed to the FM

As seen in the code above, the FM `Y_DETERMINE_SCREEN_NUMBER` is used to get the physical screen `pscrn` from the logical screen `lscrn`. To determine the physical screen from the logical screen we can make use of standard FM `DETERMINE_SCREEN_NUMBER`

The data to be passed to the different fields of the screen is populated into the fields from the importing parameters before making a call to the screen.

After the screen is called the data is passed from the fields of the screen to the exporting parameters of the FM.

The screen which will be called by the FM as shown:

The screenshot shows a mobile RF interface with the following elements:

- Top bar: F1 Save, F2 Clr, F3 Back
- Field: HU Barcode (linked to VEKP-EXIDV)
- Section: Material information
 - Field: Material (linked to RLMOB-CMATN)
 - Field: Quantity (linked to RLMOB-CQTY)
 - Field: Batch

Programming of the Screen

The screen will contain PBO and PAI modules as in a normal module pool program.

Screen Painter: Display Screen for SAPLYLMOB

The screenshot displays the SAP Screen Painter interface for the screen SAPLYLMOB. The screen number is 9712 and it is active. The 'Flow logic' tab is selected, showing the following modules:

```

process before output.
  module y_status_scr.
  module y_set_cursor.
  module y_change_verific_rfidpe.

process after input.
  field ok_code module y_user_command1.
  
```

As seen the PBO contains modules for setting the PF status, the cursor positioning and the verification field settings. The PAI contains the module for handling the user command.

The PAI module is as follows:

```

Include          LYLMOBI03          Active

module y_user_command1 input.
  get cursor field y_v_current_field.
  screen_fcode = ok_code.
  case ok_code .
    when y_k_clr.
      clear: ypadest, rlmob-cexidv, rlmob-cmatnr, rlmob-cqty.
    when y_k_back.
      clear : rlmob-cmatnr, rlmob-cqty.
      leave to screen 0.
    when y_k_save or y_k_next.
      perform y_f_check_qty_is_numeric.
  endcase.

endmodule.          " y_user_command1  INPUT
*&-----*
*&      Form  y_f_check_qty_is_numeric
*&-----*
* MOVE THE QUANTITY TO A NUMERIC FIELD
* INCASE EXCEPTION OCCURS THEN GIVE ERROR MESSAGE
*-----*
* --> p1      text
* <-- p2      text
*-----*

form y_f_check_qty_is_numeric .

  data : y_lv_tmp_qty type ltap_nsola,
        y_lv_ref     type ref to cx_root.

  try.
    move rlmob-cqty to y_lv_tmp_qty.

    catch cx_sy_conversion_no_number into y_lv_ref.
      message_id = 'YL01'.
      message_number = '015'.
      message_var1 = text-013.
      perform y_error_message.
      error_code = 1.
      clear : rlmob-cqty.
      exit.
    endtry.

  leave to screen 0.
endform.          " y_f_check_qty_is_numeric

```

As seen in the PAI module 'y_user_command1 input' you can see that within the screen not much of programming is done. Just the values are passed to the various fields and then the control is sent back to screen 0. The control then comes back to the statement after CALL SCREEN in the FM Y_CALL_SCREEN_XXXX. The screen ok_code is passed back as the exporting parameter of the FM.

Subroutines to Call the Screen and Check the Data Coming from the Screen

The subroutines Y_CALL_SCREEN_XXXX and Y_CHECK_SCREEN_XXXX basically manage the interaction between the executable program and the RF screens. Once the control returns back to the executable screen from the FM Y_CALL_SCREEN_XXXX; the actual processing which is to be done for whatever action the user has performed on the screen XXXX will be done in the subroutine Y_CHECK_SCREEN_XXXX.

The check subroutine for our previous screen is as follows:

```
*&-----*
*&      Form  y_check_screen_8709
*&-----*
* Handle User Command For Quantity Check Screen
*-----*
form y_check_screen_8709 .

  case screen_fcode.
    when y_k_back.
      call_screen = y_k_8708.
      clear y_v_exidv.
    when y_k_save.
      perform y_f_mat_qty_check.
    when y_k_next.
      perform y_f_mat_qty_check.
  endcase. " CASE screen_fcode.

endform.          " y_check_screen_8709
```

As seen depending upon the value of the screen OK_CODE, the various processes will be carried out.

In case the TO has to be confirmed on pressing the SAVE; then the corresponding FM will be called within the subroutine Y_CHECK_SCREEN_XXXX and the next screen which should come up will be passed to the variable CALL_SCREEN of the executable program.

All the processing actions such as validation of materials or the HU entered or confirmation of TO etc should be carried out in the Y_CHECK_SCREEN_XXXX of that particular screen. Thus in this way the RF screens are not overloaded and the process flow is smooth and faster.

Thus in this way you can create a custom RF transaction with several screens as per the requirement to cover the various flows. All the functional processes such as creation of TO, confirmation of TO, creation of goods receipt should be achieved by making use of the appropriate standard function modules available in SAP.

Assign Transaction Code to the Program

Once the executable program is created you should assign a transaction code to it via SE93.

The screenshot displays the SAP SE93 'Display Report Transaction' interface. The main data area contains the following fields:

Transaction code	YLOPUPRE
Package	YLMOB
Transaction text	RF Putaway Preparation
Program	YLORFPUTPRER0100
Selection screen	1000
Start with variant	
Authorization object	

Below the main data area, there are two sections:

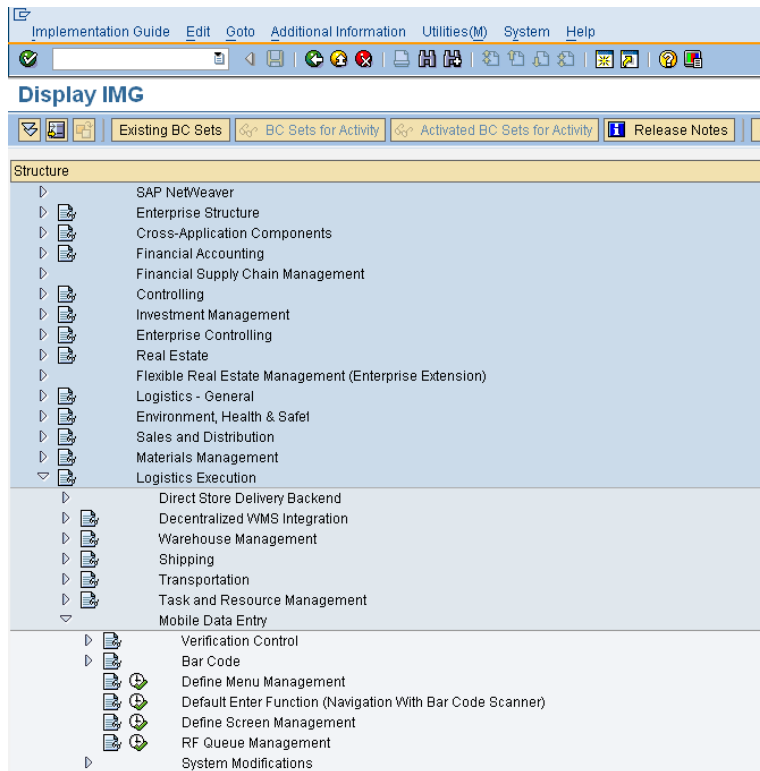
- Classification**
 - Transaction classification
 - Professional User Transaction
 - Easy Web Transaction
 - Pervasive enabled
 - Service:
- GUI support**
 - SAPGUI for HTML
 - SAPGUI for Java
 - SAPGUI for Windows

The transaction code will be of type Program and Selection Screen

Assign Transaction Code to the RF Menu

Finally assign the transaction code to the particular point in the RF menu via customizing.

Check with the functional consultants where to assign the particular transaction code.



Change View "Menu Selection": Overview

W	Dyn. menu	S	MenuTrms type	MenuTrms
012	LOAD	5	1	P2_STCKCOUNT
012	P2_PAL	1	2	YLORFOVERPACK
012	P2_PAL	2	2	LM18
012	P2_PAL	3	1	P2_STCKCOUNT
012	P2_PICK	1	2	YLORFPICK
012	P2_PICK	2	2	LM18
012	P2_PICK	3	2	LM12
012	P2_PICK	4	1	P2_STCKCOUNT
012	P2_PTW	1	2	YLOPUPRE
012	P2_PTW	2	2	YLOPUEX
012	P2_PTW	3	2	LM13
012	P2_PTW	4	2	YLORFONF
012	P2_PTW	5	1	P2_INFO
012	P2_PTW	6	1	P2_STCKCOUNT
012	P2_PTWEEXE	1	2	YLOPUEX
012	P2_PTWEEXE	2	2	LM13
012	P2_PTWEEXE	3	2	LM18
012	P2_PTWEEXE	4	1	P2_STCKCOUNT
012	P2_PTWPREP	1	2	YLOPUPRE
012	P2_PTWPREP	2	2	LM36
012	P2_PTWPREP	3	2	LM18
012	P2_PTWPREP	4	1	P2_STCKCOUNT
012	P2_SHIP	1	1	P2_PICK
012	P2_SHIP	2	1	P2_PAL
012	P2_SHIP	3	1	P2_LOAD
012	P2_STCKCOUNT	1	2	YLORFCOUNT

The concepts shown here will help you to design a custom program and transaction for any RF scenario which the customer wants to customize.

Related Content

http://help.sap.com/saphelp_47x200/helpdata/en/c9/c93237d5c2cf12e1000009b38f839/frameset.htm

http://help.sap.com/saphelp_47x200/helpdata/en/d1/801b50454211d189710000e8322d00/frameset.htm

http://help.sap.com/saphelp_47x200/helpdata/en/c6/f8386f4afa11d182b90000e829fbfe/frameset.htm

<https://www.sdn.sap.com/irj/scn/forums>

For more information, visit the [ABAP homepage](#).

Disclaimer and Liability Notice

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

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

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