PUBLIC

How-To: Enhance the MDG POWL for Multiple Work Item Processing - aka 'Mass Approval'

Applicable Releases:

All

Version 2.2

February 2023

Document History

Document Version	Description
1.0	First official release of this guide
2.0	Correction for more than two CR actions and performance optimizations
2.1	Minor changes
2.2	Adjust template, adjust sample code, adjust screenshots

Contents

1 Business Scenario

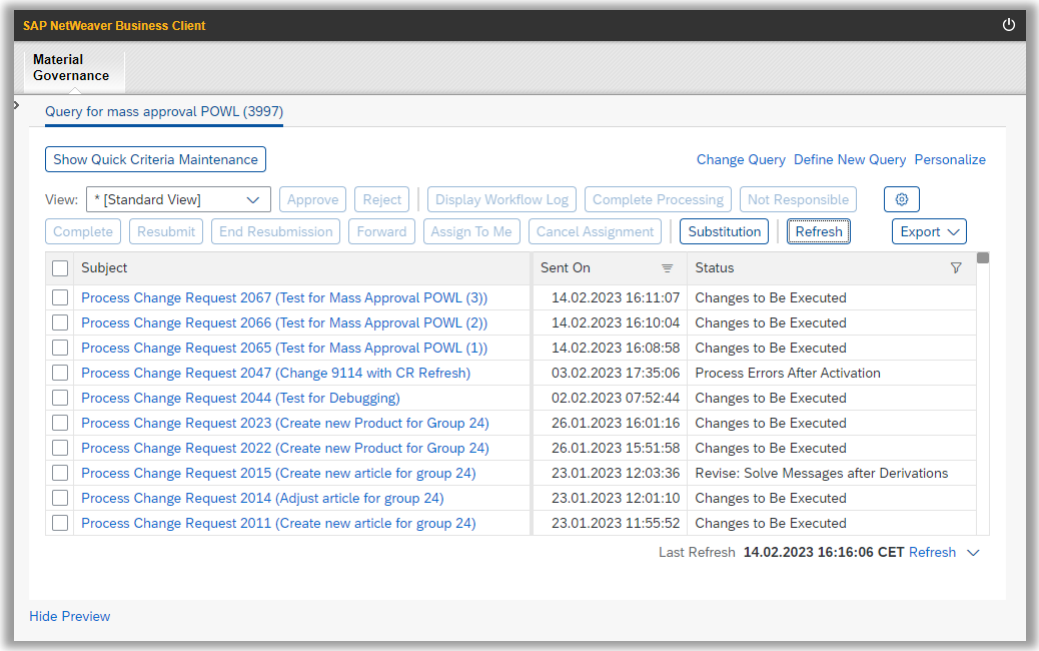
SAP Master Data Governance (MDG) provides business processes to find, create, change, and mark master data for deletion. It supports the governance of master data in a central hub and the distribution to connected operational and business intelligence systems.

The processes are workflow-driven and can include several approval and revision phases, and the collaboration of all users participating in the master data maintenance.

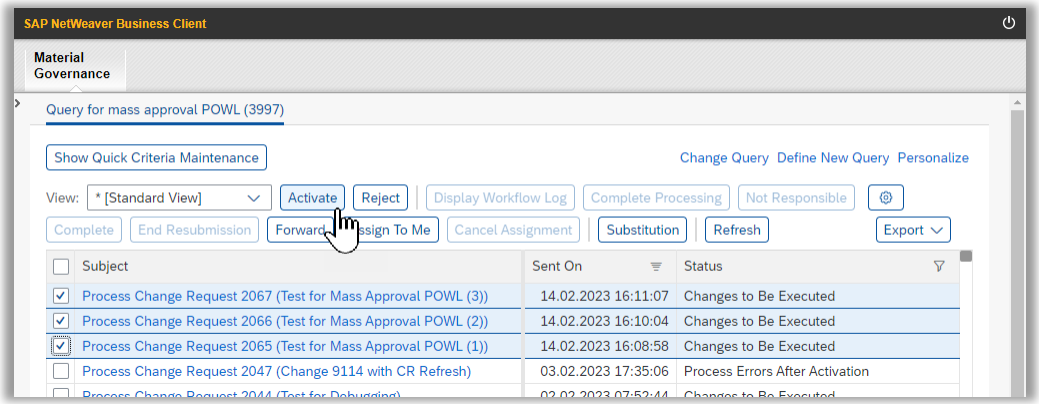
Customers with users that have many Master Data Governance tasks to fulfil asked for a more comfortable way to process multiple workflow tasks together. This is especially true for routine tasks, which customers don't want to automate but rather 'mass approve' more efficiently. This guide describes how you can define a 'Mass Approval Inbox' for SAP MDG standard domains.

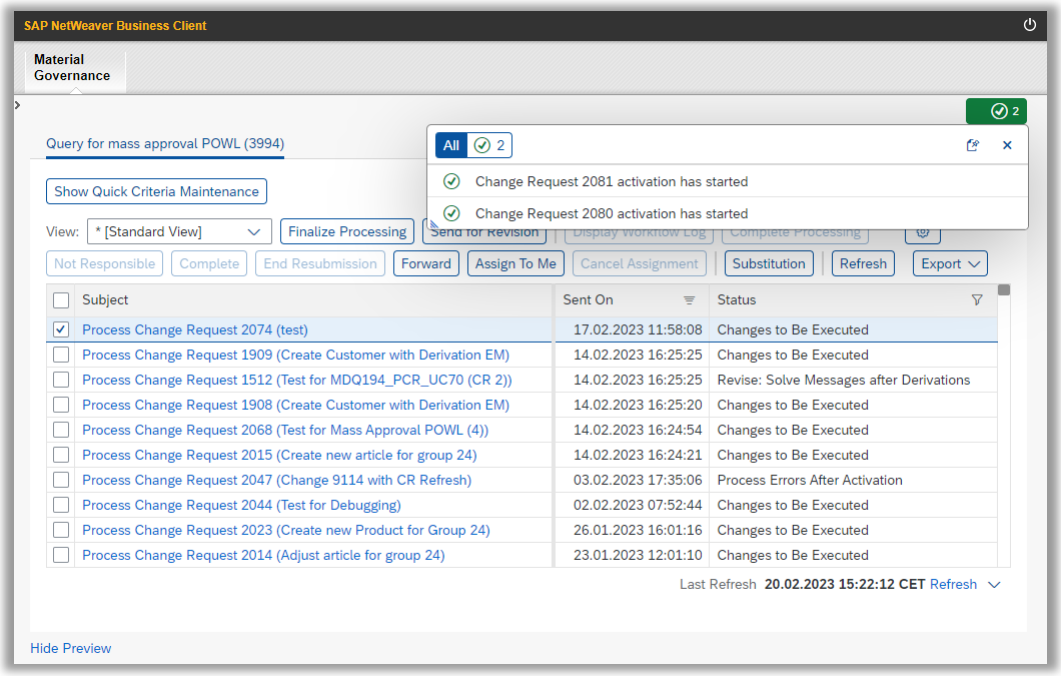
2 Expected Result

In the enhanced 'mass approval' inbox the user can select more than one task in the inbox (checkbox instead of radio button).

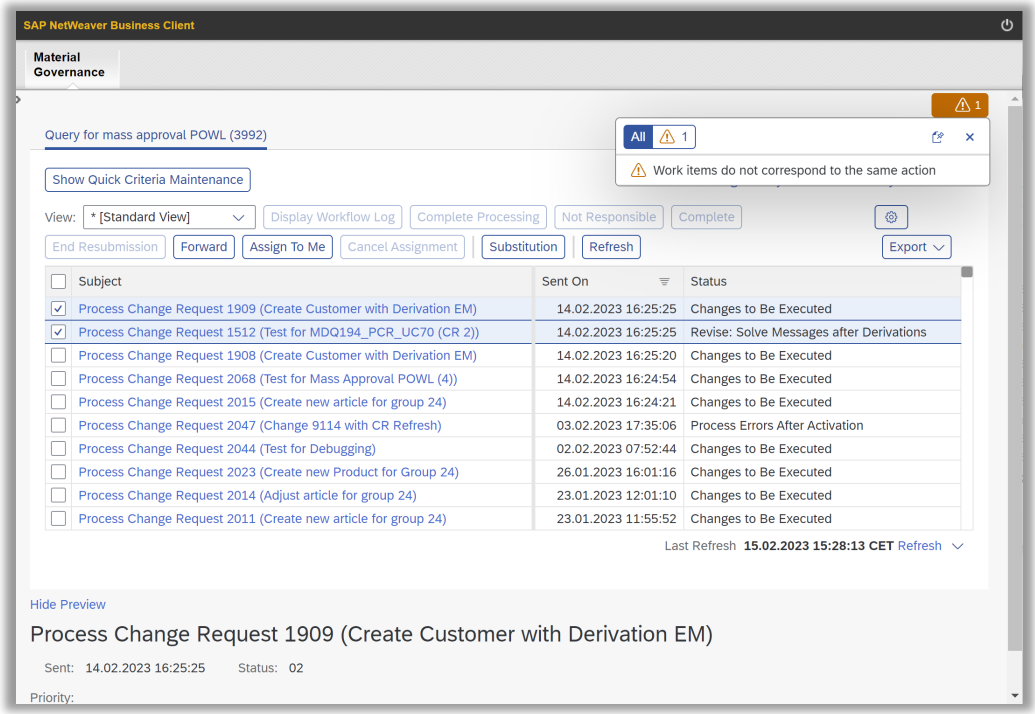


For example, multiple change requests could be approved. The user is informed which changes have been triggered.





Only tasks with identical (MDG) actions can be processed together, otherwise an error message is raised.



The mass approval inbox refreshed automatically after an action has been triggered.

3 Implementation

This chapter describes how to create a new Personal Object Worklist (POWL) configuration, related customizing, and how to enhance the coding to enable processing multiple workflow tasks in one step.

3.1 Create a new POWL configuration

3.1.1 Feeder class

Create a new empty feeder class (in this example `ZCL_CREQUEST_MASS_APRVL_POWL` with description Feeder class for Mass Approval POWL that inherits from `CL_USMD_CREQUEST_POWL` (refer to F1 and F2).

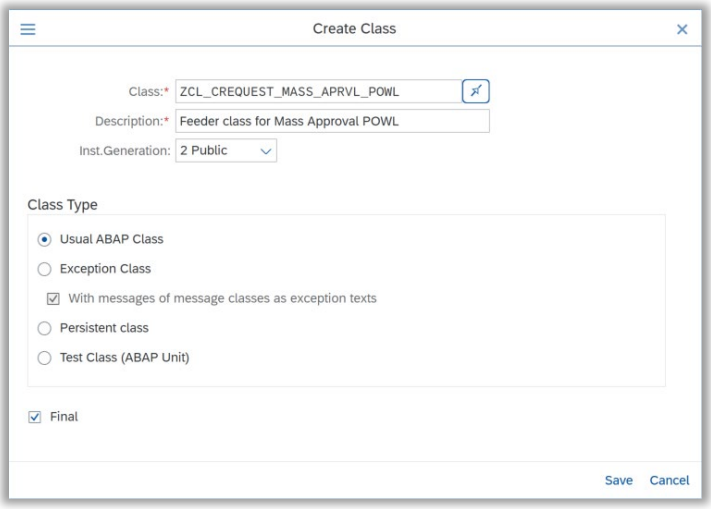


Figure 1 - Create feeder class

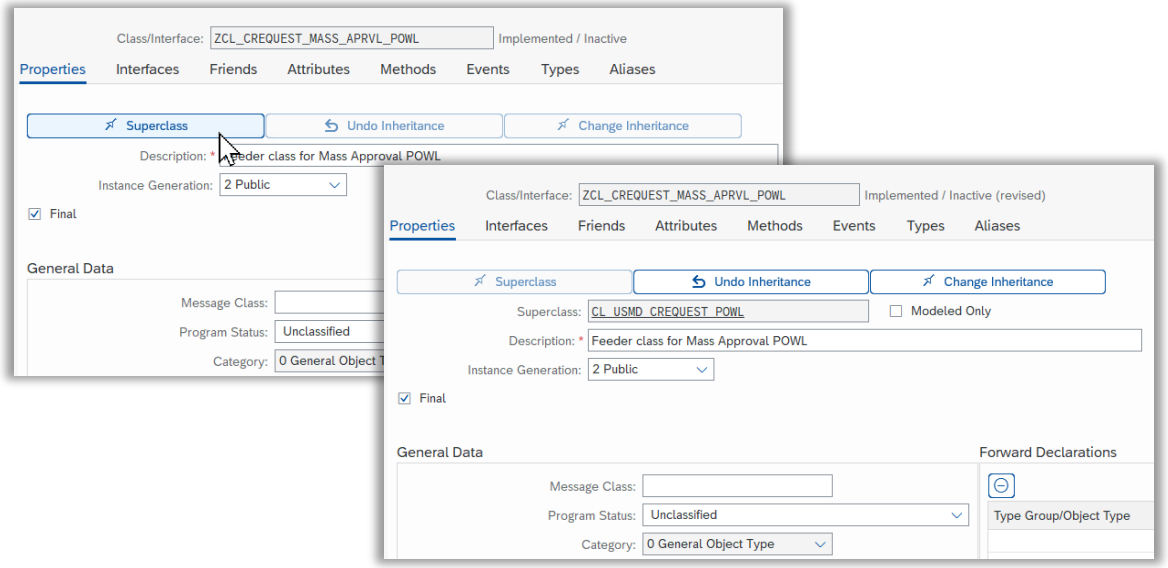
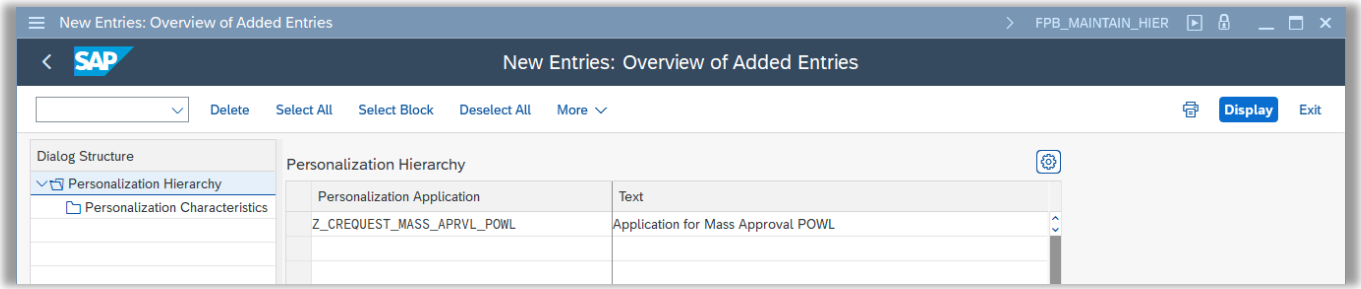


Figure 2

Save and activate the class.

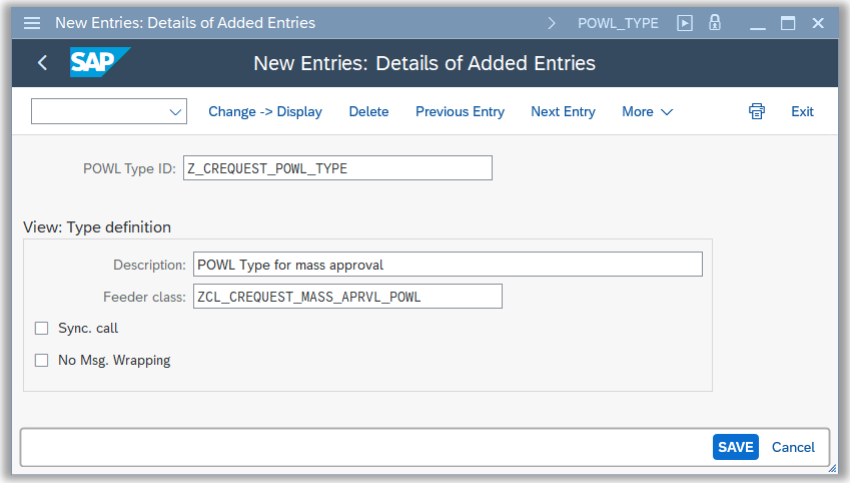
3.1.2 Application ID

Start transaction `FPB_MAINTAIN_HIER`. Create a new entry in view Personalization Hierarchy with Personalization Application is e.g. `Z_CREQUEST_MASS_APRVL_POWL` and text *Application for Mass Approval POWL*. Save your changes.



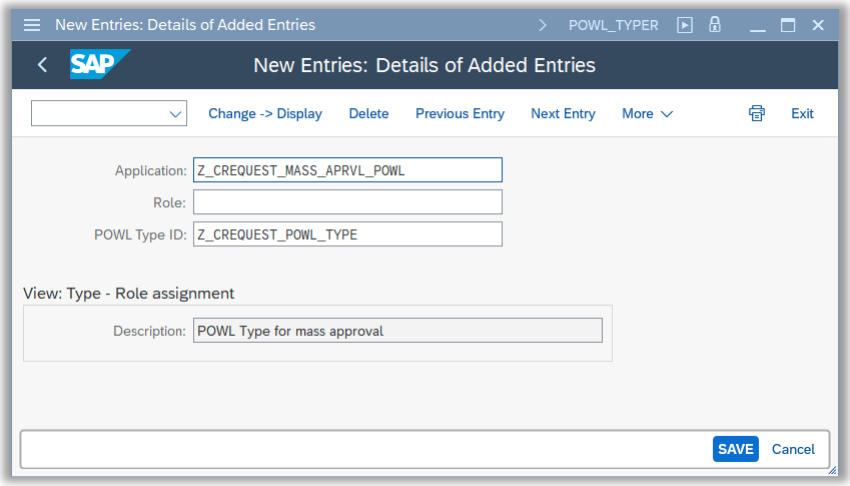
3.1.3 POWL Type

Start transaction `POWL_TYPE`. Create a new POWL Type ID, e.g. `Z_CREQUEST_POWL_TYPE` and description *POWL type for mass approval*. Fill in the previously created feeder class. Save your changes.



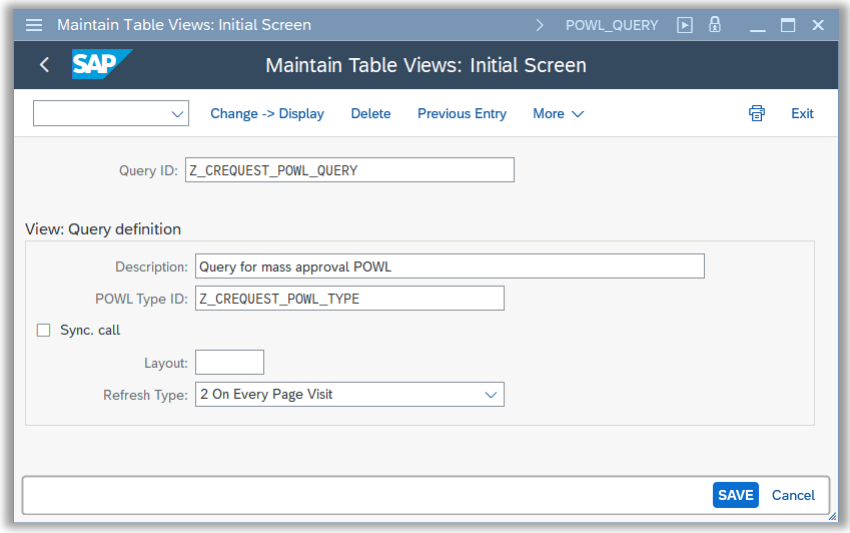
3.1.4 Application ID and POWL Type mapping

Start transaction `POWL_TYPER` (mind the ending 'R'). Create a new entry to map the previously created application ID to the POWL type. Keep the *Role* empty. Save your changes. (Unless the previous changes this change will be recorded in a customizing request.)

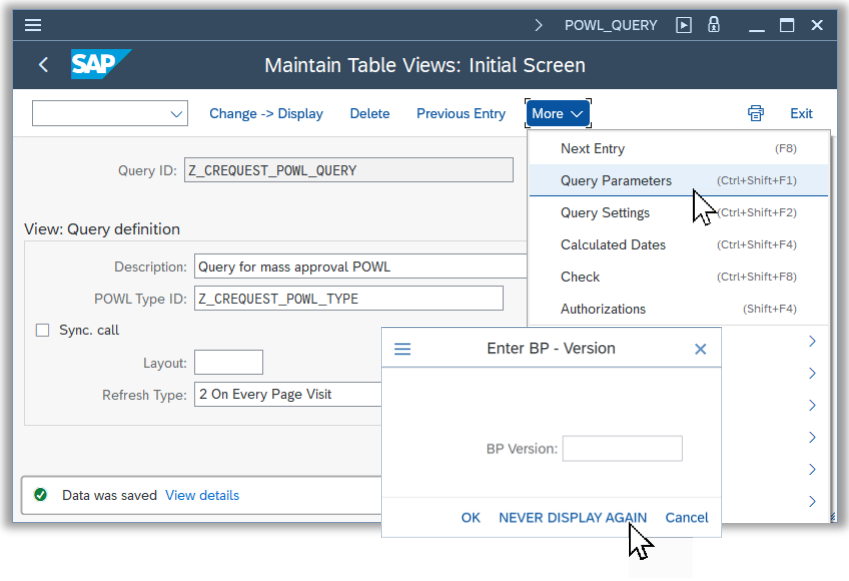


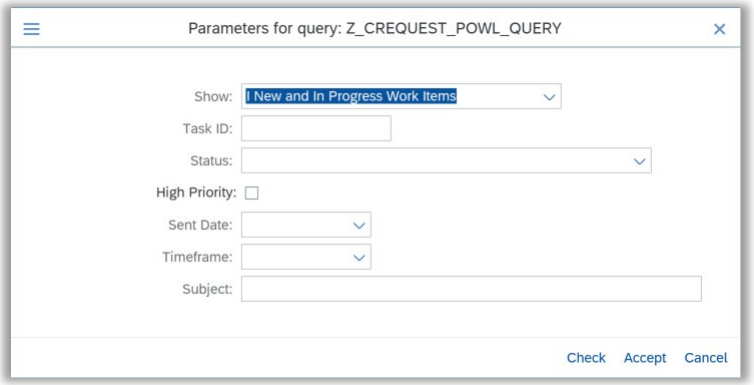
3.1.5 POWL Query

Start transaction POWL_QUERY. Create a new entry, e.g. `Z_CREQUEST_POWL_QUERY` with description *Query for mass approval POWL*. Assign the previously created POWL Type ID `Z_CREQUEST_POWL_TYPE`. For *Refresh Type* select *On Every Page Visit*. Save your changes.



Check that the query parameters are set to show work items which are in status new and in progress. In the menu press *Query Parameters*. If you are prompted at add a *BP Version* leave this input field empty and press *Never display again*.

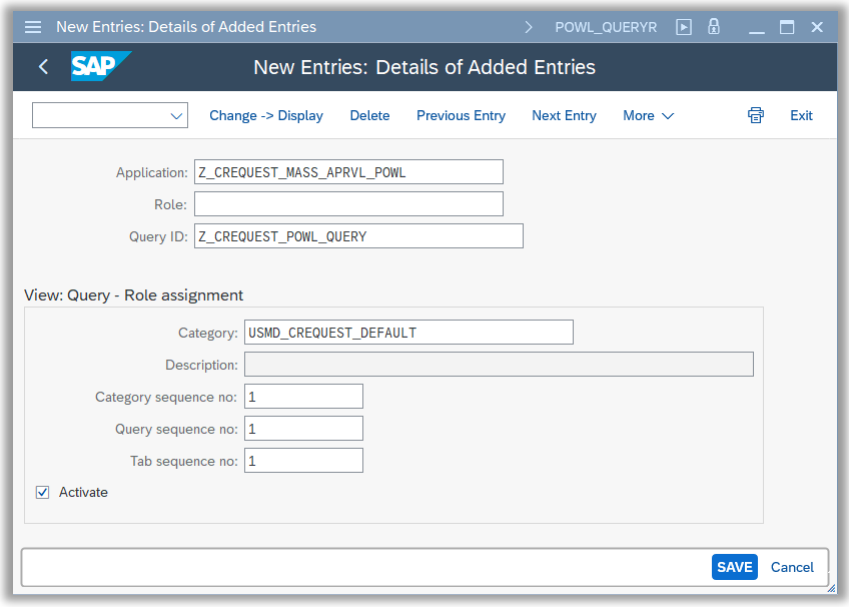




Check that *New and in Progress Work Items* is selected and press *Accept*. Again, save your changes. (If you have to delete an existing query, you can use program `POWL_D01`.)

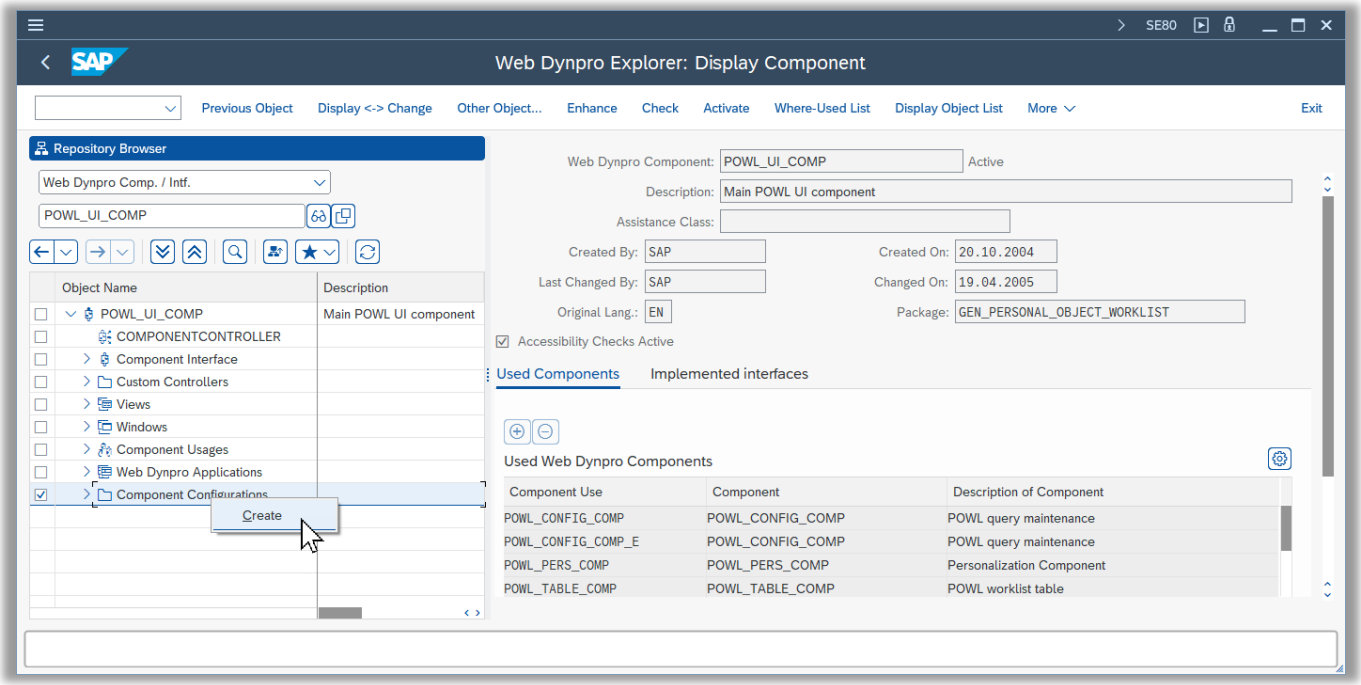
3.1.6 Application ID and Query Mapping

Start transaction `POWL_QUERYR`. Create a new entry to map the application ID `Z_CREQUEST_MASS_APRVL_POWL` to the query ID `Z_CREQUEST_POWL_QUERY`. Enter *Category* `USMD_CREQUEST_DEFAULT`. Set the *Category-*, *Query-* and *Tab sequence no.* to `1`. Select the *Activate* flag. Save your changes.

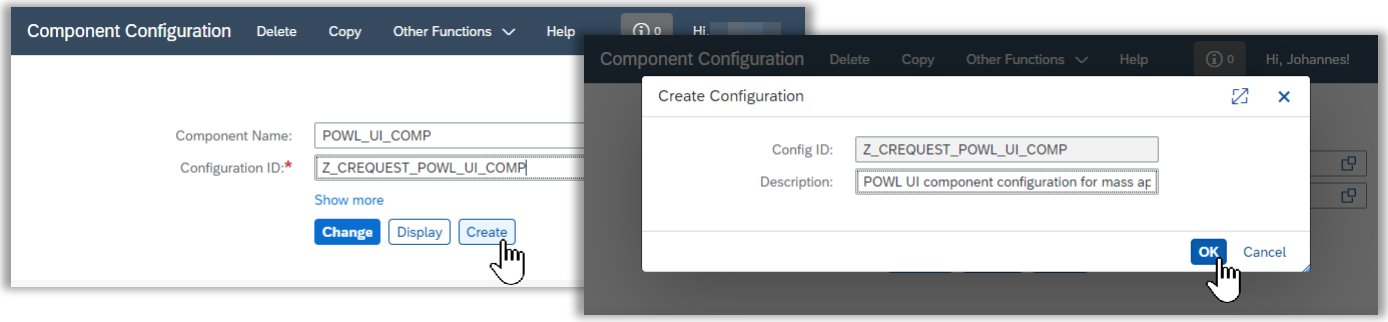


3.1.7 Component Configuration

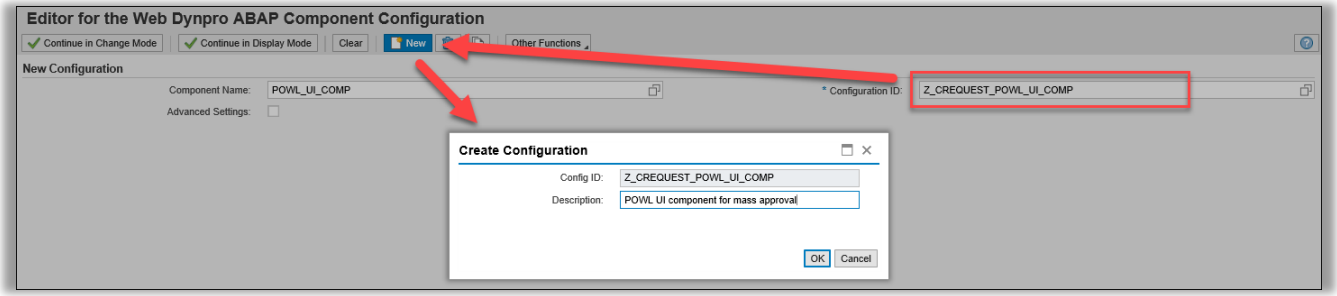
In transaction `SE80` open Web Dynpro Component `POWL_UI_COMP`. In the repository browser open the context menu on its *Component Configurations* and press *Create*.



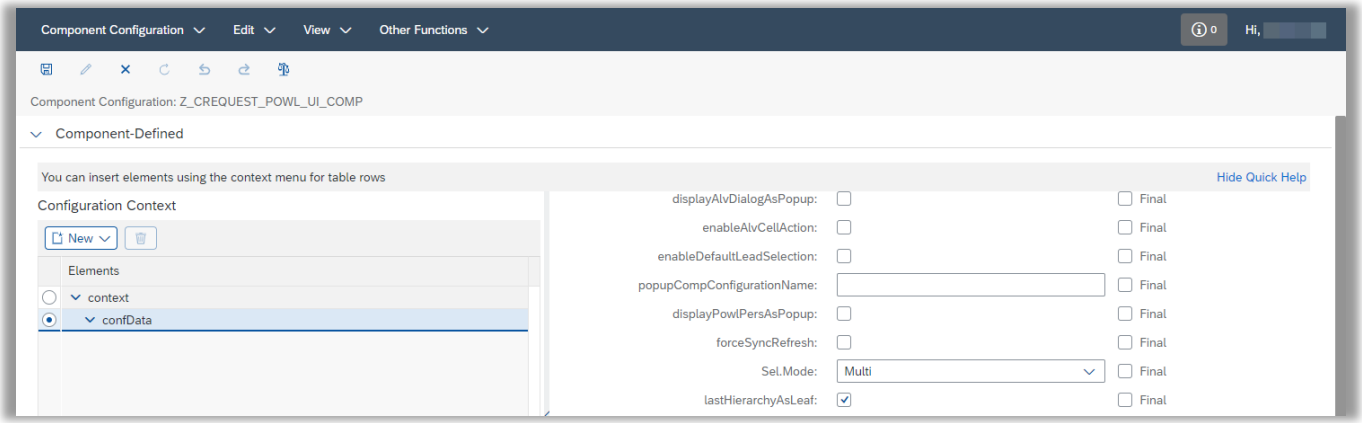
The Component Configuration is opened in a browser. Enter a *Configuration ID*, e.g. `Z_CREQUEST_POWL_UI_COMP`. Press *Create*. Enter a description, e.g. *POWL UI component configuration for mass approval*. Chose a suitable package.



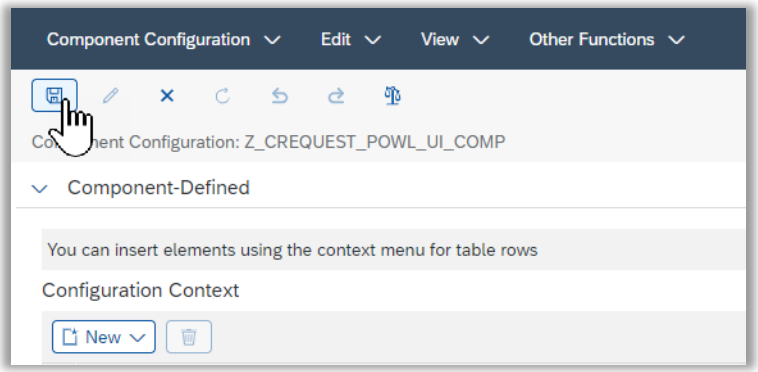
If you are not using a SAP Fiori compatible UI chose the following approach:



In the *Configuration Context* select node `confData`. Ensure the Selection Mode is *Multi*.

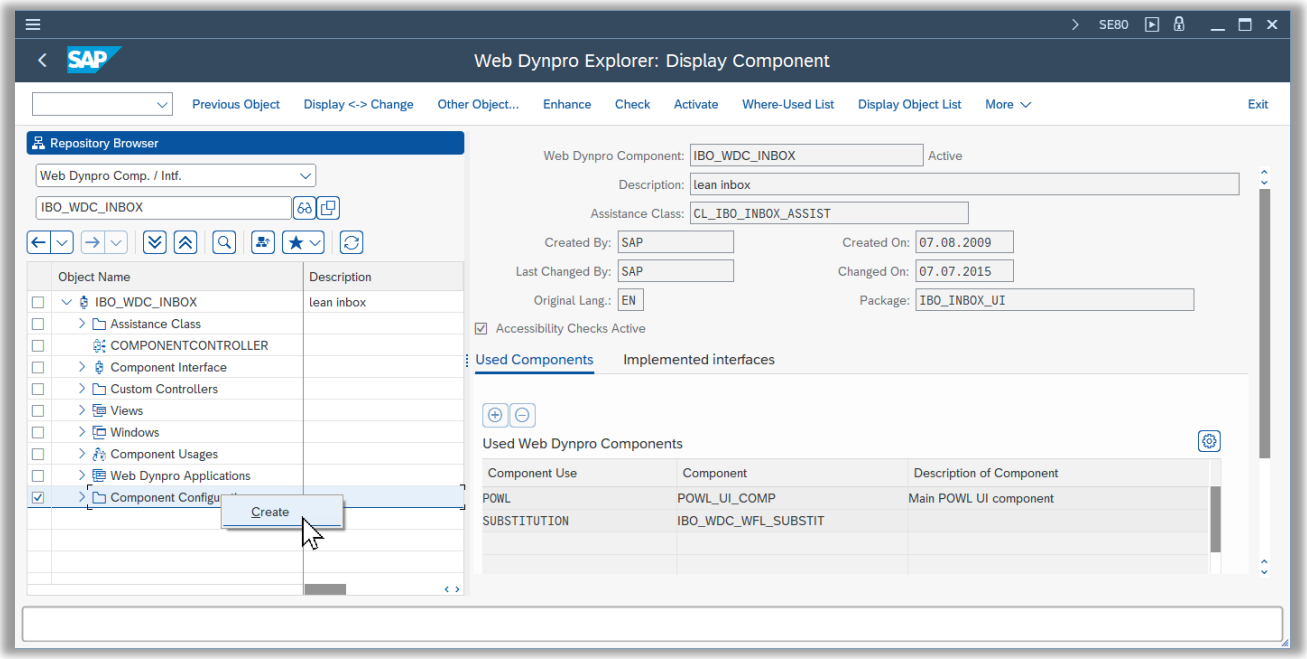


Press *Save*.

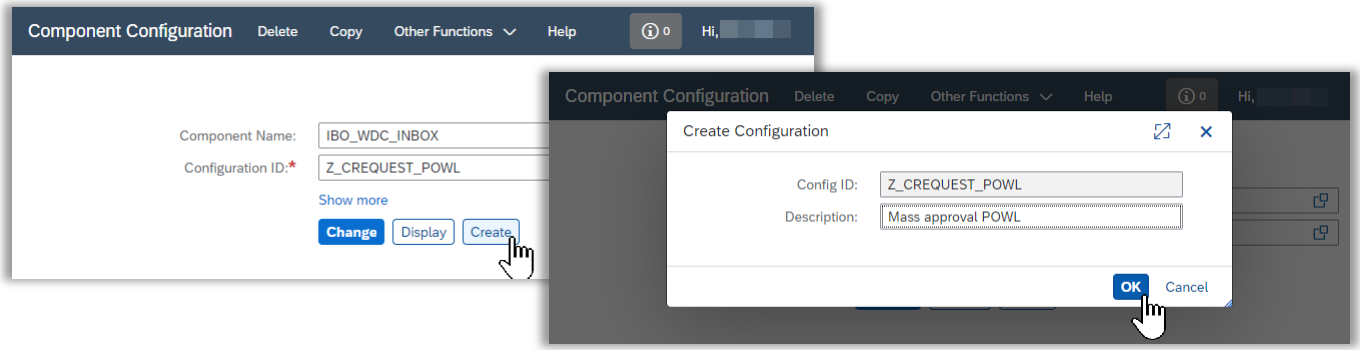


3.1.8 Integrate the POWL into the IBO_WDC_INBOX

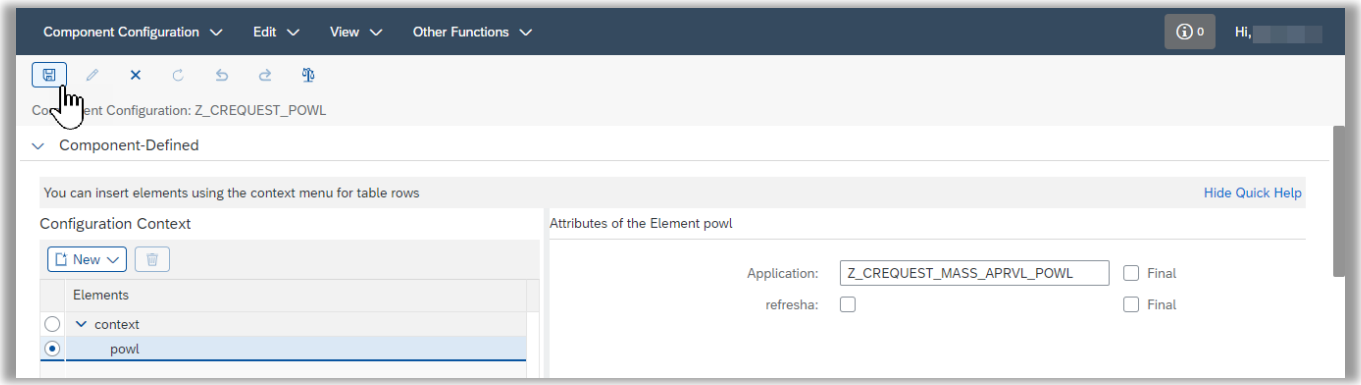
Start transaction `SE80`. Open Web Dynpro Component `IBO_WDC_INBOX`. Open the context menu on *Component Configurations* and press *Create*.



The Component Configuration application opens in a browser. Enter a configuration ID, e.g. `Z_CREQUEST_POWL`. Press *Create*. Enter a description, e.g. *Mass approval POWL*. Chose a suitable package.

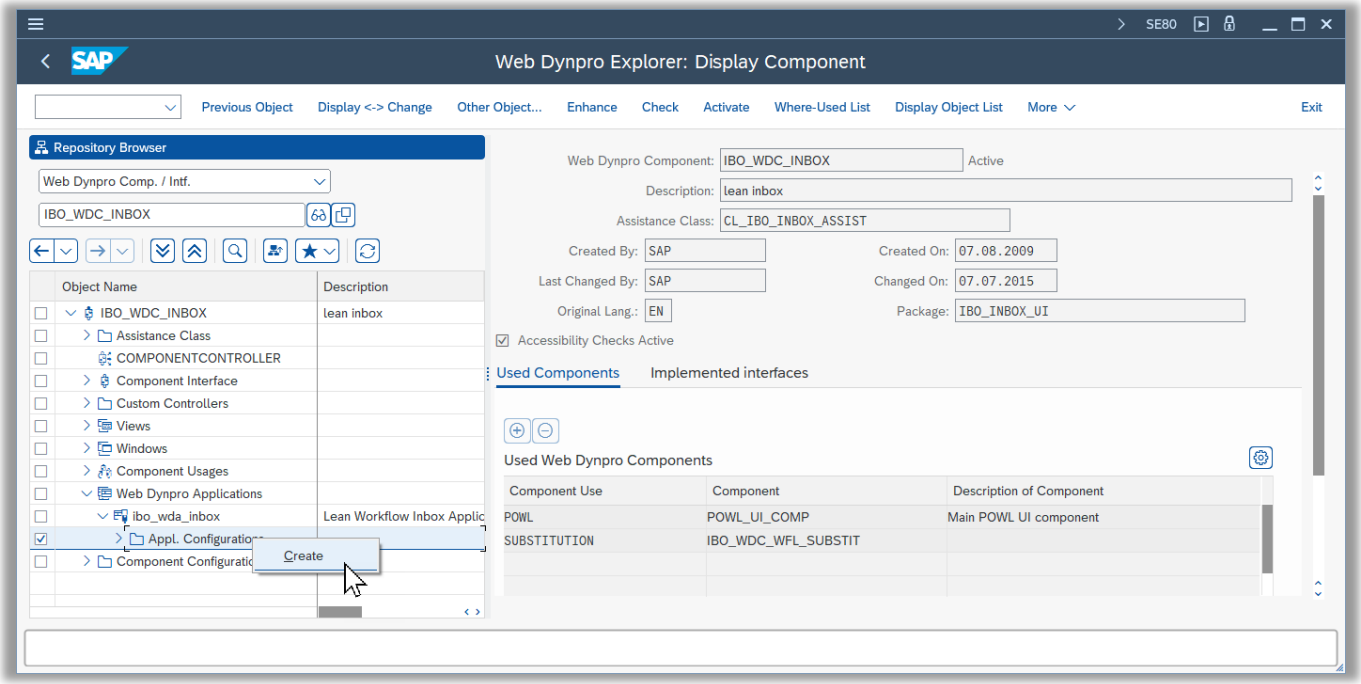


At the Configuration Context select node `powl`. Enter application which you created in 3.1.2, (in this example `Z_CREQUEST_MASS_APRVL_POWL`). Press *Save*.

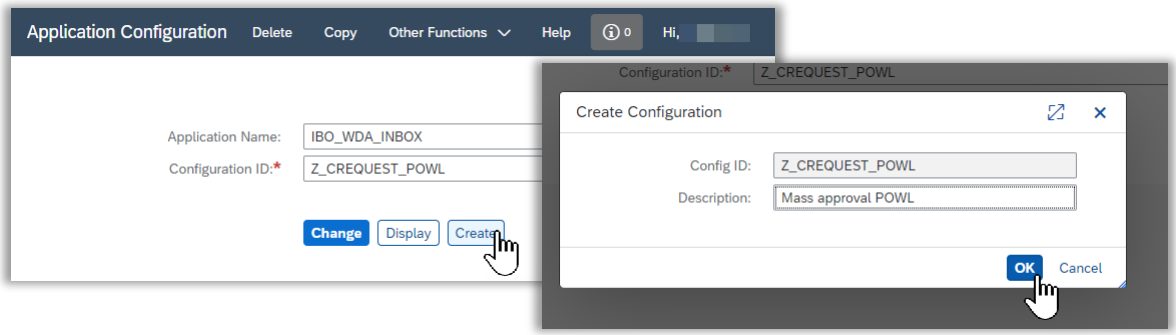


3.1.9 Create an Application Configuration

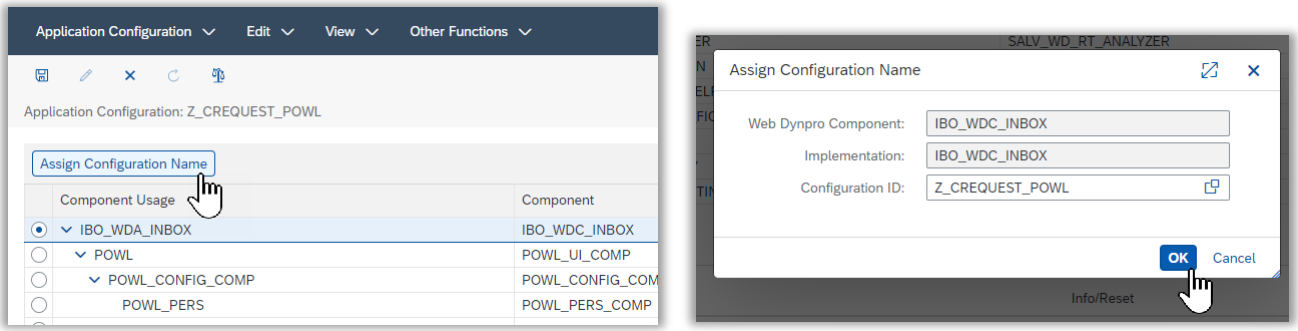
Go back to transaction `SE80` for Web Dynpro Component `IBO_WDC_INBOX`. Create a new application configuration for the Web Dynpro application `IBO_WDA_INBOX` using the context menu.



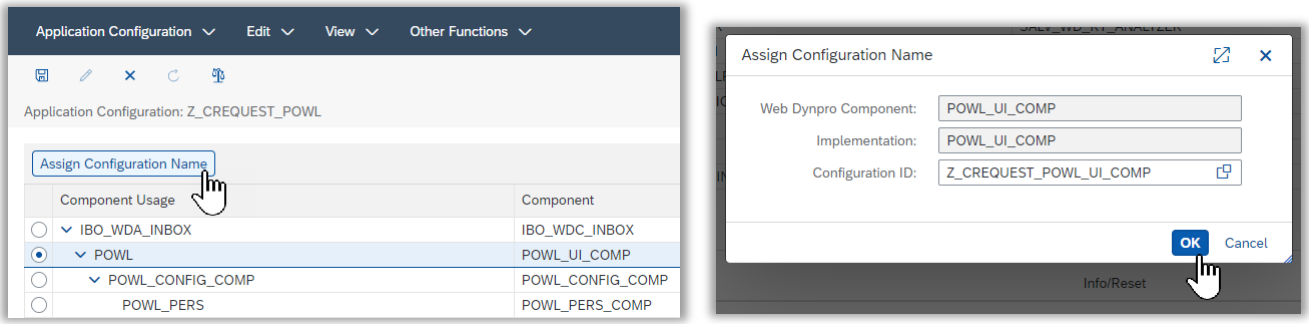
The Application Configuration opens in a browser. Enter *Application Name* `IBO_WDA_INBOX`. Enter a configuration ID, e.g. `Z_CREQUEST_POWL`. Press *Create*. Enter a description, e.g. *Mass approval POWL*. Chose a suitable package.



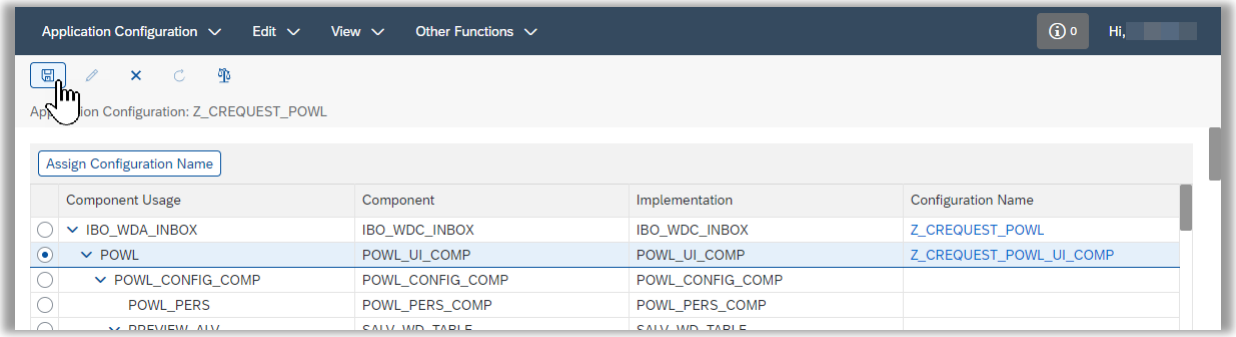
Make sure the *Component Usage* `IBO_WDA_INBOX` is selected. Press *Assign Configuration Name* assign the component configuration to `IBO_WDA_INBOX`.



Select *Component Usage* `POWL`. Press *Assign Configuration Name* assign the component configuration to `IBO_WDA_INBOX` which you create in 3.1.7 (in this example `Z_CREQUEST_POWL_UI_COMP`).



Save your configuration.



3.2 Inbox Customizing

Association of Workflow Task ID to POWL Type (`IBO_C_WF_TA_P_SC`)

Start transaction SM30. Enter table `IBO_C_WF_TA_P_SC` and press *Maintain*. Create entries listed in the following. This defines which workflow tasks should be visible in the POWL.

Table 1 Association of Workflow Task ID to POWL Type

Application	POWL Type ID	Workflow Task ID
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS00008267
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS53200002
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS54307924
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS54307925
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS54307931
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS54307937
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS60807954
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS60807991
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS60808004
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS60808005
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS60808007
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75707943
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75707963
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75707979
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75707980
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75707981
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717964
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717969
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717971
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717972
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717974
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717976
Z_CREQUEST_POWL	Z_CREQUEST_POWL_TYPE	TS75717977

Action of the Inbox (IBO_C_WF_ACC)

In transaction SM30 enter table IBO_C_WF_ACC and press *Maintain*. Create entries listed in the following. This defines which actions should be visible in the POWL.

Table 2 Action of the Inbox

APPL_ID	ACTION_ID	ACTION_TYPE	BUTTON_ID
Z_CREQUEST_POWL	USMDCOMPLSGLSTEP	FUNCTIONMODULEACTIONHANDLER	USMDCOMPLSGLSTEP
Z_CREQUEST_POWL	USMDCREQUESTPROCESS	OBJECTNAVIGATIONLAUNCHER	USMDCREQUESTPROCESS
Z_CREQUEST_POWL	USMDWFPROTOCOL2	OBJECTNAVIGATIONLAUNCHER	USMDWFPROTOCOL2

Tasks for Inbox Configuration (IBO_C_WF_TAC)

In transaction SM30 enter table IBO_C_WF_TAC and press *Maintain*. Create entries listed in the following. This defines the standard action that is to be executed on a task.

Table 3 Tasks for Inbox Configuration

Application	Workflow Task ID	Task Compl	Action Name
Z_CREQUEST_POWL	TS53200002		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307924		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307925		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307928		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307931		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307937		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60807954		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808004		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808005		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808007		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707943		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707979		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707980		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707981		USMDCREQUESTPROCESS

List of Action Allowed Per Task (`IBO_C_WF_TTAC`)

In transaction SM30 enter table `IBO_C_WF_TTAC` and press *Maintain*. Create entries listed in the following. This defines define which inbox actions are available for a task in the inbox.

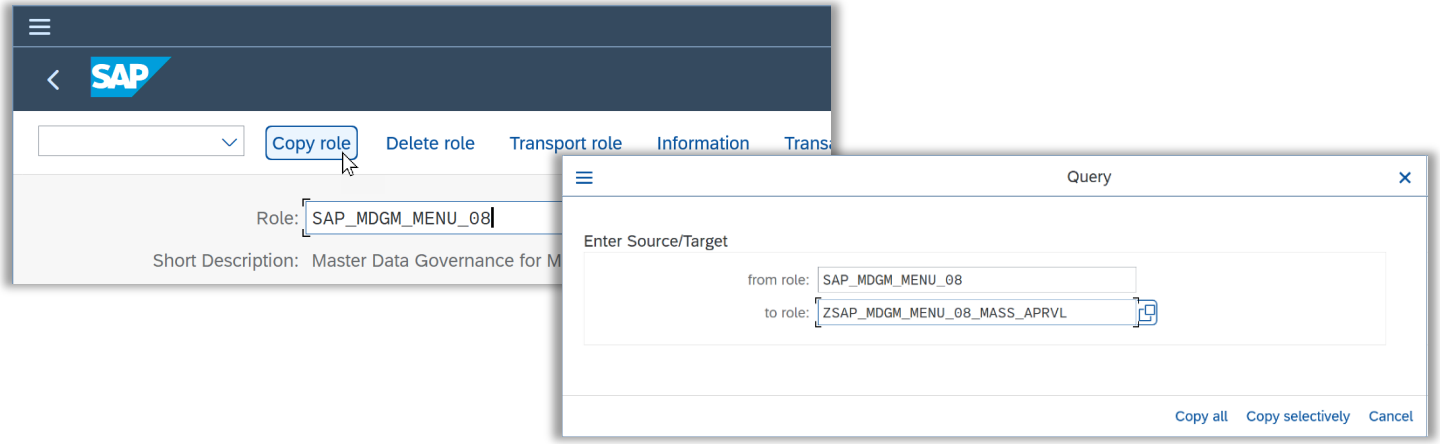
Table 4 Action Allowed Per Task

Application	Workflow Task ID	Task Compl	Action Name
Z_CREQUEST_POWL	TS53200002		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS53200002		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307924		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307924		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307924	D	USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307925		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307925		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307928		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307928		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307931		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307931		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS54307937		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS54307937		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS60807954		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60807954		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS60807991		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60807991		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS60808004		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808004		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS60808005		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808005		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS60808007		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS60808007		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS75707943		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707943		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS75707963		USMDCOMPLSGLSTEP
Z_CREQUEST_POWL	TS75707963		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707963		USMDWFPROTOCOL2

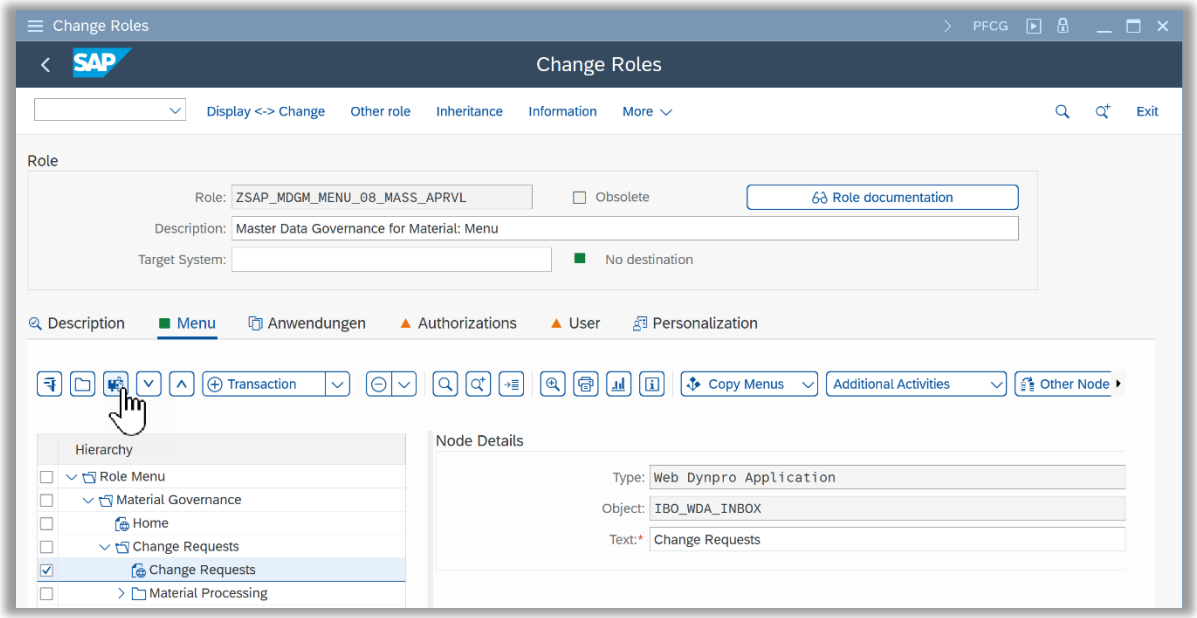
Z_CREQUEST_POWL	TS75707979		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707979		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS75707980		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707980		USMDWFPROTOCOL2
Z_CREQUEST_POWL	TS75707981		USMDCREQUESTPROCESS
Z_CREQUEST_POWL	TS75707981		USMDWFPROTOCOL2

3.3 Link the POWL to a PFCG role

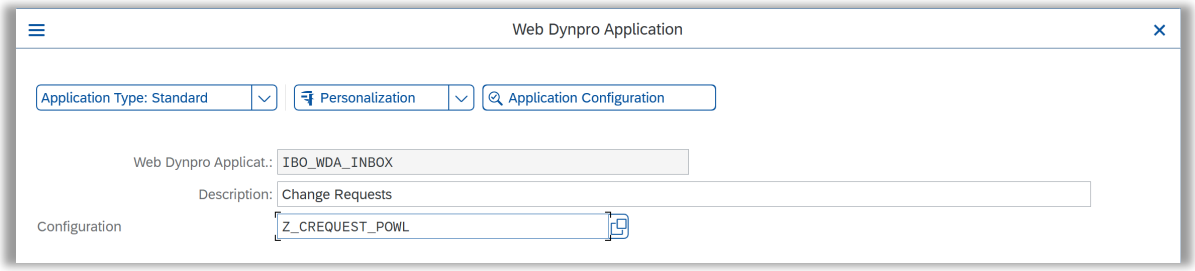
Start transaction `PFCG`. Copy or open an existing PFCG role that has a worklist attached, e.g. `SAP_MDGM_MENU_08` when working with material master data.



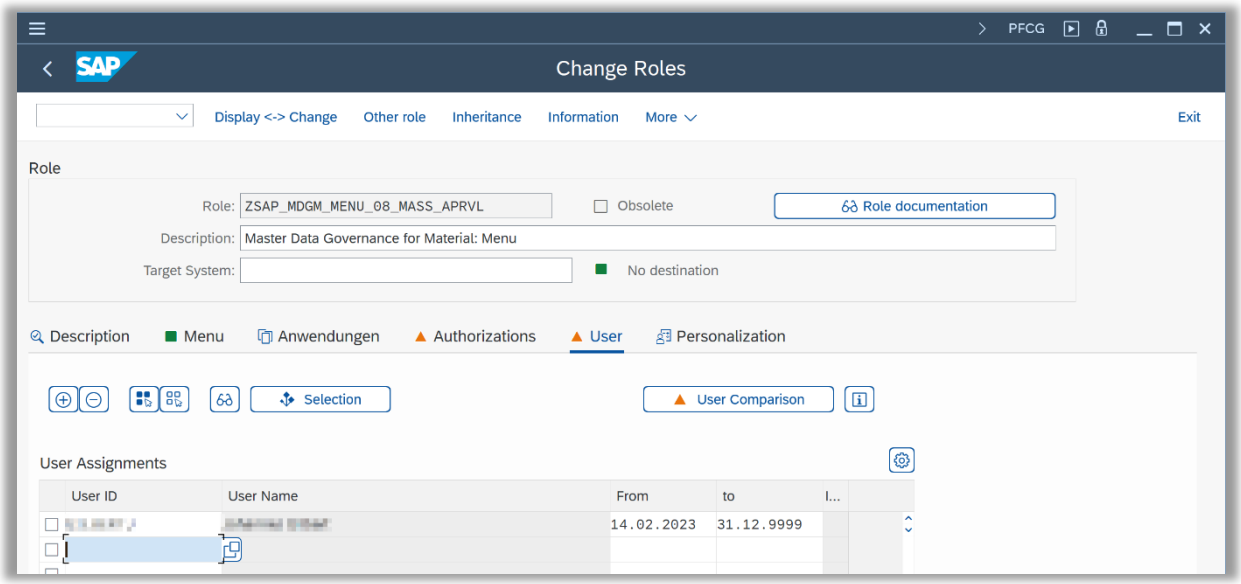
In the *Hierarchy* view navigate to *Role Menu* > *Material Governance* > *Change Requests*. Double-click the entry *Change Requests*. Press *Details*.



Open the details of this worklist and change the configuration to the newly created one (`Z_CREQUEST_POWL`). Alternatively, you can also create a new entry of “Web Dynpro Application” type.



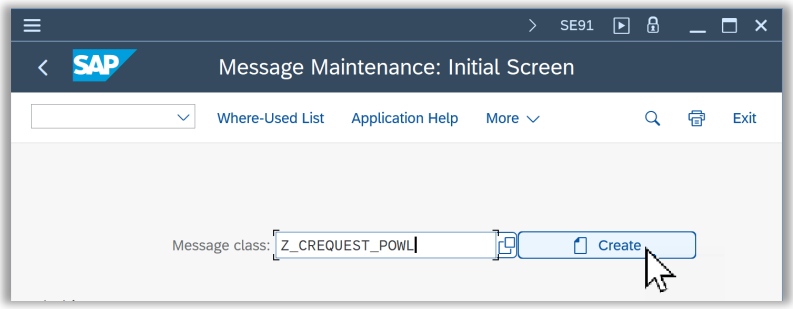
Attach the role to your user. Save the role.



Now you can already check that you can access the new POWL (which is not yet enhanced). It refreshes automatically to show all change requests for which you are authorized.

3.4 Message Class

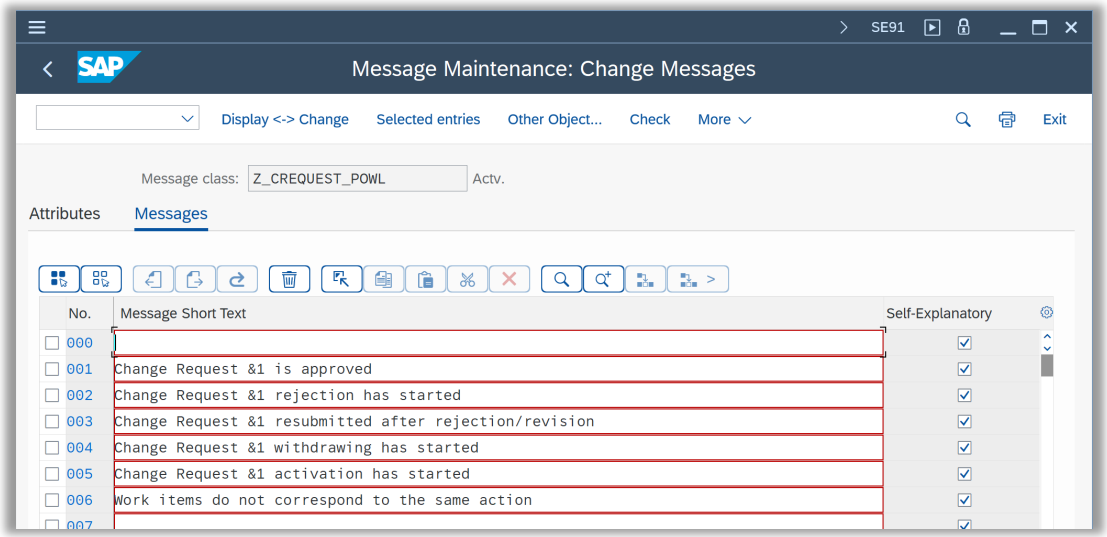
Start transaction SE91 to create a new message class. Enter `Z_CREQUEST_POWL` and press *Create*.



Enter the following messages (starting with number 00**1**):

Table 5 Messages of Message Class Z_CREQUEST_POWL

No.	Message Short Text
001	Change Request &1 is approved
002	Change Request &1 rejection has started
003	Change Request &1 resubmitted after rejection/revision
004	Change Request &1 withdrawing has started
005	Change Request &1 activation has started
006	Work items do not correspond to the same action



Save your changes.

3.5 Feeder Class Coding

Switch to source code-based editing. In the public section create a constructor (mind the importing parameter) and redefine method `if_powl_feeder~handle_action`. In the protected section redefine method `read_cust_actions`.

```
"! <p class="shorttext synchronized" lang="en">Feeder class for Mass Approval POWL</p>
CLASS zcl_crequest_mass_aprvl_powl DEFINITION
  PUBLIC
  INHERITING FROM cl_usmd_crequest_powl
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.
    CLASS-METHODS class_constructor.
    METHODS constructor
      IMPORTING !io_cust_helper TYPE REF TO object OPTIONAL .
```

```
    METHODS if_powl_feeder~handle_action
        REDEFINITION .

  PROTECTED SECTION.

    METHODS read_cust_actions
        REDEFINITION .

  PRIVATE SECTION.
    TYPES:
      "! <p class="shorttext synchronized" lang="en">Type for the no., model and work item ID of a change request</p>
      BEGIN OF ty_cr_model_wiid,
        crequest TYPE usmd_crequest,
        model    TYPE usmd_model,
        "! <p class="shorttext synchronized" lang="en">Work item ID</p>
        wi_id    TYPE sww_wiid,
      END OF ty_cr_model_wiid.
    "! <p class="shorttext synchronized" lang="en">Table of no., model and work item ID of a change requests</p>
    TYPES ty_cr_model_wiid_tab TYPE STANDARD TABLE OF ty_cr_model_wiid WITH DEFAULT KEY.
    "! <p class="shorttext synchronized" lang="en">Table type for Action Assignments to Workflow Step Types</p>
    TYPES ty_action_asg_to_step_type_tab TYPE STANDARD TABLE OF usmd2301 WITH DEFAULT KEY.
    "! <p class="shorttext synchronized" lang="en">Table type for Texts for Actions/p>
    TYPES ty_texts_of_action_tab TYPE STANDARD TABLE OF usmd220t WITH DEFAULT KEY.
    TYPES:
      "! <p class="shorttext synchronized" lang="en">Type CR step type, action, seq. no. and action texts</p>
      BEGIN OF ty_stype_action_texts,
        usmd_cr_stype    TYPE  usmd_crequest_as_type,
        usmd_cr_action   TYPE  usmd_crequest_action,
        usmd_sequence_nr TYPE usmd_sequence_nr,
        texts            TYPE ty_texts_of_action_tab,
      END OF ty_stype_action_texts.
    "! <p class="shorttext synchronized" lang="en">Table type for CR step type, action, seq. no. & action texts</p>
    TYPES ty_stype_action_texts_tab TYPE SORTED TABLE OF ty_stype_action_texts WITH UNIQUE KEY usmd_cr_stype usmd_cr_action.
    "! <p class="shorttext synchronized" lang="en">Table type for CR step type, action, seq. no. & action texts</p>
    TYPES ty_stype_action_texts_nsrt_tab TYPE STANDARD TABLE OF ty_stype_action_texts WITH EMPTY KEY.

    CONSTANTS:
      "! <p class="shorttext synchronized" lang="en">Relevant CR actions delivered by SAP</p>
      BEGIN OF cr_actions,
        approve  TYPE usmd_crequest_action VALUE '03',
        reject   TYPE usmd_crequest_action VALUE '04',
        resubmit TYPE usmd_crequest_action VALUE '07',
        withdraw TYPE usmd_crequest_action VALUE '08',
      END OF cr_actions.

    CONSTANTS:
      "! <p class="shorttext synchronized" lang="en">Message numbers of class Z_CREQUEST_POWL</p>
      BEGIN OF msg_numbers,
        "! <p class="shorttext synchronized" lang="en">Change Request &1 is approved</p>
        cr_is_approved               TYPE symsgno VALUE '001',
        "! <p class="shorttext synchronized" lang="en">Change Request &1 rejection has started</p>
        cr_rejection_started         TYPE symsgno VALUE '002',
        "! <p class="shorttext synchronized" lang="en">Change Request &1 resubmitted after rejection/revision</p>
        cr_resubmitd_after_rej_or_rev TYPE symsgno VALUE '003',
        "! <p class="shorttext synchronized" lang="en">Change Request &1 withdrawing has started</p>
        cr_withdraw_started          TYPE symsgno VALUE '004',
        "! <p class="shorttext synchronized" lang="en">Change Request &1 activation has started</p>
        cr_activation_started        TYPE symsgno VALUE '005',
      END OF msg_numbers.

    "! <p class="shorttext synchronized" lang="en">Determine the CR no., model and work item ID</p>
    "! Given the selection indexes the corresponding CR data is determined.
    "! @parameter c_selected | <p class="shorttext synchronized" lang="en">Selection indexes</p>
    "! @parameter c_result_tab | <p class="shorttext synchronized" lang="en">Generic table of type USMD_T_CREQUEST_POWL</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">CR no., model & work item ID corresponding to the selectio
n</p>
    METHODS get_data_of_sel_crequests
      IMPORTING c_selected    TYPE rstabixtab
                c_result_tab  TYPE INDEX TABLE
```

```
      RETURNING VALUE(result) TYPE ty_cr_model_wiid_tab.

    "! <p class="shorttext synchronized" lang="en">Control the available actions for the selected CR(s)</p>
    "!
    "! @parameter selected_crequests | <p class="shorttext synchronized" lang="en">Selected change requests</p>
    "! @parameter i_actionid | <p class="shorttext synchronized" lang="en">Current action</p>
    "! @parameter c_action_defs | <p class="shorttext synchronized" lang="en">Current action definitions</p>
    "! @parameter e_messages | <p class="shorttext synchronized" lang="en">Current messages</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Newly created action for the selected CR(s)</p>
    METHODS control_available_actions
      IMPORTING selected_crequests TYPE ty_cr_model_wiid_tab
                i_actionid         TYPE powl_actionid_ty
      CHANGING  c_action_defs      TYPE powl_actdescr_tty
                e_messages         TYPE powl_msg_tty
      RETURNING VALUE(result)      TYPE powl_actdescr_tty.

    "! <p class="shorttext synchronized" lang="en">Determine the available actions for the selected CR(s)</p>
    "!
    "! @parameter selected_crequests | <p class="shorttext synchronized" lang="en">Selected change requests</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Available actions</p>
    METHODS get_actions_for_creqs
      IMPORTING selected_crequests TYPE ty_cr_model_wiid_tab
      RETURNING VALUE(result)      TYPE ty_stype_action_texts_tab.

    "! <p class="shorttext synchronized" lang="en">Check the selected CRs share the same actions</p>
    "!
    "! @parameter selected_crequests | <p class="shorttext synchronized" lang="en">Selected change requests</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">'X' (true): sel. CRs share the same actions, else '' (fals
e)</p>
    METHODS check_sel_creq_have_same_actns
      IMPORTING selected_crequests TYPE ty_cr_model_wiid_tab
      RETURNING VALUE(result)      TYPE abap_bool.

    "! <p class="shorttext synchronized" lang="en">Create actions meta data</p>
    "! Given the CR actions (which are assigned to the selected CRs) the action meta
    "! data is created which is handed to the POWL (to render corresponding buttons).
    "! @parameter assigned_actions | <p class="shorttext synchronized" lang="en">Change request actions</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Action meta data</p>
    METHODS create_action_meta_data
      IMPORTING assigned_actions TYPE ty_stype_action_texts_tab
      RETURNING VALUE(result)    TYPE powl_actdescr_tty.

    "! <p class="shorttext synchronized" lang="en">Adjust the placement of actions</p>
    "! @parameter actions | <p class="shorttext synchronized" lang="en">Actions to adjust</p>
    "! @parameter shift_count | <p class="shorttext synchronized" lang="en">Integer to shift/ increase the placement </p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Adjusted actions</p>
    METHODS adjust_actions_placement
      IMPORTING actions       TYPE powl_actdescr_tty
                shift_count   TYPE i
      RETURNING VALUE(result) TYPE  powl_actdescr_tty.

    "! <p class="shorttext synchronized" lang="en">Check if the given action is a dynamically created one</p>
    "!
    "! @parameter actions | <p class="shorttext synchronized" lang="en">All actions' data</p>
    "! @parameter actionid | <p class="shorttext synchronized" lang="en">ID of the action to check</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">'X' (true) if it is a dynamically created action, else ''<
/p>
    METHODS is_newly_created_action
      IMPORTING actions       TYPE powl_actdescr_tty
                actionid      TYPE powl_actionid_ty
      RETURNING VALUE(result) TYPE abap_bool.

    "! <p class="shorttext synchronized" lang="en">Handle the given action (ID)</p>
    "! Perform the action related to the given action ID
    "! @parameter actionid | <p class="shorttext synchronized" lang="en">ID of the action to perform</p>
    "! @parameter selected_crequests | <p class="shorttext synchronized" lang="en">Selected change requests</p>
    "! @parameter messages | <p class="shorttext synchronized" lang="en">Current messages</p>
    "! @parameter do_refresh | <p class="shorttext synchronized" lang="en">Indicator to refresh the POWL</p>
    METHODS handle_action_internal
      IMPORTING actionid           TYPE powl_actionid_ty
```

```
                selected_crequests TYPE ty_cr_model_wiid_tab
      CHANGING
                messages           TYPE powl_msg_tty
                do_refresh         TYPE powl_xflag_ty.

    "! <p class="shorttext synchronized" lang="en">Convert messages from MDG structure to POWL structure</p>
    "!
    "! @parameter messages | <p class="shorttext synchronized" lang="en">Messages in MDG structure</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Messages in POWL structure</p>
    METHODS convert_messages
      IMPORTING messages      TYPE usmd_t_message
      RETURNING VALUE(result) TYPE powl_msg_tty.

    "! <p class="shorttext synchronized" lang="en">Create an info message for the given action and CR</p>
    "! The message informs which action has been triggered for the change request
    "! @parameter cr_action | <p class="shorttext synchronized" lang="en">Change request action</p>
    "! @parameter crequest | <p class="shorttext synchronized" lang="en">Change request</p>
    "! @parameter result | <p class="shorttext synchronized" lang="en">Info message</p>
    METHODS create_message_for_action
      IMPORTING cr_action     TYPE usmd_crequest_action
                crequest      TYPE usmd_crequest
      RETURNING VALUE(result) TYPE powl_msg_sty.

    "! <p class="shorttext synchronized" lang="en">Customizing USMD2301 Action Assignments to Workf. Step Types</p>
    CLASS-DATA actions_asgs_to_wf_step_types TYPE ty_action_asg_to_step_type_tab.
    "! <p class="shorttext synchronized" lang="en">Customizing USMD220T Texts for Actions (in user's language)</p>
    CLASS-DATA texts_of_actions TYPE ty_texts_of_action_tab.
    "! <p class="shorttext synchronized" lang="en">Range of all CR actions from USMD2301</p>
    CLASS-DATA action_asgs_to_wf_stepty_range TYPE RANGE OF usmd_crequest_action.
ENDCLASS.
```

The implementation part of the class has the following content:

```
CLASS ZCL_CREQUEST_MASS_APRVL_POWL IMPLEMENTATION.

  METHOD class_constructor.
    "Get all action assignments to workflow step types as well as the texts of all actions.
    SELECT * FROM usmd2301 INTO TABLE @actions_asgs_to_wf_step_types. "#EC CI_GENBUFF.
    SELECT * FROM usmd220t INTO TABLE @texts_of_actions WHERE langu = @sy-langu.

    "Build a range table containing all assigned actions.
    LOOP AT actions_asgs_to_wf_step_types ASSIGNING FIELD-SYMBOL(<action_asg_to_wf_step_type>).
      INSERT VALUE #( sign = 'I' option = 'EQ' low = <action_asg_to_wf_step_type>-
usmd_cr_action ) INTO TABLE action_asgs_to_wf_stepty_range.
    ENDLOOP.
    SORT action_asgs_to_wf_stepty_range.
    DELETE ADJACENT DUPLICATES FROM action_asgs_to_wf_stepty_range.
  ENDMETHOD.

  METHOD constructor.
    super->constructor( io_cust_helper ).
    initialize_feeder( ).
  ENDMETHOD.

  METHOD if_powl_feeder~handle_action.
    super->if_powl_feeder~handle_action(
      EXPORTING
        i_username                = sy-uname
        i_applid                  = i_applid
        i_type                    = i_type
        i_actionid                = i_actionid
        i_changed                 = i_changed
        i_action_index            = i_action_index
        i_action_conf             = i_action_conf
        i_langu                   = sy-langu
        i_additional_data         = i_additional_data
        i_visible_fields          = i_visible_fields
```

```
      IMPORTING
        e_portal_actions           = e_portal_actions
        e_messages                 = e_messages
        e_do_refresh               = e_do_refresh
        e_result_lines_changed     = e_result_lines_changed
        e_changes_processed        = e_changes_processed
        e_selected_changed         = e_selected_changed
        e_actions_changed          = e_actions_changed
      CHANGING
        c_selected                 = c_selected
        c_result_tab               = c_result_tab
        c_workflow_result_count    = c_workflow_result_count
        c_action_defs              = c_action_defs
        c_first_visible_row        = c_first_visible_row
        c_first_visible_scroll_col = c_first_visible_scroll_col ).

    "Disable the usual resubmit-button. There might be a button for action resubmit depending on the state
    "of the selected CRs.
    DELETE c_action_defs WHERE actionid = 'RESUBMIT'.

    "Determine the Create a list of the selected CRs with model
    DATA(selected_crequests) = get_data_of_sel_crequests( c_selected = c_selected c_result_tab = c_result_tab ).

    DATA(meta_data_of_new_actions) = control_available_actions( EXPORTING selected_crequests = selected_crequests
                                                                          i_actionid         = i_actionid
                                                                CHANGING  c_action_defs      = c_action_defs
                                                                          e_messages         = e_messages ).
    "In case no CRs are selected there is no need to handle to
    CHECK selected_crequests IS NOT INITIAL.

    "Check if the current action is one of the actions created by this implemented.
    "If so we want to handle it.
    CHECK is_newly_created_action( actions = meta_data_of_new_actions  actionid = i_actionid ) = abap_true.
    handle_action_internal(
      EXPORTING
        actionid           = i_actionid
        selected_crequests = selected_crequests
      CHANGING
        messages           = e_messages
        do_refresh         = e_do_refresh
    ).
  ENDMETHOD.

  METHOD get_data_of_sel_crequests.
    DATA(selected_crequests) = VALUE ty_cr_model_wiid_tab( ).
    LOOP AT c_selected ASSIGNING FIELD-SYMBOL(<selected_entry_index>).
      DATA(cr_no_work_item_id_and_model) = VALUE ty_cr_model_wiid( ).

      "The type of C_RESULT_TAB is USMD_T_CREQUEST_POWL. Get the work item ID of the selected entry.
      READ TABLE c_result_tab ASSIGNING FIELD-SYMBOL(<wa_result_tab>) INDEX <selected_entry_index>-tabix.
      ASSIGN COMPONENT 'WI_ID' OF STRUCTURE <wa_result_tab> TO FIELD-SYMBOL(<fs_wi_id>).
      cr_no_work_item_id_and_model-wi_id = <fs_wi_id>.

      "Get the CR for the work item ID
      cl_usmd_wf_service=>get_wi_crequest( EXPORTING id_wi_id = <fs_wi_id> IMPORTING ed_crequest = cr_no_work_item_id_and_model-
crequest ).
      CHECK cr_no_work_item_id_and_model-crequest IS NOT INITIAL.
      "Get the model of the CR.
      cl_usmd_crequest_util=>get_model_by_cr( EXPORTING i_crequest = cr_no_work_item_id_and_model-
crequest IMPORTING e_model = cr_no_work_item_id_and_model-model ).

      INSERT cr_no_work_item_id_and_model INTO TABLE selected_crequests.
    ENDLOOP.
    result = selected_crequests.
  ENDMETHOD.

  METHOD control_available_actions.
    "Delete the all actions which are in the customizing. The actions need special handling (done later).
    DELETE c_action_defs WHERE actionid IN action_asgs_to_wf_stepty_range.
```

```
    DATA(selected_creqs_have_same_actns) = check_sel_creq_have_same_actns( selected_crequests ).
    IF selected_creqs_have_same_actns = abap_false.
      MESSAGE w006(z_crequest_powl) INTO DATA(dummy) ##NEEDED.
      INSERT VALUE #( msgid = sy-msgid msgnumber = sy-msgno msgtype = sy-msgty ) INTO TABLE e_messages.
    ENDIF.

    DATA(assigned_actions) = get_actions_for_creqs( selected_crequests ).

    "Get rid of message: Work item &1 of task type "&2" does not support action &3
    "Example: Work item 000003425664 of task type "Dialog Processing" does not support action 04
    DELETE e_messages WHERE msgtype = 'W' AND msgid = 'IBO_INBOX_FEEDER' AND msgnumber EQ '013' AND message_v3 = i_actionid.

    "Only add additional action buttons in case there are no messages.
    IF e_messages IS INITIAL.
      DATA(meta_data_of_new_actions) = create_action_meta_data( assigned_actions ).
      c_action_defs = adjust_actions_placement( actions = c_action_defs shift_count = lines( meta_data_of_new_actions ) ).
      INSERT LINES OF meta_data_of_new_actions INTO TABLE c_action_defs.
    ENDIF.
    result = meta_data_of_new_actions.
  ENDMETHOD.

  METHOD get_actions_for_creqs.
    DATA(tmp) = VALUE ty_stype_action_texts_tab( ).
    DATA(processing_1st_sel_crequest) = abap_true.

    DATA(usmd_wf_service) = cl_usmd_wf_service=>get_instance( ).
    "Process each selected CR
    LOOP AT selected_crequests ASSIGNING FIELD-SYMBOL(<selected_crequest>).
      "Get the step and step type of this CR
      usmd_wf_service->get_crequest_step(
        EXPORTING
          id_crequest       = <selected_crequest>-crequest
          id_model          = <selected_crequest>-model
          id_wi_id          = <selected_crequest>-wi_id
          if_ignore_globals = abap_on
        IMPORTING
          ed_step           = DATA(step)
          ed_step_type      = DATA(step_type) ).

      "Process all customized assignments for the step type of the current CR
      LOOP AT actions_asgs_to_wf_step_types ASSIGNING FIELD-SYMBOL(<action_asg_to_wf_step_type>) WHERE usmd_cr_stype = step_type.
        IF processing_1st_sel_crequest = abap_false.
          "Check if the same actions are available for this CR and for all other selected CRs.
          READ TABLE tmp WITH TABLE KEY usmd_cr_stype = step_type usmd_cr_action = <action_asg_to_wf_step_type>-
usmd_cr_action TRANSPORTING NO FIELDS.
          IF sy-subrc <> 0.
            "This combination of step type and action is not available for the previously processed CRs.
            "The selected CRs do not share the same set of actions.
            EXIT.
          ENDIF.
        ENDIF.

        DATA(entry) = VALUE ty_stype_action_texts( usmd_cr_stype = step_type
                        usmd_cr_action = <action_asg_to_wf_step_type>-usmd_cr_action
                        usmd_sequence_nr = <action_asg_to_wf_step_type>-usmd_sequence_nr ).
        "Get the corresponding action texts (description, pushbutton text, quick info text).
        READ TABLE texts_of_actions ASSIGNING FIELD-SYMBOL(<action_texts>) WITH KEY langu = sy-
langu usmd_cr_action = <action_asg_to_wf_step_type>-usmd_cr_action.
        INSERT <action_texts> INTO TABLE entry-texts.
        INSERT entry INTO TABLE tmp.
      ENDLOOP.

      processing_1st_sel_crequest = abap_false.
    ENDLOOP.
    result = tmp.
  ENDMETHOD.

  METHOD check_sel_creq_have_same_actns.
    DATA(action_ags_to_wf_stptyps_tmp) = VALUE ty_action_asg_to_step_type_tab( ).
    DATA(usmd_wf_service) = cl_usmd_wf_service=>get_instance( ).
```

```
    "Process each selected CR
    LOOP AT selected_crequests ASSIGNING FIELD-SYMBOL(<selected_crequest>).
      DATA(step) = VALUE usmd_crequest_appstep( ).
      DATA(step_type) = VALUE usmd_crequest_as_type( ).

      "Get the step and step type of this CR
      usmd_wf_service->get_crequest_step(
        EXPORTING
          id_crequest       = <selected_crequest>-crequest
          id_model          = <selected_crequest>-model
          id_wi_id          = <selected_crequest>-wi_id
          if_ignore_globals = abap_on
        IMPORTING
          ed_step           = step
          ed_step_type      = step_type ).

      "Get all actions assigned to this step type and their texts.
      DATA(assigned_actions_tmp) = VALUE ty_action_asg_to_step_type_tab( ).
      "Process all customized assignments for the step type of the current CR
      LOOP AT actions_asgs_to_wf_step_types ASSIGNING FIELD-SYMBOL(<action_asg_to_wf_step_type>) WHERE usmd_cr_stype = step_type.
        INSERT <action_asg_to_wf_step_type> INTO TABLE assigned_actions_tmp.
      ENDLOOP.

      "Check that all selected work items share the same actions
      IF action_ags_to_wf_stptyps_tmp IS NOT INITIAL.
        LOOP AT assigned_actions_tmp ASSIGNING FIELD-SYMBOL(<assigned_action>).
          READ TABLE action_ags_to_wf_stptyps_tmp ASSIGNING FIELD-
SYMBOL(<2301_prev>) WITH KEY usmd_cr_action = <assigned_action>-usmd_cr_action.
          IF sy-subrc <> 0.
            result = abap_false.
            RETURN.
          ENDIF.
        ENDLOOP.
      ENDIF.
      action_ags_to_wf_stptyps_tmp[] = assigned_actions_tmp.
    ENDLOOP.
    result = abap_true.
  ENDMETHOD.

  METHOD create_action_meta_data.
    "Copy the given actions to a standard table to be able to sort it by sequence no.
    DATA(assigned_actions_tmp) = VALUE ty_stype_action_texts_nsrt_tab( ).
    assigned_actions_tmp = assigned_actions.

    "Process each action in the order of the sequence number.
    DATA(tmp) = VALUE powl_actdescr_tty( ).
    SORT assigned_actions_tmp BY usmd_sequence_nr.
    LOOP AT assigned_actions_tmp ASSIGNING FIELD-SYMBOL(<assigned_action>).
      "In case of the last action: Add a separator after this action
      IF sy-tabix = lines( assigned_actions_tmp ).
        DATA(add_separator) = abap_true.
      ENDIF.

      READ TABLE <assigned_action>-texts ASSIGNING FIELD-SYMBOL(<texts>) INDEX 1.

      DATA(powl_action_definition) = VALUE powl_actdescr_sty(
        add_separator = add_separator
        actionid = <assigned_action>-usmd_cr_action
        cardinality = 'S' "S = At least one object has to be selected
        enabled = abap_true
        text = <texts>-usmd_btn_txt
        placement = 'B' "B = Toolbar
        tooltip = <texts>-usmd_btn_tooltip
        placementindx = <assigned_action>-usmd_sequence_nr+2(1) ).

      INSERT powl_action_definition INTO TABLE tmp.
    ENDLOOP.
    result = tmp.
  ENDMETHOD.
```

```
  METHOD adjust_actions_placement.
    DATA(actions_tmp) = actions.
    "Increase the placement index by the given count.
    LOOP AT actions_tmp ASSIGNING FIELD-SYMBOL(<powl_action_definition>).
      IF <powl_action_definition>-placementindx < 9000.
        <powl_action_definition>-placementindx = <powl_action_definition>-placementindx + shift_count.
      ENDIF.
      MODIFY actions_tmp FROM <powl_action_definition> INDEX sy-tabix.
    ENDLOOP.
    result = actions_tmp.
  ENDMETHOD.

  METHOD is_newly_created_action.
    result = abap_false.
    READ TABLE actions WITH TABLE KEY actionid = actionid TRANSPORTING NO FIELDS.
    CHECK sy-subrc = 0.
    result = abap_true.
  ENDMETHOD.

  METHOD handle_action_internal.
    DATA(cr_action) = CONV usmd_crequest_action( actionid ). "POWL_LEAD_SEL
    DATA(usmd_wf_service) = cl_usmd_wf_service=>get_instance( ).

    "Process each selected CR.
    LOOP AT selected_crequests INTO DATA(cr_no_work_item_id_and_model).
      "Get a model instance.
      cl_usmd_model=>get_instance( EXPORTING i_usmd_model = cr_no_work_item_id_and_model-
model IMPORTING eo_instance = DATA(model) et_message = DATA(lt_message) ).
      "Get an instance of the governance API.
      DATA(lo_usmd_gov_api) = cl_usmd_gov_api=>get_instance( cr_no_work_item_id_and_model-model ).

      "Remove delete leading zeros at the CR number.
      SHIFT cr_no_work_item_id_and_model-crequest LEFT DELETING LEADING '0'.

      TRY.
          lo_usmd_gov_api->if_usmd_gov_api_process~check_complete_data( cr_no_work_item_id_and_model-crequest ).

          usmd_wf_service->complete_crequest_wfstep(
            EXPORTING
              id_crequest       = cr_no_work_item_id_and_model-crequest
              id_wi_id          = cr_no_work_item_id_and_model-wi_id
              id_action         = cr_action
              io_model          = model
              if_ignore_globals = abap_true
              if_commit_work    = abap_true
            IMPORTING
              et_message        = lt_message ).

          "Since at least one workflow step was completed a refresh is required.
          do_refresh = abap_true.

          "In case there are messages display them. Otherwise inform the user about the successful action.
          IF lt_message IS NOT INITIAL.
            INSERT LINES OF convert_messages( lt_message ) INTO TABLE messages.
          ELSE.
            INSERT create_message_for_action( cr_action = cr_action   crequest = cr_no_work_item_id_and_model-
crequest ) INTO TABLE messages.
          ENDIF.

        CATCH cx_usmd_gov_api_core_error cx_usmd_gov_api INTO DATA(gov_api_exception).
          "In case of exceptions leverage all exception messages to the UI.
          INSERT LINES OF convert_messages( gov_api_exception->mt_messages ) INTO TABLE messages.
      ENDTRY.
    ENDLOOP.
  ENDMETHOD.

  METHOD convert_messages.
    result = CORRESPONDING #( messages
                              MAPPING msgtype = msgty
                                      msgnumber = msgno
```

```
                                      message_v1 = msgv1
                                      message_v2 = msgv2
                                      message_v3 = msgv3
                                      message_v4 = msgv4 ).
  ENDMETHOD.

  METHOD create_message_for_action.
    DATA(powl_msg) = VALUE powl_msg_sty( msgtype = 'I' msgid = 'Z_CREQUEST_POWL' message_v1 = crequest ).
    IF cr_action = cr_actions-approve.
      powl_msg-msgnumber = msg_numbers-cr_is_approved.
    ELSEIF cr_action = cr_actions-reject.
      powl_msg-msgnumber = msg_numbers-cr_rejection_started.
    ELSEIF cr_action = cr_actions-resubmit.
      powl_msg-msgnumber = msg_numbers-cr_resubmitd_after_rej_or_rev.
    ELSEIF cr_action = cr_actions-withdraw.
      powl_msg-msgnumber = msg_numbers-cr_withdraw_started.
    ELSE.
      powl_msg-msgnumber = msg_numbers-cr_activation_started.
    ENDIF.
    result = powl_msg.
  ENDMETHOD.

  METHOD read_cust_actions.
    "Called when loading the POWL the first time.
    TRY.
        super->read_cust_actions(
          EXPORTING
            iv_type    = iv_type
            iv_langu   = iv_langu
          IMPORTING
            et_actions = et_actions ).
      CATCH cx_ibo_powl_no_action_setting ##NO_HANDLER.
    ENDTRY.

    "Delete the 'process change request' button because it will trigger a single object maintenance and cannot be used for multip
le CRs.
    CONSTANTS c_action_crequest_process TYPE powl_actionid_ty VALUE 'USMDCREQUESTPROCESS' ##NO_TEXT.
    READ TABLE et_actions ASSIGNING FIELD-SYMBOL(<action>) WITH KEY actionid = c_action_crequest_process.
    IF sy-subrc EQ 0.
      DELETE et_actions WHERE actionid = c_action_crequest_process.
    ENDIF.
  ENDMETHOD.
ENDCLASS.
```

4 Known differences between standard workflow inbox and this mass approval inbox

Table 6 Known differences between standard workflow inbox and this mass approval inbox

Topic	Standard Workflow Inbox	Mass Approval Inbox
Visibility of changes	Highlighted changes and Side Panel allow you to easily identify the delta.	Delta is not visible. The mass approval POWL could be enhanced further to list entity details. It is not part of this guide.
Notes	If notes are mandatory, users will be prompted in the change request to create a note.	Note settings are ignored while CR is processed in the background. Customers should define filters to not include such decision steps in the mass approval POWL.
Rejection Reason	If rejection reasons are mandatory, uses will be prompted in the change request to specify a rejection reason.	Rejection reason settings are ignored while CR is processed in the background. Customers should define filters to not include such decision steps in the mass approval POWL.
Selection Focus	The inbox refreshes automatically after processing a change request. The selected line will still be selected and still be the focus. This is usually the best selection if you process your work in sequence.	The inbox refreshes automatically. The selected lines will still be selected and still be the focus. In many cases this selection cannot be reused for the next step and a new selection is required.

5 Additional Information

5.1 Further Reading

5.1.1 Personal Object Worklist (POWL)

- POWL Component (help.ap.com)
- POWL (SAP Community Wiki)

5.1.2 Information on SAP MDG on SAP S/4HANA

- Exchange knowledge: SAP Community | Q&A | Blog
- Try SAP Master Data Governance on S/4HANA for free: Trial Version
- Learn more: Latest Release | Webinars | Help Portal | How-to Information | Key Presentations

5.1.3 SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

5.1.4 Related Information

- Learn more: Floorplan Manager for Web Dynpro ABAP | How to Adapt FPM | FPM Blog | How-to Information | Service Mapping Tool | SAP S/4HANA Cookbook CVI

5.2 SAP Notes

In addition to the detailed explanations written in this document, please see the following SAP Notes for further important information.

Note	Description
1619534	How to Create, Enhance and Adapt FPM Applications
1637249	MDG: Information for efficient message processing
2105467	MDG Performance
2561461	Scope of support for SAP Master Data Governance (MDG)