

Version: 1.0 (Draft)

Date: Personal object
work list, POWL

Doc. No.:

Language: Personal
object work list, POWL



POWL Cookbook

Copyright

Copyright © 2013 SAP AG. All rights reserved.

No part of this documentation may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG.

SAP reserves the right to change the information contained in this document without prior notice.

Author(s)

Miro Vins

Person responsible

Miro Vins

History

Version	Status	Date
1.0	Draft	June 3 rd , 2005

1 DOCUMENT ORGANIZATION.....	5
1.1 Authors.....	5
1.2 History.....	5
1.3 Intended Group of Readers.....	5
1.4 References.....	5
1.4.1 Related Documents.....	5
1.4.2 Documentation within Programming Environment.....	5
1.4.3 User documentation.....	5
1.5 Glossary.....	5
1.6 Motivation and semantic separation from Universal Worklist (UWL).....	6
2 HOW-TO.....	7
2.1 Context definition (Applid).....	7
2.2 Feeder(type) implementation.....	8
2.2.1 Create work list class with interface IF_POWL_FEEDER.....	8
2.2.1.1 Define list output structure.....	8

2.2.1.2 Define work list actions.....	8
2.2.1.3 Optional confirmations for defined actions.....	8
2.2.1.4 Define selection criteria.....	8
2.2.1.5 Define field catalogue.....	9
2.2.1.6 Implement action handler.....	10
2.2.1.7 Implement data retrieval.....	10
2.2.1.8 Optional WD ABAP detail component	10
2.2.1.9 Interface description.....	10
2.2.2 Create detail component (optional).....	15
2.3 Customizing	15
2.3.1 Feeder type repository & Context.....	15
2.3.2 Settings.....	15
2.3.3 Default Queries (Work lists).....	16
2.4 Authorities.....	16
2.4.1 Authority object CA_POWL.....	16
2.4.2 Application specific checks.....	17
2.5 Embedding.....	17
2.6 Portal.....	17
2.7 WD ABAP application.....	17
2.8 Query Scheduler.....	17
3 HOW WILL THE POWL TREAT MY FEEDER.....	18

1 Document Organization

1.1 Authors

Name	Group	Project Role(s) / Comments
Miro Vins		Project Lead

1.2 History

Date	Chapter	Name	Change/enhancement	Agreed with

1.3 Intended Group of Readers

- Application developer
- Development architects

1.4 References

1.4.1 Related Documents

1.4.2 Documentation within Programming Environment

1.4.3 User documentation

1.5 Glossary

Term	Definition
POWL	Personal Object Work List
Feeder	Application class describing the look alike and the behavior of the work list.
Query	Selection variant

1.6 Motivation and semantic separation from Universal Worklist (UWL)

Although there is already a generic worklist tool offered in Enterprise Portal, called the Universal Worklist (UWL), the special need for user-definable worklists is not satisfied. This is because the UWL has just an opposite paradigm: the user is shown those objects he is *obliged* to process or to notice, e.g. workflow items and alerts from an attached ERP system (as of today, the UWL does not officially support other object types anyway). The creation of a custom query that delivers a list of objects (of an arbitrary type) the respective user is merely *interested* in or he has to select *on its own* for his daily work is therefore not in scope of the UWL.

This is where the Personal Object Work List (POWL) comes in: It is aimed at providing the user with a portal-based interface that he can use to define (and optionally store) queries in a similar way as he is used to from R/3 selection screen variants. Moreover, it is possible to flag such queries as “activated” so their results are immediately visible when navigating to a portal page/workset containing the respective POWL iView. This way, the user has its custom “workload” right at hand (as it is the case in UWL with the user’s mandatory, externally “pushed” workload).

2 How-To

Main task = create class implementing our interface 'IF_POWL_FEEDER' . This class will tell us how your work list will look like . Actions, Selection criteria , Output structure , Detail etc.

Your class is the App ! We build the UI (except the detail comp. ☺).

2.1 Context definition (Applid)

In order to identify your POWL and the content (feeder types) in the portal role you need to create a context ID (Application ID = APPLID) . This Application ID is necessary for the IView

creation , for assigning your feeder types to the POWL (visibility of work lists types) , assigning default queries (work lists) .

Application ID:Feeder type = n:m

Application ID:Feeder query = n:m (Prerequisite : APPLID – Feeder type assignment)

Without Application ID -> no POWL content !

2.2 Feeder(type) implementation

2.2.1 Create work list class with interface IF_POWL_FEEDER

2.2.1.1 Define list output structure

2.2.1.2 Define work list actions

2.2.1.3 Optional confirmations for defined actions

2.2.1.4 Define selection criteria

get_sel_criteria: this method has to provide the definitions of all selection criteria available for query creation for the combination of user ID and POWL iView (reflected by the importing parameters *bname* and *applid*).

The POWL runtime expects the selection criteria definitions to be returned as internal table with line type **POWL_SELCRIT_STY**:

Component	Type	Description
FIELDNAME	CHAR(30)	Internal Name of the selection criteria
CRITTYPE	POWL_CRITTYPE_TY	Display type of the selection criteria
MANDATORY	POWL_XFLAG_TY	Criteria is mandatory
DOMNAME	DOMNAME	Name of a DDIC domain
DATATYPE	DATATYPE	Name of a DDIC datatype
REF_TABLE	CHAR(30)	Name of a DDIC reference table
REF_FIELD	CHAR(30)	Name of a reference field in the reference table
INTTYPE	CHAR(1)	ABAP built-in type (C,D,N...)
OUTPUTLEN	NUMC(6)	Output length
CRITTEXT	CHAR(40)	Output text for the criteria
TOOLTIP	CHAR(40)	Tooltip for the criteria
HEADER	CHAR(40)	Headline for the criteria
VALUESET	POWL_NAMEVALUE_STY	set of valid values with corresp. descriptions

CRITTYPE defines the display style of the criteria. The following styles are supported:

- select option field with interval and multi-selection

- select option field with interval, but without multi selection
- select option field with multi selection, but without interval
- parameter field (i.e. simple input field)
- dropdown list
- checkbox

By means of HEADER, the individual criteria can be separated by intermediate headlines. This way, semantic coherent groups of selection criteria can be shown to the user.

The VALUESET field can be used to define an F4 help which is not dictionary-based. Each entry in the value set table consists of an internal name (which is interpreted as the actual value) and a descriptive text which is displayed to the user.

2.2.1.5 Define field catalogue

If you do not specify the field catalogue (look a like) of your output table we will create a default
-> simple text view , every field in output structure = column with text view in ALV = visible

You don't want this -> tell us how it should look like :

Hint: when you specify it on your own, tell us as much as possible !

Component	Type	Description
COLID	CHAR(30)	ID of the column (also key of the fieldcat table type)
COLPOS	INT4	defines order for positioning of the columns
WIDTH	NUMC(6)	Output length of the field
HEADER	CHAR(30)	Column header text
HEADER_DDIC	DATATYPE	DDIC Element supplying the header text
DISPLAY_TYPE	POWL_DISPLAY_TY	Display type: text, icon, checkbox, link (for action binding)
H_ALIGN	POWL_ALIGN_TY	Horizontal output alignment of the column (left, center, right)
FIXED	POWL_XFLAG_TY	Fixed column
TEXT	STRING	Text to be displayed (max length depending on display type)
EDITABLE	POWL_XFLAG_TY	Set Column editability
EDITABLE_REF	CHAR(30)	Output table reference field determining editability status
WRAPPING	POWL_XFLAG_TY	Enable text wrapping
WRAPPING_REF	CHAR(30)	Output table reference field for text wrapping enabling
ICON_SRC	STRING	Source for icon display
ICON_SRC_REF	CHAR(30)	Output table reference field for icon display source
ICON_ALT	POWL_XFLAG_TY	Icon display alternative text
ICON_ALT_REF	CHAR(30)	Output table reference field for icon display alternative text
ICON_FIRST	POWL_XFLAG_TY	Output Icon before text
ICON_FIRST_REF	CHAR(30)	Output table reference field for icon positioning

		before text
CBOX_CHECK	POWL_XFLAG_TY	Checkbox checked status
CBOX_CHECK_REF	CHAR(30)	Output table reference field for checkbox checked status
COL_VISIBLE	POWL_XFLAG_TY	Set visibility of entire column
ENABLED	POWL_XFLAG_TY	Enable or disable entire column
ENABLE_REF	CHAR(30)	Output table reference field for enabling or disabling
TOOLTIP	CHAR(100)	Tooltip for the entire column
TOOLTIP_REF	CHAR(30)	Output table reference field for tooltip
ALLOW_FILTER	POWL_XFLAG_TY	Allow filtering of column
ALLOW_SORT	POWL_XFLAG_TY	Allow sorting of column
CQ_REF	CHAR(30)	Output table reference field for either currency or quantity
SORT_REF	CHAR(30)	Output table reference field containing sort index
FILTER_REF	CHAR(30)	Output table reference field containing filter value

2.2.1.6 Implement action handler

You have define Action ID's in chapter 2.2.1.2 . Now you have to handle them . When ever the user triggers one of your actions , we will dispatch it to your action handler.

Then you can decide whether :

- Navigate to another portal target
- Fire portal event
- Do something in backend
- Fire WD ABAP event (only necessary in case of direct embedding)

2.2.1.7 Implement data retrieval

Select your data here . We will pass the users selection criteria to this method .

2.2.1.8 Optional WD ABAP detail component

2.2.1.9 Interface description

- GET_ACTIONS: defines all explicit actions. An action is explicit in POWL if it is not defined by a field catalog cell renderer (c.f. GET_FIELD_CATALOG).
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - importing I_SELCRIT_PARA: the current selection criteria values of the query adhering to the definitions delivered by GET_SEL_CRITERIA
 - changing C_ACTION_DEFS: supplies the current explicit action definitions for the query to the Feeder. The line type of this table is POWL_ACTDESCR_STY.

- exporting E_ACTIONS_CHANGED: if none of the action definitions supplied to the Feeder via C_ACTION_DEFS had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.
- GET_ACTION_CONF: defines an action confirmation message for an (explicit or implicit) action.
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - importing I_ACTIONID: the ID of the respective action
 - importing I_RESULT_TAB: the current query results adhering to the type definition supplied by GET_OBJECT_DEFINITION
 - importing I_SELECTED: the table line indices of the query results currently selected (i.e. marked) by the user. The first line entry corresponds to the lead selection of the query results table.
 - importing I_CHANGED: --only relevant for editable query results table-- change information on those query results changed by the user since most recent enabling of POWL "dirty" state (c.f. HANDLE_ACTION for details)
 - exporting E_CONF_MESSAGE: the confirmation message to be displayed before actual execution of the action identified by I_ACTIONID
- GET_SEL_CRITERIA: defines selection criteria that can be used to define new queries against this Feeder. Moreover, default values for the selection criteria can be supplied by this method
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - changing C_SELCRIT_DEFS: supplies the current selection criteria definitions to the Feeder. Note that selection criteria definitions are cached on database upon query refresh. The line type of this table is POWL_SELCRIT_STY.
 - changing C_DEFAULT_VALUES: supplies the current selection criteria default values to the Feeder. Note that selection criteria default values are cached on database upon each query refresh.
 - exporting E_SELCRIT_DEFS_CHANGED: if none of the selection criteria definitions supplied to the Feeder via C_SELCRIT_DEFS had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.
 - exporting E_DEFAULT_VAL_CHANGED: if none of the selection criteria default values supplied to the Feeder via C_DEFAULT_VALUES had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.

- GET_FIELD_CATALOG: defines the field catalog to be used for query results table display.
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - importing I_SELCRIT_VALUES: the selection criteria values of the current query
 - changing C_FIELDCAT: supplies the current field catalog to the Feeder.

Each entry in the field catalog is a reference to one column of the actual (internal) query results table as delivered by GET_OBJECTS and defines the rendering of this column. The set of available columns is defined by GET_OBJECT_DEFINITION. Note that each internal results column not referenced by a field catalog entry will be rendered as text view per default.

There are passive cell renderers (as text views, input fields or checkboxes) and active ones (button and link to action). If the user clicks on a table cell with active cell renderer, this will trigger a call of HANDLE_ACTION, supplying the column ID as action ID. Such actions are defined as "implicit actions". (Therefore, the column IDs in the field catalog and the action IDs receivable by HANDLE_ACTION have the same domain).

The line type of this table is POWL_FIELDCAT_STY.

- exporting E_FIELDCAT_CHANGED: if none of the field catalog entries supplied to the Feeder via C_FIELDCAT had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.
- GET_OBJECT_DEFINITION: defines the datatype of the actual (internal) query results table as to be delivered by GET_OBJECTS.
 - importing I_TYPE: the POWL type ID as registered
 - returning E_OBJECT_DEF: the table type definition of the actual (internal) query results table as instance of CL_ABAP_TABLEDESCR. Note that the creation of the CL_ABAP_TABLEDESCR instance is quite easy in most cases by using one of the static methods DESCRIBE_BY_DATA or DESCRIBE_BY_NAME of this class respectively
- GET_OBJECTS: responsible for the actual query execution and delivery of the query results table which must adhere to the type definition given by GET_OBJECT_DEFINITION.
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - importing I_SELCRIT_VALUES: the selection criteria values of the current query

- exporting E_RESULTS: the actual query results for the criteria values supplied to the Feeder via I_SELCRIT_VALUES
 - exporting E_MESSAGES: messages that occurred during query execution/results determination and are to be displayed to the user. The line type of this table is POWL_MSG_STY.
- GET_DETAIL_COMP: optionally, custom details can be displayed for one query result at a time. If this is desired, this method has to deliver the necessary meta-information to the POWL.
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - exporting E_DETAIL_COMP: the name of a Web Dynpro Component which implements the component interface POWL_DETAIL_COMP_IF. This component will be instantiated at runtime and has to manage the actual details display; the respective "master result" data (adhering to GET_OBJECT_DEFINITION) will be supplied by the POWL to the detail component via method UPDATE_DETAIL_DATA as defined in POWL_DETAIL_COMP_IF.

If you don't want to display result details, just leave this parameter blank.

- exporting E_DETAIL_TYPE: supply a type ID defined in transaction POWL_TYPE in order to render a default detail view filled with data from the respective Feeder (i.e. the GET_OBJECTS method will be called with the data of the current query result as selection criteria values).
- !!! NOTE:** As of today, the availability of this feature in the first delivered POWL version is not committed !!!
- HANDLE_ACTION: has to handle all implicit and explicit actions defined by this Feeder instance.
 - importing I_APPLID: the application ID identifying the current POWL iView
 - importing I_USERNAME: the user ID the current portal user is mapped to
 - importing I_TYPE: the POWL type ID as registered
 - importing I_ACTIONID: the ID of the action triggered by the user (i.e. either the ID of one of the explicit actions as defined by GET_ACTIONS or the results table column ID of an implicit action as defined by the field catalog by an active cell renderer).
 - importing I_CHANGED: --only relevant for editable query results table-- change information on those query results changed by the user since most recent enabling of POWL "dirty" state.
 - importing I_ACTION_INDEX: in case an implicit action, this parameter will supply the results table line index where the action was triggered to the Feeder.

- importing I_ACTION_CONF: in case there was an action confirmation message defined by GET_ACTION_CONF, this parameter supplies the confirmation result (i.e. Y for Yes or N for No) to the Feeder. The default value (which is also supplied in case there was no confirmation message) is Yes.
- exporting E_PORTAL_ACTIONS: supplies a portal action to be performed upon the HANDLE_ACTION call. Currently, the following portal actions are supported:

Absolute navigation: just set the component PORTAL_PATH to the respective navigation target. Note that all parameters supplied in component PARAMETERS will be transported as URL-encoded value string of URL parameter 'DynamicParameter'.

Object based navigation: all components with prefix BO_ are to be filled according to the business object operation to be triggered. If you want to fire the default operation of the resp. business object, you may omit the component BO_OP_NAME.

Portal client-side event: all components with prefix CS_ have to be filled accordingly.

The structure type of this parameter is POWL_FOLLOW_UP_STY.

- exporting E_MESSAGES: messages to be displayed to the user. The line type of this table is POWL_MSG_STY.
- exporting E_DO_REFRESH: if this flag is set, a refresh of the query will be enforced upon the HANDLE_ACTION call.
- changing C_RESULT_TAB: supplies the current query results table (including all changes done by the user if the results table is editable) to the Feeder.
- exporting E_RESULT_LINES_CHANGED: if none of the results supplied to the Feeder via C_RESULTS had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.
- exporting E_CHANGES_PROCESSED: --only relevant for editable query results table-- if all results changes done by the user have been processed by the Feeder, you can set this flag to 'X' in order to tell the POWL to suppress data loss confirmation messages (like "Unsaved data will be lost. Continue?"), i.e. to reset the POWL "dirty state". A typical action which would set this flag is something like "Save data".
- changing C_SELECTED: supplies the indices of all results table lines selected/marked by the user to the Feeder. The first entry of this table corresponds to the lead selection index.
- exporting E_SELECTED_CHANGED: if none of the selected result line indices supplied to the Feeder via C_SELECTED had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.
- changing C_ACTION_DEFS: supplies the current explicit action definitions for the query to the Feeder. The line type of this table is POWL_ACTDESCR_STY.
- exporting E_ACTIONS_CHANGED: if none of the action definitions supplied to the Feeder via C_ACTION_DEFS had to be changed, you can leave this flag unset. Otherwise, you have to set it to 'X'.

- exporting C_FIRST_VISIBLE_ROW: supplies the index of the first visible results table row to the Feeder (i.e. the current scroll position of the results table). Normally, you will only have to adapt this index if result table lines were added and/or deleted.

2.2.2 Create detail component (optional)

2.3 Customizing

2.3.1 Feeder type repository & Context

Transaction **POWL_TYPE** Maintain POWL Type definition

Use this transaction to register your feeder class . Create a new type , assign your class (implementing interface IF_POWL_FEEDER) and add a description . The description will be used within the POWL dialog for the user specific query definition.

In addition you can set the attribute 'Sync' -> this will force a synchronous refreshing of this type. By default the POWL refreshes asynchronous.

Transaction **FPB_MAINTAIN_HIER** Personalization: Hierarchy Maint.

Use this transaction to create your context ID (Applid). You can setup a hierarchy of application ID's which can represent for e.g. you role or a set of roles.

e.g.

Applid ID	Description	Parent
OPS	Operations	
OPS-SALESREP	Sales Representative	OPS
OPS-SALESREP-W1	Worklist 1	OPS-SALESREP
OPS-SALESREP-W2	Worklist 2	OPS-SALESREP
OPS-SALESREP-W3	Worklist 3	OPS-SALESREP
OPS-.....

2.3.2 Settings

Transaction **POWL_CAT** Maintain POWL categories

Use this transaction to create or deliver admin categories. Categories are used in the POWL dialog to organize/classify work lists.

Transaction **POWL_TYPER** Maintain POWL Type role assignment

Use this transaction to tell the POWL which feeder types are available for Context(APPLID) , Role or both. If there are no entries for your type in this table or the following transaction POWL_TYPEU , your type will not be available within the POWL + all queries based on this type won't be available too .

Transaction POWL_TYPEU Maintain POWL Type user assignment

See transaction POWL_TYPER . But this time at user level.

2.3.3 Default Queries (Work lists)

Transaction POWL_QUERY Maintain POWL Query definition

Use this transaction to create or change 'admin' work lists. This queries will be , when assigned to a user or a context , used for derivation . This is a simple way for the administrator to define 'template' work lists for n users. These templates queries can contain predefined selection values , special settings like read-only, quick search .. etc . plus you can create admin layout variants (now called views) for embedded WD ALV.

Transaction POWL_QUERYR Maintain POWL Query role assignment

Similar to POWL_TYPER , but this time for queries(work lists) . Create or change an entry for a admin query and assign it to a application context. These queries will be automatically derivated when a user starts his POWL the first time . You can change the sequence of queries , assign categories , activate/deactivate you assignments.

Transaction POWL_QUERYU Maintain POWL Query user assignment

See POWL_QUERYR , this time at user level.

2.4 Authorities

2.4.1 Authority object CA_POWL

Use this to restrict the POWL functionality:

- POWL_APPID: application ID of POWL iView (as specified in Application Parameters in the iView properties)
- POWL_QUERY: determines the authorities of a user with respect to query creation.
 - 01: the user is allowed to create/change/delete own queries for all POWL object types assigned to him (c.f. customizing tables POWL_TYPE_USR and POWL_TYPE_ROL).
 - 02: the user is only allowed to create own queries on the basis of admin queries assigned to him via customizing tables POWL_QUERY_USR and POWL_QUERY_ROL respectively. (Note: this is also subjected to the user - POWL object type assignments)
 - 03 (and other values): the user is only allowed to change admin queries assigned to him with respect to the select options restrictions of those admin queries (thus creating one own "derivation" per admin query transparently)
- POWL_CAT: determines the authorities of a user with respect to query categories
 - 01: the user is allowed to create/change/delete own categories and assign queries to them

- 02: the user is only allowed to assign queries to the existing categories and change the order of queries
- 03 (and other values): the user is not allowed to re-assign queries or change the query order.
Note: if field POWL_QUERY is set to 01 or 03, setting POWL_CAT to 03 is not sensible. Therefore, the value will be set to '02' implicitly in that case.
- POWL_LSEL: determines if the user is allowed to select the layout style (either one entry in a hyper-link matrix or one tabstrip per query) for the POWL iView
- POWL_TABLE: determines if the user is allowed to personalize the query result table settings (define column order, hide columns, etc).
- POWL_RA_AL: determines if the user gains access to a "Refresh all" button, which triggers a parallelized refresh for all queries which are active on the POWL iView identified by POWL_APPLID. Note this may cause high system load on the application server group used for refreshes on this POWL iView.

2.4.2 Application specific checks

Application / work list specific authority checks must be implemented in the respective feeder class. E.g.: Actions, field catalogue, action handling and object selection. Use the corresponding interface method for implementation.

2.5 Embedding

2.6 Portal

Create an portal iView for Web Dynpro ABAP foundation with the following parameters:

- Portal system alias (select the system where you have registered your feeder class)
- Application name = POWL
- Namespace = SAP
- Application parameters = 'APPLID={your context ID}'

2.7 WD ABAP application

2.8 Query Scheduler

All user queries(work lists) results within a POWL context are cached into an internal cluster table. So every time the user hit the refresh link below his work list the result of the feeder class method 'GET_OBJECTS' are stored into this cache . We (the POWL) always read the cache ! Independent from the 'Sync' setting in the type repository or query definition.

This cache enables us to provide the possibility of a scheduled work list creation by the admin. E.g. nightly work list build

User the report **POWL_WLOAD** to schedule the work list creation.

3 How will the POWL treat my feeder

[Questions, comments, missing information etc.]

This chapter exists during work-in-progress status only and must be deleted once the document is released!]