BDEx Developers Cookbook

Manual

Release 4 — Last update: 2015/08/27

Basis Technologies

Table of Contents

Introduction

BDEx provides a number of areas where functionality can be enhanced and customized beyond the options provided in the configuration settings.

This includes a number of BAdIs provided to include custom mappings between work request and customer data, improve performance by using custom indexing and the method by which custom actions can be added to work requests and master data.

Adding Custom Actions

The list of actions available when the user right-clicks master data or a work request can be customized to provide additional or alternative actions. To enable a custom action, the following steps must be followed.

Action Header

The table /BTI/MDE_C_ACT contains the master information for all actions.

All custom actions must be included in this table.

Field name	Description	Notes
MANDT	Client	Client
ACTIONID	MDE:Action identifier	Primary key for all actions. Generally defined to be identifiable to the primary master data parameter (e.g. BP0001 for the first action for business partner). Use an Action ID in the customer namespace to ensure compatibility with future BDEx versions (e.g. ZBP001)
FCODE	Function code	Function code (type UCOMM) that is passed to the relevant BAdI
CLASSNAME	Reference type	Method class
METHODNAME	Full Component Name	Method
ICONNAME	Name of an Icon	Method icon
REFRESH	Refresh actions	Whether or not BDEx should refresh the master data and work requests after this action is executed.
INACTIVE	Active / Inactive	Shows/hides the action in the right-click context menu and work request detail view.

Table 1 - /BTI/MDE_C_ACT

Work request/master data linkage

There are two different tables for linking the action header to the relevant context menu, /BTI/MDE_C_EXACT for work request actions, and /BTI/MDE_C_MDACT for master data actions.

One of these tables will require an entry:

Field name	Description	Notes
MANDT	Client	Client.
MDOBJECTID	MDE:Master data object identifier	Primary key for master data actions. Corresponds to an ACTIONID in /BTI/MDE_C_ACT.
ACTIONID	MDE:Action identifier	Function code (type UCOMM) that is passed to the relevant BAdI.
SEQUENCE	MDE:Action sequence	The action's sequence number within the relevant menu.

Table 2 - /BTI/MDE_C_MDACT

Field name	Description	Notes
MANDT	Client	Client.
CLASS	/BTI/MDE_DE_WR_CLASS	Action class.
SUBCLASS	/BTI/MDE_DE_WR_SUBCLASS	Action method.
STATUS	/BTI/MDE_DE_WR_STATUS_ID	Allows the action to be limited to certain work request statuses (e.g. BPEMs in status "new").
ACTIONID	/BTI/MDE_DE_GUI_ACTION_ID	Function code (type UCOMM) that is passed to the relevant BAdI.
SEQUENCE	/BTI/MDE_DE_GUI_ACTION_SEQ	The action's sequence number within the relevant menu.

Table 3 - /BTI/MDE_C_EXACT

Function code handling

Once the table entries have been added, the function code resulting from the action being selected from the menu must be handled.

There are two separate BAdIs that handle this, /BTI/MDE_BADI_ACTION for work request actions, and /BTI/MDE_BADI_MD_ACTION for master data actions, calling a PROCESS_ACTIONS method.

PROCESS ACTIONS takes the action class and the work request (or master data) as input parameters, and should be used to call custom code depending on the parameters passed in.

Example implementations are provided in each BAdI if guidance is needed.

Example

A custom action, “Correct Plausible Meter Reading Results” is required. This corresponds to transaction EL29, and for simplicity’s sake this example will use the function module ISU_S_METERREAD_CHANGE.

1. **Add the new action to the action header table**

Field	Value	Rationale
ACTIONID	ZDE0001	First available action ID in the customer name space corresponding to device
FCODE	CORRPLAUSREAD	Meaningful function code to aid with debugging
CLASSNAME	ZCL_EXAMPLE	Class name, although this field is not used for BAdI methods.
METHODNAME	CORRECT_PLAUS_READ	Method name, although this field is not used for BAdI methods.
ICONNAME	ICON_EXECUTE_OBJECT	Icon name.
REFRESH	X	EL29 makes changes to the customer’s account, therefore the master data should be refreshed when returning to BDEx.
INACTIVE		Action is active.

Table 4 - /BTI/MDE_C_ACT example

2. **Add the new action to the relevant action table**

   As a plausible meter read will not correspond to an outstanding work request, this action will be attached to the device master data object. Therefore in this example the /BTI/MDE_C_MDACT table will be used.

Field	Value	Rationale
MDOBJECTID	ISU0009	Corresponds to Device
ACTIONID	ZDE0001	Same as ACTIONID in previous step
SEQUENCE	22	22nd item in list for Device assuming no other items have been disabled

Table 5 - /BTI/MDE_C_MDACT example

3. If necessary, implement the relevant BAdI class
Create the enhancement implementation for BAdI /BTI/MDE_BADI_MD_ACTION. The example implementation /BTI/MDE_ACTION_EXAMPLE is provided to assist with this.

4. Add the logic to handle the user command
Use the logic in /BTI/MDE_ACTION_EXAMPLE as a template to implement the custom action.

Here is an example of what the final code in ZCL_EXAMPLE may look like:

```
METHOD /bti/mde_if_md_action~process_actions.
  DATA
    lv_fcode TYPE sy-ucomm,
    lv_equnr TYPE equnr.

  CASE x_action->get_fcode( ).
    WHEN 'CORRPLAUSREAD'.
      lv_equnr = x_md_class->get_key( ).
      CALL METHOD zcl_example->corr_plaus_read
        EXPORTING
          X_EQUNR        = lv_equnr

      CALL METHOD x_action->set_action_handled.
    WHEN OTHERS.
  ENDCASE.
ENDMETHOD.
```

```
METHOD corr_plaus_read.
  CALL FUNCTION 'ISU_S_METERREAD_CHANGE'
    EXPORTING
      X_UPD_ONLINE              = 'X'
      X_EQUNR                   = pv_equnr
ENDMETHOD.
```

Note that the line of code: “x_action->set_action_handled” is used to ensure that the action is considered “finished” so that the action can be logged, and the customer account refreshed as per the configuration.

Calling BDEx from an unsupported BOR object

Many BOR objects are already linked to master data and work request classes via the /BTI/MDE_C_MDBOR and /BTI/MDE_C_WRBOR tables.

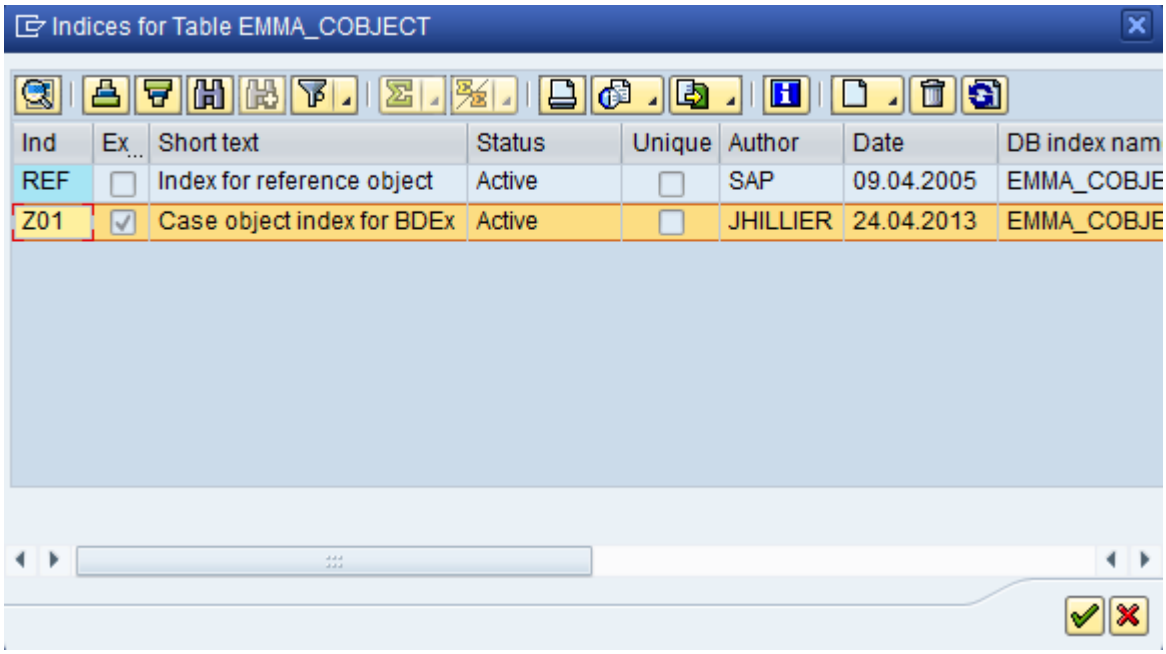
Certain BOR objects do not directly link to master data or work requests in this way, and some custom logic must be added.

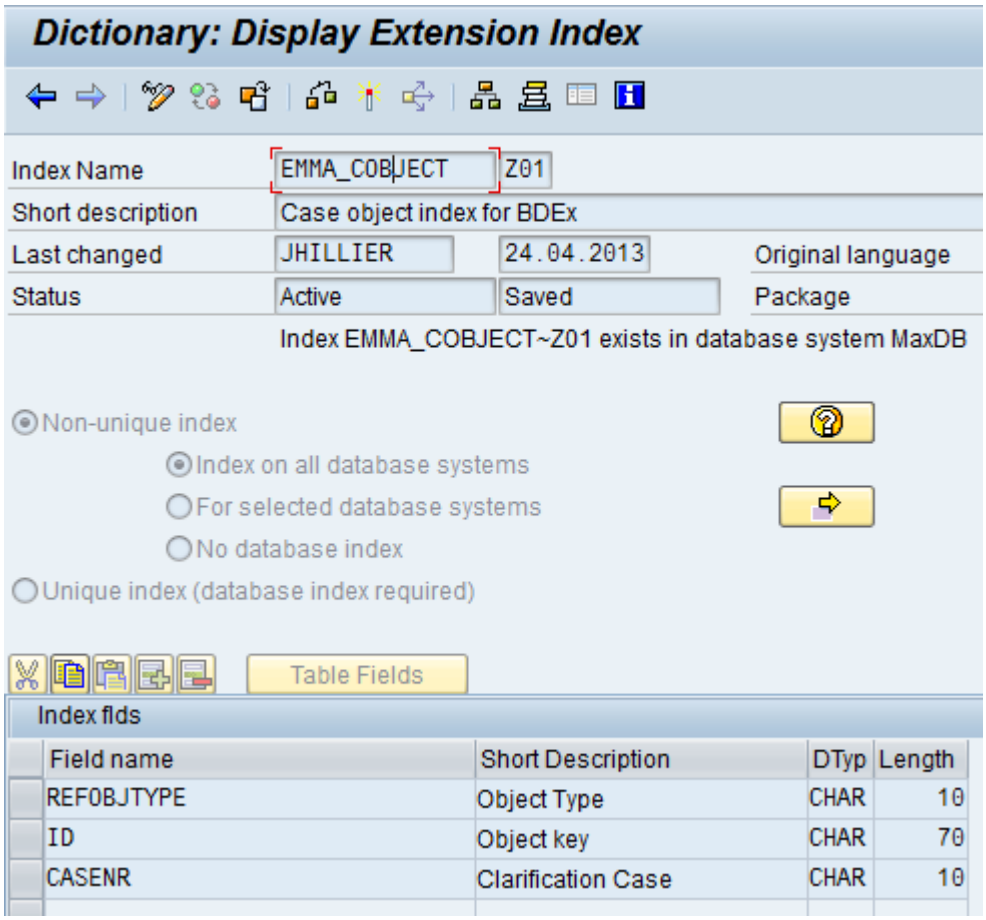
BAdI /BTI/MDE_BADI_GOS can be used to provide a linkage in these instances.

BPEM Customising

Standard configuration

BPEMs are linked to master data through their attached objects and the /BTI/MDE_C_WRBOR table configuration. For performance reasons, a new custom (Z/Y) extension index must be added to the table EMMA_COBJECT in order for this exception to work under high volumes. This must contain the fields REFOBJTYPE, ID and optionally (but preferably) CASENR.




Dictionary: Display Extension Index
Index Name
EMMA_COBJECT
Z01
Short description
Case object index for BDEx
Last changed
JHILLIER
24.04.2013
Original language
Status
Active
Saved
Package
Index EMMA_COBJECT~Z01 exists in database system MaxDB
Non-unique index
Index on all database systems
For selected database systems
No database index
Unique index (database index required)
Table Fields
Index flds
Field name
Short Description
DTyp
Length
REFOBJTYPE
Object Type
CHAR
10
ID
Object key
CHAR
70
CASENR
Clarification Case
CHAR
10

Custom linking logic

Many implementations add custom fields to the standard BPEM structure, providing additional information to the EMMA_CASE line item.

If a master data key is included in these additional fields, such as business partner or contract account number, and this field is indexed, then greater performance can be gained by using this linkage instead of the BOR object.

The BAdI /BTI/MDE_BADI_WR_ISU_BPEM allows custom fields to be used directly in BPEM logic.

Method name	Description
GET_BPEM_CUST	Retrieve non-completed BPEMs that correspond to a table of BOR objects
GET_BPEM_HIST	Retrieve completed BPEMs for a given table of BOR objects
GET_BOR_CUST	Construct a table of BOR objects for a given BPEM object

Table 6 - /BTI/MDE_BADI_WR_ISU_BPEM methods

See the BAdI implementation example class /BTI/MDE_CL_WR_ISU_BPEM_BADI_X for code examples.

BPEM Enhancement Implementation (CL_EMMA_DBL46)

This enhancement is required to connect the Dynamic Work Centre with the standard BPEM functionality provided by SAP and/or the Assignments History feature of both the DWC and BDEx Customer-centric Hub. It is essential that this enhancement catches and maintains the underlying BWC database tables in line with updates to the EMMA Case Header fields and assignments changes otherwise the DWC and/or the Assignments History feature will become disconnected from the work routed to agents.

Since SAP has built their BPEM Solution to allow for Cases to be created manually and also in batch the additional logic required to keep the BWC tables up to date requires the following Methods to be adjusted:

CREATE_CASES – Post Exit needed to ensure a new entry is added to the main BWC Work Request Header Table /BTI/MDE_BWC_WRH for each new BPEM Case being created.
Note: When called in Batch mode this Method may also be provided Agent Assignments up front which will need to be captured in the BWC Assignments table /BTI/MDE_BWC_ASN.

CHANGE_CASE – Post Exit required to modify the existing entry in the BWC table /BTI/MDE_BWC_WRH and keep it in line with any changes made to the EMMA Case header fields. If any Agent Assignment changes have occurred these will also need to be stored in the BWC Assignments table /BTI/MDE_BWC_ASN.

CHANGE_CASE_HEADER – Post Exit required to modify the existing entry in the BWC table /BTI/MDE_BWC_WRH and keep it in line with any changes made to the EMMA Case header fields. If any Agent Assignment changes have occurred these will also need to be stored in the BWC Assignments table /BTI/MDE_BWC_ASN.

CHANGE_CASE_ACTORS – Post Exit required to capture any Agent Assignment changes and cascade them to the BWC Assignments table /BTI/MDE_BWC_ASN.

DELETE_CASES – An Overwrite Exit is required to ensure database integrity and sequencing of updates to allow for the BPEM and BWC tables to be deleted in a logical order.

Method Type Description
CHANGE_CASE PostExit Update BWC Tables with new details and possibly new Processors if required
CHANGE_CASE_HEADER PostExit Update BWC Tables with new details and possibly new Processors if required
CREATE_CASES PostExit Update BWC Tables with new details and Actors

DELETE_CASES Overwrite-Exit Delete BWC Table entries in line with updates to EMMA Tables
CHANGE_CASE_ACTORS PostExit Update BWC Assignments tables

The following sample code should be used as a guide:

CLASS lcl_zbwc_emma_dbl46 DEFINITION DEFERRED.
CLASS cl_emma_dbl46 DEFINITION LOCAL FRIENDS lcl_zbwc_emma_dbl46.
——————————————————————————————

- CLASS LCL_ZBWC_EMMA_DBL46 DEFINITION

——————————————————————————————

*

——————————————————————————————

CLASS lcl_zbwc_emma_dbl46 DEFINITION. PUBLIC SECTION. CLASS-DATA obj TYPE REF TO lcl_zbwc_emma_dbl46. “#EC NEEDED DATA core_object TYPE REF TO cl_emma_dbl46 . “#EC NEEDED INTERFACES: IPO_ZBWC_EMMA_DBL46, IOW_ZBWC_EMMA_DBL46. METHODS: constructor IMPORTING core_object TYPE REF TO cl_emma_dbl46 OPTIONAL. ENDCLASS. “LCL_ZBWC_EMMA_DBL46 DEFINITION

——————————————————————————————

- CLASS LCL_ZBWC_EMMA_DBL46 IMPLEMENTATION

——————————————————————————————

*

——————————————————————————————

CLASS lcl_zbwc_emma_dbl46 IMPLEMENTATION. METHOD constructor. me->core_object = core_object. ENDMETHOD. “CONSTRUCTOR

METHOD iow_zbwc_emma_dbl46~delete_cases.
“—————————————————————————————— *” Declaration of Overwrite-method, do not insert any comments here please! *” *“methods DELETE_CASES *” importing *” !IV_EMMARUNID type EMMA_RUNID optional *” !IV_INTNR type EMMA_INTNR optional *” exceptions *” NOT_DELETED *” NOT_FOUND .
“—————————————————————————————— DATA lt_emma_case TYPE STANDARD TABLE OF emma_case WITH NON-UNIQUE DEFAULT KEY INITIAL SIZE 0. FIELD-SYMBOLS: TYPE emma_case. DATA: ls_emma_hdr TYPE emma_hdr. DATA: co_obj TYPE balobj_d VALUE ‘EMMA’. DATA: co_subobj TYPE balsubobj VALUE ‘CASE’. DATA: lv_extnum TYPE balnrext. DATA: lt_emma_int TYPE STANDARD TABLE OF emma_int WITH NON-UNIQUE DEFAULT KEY INITIAL SIZE 0. DATA: ls_emma_int TYPE emma_int. DATA:

ls_bal_fil TYPE bal_s_lfil, lt_extnum TYPE bal_r_extn, ls_extnum TYPE bal_s_extn, lt_obj TYPE bal_r_obj, ls_obj TYPE bal_s_obj, lt_subobj TYPE bal_r_sub, ls_subobj TYPE bal_s_sub, lt_balhdr TYPE balhdr_t, ls_balhdr TYPE balhdr, ls_log TYPE bal_s_log, ls_data TYPE emma_case, ls_context TYPE emma_case_context, lt_loghandle TYPE bal_t_logh, lv_biw TYPE boole_d, ls_biw TYPE biw_emma_case, lt_biw TYPE TABLE OF biw_emma_case, lv_wr_key TYPE /bti/mde_de_wr_key.

- update EMMA_HDR and EMMA_INT only as long as classic EMMA is still calling method SELECT SINGLE * FROM emma_hdr INTO ls_emma_hdr WHERE runid = iv_emmarunid.

IF iv_intnr IS INITIAL.

- read all intervals SELECT * INTO TABLE lt_emma_case FROM emma_case WHERE runid = iv_emmarunid. ELSE. SELECT * INTO TABLE lt_emma_case FROM emma_case WHERE runid = iv_emmarunid AND intnr = iv_intnr.
- read only one interval ENDIF. IF sy-subrc <> 0. IF ls_emma_hdr-version < cl_emma_job=>co_version_5.
- update EMMA_HDR and EMMA_INT only as long as classic EMMA is still calling method UPDATE emma_hdr SET emma_case = 0 cgen_status = ‘0’ WHERE runid = iv_emmarunid. UPDATE emma_int SET emma_case = 0 cgen_status = ‘0’ WHERE runid = iv_emmarunid. ENDIF. RAISE not_found. ENDIF.
- don’t delete manual cases IF ls_emma_hdr-emma_jobrustatus = ‘4’. “cl_emma_job=>co_job_status_comp_no_del. RAISE not_deleted. ENDIF.
- all cases need to be in status NEW or CANCELLED LOOP AT lt_emma_case ASSIGNING . CHECK -status NE cl_emma_case=>co_status_new AND -status NE cl_emma_case=>co_status_canc. RAISE not_deleted. ENDLOOP.

DELETE emma_case FROM TABLE lt_emma_case. LOOP AT lt_emma_case ASSIGNING . DELETE FROM emma_cactor WHERE casenr = -casenr. DELETE FROM emma_cmsg_link WHERE casenr = -casenr. DELETE FROM emma_cobject WHERE casenr = -casenr. DELETE FROM emma_csolp WHERE casenr = -casenr. DELETE FROM stxh WHERE tdobject = cl_emma_case=>co_text_obj AND tdname = -casenr AND tdid = cl_emma_case=>co_text_idcase. DELETE FROM stxb WHERE relid = ‘TX’ AND tdobject = cl_emma_case=>co_text_obj AND tdname = -casenr AND tdid = cl_emma_case=>co_text_idcase. DELETE FROM stxl WHERE relid = ‘TX’ AND tdobject = cl_emma_case=>co_text_obj AND tdname = -casenr AND tdid = cl_emma_case=>co_text_idcase.

- Update the BWC tables to cancel any deleted cases. lv_wr_key = -casenr.

UPDATE /bti/mde_bwc_wrh SET status = /bti/mde_cl_bwc_wr_core=>pc_status_cancelled WHERE class = /bti/mde_cl_wr_isu_bpem=>pc_wrclass AND wr_key = lv_wr_key. DELETE FROM /bti/mde_bwc_asn WHERE class = /bti/mde_cl_wr_isu_bpem=>pc_wrclass AND wr_key = lv_wr_key. DELETE FROM /bti/mde_bwc_fwd WHERE class = /bti/mde_cl_wr_isu_bpem=>pc_wrclass AND wr_key = lv_wr_key. ENDLOOP. IF ls_emma_hdr-version < cl_emma_job=>co_version_5.

- update EMMA_HDR and EMMA_INT only as long as classic EMMA is still calling method UPDATE emma_hdr SET emma_case = 0 cgen_status = ‘0’ WHERE runid = iv_emmarunid. UPDATE emma_int SET emma_case = 0 cgen_status = ‘0’ WHERE runid = iv_emmarunid. ENDIF.

- check if DataSource 0FCEMMA_CASE is initialized SELECT initstate FROM roosprmsc INTO lv_biw WHERE oltpsource = cl_emma_case=>co_datasource AND initstate = cl_emma_case=>co_true. EXIT. ENDSELECT. IF lv_biw = cl_emma_case=>co_true. “DataSource initialized
- add delete records to delta table LOOP AT lt_emma_case ASSIGNING . -changed_date = sy-datum. -changed_time = sy-uzeit. -changed_by = sy-uname.
- fill deltaqueue table for BW MOVE-CORRESPONDING TO ls_biw.
- add business process area IF NOT -bpcode IS INITIAL. SELECT SINGLE bparea FROM emma_bpc INTO ls_biw-bparea WHERE bpcode = -bpcode. ENDIF.
- draw new GUID CALL FUNCTION ‘GUID_CREATE’ IMPORTING ev_guid_16 = ls_biw-delta.
- mark case as deleted ls_biw-updmode = ‘D’. APPEND ls_biw TO lt_biw. ENDLOOP.
- insert BW delta records INSERT biw_emma_case FROM TABLE lt_biw. ENDIF.

IF iv_intnr IS INITIAL. UPDATE emma_hdr SET emma_case = 0 cgen_status = 0 WHERE runid = iv_emmarunid. ENDIF.

- try to find the existing action log for the cases
- build ranges EXTERNAL_NUMBER CLEAR ls_extnum. ls_extnum-sign = ‘I’. ls_extnum-option = ‘EQ’. ls_extnum-low = iv_emmarunid. APPEND ls_extnum TO lt_extnum.
- build ranges OBJECT CLEAR ls_obj. ls_obj-sign = ‘I’. ls_obj-option = ‘EQ’. ls_obj-low = co_obj. APPEND ls_obj TO lt_obj.
- build ranges SUBOBJECT CLEAR ls_subobj. ls_subobj-sign = ‘I’. ls_subobj-option = ‘EQ’. ls_subobj-low = co_subobj. APPEND ls_subobj TO lt_subobj.
- build filter structure CLEAR ls_bal_fil. ls_bal_fil-extnumber = lt_extnum. ls_bal_fil-object = lt_obj. ls_bal_fil-subobject = lt_subobj.

CALL FUNCTION ‘BAL_DB_SEARCH’ EXPORTING i_s_log_filter = ls_bal_fil IMPORTING e_t_log_header = lt_balhdr EXCEPTIONS log_not_found = 1 no_filter_criteria = 2 OTHERS = 3. CHECK sy-subrc = 0. CALL FUNCTION ‘BAL_DB_DELETE’ EXPORTING i_t_logs_to_delete = lt_balhdr EXCEPTIONS no_logs_specified = 1 OTHERS = 2. ENDMETHOD. “IOW_ZBWC_EMMA_DBL46~DELETE_CASES METHOD ipo_zbwc_emma_dbl46~create_cases. “═══ *” Declaration of POST-method, do not insert any comments here please! *” *“methods CREATE_CASES *” changing *” !CT_CASES type EMMA_CL_CASE_T *” !EV_CASECNT type I *” !EV_OBJCNT type I *” !EV_ACTCNT type I *” !EV_MSGCNT type I *” !EV_SOLPCNT type I *” exceptions *” NO_CASES_CREATED *” ERROR_DRAWING_NEW_CASE_NUMBER *” ERROR_SAVING_TEXT . “═══ DATA: lr_db TYPE REF TO /bti/mde_cl_bwc_db, lr_case TYPE REF TO cl_emma_case, ls_data TYPE emma_case, lt_act TYPE tswhactor, lr_wr TYPE REF TO /bti/mde_cl_bwc_wr_core. FIELD-SYMBOLS: TYPE REF TO cl_emma_case. CREATE OBJECT lr_db. LOOP AT ct_cases ASSIGNING . ls_data = ->get_data( ). lr_case = core_object->read_case_detail(ls_data-casenr). ls_data = lr_case->get_data( ). lt_act = lr_case->get_actors( ). CALL METHOD lr_db->load_from_bpem EXPORTING is_case = ls_data it_actors = lt_act RECEIVING rr_bwc_wr = lr_wr. CALL METHOD lr_db->update_db EXPORTING ir_wr = lr_wr EXCEPTIONS update_header_failed = 1 update_assignments_failed = 2 OTHERS = 3. IF sy-subrc <> 0.

MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF. CLEAR: ls_data, lt_act[]. ENDLOOP. ENDMETHOD. “IPO_ZBWC_EMMA_DBL46~CREATE_CASES METHOD ipo_zbwc_emma_dbl46~change_case_header. “———————————————————————————————— *” Declaration of POST-method, do not insert any comments here please! *” *“methods CHANGE_CASE_HEADER *” importing *” !IS_DBCASE type EMMA_CASE *” !IV_WRITE_CDOC type XFELD *” changing *” !CS_CASE type EMMA_CASE *” exceptions *” UPDATE_FAILED . “———————————————————————————————— DATA: lr_db TYPE REF TO /bti/mde_cl_bwc_db, ls_bwc_wrh TYPE /bti/mde_bwc_wrh, lt_proclog TYPE /bti/mde_tt_bwc_proclog, lr_wr TYPE REF TO /bti/mde_cl_bwc_wr_core, lt_actors TYPE tswhactor.

- Retrieve the latest case actors SELECT * FROM emma_cactor INTO CORRESPONDING FIELDS OF TABLE lt_actors WHERE casenr = cs_case-casenr.
- Update the BWC tables CREATE OBJECT lr_db.

CALL METHOD lr_db->load_from_bpem EXPORTING is_case = cs_case is_dbcase = is_dbcase it_actors = lt_actors RECEIVING rr_bwc_wr = lr_wr. ls_bwc_wrh = lr_wr->get_header( ). IF NOT cs_case-currproc IS INITIAL AND NOT cs_case-currproc EQ is_dbcase-currproc. lt_proclog = lr_wr->get_processor_log( ). ENDIF. CALL METHOD lr_db->update_header EXPORTING is_header = ls_bwc_wrh it_processors = lt_proclog EXCEPTIONS update_failed = 1 OTHERS = 2. IF sy-subrc <> 0. MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF. ENDMETHOD. “IPO_ZBWC_EMMA_DBL46~CHANGE_CASE_HEADER METHOD ipo_zbwc_emma_dbl46~change_case. “———————————————————————————————— *” Declaration of POST-method, do not insert any comments here please! *” *“methods CHANGE_CASE *” importing *” !IR_CASE type ref to CL_EMMA_CASE *” !IR_DBCASE type ref to CL_EMMA_CASE *” exceptions *” ERROR_INSERTING_OBJECTS *” ERROR_UPDATING_CASE *” ERROR_SAVING_TEXT . “———————————————————————————————— DATA: lr_case TYPE REF TO cl_emma_case, ls_data TYPE emma_case, ls_dbdata TYPE emma_case, lt_act TYPE tswhactor, lr_db TYPE REF TO /bti/mde_cl_bwc_db, lr_wr TYPE REF TO /bti/mde_cl_bwc_wr_core. ls_data = ir_case->get_data( ).

- Need most up to date version of case, so retrieve from db instead of using ir_case lr_case = core_object->read_case_detail(ls_data-casenr). ls_data = lr_case->get_data( ).

ls_dbdata = ir_dbcase->get_data( ). lt_act = lr_case->get_actors( ).

- Use this data to update BWC tables CREATE OBJECT lr_db. CALL METHOD lr_db->load_from_bpem EXPORTING is_case = ls_data is_dbcase = ls_dbdata it_actors = lt_act RECEIVING rr_bwc_wr = lr_wr.

CALL METHOD lr_db->update_db EXPORTING ir_wr = lr_wr EXCEPTIONS update_header_failed = 1 update_assignments_failed = 2 OTHERS = 3. IF sy-subrc <> 0. MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF. ENDMETHOD. “IPO_ZBWC_EMMA_DBL46~CHANGE_CASE METHOD IPO_ZBWC_EMMA_DBL46~CHANGE_CASE_ACTORS.

“—— *” Declaration of POST-method, do not insert any comments here please! *” *“methods CHANGE_CASE_ACTORS *” importing *” !IR_CASE type ref to CL_EMMA_CASE *” !IT_ACTORS type TSWHACTOR *” exceptions *” ERROR_UPDATING_ACTORS . “—— DATA: lr_case TYPE REF TO cl_emma_case, ls_data TYPE emma_case, ls_dbdata TYPE emma_case, lt_act TYPE tswhactor, lr_db TYPE REF TO /bti/mde_cl_bwc_db, lr_wr TYPE REF TO /bti/mde_cl_bwc_wr_core. ls_data = ir_case->get_data( ).

- Need most up to date version of case, so retrieve from db instead of using ir_case lr_case = core_object->read_case_detail(ls_data-casenr). ls_data = lr_case->get_data( ). lt_act = lr_case->get_actors( ).
- Use this data to update BWC tables CREATE OBJECT lr_db. CALL METHOD lr_db->load_from_bpem EXPORTING is_case = ls_data it_actors = lt_act RECEIVING rr_bwc_wr = lr_wr.

CALL METHOD lr_db->update_db EXPORTING ir_wr = lr_wr EXCEPTIONS update_header_failed = 1 update_assignments_failed = 2 OTHERS = 3. IF sy-subrc <> 0. MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4. ENDIF.

ENDMETHOD.

ENDCLASS.

BPEM Inbox Integration

BDEx can be tightly integrated with the standard EMMACL and EMMACLS BPEM inboxes with very little effort required, allowing BDEx to be called directly from a BPEM case in the worklist. To enable this functionality, perform the following:

Implement the BAdI BADI_EMMA_CASE in enhancement spot EMMA_CASE.

This BAdI is called when a user views or changes a BPEM case from EMMACL and EMMACLS, and can be used to launch BDEx instead. It includes a check to a flag held in memory, so that BDEx is only called under circumstances defined in the next section.

The following code should be used to implement the BAdI logic:

```
class zbdex_cl_emma_case definition
  public
  final
  create public .

  public section.

    interfaces if_badi_emma_case .
    interfaces if_badi_interface .

    constants co_bdex_mem_id type char20 value '/BTI/MDE_CASELIST' ##NO_TEXT.
  protected section.
  private section.
ENDCLASS.



CLASS ZBDEX_CL_EMMA_CASE IMPLEMENTATION.


*
<SIGNATURE>-------------------------------------------------------------------------------+
* | Static Public Method ZBDEX_CL_EMMA_CASE=>IF_BADI_EMMA_CASE~TRANSACTION_START
*
+-------------------------------------------------------------------------------------------
* | [--->] IV_CASENR                      TYPE        EMMA_CNR(optional)
* | [--->] IV_CCAT                        TYPE        EMMA_CCAT(optional)
* | [--->] IV_TEMPLATE_CASE               TYPE        EMMA_CNR(optional)
```

```
* | [--->] IV_WMODE                        TYPE        EMMA_CTXN_WMODE
* | [--->] IV_ALLOW_TOGGLE_DISPCHAN        TYPE        FLAG (default =' ')
* | [--->] IV_NEXT_PREV_CASE               TYPE        NUM1 (default ='0')
* | [<---] EV_CASENR                       TYPE        EMMA_CNR
* | [<---] EV_OKCODE                       TYPE        SYUCOMM
* | [EXC!] CASE_NOT_FOUND
* | [EXC!] INCORRECT_WORKMODE
* | [EXC!] INCORRECT_PARAMETERS
*
+-------------------------------------------------------------------------------</SIGNATU
  method if_badi_emma_case~transaction_start.


    data:
      lv_bdex   type flag,
      lv_called type flag,
      lo_bpem   type ref to /bti/mde_cl_wr_isu_bpem,
      lo_md     type ref to /bti/mde_cl_md_object_base,
      lo_action type ref to /bti/mde_cl_action.

* Check whether this call is from an emma caselist hotspot click
    import lv_bdex = lv_bdex from memory id co_bdex_mem_id.

    if lv_bdex is not initial.
      free memory id co_bdex_mem_id.

      call method /bti/mde_cl_wr_isu_bpem=>get_object
        exporting
          x_key                    = iv_casenr
          x_ignore_existence_check = space
        receiving
          r_object                 = lo_bpem
        exceptions
          ex_object_not_found      = 1
          others                   = 2.
      if sy-subrc <> 0.
        raise case_not_found.
      endif.

      lo_md = lo_bpem->get_primary_md_object( ).
      if lo_md is not initial.

        call method /bti/mde_cl_action=>factory
          exporting
            actionid  = /bti/mde_cl_application_main=>pc_actionid_bdex_wr
            processor = lo_bpem
          receiving
            action    = lo_action.

        if lo_action is not initial.
```

```
          lo_action->execute( exporting x_no_log = 'X' ).
          lv_called = 'X'.

        endif.
      endif.
    endif.

* If BDEx wasn't called successfully; either becase the flag wasn't set, or
* it was set, but BDEx couldn't determine the customer, just call the regular
transaction.
    if lv_called is initial.
      call function 'EMMA_CASE_TRANSACTION_START'
        exporting
          iv_casenr                = iv_casenr
          iv_ccat                  = iv_ccat
          iv_template_case         = iv_template_case
          iv_wmode                 = iv_wmode
          iv_allow_toggle_dispchan = iv_allow_toggle_dispchan
          iv_next_prev_case        = iv_next_prev_case
        importing
          ev_casenr                = ev_casenr
          ev_okcode                = ev_okcode
        exceptions
          case_not_found           = 1
          incorrect_workmode       = 2
          incorrect_parameters     = 3.
      case sy-subrc.
        when 1. raise case_not_found.
        when 2. raise incorrect_workmode.
        when 3. raise incorrect_parameters.
      endcase.
    endif.

  endmethod.
ENDCLASS.
```

Note that if BDEx is not called successfully using this method, it will fall back gracefully to the standard BPEM case transaction.

Enhance program REMMACASELIST

Add a code enhancement at the start of form ALV_USER_COMMAND (found around line 386), setting the BDEx memory flag if the action was a hotspot or double-click and add the following code:

```
  if iv_ucomm eq '&IC1'.
*   Export flag to indicate hotspot click or double-click.
```

```
*   This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
    export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
  endif.
```

The enhanced form will look like this:

```
*&---------------------------------------------------------------------*
*&      Form  alv_user_command
*&        DRILL DOWN WORK
*----------------------------------------------------------------------*
FORM alv_user_command USING iv_ucomm    LIKE sy-ucomm
                            cs_selfield TYPE slis_selfield. "#EC CALLED
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start:
(1)--------------------------------------------------------------------------$*$*
ENHANCEMENT 1  ZBDEX_ENH_EMMACASELIST.    "active version

  if iv_ucomm eq '&IC1'.
*   Export flag to indicate hotspot click or double-click.
*   This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
    export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
  endif.

ENDENHANCEMENT.
*$*$-End:
(1)--------------------------------------------------------------------------$*$*

  DATA: lt_rows TYPE lvc_t_row,
...
```

Enhance program REMMACASELIST_SHL

Add a code enhancement at the start of method HANDLE_HOTSPOT of class LC_CLALV_EVENT_RECEIVER (found near line 163) with the following logic:

```
if e_column_id eq 'CASENR'. "#EC NO_TEXT
*  Export flag to indicate hotspot click or double-click.
*  This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
  export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
endif.
```

The enhanced method will look like this:

```
CLASS lc_clalv_event_receiver IMPLEMENTATION.
  METHOD handle_hotspot.
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start:
(1)----------------------------------------------------------------------$*$*
ENHANCEMENT 1  ZBDEX_ENH_EMMACLS.    "active version
  if e_column_id eq 'CASENR'. "#EC NO_TEXT
*  Export flag to indicate hotspot click or double-click.
*  This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
    export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
  endif.
ENDENHANCEMENT.
*$*$-End:
(1)----------------------------------------------------------------------$*$*

    DATA: ls_outtab LIKE LINE OF gt_outtab.
...
```

Add a code enhancement at the start of method HANDLE_DOUBLE_CLICK of class LC_CLALV_EVENT_RECEIVER (found near line 263) with the following logic:

```
* Export flag to indicate hotspot click or double-click.
* This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
  export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
```

The enhanced method should look like this:

```
*&---------------------------------------------------------------------*
*&    Class lc_dcalv_event_receiver implementation
*&  Handles double click on case list alv grid
*&---------------------------------------------------------------------*
CLASS lc_dcalv_event_receiver IMPLEMENTATION.
  METHOD handle_double_click.
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
*$*$-Start:
(2)----------------------------------------------------------------------$*$*
ENHANCEMENT 2  ZBDEX_ENH_EMMACLS.    "active version
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
A
* Export flag to indicate hotspot click or double-click.
* This can then be checked in the BPEM transaction BAdI, in order to call BDEx
if necessary
  export lv_bdex = abap_true to memory id zbdex_cl_emma_case=>co_bdex_mem_id.
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
ENDENHANCEMENT.
```

```
*$*$-End:
(2)----------------------------------------------------------------------$*$*
    DATA: ls_outtab LIKE LINE OF gt_outtab.
```

With these enhancements in place, clicking a hotspot or double-clicking a line item in EMMACL and EMMACLS will attempt to call BDEx directly.

Get Work button in EMMACL/EMMACLS

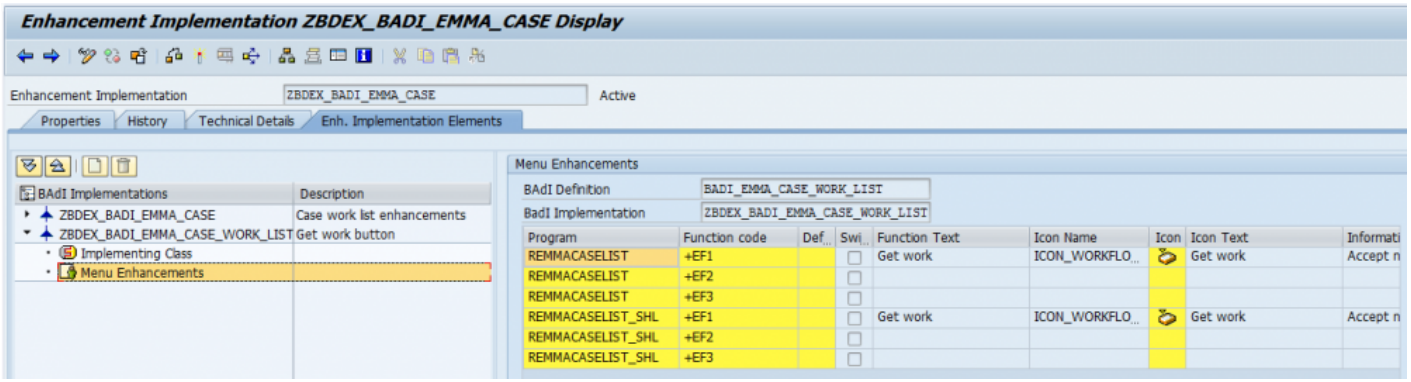
A get work button can be added to EMMACL and EMMACLS very easily, allowing a user to quickly accept work according to predefined business rules. To do so, perform the following steps:

Implement the BADI_EMMA_CASE_WORK_LIST BAdI

The BADI_EMMA_CASE_WORK_LIST business add-in allows up to three custom buttons to be added to EMMACL and EMMACLS. It also allows standard functions to be disabled, which can be useful if agents should only accept work via the Get Work button.

Add Get Work buttons to the menu enhancements

Once the BAdI has been implemented the next step is to add the actual buttons to the screen. REMMACASELIST refers to EMMACL, and REMMACASELIST_SHL refers to EMMACLS. The following image contains an example of a Get Work button defined in each:



Handle the button function-codes

The next step is to handle the button-clicks, which produce the function codes ‘+EF1’, ‘+EF2’ and ‘+EF3’ when pressed.

To do this implement the BAdI method IF_BADI_EMMA_CASE_WORK_LIST~CALL_ENHANCEMENT_FUNCTION. This may look something like the following:

```
method if_badi_emma_case_work_list~call_enhancement_function.

  case iv_ucomm.
    when '+EF1'. "Get work
      get_work( ).
    when '+EF2'. "Not yet bound
    when '+EF3'. "Not yet bound
  endcase.

endmethod.
```

Once this is done the final step is to implement the `get_work` method. This is generally divided into three sections; identifying the cases to be accepted, accepting them, then refreshing the case list.

```
method get_work.

  data:
    casenrs type table of emma_case-casenr.

  field-symbols:
    <casenr> like line of casenrs.

* Retrieve some cases
  select casenr from emma_case
    inner join emma_cactor
    into table casenrs
    up to 5 rows
    where ...
    order by ....

* Accept them
  loop at casenrs assigning <casenr>.
    call function 'BAPI_EMMA_CASE_ACCEPT'
      exporting
        case               = <casenr>.
  endloop.

* Refresh the list
  call function 'SAPGUI_SET_FUNCTIONCODE'
   exporting
     functioncode          = 'REFRESH'
   exceptions
     function_not_supported = 1
     others                 = 2.

endmethod.
```

The logic for determining the cases to be accepted will differ depending on business needs and may require additional steps, though the overall structure of this method should be sufficient.

Custom work request classes

In addition to the standard work request classes supported by BDEx, custom classes can be added as required. Note that this is a complex job and should be done with the help of Basis Technologies.

The following steps can be used to create a new exception type.

1. Create a new subclass of /BTI/MDE_CL_WR_CORE

The interface /BTI/MDE_IF_ACTION_PROCESSOR will then be inherited. Additionally, interfaces /BTI/MDE_IF_WR_CORE and /BTI/MDE_IF_WR_REPORT should be implemented in the new class. These interfaces will provide the following inherited methods:
/BTI/MDE_IF_ACTION_PROCESSOR~PROCESS_ACTION
/BTI/MDE_IF_WR_CORE~RETRIEVE_WORK_REQUESTS
/BTI/MDE_IF_WR_CORE~RETRIEVE_WORK_REQUESTS_HIST
/BTI/MDE_IF_WR_REPORT~RETRIEVE_ALL

2. Define methods

At least the following methods will need to be redefined:

Method name	Description
/BTI/MDE_IF_WR_CORE~RETRIEVE_WORK_REQUESTS	Used to extract active work requests
/BTI/MDE_IF_WR_CORE~RETRIEVE_WORK_REQUESTS_HIST	Used to extract historic work requests
CONSTRUCTOR	Used to create the work request object from the data retrieved in the previous methods
LOAD_DATA	Used to retrieve any additional data used by the class
/BTI/MDE_IF_ACTION_PROCESSOR~PROCESS_ACTION	Used to provide the link between action methods and the OK codes passed back from the context menus
/BTI/MDE_IF_WR_REPORT~RETRIEVE_ALL	Retrieve all work requests for WR summary report

Table 7 - Methods to redefine

3. Configure the work request class

An entry should be added to /BTI/MDE_C_WRCLS

4. Define the actions

Create methods for the actions, using the following format:
ACTION_x
These take an import parameter of X_ACTION, TYPE REF TO /BTI/MDE_CL_ACTION.

5. Add the actions to the config tables

Entries should be added in the following tables:
/BTI/MDE_C_ACT
/BTI/MDE_C_EXACT

BDEx BAdIs

There are a number of BDEx BAdIs that can be implemented to enhance or customise BDEx.

This section provides all BAdI classes made available, along with a brief description of their use.

Enhancement Spot /BTI/MDE_ENH_WR

This enhancement spot contains BAdIs for work requests and master data objects at a generic level.

Customer actions for work requests

/BTI/MDE_BADI_ACTION provides the ability to add custom actions to work request classes.

See the Adding Custom Actions chapter for detailed instructions.

Method	Description
/BTI/MDE_IF_ACTION~PROCESS_ACTIONS	This method is used to call custom code when a custom work request action is triggered in BDEx. See the enhancement example implementation Z_ENH_ACTION_EXAMPLE1 for instructions on how to use this.

Add custom action log information

The /BTI/MDE_ACTLOG activity log table can be extended by implementing the CI_BTI_MDE_ACTLOG custom include structure.

Fields added in this way are populated using the /BTI/MDE_BADI_ACTLOG BAdI.

Method	Description
/BTI/MDE_IF_ACTLOG_CUST~UPDATE_ACTLOG	This method is used to update fields in the CI_BTI_MDE_ACTLOG append structure of the CS_ACTLOG structure. IMPORTANT: Do not update any other fields outside of the append structure, otherwise the activity logging will not work as expected. See /BTI/MDE_CL_ACTLOG_EXAMPLE for an example implementation.

Execute generic object service method

Implement /BTI/MDE_BADI_GOS to allow custom logic to be executed when calling BDEx via the generic object services dropdown.

See the example implementation class for more information.

Method	Description
/BTI/MDE_IF_GOS~GET_CUSTOMER	This method allows custom logic to be used when linking a BOR object type and key to a BDEx parameter

Change the master data ALV

Implement /BTI/MDE_BADI_GUI_MDLIST to modify how the master data will appear on the left hand side of the BDEx screen.

See the example implementation for a detailed usage guide.

Method	Description
/BTI/MDE_IF_GUI_MDLIST~CHANGE_MD_GRID	Update the master data ALV grid
/BTI/MDE_IF_GUI_MDLIST~CHANGE_MD_TREE	Update the master data ALV tree

Change the work request details tree

Implement /BTI/MDE_BADI_GUI_WRDET to modify the information that appears in the work request details screen when a work request is double-clicked.

Method	Description
/BTI/MDE_IF_GUI_WRDET~CHANGE_WRDET	Add, change, or remove lines of the work request details screen.

Change the work request ALV

Implement /BTI/MDE_BADI_GUI_WRLIST to modify how the work requests appear in the main work request list.

Method	Description
/BTI/MDE_IF_GUI_WRLIST~CHANGE_ALV	Add, remove, or update the contents of the ALV grid before output

Customer actions for master data objects

Implement /BTI/MDE_BADI_MD_ACTION to enable custom master data actions.

See the earlier chapter for more information.

Method	Description
/BTI/MDE_IF_MD_ACTION~PROCESS_ACTIONS	Process the action codes passed when custom master data actions are selected in BDEx.

Retrieve work requests

Implement /BTI/MDE_BADI_WR_RETRIEVE to modify which work requests are retrieved in BDEx.

Method	Description
/BTI/MDE_IF_WR_RET~CHANGE_WORK_REQUESTS	Change the list of work requests before output. Additional requests could be retrieved here, or existing ones removed before output.

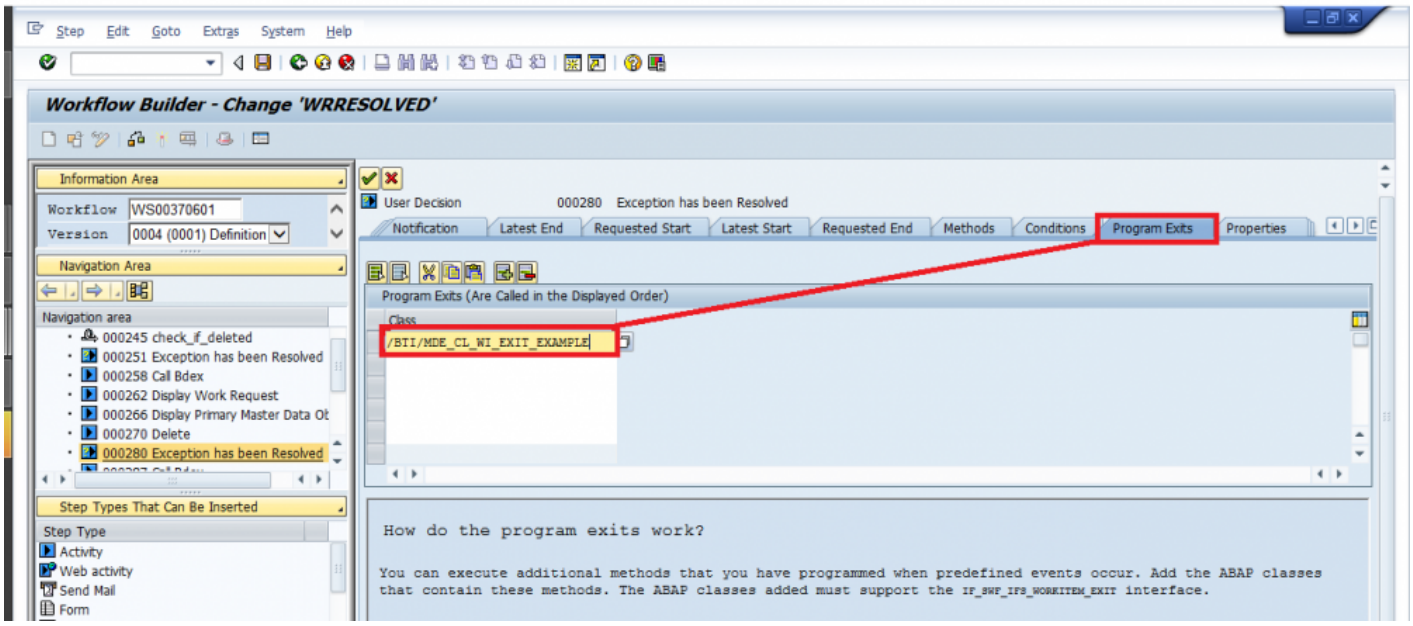
SAP Business Workflow Program Exit for Dynamic Work Center

!https://cdn.manula.com/user/3588/img/large/bwc-wi-exit.png!This enhancement is critical to ensuring that Workflow steps, typically Work Items, that are intended to be published in the BWC invoke the necessary logic to update the appropriate entries in the /BTI/MDE_BWC_WRH and /BTI/MDE_BWC_FWD tables.

As standard BTI provides the logic for this enhancement and cannot guarantee the continued functionality of the Dynamic Work Centre should the logic provided be deactivated or adjusted by non-BTI developers.

The BAdI consists of a single Method discussed here:

Essentially this BAdI seeks to implement a SAP standard Interface Method IF_SWF_IFS_WORKITEM_EXIT~EVENT_RAISED that can be activated directly from the Workflow Builder (Transaction SWDD) for any Workflow definition as shown:



To make use of the BAdI for a particular Work Item step simply add the Implementing Class Name ZCL_BWC_WORKITEM_EXIT and the code will be invoked automatically.

Method	Type	Description
IF_SWF_IFS_WORKITEM_EXIT~EVENT_RAISED	Plug-In	Work Item Event Raised catcher

Enhancement Spot /BTI/MDE_ES_WR_ISU

This enhancement spot is used for ISU specific functionality, such as ISU work requests like BPEM cases and billing and invoicing outsorts.

Billing outsort custom validations

Implement /BTI/MDE_BADI_WR_ISU_BOUT to allow the results of custom validation functions to be reflected clearly.

Method	Description
/BTI/MDE_IF_BADI_ISU_BOUT~GET_VALIDATION_TEXT	Use the billing document and validation class to determine exactly what validation failed so that it can be clearly displayed.

BPEM custom fields

Implement /BTI/MDE_BADI_WR_ISU_BPEM to integrate any custom fields or functionality with the general BDEx BPEM work request class.

See example class /BTI/MDE_CL_WR_ISU_BPEM_BADI_X for more details.

Method	Description
/BTI/MDE_IF_WR_ISU_BPEM~GET_BPEM_CUST	This method is no longer used by BDEx.
/BTI/MDE_IF_WR_ISU_BPEM~GET_BOR_CUST	Load BDEx from a BPEM screen using custom fields appended in structure CI_EMMA_CASE
/BTI/MDE_IF_WR_ISU_BPEM~GET_BPEM_HIST	This method is no longer used by BDEx.
/BTI/MDE_IF_WR_ISU_BPEM~GET_BPEMS	Retrieve BPEM cases using custom fields appended in structure CI_EMMA_CASE . Note that both active and historical BPEMs should be retrieved using this BAdI method.

Invoice outsort custom validations

Implement /BTI/MDE_BADI_WR_ISU_INVO to allow the results of custom validation functions to be reflected clearly.

Method	Description
/BTI/MDE_IF_BADI_ISU_INVO~GET_VALIDATION_TEXT	Use the invoice document and validation class to determine exactly what validation failed so that it can be clearly displayed.

No device custom logic

Implement /BTI/MDE_BADI_WR_ISU_NODV to add custom logic to the “No Device Found” work request.

This is primarily used to identify installations that do not require a device. See /BTI/MDE_CL_WR_NODV_EXAMPLE for more details.

Method	Description
/BTI/MDE_IF_BADI_ISU_NODV~FILTER_INSTALLATIONS	Use this method to filter out certain installations that should not display a “no device found” work request.

Enhancement Spot /BTI/MDE_BWC_ENH_CUSTDATA

This Enhancement Spot is used to adjust the Dynamic Work Centre and handle updates to the underlying Database Tables that support it.

Transfer customer fields to Dynamic Work Center tables

Implement BAdI /BTI/MDE_BWC_BADI_CUSTFIELDS to handle custom fields being copied across in the BWC Database Tables by means of accessing fields added to the .CI_BWC_CUST_INCL structure.

An example implementation is provided with hints.

Custom logic for ‘Get Work’ button

Implement BAdI /BTI/MDE_BWC_BADI_GET_WORK to introduce custom logic to be invoked when an Agent clicks on the ‘Get Work’ button int he Dynamic Work Center toolbar.

An example implementation is provided to demonstrate an alternative approach to relying on standard HR Organizational Assignments and make use of ‘Search Presets’.

Get Organizational Assignments

Implement BAdI /BTI/MDE_BWC_BADI_GET_ORG_ASS to introduce:

a) an alternative approach to building the Organizational Assignments of the Agent;

b) an alternative approach to determining whether the Agent should be considered a Manager and have access to the ‘Manager’s View’ within the Dynamic Work Center;

c) an alternative approach to determining the members of the Team managed by the Agent.

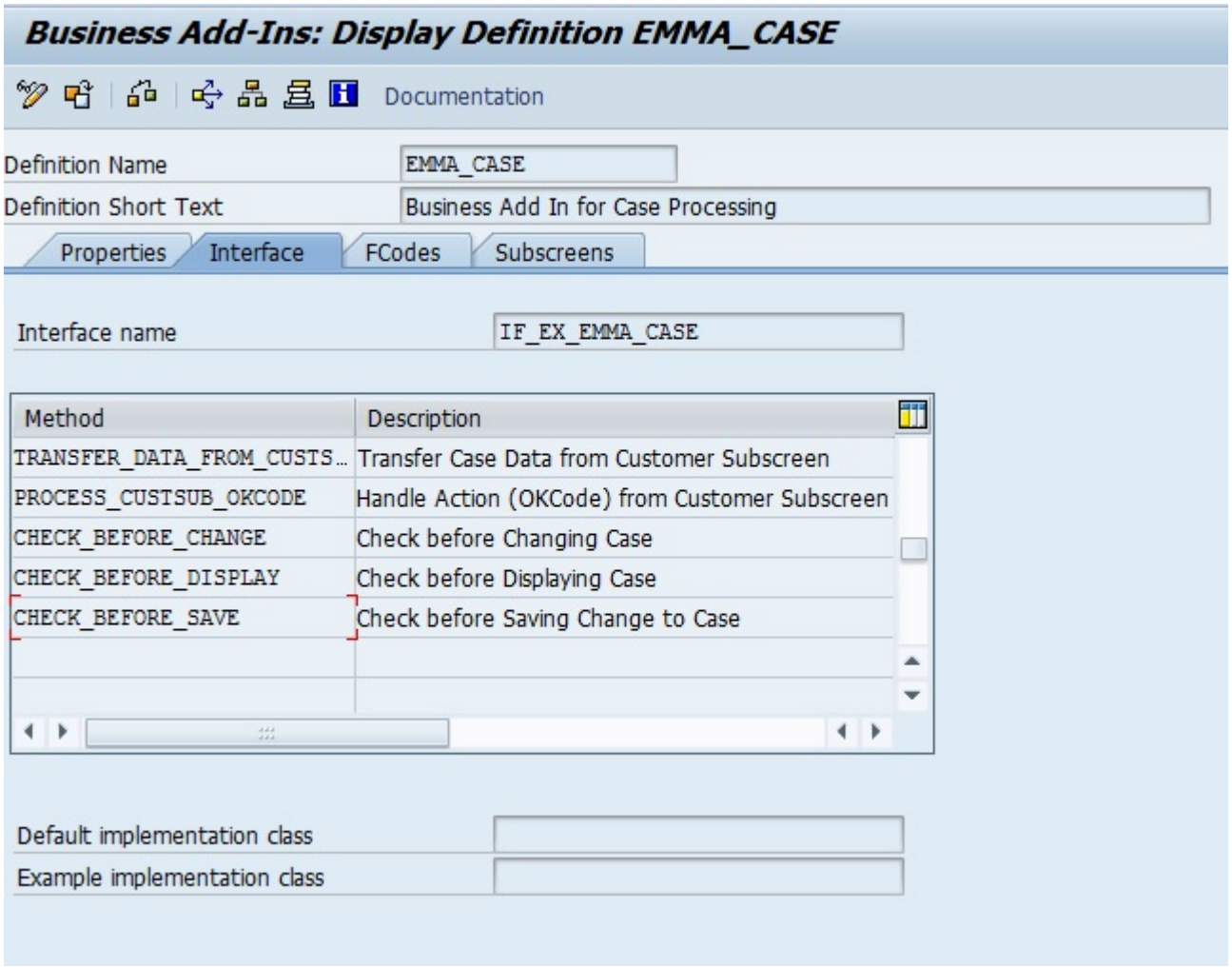
BPEM Substitute Actions BAdI

Implement this BAdI to allow for substitute custom Actions to be provided and used instead of the standard BDEx Actions for BPEMs.

An example implementation is provided with hints.

BPEM Closure Control Integration

In order to configure and implement the rules for the BPEM Closure Control Module, it is necessary to add a BDEx method call to the classic EMMA_CASE BAdI.



The BAdI needs to be implemented and the following call added to the CHECK_BEFORE_SAVE method.

Class Builder: Class ZCL_IM_EMMA_CASE_CLOSURE Display

Pattern Pretty Printer

Ty.	Parameter	Type spec.	Description
	IR_CASE	TYPE REF TO CL_EMMA_CASE	Clarification Case after Change
	IR_DBCASE	TYPE REF TO CL_EMMA_CASE	Clarification Case before Change
	ES_RETURN	TYPE BAPIRET2	Error Message
	CS_CUSTFIELDS	TYPE EMMA_CCI	Customer Fields of Case

Method IF_EX_EMMA_CASE~CHECK_BEFORE_SAVE Active

```
METHOD if_ex_emma_case~check_before_save.

  IF sy-batch IS INITIAL.

*   Check closure condition for BPEM closure restrictions
    CALL METHOD /bti/mde_cl_closure_rt=>check_closure_cond
      EXPORTING
        ir_case       = ir_case
        ir_dbcase     = ir_dbcase
      IMPORTING
        es_return     = es_return
      CHANGING
        cs_custfields = cs_custfields.

  ENDIF.

ENDMETHOD.
```