How to use Adobe Table Processing in Tax and Revenue Management

Applies to:
Tax and Revenue Management. For more information, visit the Business Rules Management homepage.

Summary
Tax and Revenue Management (TRM) supports the use of interactive Adobe form in order to lodge tax registration and return data. BRF Plus is being used to calculate and validate form data. This document describes the use of tables in the Adobe Forms and the correspondent linkage in BRF+.

Author: Frank Godeby
Company: SAP Australia
Created on: 10 February 2011

Author Bio
Frank Godeby has worked at SAP since August 1994. He joined the IBU Public Sector and worked in Development, Business development and Field Support and started his current position as Solution Manager for Tax and Revenue Management in 2008.
# Table of Contents

Introduction .............................................................................................................................. 3
Creation of a New Form Scenario .............................................................................................. 4
Adding a Table in the Adobe Form .............................................................................................. 6
Setting up the ISR BADI Call back Method Linked to the Push Button ..................................... 8
Calling the ISR BADI out of Adobe in Order to add and Delete Tables lines ............................. 9
Implementing the ISR BADI ....................................................................................................... 11
Creation of the BRFplus Application and link to the Form Bundle ........................................... 18
Process the table data in BRFplus .............................................................................................. 20
Special TPOS Considerations ....................................................................................................... 22
  Creating a Subscreen ................................................................................................................ 23
  Modify the Form ....................................................................................................................... 23
  Implement the Add and Delete into the Subscreen .................................................................... 23
  Configure TPOS to call the subscreen .................................................................................... 24
Related Content ........................................................................................................................ 25
Copyright ...................................................................................................................................... 26
Introduction

The following example describes how to enable Adobe table processing using TRM form bundles.

The example is quite simple. A form has a table with four fields (A_FIRST/LAST_NAME and A_AMOUNT_1/2). The 2 amount fields will be totaled up alternatively in Adobe or with BRFPLUS. The document only describes the steps necessary to process the tables. The document is not a step by step description for the entire setup process. The reader of this document should be familiar with Form bundle processing, Interactive Adobe Forms, BRF Plus, BADIs and Tax Payer Online Services (TPOS).

This description is completely independent from the processing of table forms in DDIC tables, which is done outside of Adobe. Table form processing in DDIC tables is used if you get a large number of line items in a tax return, which are usually loaded through files, which are visible separately from the Adobe form in the Tax Officer Work centre (TOW).

Tables within Adobe on the other side address reoccurring data within an individual tax form. Usually it is not clear how many rows in a table are needed upfront. Example: In a land transfer stamp duty form the taxpayer needs to specify all buying or selling parties. Usually it is only one buyer/seller, but there might be up to a hundred in exceptional cases.

Without proper Adobe table processing the form designer has to limit the number of instances and create many fields (BUYER_1, BUYER_2, BUYER_3, …), which inflates the number of rules within the rules engine and the size of the Adobe Form interface and context.
Creation of a New Form Scenario

First of all you need to create a form scenario. 8 fields have been defined; the first 4 fields are used in the table. For simplicity reasons only the 'A' (as entered) phase is being used in this example:

### Display View "Form view Fields": Overview

<table>
<thead>
<tr>
<th>Form View Fields</th>
<th>Char. Name</th>
<th>Char. Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A_LAST_NAME</td>
<td>Dictionary Type</td>
<td>▼ CHAR36</td>
</tr>
<tr>
<td>2</td>
<td>A_FIRST_NAME</td>
<td>Dictionary Type</td>
<td>▼ CHAR36</td>
</tr>
<tr>
<td>3</td>
<td>A_AMOUNT_1</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
<tr>
<td>11</td>
<td>A_AMOUNT_2</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
<tr>
<td>20</td>
<td>A_SUMMARY1_ADOBE</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
<tr>
<td>21</td>
<td>A_SUMMARY2_ADOBE</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
<tr>
<td>30</td>
<td>A_SUMMARY1_BRF</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
<tr>
<td>31</td>
<td>A_SUMMARY2_BRF</td>
<td>Dictionary Type</td>
<td>▼ AMOUNT_F6</td>
</tr>
</tbody>
</table>

Now you mark the first 4 fields as table fields. You do this by going into the ‘TRM Field Attributes’ section of the view. There you specify a table name (here TABLE1) for BRFPLUS:

### Display View "TRM Fields Attributes": Details

- **Costng Model**: ISR_ZML1
- **View**: A
- **Char. Name**:

You need to specify this for each field, which should be part of the table. You can invent any name. It is not a DDIC object name. The name here is used for 2 reasons: BRFPLUS will generate table expressions later and it is used in coding to distinguish between table and none table form fields.
It is possible to have multiple tables on a form; in this case the fields within each table should point to a different table name.

**Background information:**
Every field that is defined in the scenario is internally defined in the form interface as a table field. But without the instructions in this paper only the first item of each fields is being used. You can see this in the context interface, which is being generated by the scenario.

Once the scenario definition is done, you can generate your Adobe form and the interface definition via the form generation button in the scenario.
**Adding a Table in the Adobe Form**

Add a table to your form and drag and drop the fields into the table columns:

Check that the cells of the fields are bound to the table. The [*] is the indicator, that any value of the context table in the field can be displayed in the column.

Adobe tables can be static (e.g. always 3 rows) or dynamic (growing in between a min and max boundary). To create a dynamic table you need to have the “repeat row for each data item” on. In the example below the table always needs to have one row, but is not limited to a maximum. Please be aware that you need to select the rows of the table in order to alter these settings.
When placing new fields such as the summary fields below the table, Adobe might leave the table creation mode as indicator. When you drag and drop a field from the field catalogue into your form, Adobe might automatically create a header row. You can deactivate this by clicking the right mouse button in the data view. Select Data View Options and deselect “Allow Tables To Be Generated”.

![Data View Options dialog box](image)
Setting up the ISR BADI Call back Method Linked to the Push Button

Every scenario points to a BADI implementation, which you cannot see in the normal TRM scenario transaction. All scenarios, which are created with Application indicator ‘T’, automatically use the Badi from scenario ‘STRM’. You need to go into transaction ‘QISRSCENARIO’ and change the application indicator from ‘T’ to ‘S’.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZJIS</td>
<td>Tax on Sales/Purchases</td>
<td>T</td>
</tr>
<tr>
<td>ZMIT</td>
<td>Testing for MIT1</td>
<td>T</td>
</tr>
<tr>
<td>ZML1</td>
<td>Multi-line Test (Godeby)</td>
<td>S</td>
</tr>
<tr>
<td>ZOJR</td>
<td>ZOJR</td>
<td>T</td>
</tr>
</tbody>
</table>

If you go into the version of your scenario you will see the entry ‘STRM’ in the BADI. Overwrite it with your scenario name. This value will be used later on as filter in the BADI implementation.

Be aware that you do not see the scenario in the TRM transaction anymore once you have changed the scenario application indicator from ‘T’ to ‘S’. So you should change the value back to ‘T’ after you have stored the scenario with the new BADI entry.
Calling the ISR BADI out of Adobe in Order to add and Delete Tables lines

It is possible to add lines in Adobe via the direct JavaScript command, but in this case the ISR context is not updated automatically. This is why you need to update the ISR context, which then updates the size of the table automatically. You need to update the context in the BADI implementation of the Badi definition ‘QISR1’. It will be called by firing an event once the Add or Delete buttons are clicked. So add the Java Script shown below to the “click” event of the button. You are basically calling the BADI method with the value “ADD”.

Suggestion: If you have multiple tables on a form, you should concatenate the table name with add e.g. “ADD_TABLE1”. So the BADI can then distinguish which context table needs to be altered.

Below is the coding suggestion for the add button:

```
Window.xfa.record.CONTROL_PARAM.ISR_EVENT.value = "ADD";
ContainerFoundation.Info.SendMessageToContainer(event.target, "submit", "", "", "", "");
```

And for the remove button:

```
Window.xfa.record.CONTROL_PARAM.ISR_EVENT.value = "DEL";
ContainerFoundation.Info.SendMessageToContainer(event.target, "submit", "", "", "", "");
```
Obviously the remove button is a bit simplified since it just deletes the last line. In order to delete a specific line the user should be able to select which row to delete. This could be done by placing the remove button into the table cell.

The adobe summary fields are updated via the FormCalc “sum” command pointing to the field name of the table cell. The adobe summary fields will be updated immediately when entering the values in the table and leaving the field e.g. pressing the tab key.

The BRF summary fields will only be calculated when the user presses the “Check” button outside the adobe form.

```
data: Form.FIELD[1]::calculate = (FormCalc, client) -------------------------
| 1 * sum | @pacingTable1.Row[*].Value2 |
```

Please be aware that the size of the table can significantly influence the behavior of the page format. Without maximizing the number of table entries, flowing pages and page breaks need to be considered in order to avoid “writing beyond page boundaries”.
Implementing the ISR BADI

Implement the BADI of the definition QISR1. The main method for adding or deleting lines in the table context is scenario_process_user command.

The filter of the BADI (here ZML1) needs to be identical to the BADI entry in the scenario. Use the name of your scenario.

Here is the suggested coding for the method scenario_process_user_command. The user commands ‘TRM_CHANGE’ and ‘TRM_DISPLAY’ are automatically fired by the form bundle framework. The coding for these commands is not related to the table processing. The coding uses the added private methods ‘SET_FOCUS_FIELD’ and ‘SET_UI_ATTRIBUTES_FOR_VIEW’, which are also independent of the table processing. The commands ‘ADD’ and ‘DEL’ are the commands fired from the form buttons. The coding only adds/deletes a line for the fields which are linked to BRFPLUS table names in the scenario set-up. For simplicity reason no buffer has been introduced for the read of the customizing table.
This is an example of a code sample:

```abap
method IF_EX_QISR1~SCENARIO_PROCESS_USER_COMMAND.

* translation table
* removed: id_mode, id_init, ct_new_longtext
* es_return -> return
* cd_user_command -> user_command
* cs_general_data -> general_data
* ct_special_data -> special_data
* ct_external_data -> external_data
* ct_message_list -> message_list
* ct_ui_attributes -> ui_attributes

DATA: ls_external_data TYPE qisrsspecial_param,
     lv_phase TYPE phase_ps,
     lv_scenario TYPE qscenario,
     lv_version TYPE qisrdversion,
     ls_focus_field TYPE dfmca_brf_msg_wd,
     ld_scenario TYPE CK_COSTMODEL,
     ls_TFMCA_FORMFIELDS TYPE TFMCA_FORMFIELDS.

DATA: ls_special_data TYPE qisrsspecial_param,
     ls_special_top_data TYPE qisrsspecial_param,
     lt_special_data TYPE QISRTSPECIAL_PARAM,
     lv_counter TYPE i,
     ld_isr_object_name TYPE qisrdobjectname.

**----------------------------------------------------------------------
** The BADI mainly handles the following jobs
** 1. Control the edit/display mode of adobe interactive form
** 2. Control the form layout rendering, conditions are authorization and current phase
** 3. Locate the field to be focused, the field will be highlighted during form rendering
**----------------------------------------------------------------------

* retrieve the selected form scenario from general data
  lv_scenario = general_data-header-scenario.
* retrieve the selected form scenario version from general data
  lv_version = general_data-header-version.

CASE user_command.
  WHEN 'TRM_DISPLAY'.
    IMPORT trm_form_phase TO lv_phase FROM MEMORY ID 'TRM_FORM_UI_FORM_PHASE'.
    IF lv_phase IS NOT INITIAL AND lv_scenario IS NOT INITIAL.
      CALL METHOD me->set_ui_attributes_for_view
        EXPORTING
          i_actvt = '03'
          i_view = lv_phase
          i_scenario = lv_scenario
          i_version = lv_version
      CHANGING
        ct_ui_attributes = ui_attributes
        ct_special_data = special_data.
  ENDIF.
```

How to use Adobe Table Processing in Tax and Revenue Management

SAP COMMUNITY NETWORK | SDN - sdn.sap.com | BPX - bpx.sap.com | BOC - boc.sap.com | UAC - uac.sap.com
© 2011 SAP AG
IMPORT trm_form_focus_field TO ls_focus_field FROM MEMORY ID 'TRM_FORM_UI_FOCUS_FIELD'.

IF ls_focus_field-field_name IS NOT INITIAL.
CALL METHOD me->set_focus_field
EXPORTING
  i_focus_field   = ls_focus_field-field_name
CHANGING
  ct_ui_attributes = ui_attributes.
ENDIF.

WHEN 'TRM_CHANGE'.
IMPORT trm_form_phase TO lv_phase FROM MEMORY ID 'TRM_FORM_UI_FORM_PHASE'.
IF lv_phase IS NOT INITIAL AND lv_scenario IS NOT INITIAL.
CALL METHOD me->set_ui_attributes_for_view
EXPORTING
  i_actvt = '02'
  i_view = lv_phase
  i_scenario = lv_scenario
  i_version = lv_version
CHANGING
  ct_ui_attributes = ui_attributes
  ct_special_data = special_data.
ENDIF.

IMPORT trm_form_focus_field TO ls_focus_field FROM MEMORY ID 'TRM_FORM_UI_FOCUS_FIELD'.

IF ls_focus_field-field_name IS NOT INITIAL.
CALL METHOD me->set_focus_field
EXPORTING
  i_focus_field   = ls_focus_field-field_name
CHANGING
  ct_ui_attributes = ui_attributes.
ENDIF.

WHEN 'ADD'.

ld_scenario   = 'ISR_'.
ld_scenario+4 = flt_val.

* Get consistent counter

*----- Loop over table fields
  select * from TFMCA_FORMFIELDS into ls_TFMCA_FORMFIELDS
  where model = ld_scenario
  and table_group <> space.
  lv_counter = 0.

*----- Get max counter
  LOOP AT special_data INTO ls_special_data
  WHERE fieldname = ls_TFMCA_FORMFIELDS-field.
  IF ls_special_data-fieldindex > lv_counter.
    lv_counter = ls_special_data-fieldindex.
  ENDIF.
ENDLOOP.

clear ls_special_data-fieldvalue.
ls_special_data-fieldindex = lv_counter + 1.
APPEND ls_special_data TO lt_special_data.
endselect.

APPEND lines of lt_special_data TO special_data.

SORT special_data BY fieldname fieldindex.

ld_isr_object_name = general_data-header-object_name.
EXPORT trm_form_buffer FROM special_data TO MEMORY ID ld_isr_object_name.

WHEN 'DEL'.

ld_scenario = 'ISR_'.
ld_scenario+4 = flt_val.

select single * from TFMCA_FORMFIELDS into ls_TFMCA_FORMFIELDS
   where model = ld_scenario
   and table_group <> space.

if sy-subrc = 0.
   LOOP AT special_data INTO ls_special_top_data
      WHERE fieldname = ls_TFMCA_FORMFIELDS-field.
      
      IF ls_special_top_data-fieldindex > lv_counter.
         lv_counter = ls_special_top_data-fieldindex.
      ENDIF.
   ENDDO.
ENDLOOP.

DELETE special_data WHERE fieldindex = lv_counter.

ld_isr_object_name = general_data-header-object_name.
EXPORT trm_form_buffer FROM special_data TO MEMORY ID ld_isr_object_name.
endif.
ENDCASE.
endmethod.

The following coding needs to be added to the method scenario_set_additional_values:

method IF_EX_QISR1~SCENARIO_SET_ADDITIONAL_VALUES.

data: ld_isr_object_name type qisrdobjectname,
   lt_buffer type qisrtspecial_param,
   ls_buffer type qisrsspecial_param. ld_isr_object_name = general
   _data-header-object_name.

import trm_form_buffer to lt_buffer from memory id ld_isr_object_name.
if sy-subrc = 0.
   special_data = lt_buffer.
endif.
endmethod.
The signature for the method SET_UI_ATTRIBUTES_FOR_VIEW:

```
DATA: ls_ui_attributes TYPE qisrsui_attributes,
     ls_formfield  TYPE qisrsspecial_param,
     lv_fieldmod   TYPE qisrsui_attribute,
     lv_prefix     TYPE char2,
     lv_view       TYPE phase_ps,
     lt_scen_views TYPE TABLE OF tfmca_scen_view,
     lv_scen_view  TYPE tfmca_scen_view,
     lv_max_pos    TYPE numc2.
```

And the corresponding coding:

```
method SET_UI_ATTRIBUTES_FOR_VIEW.

   DATA: ls_ui_attributes TYPE qisrsui_attributes,
         ls_formfield  TYPE qisrsspecial_param,
         lv_fieldmod   TYPE qisrsui_attribute,
         lv_prefix     TYPE char2,
         lv_view       TYPE phase_ps,
         lt_scen_views TYPE TABLE OF tfmca_scen_view,
         lv_scen_view  TYPE tfmca_scen_view,
         lv_max_pos    TYPE numc2.
   *
   * retrieve all existing phases for this scenario version
   CALL FUNCTION 'FMCA_TFMCA_SCEN_VIEW_READ'
       EXPORTING
       i_scenario       = i_scenario
       i_version        = i_version
   IMPORTING
       e_tfmca_scen_view = lt_scen_views
   EXCEPTIONS
       not_found = 1
       invalid_input = 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.
   *
   * get sequence number of given view
   READ TABLE lt_scen_views WITH KEY scenario = i_scenario
       version = i_version
       form_view = i_view INTO lv_scen_view.
   IF sy-subrc = 0.
       lv_max_pos = lv_scen_view-view_seq_num.
   ENDIF.
   *
   * set the ui attributes for all views preceding or equal to the selected one
   LOOP AT lt_scen_views INTO lv_scen_view WHERE view_seq_num LE lv_max_pos.
      *
      * check for authorization
      CALL FUNCTION 'FMCA_VIEW_AUTHORITY_CHECK'
```
EXPORTING
   i_view = lv_scen_view-form_view
   i_actvt = i_actvt
EXCEPTIONS
   no_auth = 1
   OTHERS = 2.

IF sy-subrc = 0.
  * user is authorized to operate with i_actvt on fields of phase i_view
  * in case of change mode, user can input the fields, otherwise the fields are read-only
    IF i_actvt = '02'.
      lv_fieldmod = 'I'.
    ELSE.
      lv_fieldmod = 'R'.
    ENDIF.
  ELSEIF sy-subrc = 1 AND i_actvt = '02'.
    * in case of change mode user could still be authorized to see the fields
      CALL FUNCTION 'FMCA_VIEW_AUTHORITY_CHECK'
        EXPORTING
           i_view = lv_scen_view-form_view
           i_actvt = '03'
        EXCEPTIONS
           no_auth = 1
           OTHERS = 2.
      IF sy-subrc = 0.
        * user is authorized to see the fields
          lv_fieldmod = 'R'.
        ELSE.
          * user is not authorized to even see the fields
            lv_fieldmod = 'H'.
        ENDIF.
      ELSE.
        * user is not authorized to even see the fields
          lv_fieldmod = 'H'.
      ENDIF.

    * set the fieldmodifications according to authorization
      CONCATENATE lv_scen_view-form_view '_' INTO lv_prefix.

      LOOP AT ct_special_data INTO ls_formfield WHERE fieldname(2) = lv_prefix.

        ls_ui_attributes-fieldname = ls_formfield-fieldname.
        ls_ui_attributes-ui_attribute = lv_fieldmod.

      READ TABLE ct_ui_attributes
        WITH KEY fieldname = ls_formfield-fieldname TRANSPORTING NO FIELDS.

      IF sy-subrc = 0.
        MODIFY ct_ui_attributes FROM ls_ui_attributes INDEX sy-index.
      ELSE.
        APPEND ls_ui_attributes TO ct_ui_attributes.
      ENDIF.
      ENDLOOP.
    ENDLOOP.
* hide all other fields
  LOOP AT lt_scen_views INTO lv_scen_view
    WHERE view_seq_num GT lv_max_pos.

* user is not authorized to even see the fields
  lv_fieldmod = 'H'.

* set the fieldmodifications accordingly
  CONCATENATE lv_scen_view-form_view '_' INTO lv_prefix.
  LOOP AT ct_special_data INTO ls_formfield WHERE fieldname(2) = lv_prefix.
    ls_ui_attributes-fieldname = ls_formfield-fieldname.
    ls_ui_attributes-ui_attribute = lv_fieldmod.
  READ TABLE ct_ui_attributes
    WITH KEY fieldname = ls_formfield-fieldname TRANSPORTING NO FIELDS.
  IF sy-subrc = 0.
    MODIFY ct_ui_attributes FROM ls_ui_attributes INDEX sy-index.
  ELSE.
    APPEND ls_ui_attributes TO ct_ui_attributes.
  ENDIF.
  ENDDO LOOP.
endmethod.

The signature for the method SET FOCUS FIELD:

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Type spec.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>lv_focus</td>
<td>TYPE qisrsui_attributes</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>CT_UI_ATTRIBUTES</td>
<td>TYPE qisrtui_attributes</td>
<td>Internal Service Request: UI Attributes for a Field</td>
</tr>
</tbody>
</table>

method SET FOCUS FIELD.

DATA lv_focus TYPE qisrsui_attributes.

FIELD-SYMBOLS: <fs_fieldname> TYPE qisrdfieldname,
               <fs_label> TYPE qisrdfieldtxt.

* to pass the focus information to the form we use a line of table ct_ui_attributes
* fieldname = 'TRM_FOCUS_FIELD' and label = (name of field to set the focus to)
  READ TABLE ct_ui_attributes WITH KEY fieldname = 'TRM_FOCUS_FIELD' INTO lv_focus.
  IF sy-subrc <> 0.

* no focus has been set yet, add line
  lv_focus-fieldname = 'TRM_FOCUS_FIELD'.
  ASSIGN i_focus_field TO <fs_fieldname>.
  ASSIGN lv_focus-label TO <fs_label>.
  <fs_label> = <fs_fieldname>.
  APPEND lv_focus TO ct_ui_attributes.
ELSE.

* edit existing line
  ASSIGN i_focus_field TO <fs_fieldname>.
  ASSIGN lv_focus-label TO <fs_label>.
  <fs_label> = <fs_fieldname>.
  MODIFY ct_ui_attributes FROM lv_focus INDEX sy-tabix.

ENDIF.

e ndmethod.

At this stage you can check your form bundle. It should not add and delete lines in the table. The values in the table should be added up in the adobe summarization field. The values of the form should be saved once you save the form bundle. In the last stage we link a BRF Plus application to the form bundle and add up the amount fields via the rules engine.

Creation of the BRFplus Application and link to the Form Bundle

In the next steps BRF Plus is used to add up fields of the table and write the result into the summarization fields of the form.

Create a BRF Plus application via the TRM configuration (“Define BRFplus Objects for Form Bundles”).

**BRFplus Integration Manager for Form Bundles**

<table>
<thead>
<tr>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Object</td>
</tr>
<tr>
<td>Expression</td>
</tr>
<tr>
<td>Function</td>
</tr>
</tbody>
</table>

Select the “Create from scenario” option. The screen should allow you to select a table entry with the name of the specified table name in the scenario set up. You see the name ‘TABLE1’ twice since BRFPLUS always generates a structure within the table.
Create BRFplus function as entry point from the form processing:

**Display Functions**

- BRFplus Function Name: FUNC_CALCULATE
  - SYS_CONTROL
  - SYS_FORM.Bundle_HEADER
  - SYS_MESSAGE
  - ZML1

Assign the BRFplus application and BRFplus function to the form bundle type which uses the scenario created in earlier steps:

**Change View "Define Processing Rule": Details**

- Form Bundle Type: ZMT1
- Rule ID: 1
- Valid from: 01.01.2000
- Valid to: 31.12.2020
- Use BRFplus
- BRFplus Application: ZML1
- BRFplus Function: FUNC_CALCULATE
- Activate Trace
Process the table data in BRFplus

At this stage the data object of your form are visible in your application in the BRFplus workbench:

Add a "rule set" to your function:

Function FUNC_CALCULATE

Rulesets

Name: RULESET_CALCULATE
Priority: 00
Text: Total up Values
Create a rule in your rule set:

Loop at your table and add the values together:

Loop LOOP_AND_TOTAL

For each entry in table ZML1-TABLE1 with line type TABLE1

Do following operations...

(1) Perform following operations
   (1) Change ZML1-Summary_amount1 (BRF) after processing expression ADD_VALUES
The expression add_values has been created via the formula editor:

```
<table>
<thead>
<tr>
<th>Name:</th>
<th>ADD_VALUES</th>
<th>Short Text:</th>
<th>Access Level:</th>
<th>Application</th>
</tr>
</thead>
</table>
```

**General**

Result Data Object: Summary amount1 (BRF) = ZML1-Summary amount1 (BRF) + TABLE1-Amount1 in Table

It might be a good idea to clear the result data field before starting the loop in order to stay re-entrant.

At this stage you should be able to test the example in the Tax Officer Work Centre. Further adjustments might be necessary in Taxpayer Online Services.

**Special TPOS Considerations**

At the time of this test, the following adjustments needed to be in place in order for the form containing the tables working in a TPOS environment:

The binding of the individual table data cells need to be $record.A_AMOUNT_2[*].FIELD[*]:

In addition a hidden field was added to the form which contains the ISR_EVENT field of the CONTROL_PARAM structure. Without this field the system communicates this data element only from server side to client side, but it does not do it from client side to server side.

A subscreen must be created and assigned to the main screen, which holds the Interactive Adobe Form. This is needed in order to handle the control events coming from the PDF.
Creating a Subscreen

Create a new WD component and add the TPOS SubScreenInterface to the Implemented Interfaces. Let the system generate the necessary objects. Delete the default text from the view.

Note: before creating the subscreen dependency is needed to the is/cmp/etax/ui/wdif Navigation public part in the ISCETAXUI SCA.

For more details see: **TPOS Cook Book for implementing customer screens and subscreens**

Modify the Form

There is one necessary modification on the form to be able to handle the control parameters coming from the form. A new field should be added to the form with no visibility. The CONTROL_PARAM.ISR_EVENT should be bound to this field. This is needed for the data communication.

Implement the Add and Delete into the Subscreen

1. Define the table fields and variables into the other section of the controller.

```java
//@@begin others
private String scenario; //form scenario
private String version; //form version
private String[] tableFields={"A_LAST_NAME",
    "A_FIRST_NAME",
    "A_AMOUNT_1",
    "A_AMOUNT_2"}; //fields in the table
//@@end
```

2. Get the scenario and scenario version for the form (you can use the Session comp approach to determinate it)

Implement initScreen like this:

```java
public void initScreen( com.sap.tc.webdynpro.progmodel.api.IWDComponentUsage
ejbCompUsage, com.sap.tc.webdynpro.progmodel.api.IWDComponentUsage sessionCompUsage,
java.lang.String viewVariant, com.sap.tc.webdynpro.progmodel.api.IWDNode context ) {
    //@@begin initScreen()
    IWDNode childNode = context.getChildNode("FormHandling", 0);
    scenario = childNode.getElementAt(0).getAttributeAsText("pdfScenario");
    version = childNode.getElementAt(0).getAttributeAsText("pdfVersion");
    //@@end
```

3. Implement the beforeAction method of the controller like this for remove and add lines:

```java
public void beforeAction( java.lang.String actionName, com.sap.tc.webdynpro.progmodel.api.IWDNode node ) throws
com.sap.is.cmp.etax.exceptions.NavigationSuspendException {
    //@@begin beforeAction()
    //Check that the action is coming from the IAF
    if(actionName.equals("IAFSUBMIT")){
        //check scenario and scenario version
        if("ZML1".equalsIgnoreCase(scenario) &&
"00000".equalsIgnoreCase(version)){
            //Get the event parameter
            String event =
            node.getElementAt(0).node("CONTROL_PARAM").getElementAt(0).getAttributeAsText("ISR_EVENT");
            if(event.equals("ADD")){
```
Configure TPOS to call the subscreen

Use the Subscreen configuration and add you subscreen to the main screen

- com.sap.is.cmp.etax.ui.wd.registration.formbasedopencomp.FormBasedOpenComp - in case it is a registration form
- com.sap.is.cmp.etax.ui.wd.filing.filingopencomp.FilingOpenComp - in case it is a return form
Related Content

SAP Interactive Forms by Adobe
Handling Dynamic Length Tables in Adobe Forms
BRF plus
Tax and Revenue Management

For more information, visit the Business Rules Management homepage
Copyright

© Copyright 2011 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP Business ByDesign, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects S.A. in the United States and in other countries. Business Objects is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies (“SAP Group”) for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.