Calculating Property Tax with BRF+

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

Summary
With Enhancement Package 5 Tax and Revenue Management has integrated BRF+ into form based returns and registration processing. Property Tax (using RE-FX Land Use Management) is still only integrated with BRF. This cookbook describes all the necessary steps to use BRFplus for Property Tax Calculation instead of using BRF. The advantages of using BRF+ over BRF are mainly:

- BRF+ has more features than BRF and will be developed
- Consistent Rules Basis between Property and Form-based Taxes
- BRF+ is significantly faster than BRF.

Authors: Frank Godeby, Albrecht Weiss
Company: SAP Australia, SAP AG
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.

Albrecht Weiss has worked for SAP since 1998. He joined the IBU Public Sector in 2004 and started his position as Solution Manager for Tax and Revenue Management in 2009.
## Table of Contents

Introduction .................................................................................................................................................. 3

Differences between the BRF and BRF+ Integration .................................................................................. 3

One-Time Implementation Steps ............................................................................................................... 4
  1) Structure Types .................................................................................................................................... 4
  2) Table Types ......................................................................................................................................... 4
  3) Tables .................................................................................................................................................. 5
  4) Classes (see below for method coding) ............................................................................................... 6
  5) Message class ...................................................................................................................................... 8
  6) Program (see below for coding) ......................................................................................................... 8
  7) BadI ..................................................................................................................................................... 9
  8) BRFplus .............................................................................................................................................. 10

Recurring Customizing steps .................................................................................................................... 11

Notes regarding the creation of BRFplus content .................................................................................. 11

Appendix: Coding for the Class Methods ................................................................................................. 13

Related Content ......................................................................................................................................... 36

Copyright .................................................................................................................................................... 37
Introduction

The solution described in this paper basically replaces the major call into BRF with a call into BRF+. It does not replace the BRF configuration content e.g. tax rates. Instead it assumes complete set up of the calculation in BRFplus. Once you have implemented the BRFplus integration, you will not use the property tax configuration for tax calculation e.g. tax rates anymore.

The coding provided in this paper is for exemplarity purposes only.

Differences between the BRF and BRF+ Integration

The major difference between the BRF and BRFplus property tax integration is the amount of calls. While the BRFplus application/function is called once per contract, BRF is called for each parcel, time slice and condition. This requires the setup of rules in BRFplus which loop across the tables IT_PARCEL, IT_COND and IT_YEAR if the same results are to be achieved. However, as this is modeled in BRFplus, the sequence of which loop is done first, second or third is flexible. In fact you might not even need to loop across the IT_COND table at all in order to improve performance, since the IT_YEAR table holds the condition type name as well.

The BRFplus integration does not consider 2 customizing views (V_PT_TAX_COND, V_PT_TAX_RATE) anymore. In case tax rate tables are required, decision tables created in BRFplus are to be used.

Potential conditions are to be linked to a condition group which again is linked to a contract type.

While BRF calculates all values of a contract from start date to the calculation end date, the new integration allows to define a recalculation start date. From that date onwards, time slices are to be calculated. Previous time slices before the recalculation start date are read from the contract and are not changed anymore.

BRF accumulated conditions results (IT_CALC_VALUE) into one line if possible. The BRFplus integration provides a condition calc value for each time slice. No aggregation takes place.
One-Time Implementation Steps

Create a package e.g. with the name ZTRM_PT. All objects can be included into this package. Create the following objects:

1) Structure Types

<table>
<thead>
<tr>
<th>Structure Type</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZFMCA_PT_YEAR</td>
<td>ZFMCA_PT_YEAR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>Typing Method</th>
<th>Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDTYPE</td>
<td>Types</td>
<td>RECCONDTYPE</td>
</tr>
<tr>
<td>CALCYEAR</td>
<td>Types</td>
<td>CALCYEAR_PS</td>
</tr>
<tr>
<td>VALID FROM</td>
<td>Types</td>
<td>PS_VALID_FROM_DA</td>
</tr>
<tr>
<td>VALID TO</td>
<td>Types</td>
<td>PS_VALID_TO_DATE</td>
</tr>
</tbody>
</table>

ZFMCA_PT_CALC_VALUE

<table>
<thead>
<tr>
<th>Structure Type</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZFMCA_PT_CALC_VALUE</td>
<td>Calculated Values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>Typing Method</th>
<th>Component Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONDTYPE</td>
<td>Types</td>
<td>RECCONDTYPE</td>
</tr>
<tr>
<td>PARCEL</td>
<td>Types</td>
<td>RECAINTREN0</td>
</tr>
<tr>
<td>VALID FROM</td>
<td>Types</td>
<td>PS_VALID_FROM_DA</td>
</tr>
<tr>
<td>VALID TO</td>
<td>Types</td>
<td>PS_VALID_TO_DATE</td>
</tr>
<tr>
<td>CALCVALE</td>
<td>Types</td>
<td>CALCVALE_PS</td>
</tr>
</tbody>
</table>

2) Table Types

<table>
<thead>
<tr>
<th>Table Type</th>
<th>Line Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZFMCA_PT_T_YEAR</td>
<td>ZFMCA_PT_YEAR</td>
</tr>
<tr>
<td>ZFMCA_PT_T_CALC_VALUE</td>
<td>ZFMCA_PT_CALC_VALUE</td>
</tr>
</tbody>
</table>
3) Tables

ZTRM_PT_BRFPLUS

Delivery Class: C = Customizing
Display/Maintenance Allowed

Technical Settings:
Data Class: APPL0
Size Category: 0
Fully Buffered

Enter the application name and function name you want to use for a given contract type.

If you want that a trace is created set the trace to „X“.
If your calculation requires parcel information, set the READ_PARCEL to „X“.

If you want to avoid recalculations for periods which should no longer be calculated, set a recalculation start date (YYYYMMDD). From this start date onwards a recalculation will be done. E.g. if a contract starts in 2000 and you are in 2008, cash-flow range is set to two years, setting the date to 20080101 will only recalculate the values for the years 2008-2010.

ZTRM_PTLEANHIS

Delivery Class: A = Application
Display/Maintenance Allowed with Restrictions

Technical Settings:
Data Class: APPL0
Size Category: 1
No Buffering Allowed

This table stores all the created traces per contract number.
4) Classes (see below for method coding)

**ZCL_FMCA_PT_CALC** (superclass: CL_FMCA_PT_CALC)

Redefined Methods:
- CONDITION_VALUE
- CLEAR_ATTRIBUTES
- CALCULATES_VALUES

New Methods:
- CALCULATE_AMOUNT_BRFPLUS
- GET_RHYTHM_START_BRFPLUS
- SET_YEAR_VALUES_BRFPLUS

New Attributes:
- GV_BRFPLUS_EXECUTED
- GT_CALC_VALUE_BRFPLUS (not in FD3)
- GS_CONTRACT_BRFPLUS (not in FD3)

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Visibility</th>
<th>Method type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALCULATES_VALUES</td>
<td>Instance</td>
<td>MetPublic</td>
<td></td>
<td>Tax amount calculation</td>
</tr>
<tr>
<td>CLEAR_ATTRIBUTES</td>
<td>Instance</td>
<td>MetPublic</td>
<td></td>
<td>Clear global attributes</td>
</tr>
<tr>
<td>CONDITION_VALUE</td>
<td>Instance</td>
<td>MetPublic</td>
<td></td>
<td>Tax amount of a condition</td>
</tr>
<tr>
<td>CALCULATE_AMOUNT_BRFPLUS</td>
<td>Instance</td>
<td>MetPublic</td>
<td></td>
<td>CALCULATE_AMOUNT_BRFPLUS</td>
</tr>
<tr>
<td>SET_YEAR_VALUES_BRFPLUS</td>
<td>Instance</td>
<td>MetProtected</td>
<td></td>
<td>SET_YEAR_VALUES_BRFPLUS</td>
</tr>
<tr>
<td>GET_RHYTHM_START_BRFPLUS</td>
<td>Instance</td>
<td>MetProtected</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Method parameters for `CALCULATE_AMOUNT_BRFPLUS`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Pa_...</th>
<th>Typing M...</th>
<th>Associated Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_YEAR</td>
<td>Importin</td>
<td></td>
<td></td>
<td>ZFMC_F_T_T_YEAR</td>
</tr>
<tr>
<td>IT_COND</td>
<td>Importin</td>
<td></td>
<td></td>
<td>RE_T_CONDITION_REL_GROUP</td>
</tr>
<tr>
<td>IV_OBJNR</td>
<td>Importin</td>
<td></td>
<td></td>
<td>RECDOBJNRNCALC</td>
</tr>
<tr>
<td>IO_OBJECT</td>
<td>Importin</td>
<td></td>
<td></td>
<td>OBJECT</td>
</tr>
<tr>
<td>IO_MESSAGE_LIST</td>
<td>Importin</td>
<td></td>
<td></td>
<td>IF_RCMA_MESSAGE_LIST</td>
</tr>
<tr>
<td>IT_MAIN_PARTNER</td>
<td>Importin</td>
<td></td>
<td></td>
<td>BAPI_RE_T_PARTNER_INT</td>
</tr>
<tr>
<td>IT_OBJECT_REL</td>
<td>Importin</td>
<td></td>
<td></td>
<td>BAPI_RE_T_OBJECT_REL_INT</td>
</tr>
<tr>
<td>LS_CONTRACT</td>
<td>Importin</td>
<td></td>
<td></td>
<td>BAPI_RE_CONTRACT_INT</td>
</tr>
<tr>
<td>IT_COND_CALC_OLD</td>
<td>Importin</td>
<td></td>
<td></td>
<td>BAPI_RE_T_COND_CALC_INT</td>
</tr>
<tr>
<td>BRFPLUS_FUNCTION_ID</td>
<td>Importin</td>
<td></td>
<td></td>
<td>IF_FDT_TYPES=&gt;ID</td>
</tr>
<tr>
<td>IS_ZTRM_PT_BRFPLUS</td>
<td>Importin</td>
<td></td>
<td></td>
<td>ZTRM_PT_BRFPLUS</td>
</tr>
<tr>
<td>CT_CALC_VALUE</td>
<td>Changin</td>
<td></td>
<td></td>
<td>ZFMC_F_T_T_CALC_VALUE</td>
</tr>
<tr>
<td>CT_CALC_USED_OBJECTS</td>
<td>Changin</td>
<td></td>
<td></td>
<td>RE_T_OBJNR</td>
</tr>
</tbody>
</table>
Get_Rhythm_start_brfplus needs an exception parameter as well.

<table>
<thead>
<tr>
<th>Method</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_RHYTHM_START_BRFPLUS</td>
<td></td>
</tr>
</tbody>
</table>

### ZCL_FMCA_PT_CALC_RULE (superclass: CL_FMCA_PT_CALC_RULE)

Redefined Methods:
- IF_EX_RECD_CALC_RULE~GET_ATTRIBUTES
- IF_EX_RECD_CALC_RULE~GET_VALUES

### ZCL_FMCA_PT_RECN_CONTRACT (superclass: CL_FMCA_PT_RECN_CONTRACT)

Redefined Methods:
- IF_EX_RECN_CONTRACT~SUBSTITUTE
- UPDATE_CONDITIONS
5) Message class

Message class: ZTRM_PT

<table>
<thead>
<tr>
<th>Message</th>
<th>Message shortened</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Noibriplus function foun for contract type A1</td>
</tr>
<tr>
<td>001</td>
<td>Error during execution of BRFplus function (104 K1 )</td>
</tr>
<tr>
<td>002</td>
<td>Error during the storage of the BRFplus trace</td>
</tr>
<tr>
<td>003</td>
<td>Error during the storage of the trace reference</td>
</tr>
<tr>
<td>004</td>
<td>No unique briplus function id found</td>
</tr>
</tbody>
</table>

6) Program (see below for coding)

ZTRM_PT_DISPLAY_TRACE

This program allows displaying any created traces for a real estate contract. The Entry Parameter to start this program are Company Code and RE-FX Contract. The next screen shows a list of every persisted trace:

Double-click to show the result of the trace:
7) BADI

Create a new BADI implementation based on `ZCL_FMCA_PT_CALC_RULE`. Deactivate `CL_FMCA_PT_CALC_RULE`. The creation of the BADI should be done via IMG. It is a 2 step process. First use the create button to create a new BADI pointing to the class created before; and secondly the deactivation of the existing implementation.

(IMG -> Flexible Real Estate Management (RE-FX) -> Conditions and Flows -> Calculation and Distribution Formulas -> Implement Enhancements (BAdI) -> Calculation Formula for Conditions)

<table>
<thead>
<tr>
<th>Implementations for BAdI Definition BADI_RECN_CALC_RULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active(MG)</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
</tr>
</tbody>
</table>

This is how the implementation will look like in SE80:

Create a new BADI implementation based on `ZCL_FMCA_PT_RECN_CONTRACT`. Deactivate `CL_FMCA_PT_RECN_CONTRACT`. The creation of the BADI should be done via IMG. It is a 2 step process. First use the create button to create a new BADI pointing to the class created before; and secondly the deactivation of the existing implementation.

(IMG -> Flexible Real Estate Management (RE-FX) -> Contract -> Implement Enhancements (BAdI) -> Number Assignment, Validation, Substitution)
This is how the implementation will look like in SE80:

8) BRFplus

Upload BRFplus template application “TRM_PT_TEMPLATE” via xml-import in the BRFplus workbench. The template file should have been delivered with this cook book. As a result of the upload, you will find all the elements, structures and tables necessary for the interface between the calling programs and BRF+ in the BRF+ workbench. In order to upload the file, you need to switch to expert mode in the BRF+ workbench (Workbench->User Mode->Expert Mode). Then chose Tools->XML Import.

The BRFplus function is fed with the following data:

IS_CONTRACT (type: BAPI_RE_CONTRACT_INT) – contains attributes of the real estate contract
IT_DFACTS (type: DFACTS_T_TYPE) – contains attributes of facts which are linked to a parcel
IT_PARCEL (type: BAPI_RE_T_PARCEL_LAND_INT) – contains attributes on parcels which are linked to the contract
IT_COND (type: RE_T_CONDITION_REL_GROUP) – contains all conditions which are linked to a real estate contract type
IT_CONTRACT_PARTNER (type: BAPI_RE_T_PARTNER_INT) – contains contract partners of a real estate contract
IT_YEAR (type: ZFMCA_PT_T_YEAR) – contains timeslices for which a calculation should take place

The result is to be provided to the result table:

“IT_CALC_VALUE” (type: ZFMCA_PT_T_CALC_VALUE)
Recurring Customizing steps

- Copy Property Tax BRFplus application template
- Build rules in the copied BRFplus application which computes records which feed the result table “IT_CALC_VALUE”.
- Insert an entry in table “ZTRM_PT_BRFPLUS” E.g.

<table>
<thead>
<tr>
<th>MANOF</th>
<th>RECTYPE</th>
<th>APPLICATION</th>
<th>FUNCTION</th>
<th>TRACE</th>
<th>READ_PARCEL</th>
<th>RECALCSTARTDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>865</td>
<td>FS01</td>
<td>ZTRM_PT_TEST2</td>
<td>PT_CALCULATION</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

- Execute code_generation program in case it is not executed automatically after the first call (sa38 -> “FDT_GENERATION_TOOL”)

Notes regarding the creation of BRFplus content

After you have copied the application template into your application, you should have additional BRF+ content (elements, structures, tables) available in your application:

- XNAME
- XPARNT
- XPL
- BAPI_RE_PARCEL_LAND_INT
- BAPI_RE_PARTNER_INT
- CALCVLUE_STRUCTURE
- DFAGTS
- FMCA_PT_CALC_VALUE
- CONTRACT
- RECOD_CONDITION_REL_GROUP
- ZMCA_PT_CALC_VALUE
- ZMCA_PT_YEAR
- T_CALC_VALUE
- T_COND
- T_CONTRACT_PARTNER
- T_DFAGTS
- T_PARCEL
- T_YEAR
- PT_CALCULATION

- Ruleset
  - CALCULATE_PT
It is important to understand that all the conditions, time-slices and parcels are transferred into BRF+. So this usually results in a nested loop. For performance reasons it might be important to immediately check, which conditions apply, before moving into additional loops. The following screen shoot should only describe a possibility (closed to standard logic using BRF instead of BRF+).
Appendix: Coding for the Class Methods

Program: ZTRM_PT_DISPLAY_TRACE

REPORT ZTRM_PT_DISPLAY_TRACE.

TABLES: ztrm_pt_leanhis.

data: it_ztrm_pt_leanhis type table of ztrm_pt_leanhis.

TYPE-POOLS: SLIS.

SELECTION-SCREEN BEGIN OF BLOCK f01 WITH FRAME TITLE text-f01.
SELECT-OPTIONS:
  BUKRS for ztrm_pt_leanhis-BUKRS no intervals NO-EXTENSION OBLIGATORY,
  RECNNR for ztrm_pt_leanhis-RECNNR no intervals NO-EXTENSION OBLIGATORY.

SELECTION-SCREEN END OF BLOCK f01.

DATA: V_REPID LIKE SY-REPID.
DATA: I_FIELDCAT TYPE SLIS_T_FIELDCAT_ALV,
     WA_FIELDCAT TYPE SLIS_FIELDCAT_ALV.
DATA: V_EVENTS TYPE SLIS_T_EVENT,
     WA_EVENT TYPE SLIS_ALV_EVENT.

INITIALIZATION.

  V_REPID = SY-REPID.
  PERFORM BUILD_FIELDCATLOG.

  CALL FUNCTION 'REUSE_ALV_EVENTS_GET'
        EXPORTING
          I_LIST_TYPE = 0
        IMPORTING
          ET_EVENTS   = V_EVENTS.

  READ TABLE V_EVENTS INTO WA_EVENT WITH KEY NAME = 'USER_COMMAND'.
  IF SY-SUBRC EQ 0.
    WA_EVENT-FORM = 'USER_COMMAND'.
    MODIFY V_EVENTS FROM WA_EVENT TRANSPORTING FORM WHERE NAME = WA_EVENT-NAME.
  ENDF.

START-OF-SELECTION.

  select * from ztrm_pt_leanhis into table IT_ztrm_pt_leanhis
where BUKRS in BUKRS
and RECNNR in RECNNR.

sort IT_ztrm_pt_leanhis by timestamp.

PERFORM DISPLAY_ALV_REPORT.

*---------------------------------------------------------------------*
* & Form BUILD_FIELDCATLOG                                           *
*---------------------------------------------------------------------*
FORM BUILD_FIELDCATLOG.
WA_FIELDCAT-TABNAME = 'IT_ZTRM_PT_LEANHIS'.
WA_FIELDCAT-FIELDNAME = 'RECNNR'.
WA_FIELDCAT-SELTEXT_M = 'Contract ID'.
WA_FIELDCAT-outputlen = 20.
APPEND WA_FIELDCAT TO I_FIELDCAT.
CLEAR WA_FIELDCAT.

WA_FIELDCAT-TABNAME = 'IT_ZTRM_PT_LEANHIS'.
WA_FIELDCAT-FIELDNAME = 'BUKRS'.
WA_FIELDCAT-SELTEXT_M = 'Company Code'.
WA_FIELDCAT-outputlen = 20.
APPEND WA_FIELDCAT TO I_FIELDCAT.
CLEAR WA_FIELDCAT.

WA_FIELDCAT-TABNAME = 'IT_ZTRM_PT_LEANHIS'.
WA_FIELDCAT-FIELDNAME = 'TIMESTAMP'.
WA_FIELDCAT-SELTEXT_M = 'Date/Time'.
WA_FIELDCAT-outputlen = 20.
APPEND WA_FIELDCAT TO I_FIELDCAT.
CLEAR WA_FIELDCAT.

ENDFORM. "BUILD_FIELDCATLOG"

*---------------------------------------------------------------------*
* & Form display_alv_report                                          *
*---------------------------------------------------------------------*
FORM DISPLAY_ALV_REPORT.

CALL FUNCTION 'REUSE_ALV_GRID_DISPLAY'
EXPORTING
    I_CALLBACK_PROGRAM = V_REPID
    I_CALLBACK_USER_COMMAND = 'USER_COMMAND'
    IT_FIELDCAT = I_FIELDCAT[]
    IT_EVENTS = V_EVENTS
TABLES
    T_OUTTAB = IT_ztrm_pt_leanhis.

ENDFORM. "display_alv_report"

*---------------------------------------------------------------------*
* & Form USER_COMMAND                                                 *
*---------------------------------------------------------------------*
FORM USER_COMMAND USING R_UCOMM LIKE SY-UCOMM
RS_SELFIELD TYPE SLIS_SELFIELD.

DATA: lt_parameters TYPE tihttpnvp,
    ls_parameter TYPE ihttpnvp,
    wa_ztrm_pt_leanhis type ztrm_pt_leanhis.

CASE R_UCOMM.
    WHEN '&IC1'.
        READ TABLE it_ztrm_pt_leanhis INTO wa_ztrm_pt_leanhis INDEX RS_SELFIELD-
            TABINDEX.

        ls_parameter-name = 'IV_TRACE_ID'.
        move wa_ztrm_pt_leanhis-leantrace_id to ls_parameter-value.
        APPEND ls_parameter TO lt_parameters.

        ls_parameter-name = 'IV_FUNCTION_ID'.
        move wa_ztrm_pt_leanhis-function_id to ls_parameter-value.
        APPEND ls_parameter TO lt_parameters.

        ls_parameter-name = 'IV_TRACE_TABLE'.
        ls_parameter-value = 'FDT_TRACE_0100'.
        APPEND ls_parameter TO lt_parameters.
        CALL FUNCTION 'WDY_EXECUTE_IN_PLACE' 
            EXPORTING
                internalmode = 'X'
                application = 'FMCA_WD_LEAN_TRACE'
                parameters = lt_parameters.

    END_CASE.

ENDFORM.

Class: ZCL_FMCA_PT_RECN_CONTRACT
Method: IF_EX_RECN_CONTRACT~SUBSTITUTE

method IF_EX_RECN_CONTRACT~SUBSTITUTE.

    DATA: lt_tax_cond TYPE fmca_pt_t_tax_cond.

    * No property tax caluclation, if condition update is disabled
      IF ch_recn_contract=>is_condition_update_disabled( ) = 'X'.
          RETURN.
      ENDIF.

    * No property tax caluclation for the following transaction during saving a contract
      IF id_event_type = 'S'
          AND ( sy-tcode = 'RECARG'  " update contracts by worklist
                  OR sy-tcode = 'RECNCHECK'  " check in change mode/save maybe
                  OR sy-tcode = 'RECDCGOL'  " cashflow update
                  OR sy-tcode = 'RERAPP'  " periodic postings contracts
                  OR sy-batch = abap_true).
          RETURN.
      ENDIF.
IF  id_activity NE '01'
   OR  id_activity EQ '01'
   AND  id_event_type = 'S'.

   CALL METHOD update_conditions
      EXPORTING
         io_object = io_object.
   ENDIF.
endmethod.

Method: UPDATE_CONDITIONS

method UPDATE_CONDITIONS.

   DATA:
      lr_fmca_pt_calc TYPE REF TO zcl_fmca_pt_calc.

   CREATE OBJECT lr_fmca_pt_calc.

   *clear global attributes
   CALL METHOD lr_fmca_pt_calc->clear_attributes.

   *build condition list and update condition list of contract
   CALL METHOD lr_fmca_pt_calc->add_condition_list
      EXPORTING
         io_object = io_object.

   * Store any messages issued during calculations into an itab that
   * will be checked during the CHECK_ALL method:
      lr_fmca_pt_calc->go_msglist->get_list( IMPORTING et_list = gt_msg ).

      lr_fmca_pt_calc->go_msglist->free( ).
   endmethod.

Class: ZCL_FMCA_PT_CALC_RULE

Method: IF_EX_RECD_CALC_RULE~GET_VALUES

method IF_EX_RECD_CALC_RULE~GET_VALUES.

   * Only for calculation rule type 'PPTX':
   CHECK id_calcruleext = 'PPTX'.

   DATA:
      lr_fmca_pt_calc TYPE REF TO zcl_fmca_pt_calc.

   CREATE OBJECT lr_fmca_pt_calc.

   *calculate tax amounts for one condition
   CALL METHOD lr_fmca_pt_calc->condition_value
      EXPORTING
         id_abs_from          = id_abs_from
         id_abs_to            = id_abs_to
Calculating Property Tax with BRF+

Method: IF_EX_RECD_CALC_RULE~GET_ATTRIBUTES

method IF_EX_RECD_CALC_RULE~GET_ATTRIBUTES.

* Only for calculation rule type 'PPTX':
CHECK id_calcruleext = 'PPTX'.

  cf_distribute = 'X'. " Can the calculation also distribute?
  cf_adjustable = ' '. " Can the condition be adjusted by the adjustment?
  cf_unitprice_hide = 'X'. " Should the unit price be hidden?
  cf_depend_condition = ' '. " Is the calculation dependent on conditions?
  cf_depend_object = 'X'. " Is the calculation dependent on object data?
  CLEAR cf_unique_values. " Does the calculation result in one-time amounts?
  CLEAR cf_unique_values_multi. " Does the calculation result in multiple one-time amounts?

  * cd_gui_fm_para_pbo = 'FMCA_PT_GUI_CALC_RULE_PBO'. " PBO function module for maintenance of formula parameters
  * cd_gui_fm_para_pai = 'FMCA_PT_GUI_CALC_RULE_PAII'. " PAI function module for maintenance of formula parameters

endmethod.
Class: ZCL_FMCA_PT_CALC
Attribute: GV_BRFPLUS_EXECUTED / GT_CALC_VALUE_BRFPLUS / GS_CONTRACT_BRFPLUS

Method: SET_YEAR_VALUES_BRFPLUS

method SET_YEAR_VALUES_BRFPLUS.

DATA:
lv_origduedate TYPE sy-datum,
lsPosting_rh TYPE retm_posting_rh,
ls_condition_x TYPE recd_condition_x,
lv_next_from TYPE ps_valid_to_date,
lv_period_faktor TYPE int4,
lv_period TYPE fkk_fkdate_period,
ls_year TYPE zfmca_pt_year,
lv_date TYPE key_date_ps,
lv_year TYPE calcyear_ps,
key_date type key_date_ps.

CLEAR ct_year.

lv_period_faktor = is_rhythm-frequency.

CASE is_rhythm-frequencyunit.
    WHEN 0.
        *Months
            lv_period = 'M'.
            WHEN 1.
                *Years
                    lv_period = 'Y'.
                    WHEN 2.
                        *Days
                            lv_period = 'D'.
                    ENDCASE.
    ls_year-valid_from = iv_start_date.
    DO.

    CALL FUNCTION 'FKK_DTE_ADD'
        EXPORTING
            i_base_date = ls_year-valid_from
            i_periode = lv_period
            i_period_factor = lv_period_faktor
        IMPORTING
            e_datum = lv_next_from.
    ls_year-valid_to = lv_next_from - 1.

    * Determine taxation year based on: tax year starts based on frequency start
    IF ls_year-valid_from+4(4) >= iv_start_mmdd.
ls_year-calcyear = ls_year-valid_from(4).
ELSE.
   ls_year-calcyear = ls_year-valid_from(4) - 1.
ENDIF.

* Determine key date - same date used all through the tax year:
   CONCATENATE ls_year-calcyear iv_start_mmd INTO key_date.
   IF key_date > iv_end_date.
      EXIT.
   ENDIF.

* Set due date
   * ls_year-due_date = ls_year-key_date.

   * set condtype //condvalid_from
      move condtype to ls_year-condtype.
      APPEND ls_year TO ct_year.
      ls_year-valid_from = lv_next_from.
      CLEAR ls_year-valid_to.
      ENDOO.
   endmethod.

Method: GET_RHYTHM_START_BRFPLUS

method GET_RHYTHM_START_BRFPLUS.

* Based on the frequency settings, determine the starting month/day
  for conditions
*
   CLEAR ev_start_mmd.
   CASE is_rhythm-monthfrom.
      WHEN '14'.
         * Start of Calendar Year
            ev_start_mmd = '0101'.
      WHEN '13'.
         * start of contract
      WHEN '01' OR '02' OR '03' OR '04' OR '05' OR '06' OR '07' OR '08' OR '09' OR '10' OR '11' OR '12'.
         ev_start_mmd = iv_contract_valid_from+4(4).
CONCATENATE is_rhythm-monthfrom '01' INTO ev_start_mmdd.

WHEN '40'.
  Custom
  ev_start_mmdd = is_rhythm-rhythmbegin+4(4).
  WHEN OTHERS.
ENDCASE.

IF ev_start_mmdd IS INITIAL.
  MESSAGE e015(fmca_pt) WITH is_rhythm-monthfrom
  RAISING error.
ENDIF.
endmethod.

Method: CALCULATE_AMOUNT_BRFPLUS

CONSTANTS iv_tracetable type TABNAME value 'FDT_TRACE_0100'.

TYPES: 
  BEGIN OF ty_parcel_ci_buf, 
    objid TYPE dfacts-objid, 
    parcel_ci TYPE recn_contract_ci, 
  END OF ty_parcel_ci_buf.

TYPES:
  ty_t_parcel_ci_buf TYPE STANDARD TABLE OF ty_parcel_ci_buf.

DATA: l_bapiret2 TYPE bapiret2,
  ls_parcel TYPE bapi_re_parcel_land_int,
  lt_parcel TYPE table of bapi_re_parcel_land_int,
  ls_parcel_object TYPE bapi_re_object_rel_int,
  ls_parcel_ci TYPE relm_parcel_of_land_ci,
  ls_parcel_ci_buf TYPE ty_parcel_ci_buf,
  lt_parcel_ci_buf TYPE table of ty_parcel_ci_buf,
  lt_facts_buffer TYPE dfacts_t_type.

data:
  r_parcel_object TYPE RANGE OF dfacts-OBJID,
  r_parcel_line LIKE LINE OF r_parcel_object.

data:
  is_CALC_VALUE Type ZFMCA_PT_CALC_VALUE,
  it_CALC_VALUE Type table of ZFMCA_PT_CALC_VALUE,
  is_CALC_USED_OBJECTS Type RECAOBJNR.

DATA lv_result_string TYPE fmca_pt_t_calc_value.
DATA lv_timestamp TYPE timestamp.

DATA: lo_factory TYPE REF TO if_fdt_factory,
  lo_function TYPE REF TO if_fdt_function,
lo_context TYPE REF TO if_fdt_context,
lo_trace  TYPE REF TO if_fdt_trace,
lo_leantrace TYPE REF TO if_fdt_lean_trace,
1cx_fdt   TYPE REF TO cx_fdt.

data: lt_name_value TYPE abap_parmbind_tab,
    ls_name_value TYPE abap_parmbind.

data: ER_DATA type ref to data.

* Make message handler available to all methods:
go_msglist = io_msglist.

if is_ztrm_pt_brfplus-READ_PARCEL = 'X'.
*read parcels
*fill buffer
LOOP AT it_object_rel INTO ls_parcel_object
  WHERE objtypecn = 'I8'.
* Get parcel data needed for calculation:
  CALL FUNCTION 'API_RE_PL_GET_DETAIL'
    EXPORTING
      id_objnr       = ls_parcel_object-objnrcn
    IMPORTING
      es_parcel_of_land = ls_parcel
      es_ci_data      = ls_parcel_ci
    EXCEPTIONS
      error          = 1
      OTHERS         = 2.
  IF sy-subrc <> 0.
    go_msglist->add_symsg().
    RETURN.
  ENDIF.
  APPEND ls_parcel TO lt_parcel.
  MOVE-CORRESPONDING ls_parcel_ci TO ls_parcel_ci_buf.
  ls_parcel_ci_buf-objid = ls_parcel_object-objnrcn.
  APPEND ls_parcel_ci_buf TO lt_parcel_ci_buf.
  clear r_parcel_line.
  r_parcel_line-sign = 'I'.
  r_parcel_line-option = 'EQ'.
  move ls_parcel-intreno to r_parcel_line-low.
  r_parcel_line-high = ' '.
  append r_parcel_line to r_parcel_object.
ENDLOOP.

*Select facts of parcel
  CLEAR lt_facts_buffer.
  SELECT * FROM dfacts INTO TABLE lt_facts_buffer
    WHERE obart = 'I8'
    AND objid in r_parcel_object.
  * APPEND LINES OF lt_facts_buffer TO gt_facts_buffer.
ENDIF.
GET TIME STAMP FIELD lv_timestamp.

MOVE: 'IS_CONTRACT' TO ls_name_value-name,
'S' TO ls_name_value-kind.
GET REFERENCE OF ls_contract INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
 IR_DATA = er_data
 IV_FUNCTION_ID = brfplus_function_id
 IV_DATA_OBJECT = 'IS_CONTRACT'
 IV_TIMESTAMP = lv_timestamp
 IV_TRACE_GENERATION = is_ztrm_pt_brfplus-trace
 IV_HAS_DDIC_BINDING = 'X'
IMPORTING
 ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.

MOVE: 'IT_CONTRACT_PARTNER' TO ls_name_value-name,
'T' TO ls_name_value-kind.
GET REFERENCE OF it_main_partner[] INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
 IR_DATA = er_data
 IV_FUNCTION_ID = brfplus_function_id
 IV_DATA_OBJECT = 'IT_CONTRACT_PARTNER'
 IV_TIMESTAMP = lv_timestamp
 IV_TRACE_GENERATION = is_ztrm_pt_brfplus-trace
 IV_HAS_DDIC_BINDING = 'X'
IMPORTING
 ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.

MOVE: 'IT_YEAR' TO ls_name_value-name,
'T' TO ls_name_value-kind.
GET REFERENCE OF it_year[] INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
 IR_DATA = er_data
 IV_FUNCTION_ID = brfplus_function_id
 IV_DATA_OBJECT = 'IT_YEAR'
 IV_TIMESTAMP = lv_timestamp
 IV_TRACE_GENERATION = is_ztrm_pt_brfplus-trace
 IV_HAS_DDIC_BINDING = 'X'
IMPORTING
 ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.

MOVE: 'IT_DFACTS' TO ls_name_value-name,
'T' TO ls_name_value-kind.
GET REFERENCE OF lt_facts_buffer[] INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
 IR_DATA = er_data
 IV_FUNCTION_ID = brfplus_function_id
 IV_DATA_OBJECT = 'IT_DFACTS'
 IV_TIMESTAMP = lv_timestamp
 IV_TRACE_GENERATION = is_ztrm_pt_brfplus-trace
 IV_HAS_DDIC_BINDING = 'X'
IMPORTING
 ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.
MOVE: 'IT_COND' TO ls_name_value-name,
'T' TO ls_name_value-kind.

GET REFERENCE OF IT_COND[] INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
IR_DATA = er_data
IV_FUNCTION_ID = brfplus_function_id
IV_DATA_OBJECT = 'IT_COND'
IV_TIMESTAMP = lv_timestamp
IV_TRACE_GENERATION = is_ztrm_pt_brpfplus-trace
IV_HAS_DDIC_BINDING = 'X'
IMPORTING
ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.

MOVE: 'IT_PARCEL' TO ls_name_value-name,
'T' TO ls_name_value-kind.

GET REFERENCE OF IT_PARCEL[] INTO er_data.
CALL METHOD CL_FDT_FUNCTION_PROCESS=>MOVE_DATA_TO_DATA_OBJECT
EXPORTING
IR_DATA = er_data
IV_FUNCTION_ID = brfplus_function_id
IV_DATA_OBJECT = 'IT_PARCEL'
IV_TIMESTAMP = lv_timestamp
IV_TRACE_GENERATION = is_ztrm_pt_brpfplus-trace
IV_HAS_DDIC_BINDING = 'X'
IMPORTING
ER_DATA = ls_name_value-value.
INSERT ls_name_value INTO TABLE lt_name_value.

TRY.

IF is_ztrm_pt_brpfplus-trace EQ 'X'.

  process the function
  cl_fdt_function_process=>process(
    EXPORTING
    iv_function_id = brfplus_function_id
    iv_timestamp = lv_timestamp
    iv_trace_mode = if_fdt_constants=>gc_trace_mode_lean
    IMPORTING
    eo_trace = lo_trace
    ea_result = it_calc_value
    CHANGING
    ct_name_value = lt_name_value ).
  
else.

  process the function
  cl_fdt_function_process=>process(
    EXPORTING
    iv_function_id = brfplus_function_id
    iv_timestamp = lv_timestamp
    IMPORTING
    ea_result = it_CALC_VALUE
    CHANGING ct_name_value = lt_name_value ).

endif.

CATCH cx_fdt INTO lcx_fdt.

CALL METHOD go_msglist->add
EXPORTING
id_msgty = 'E'
Calculating Property Tax with BRF+

id_msgid = 'ZTRM_PT'
id_msgno = '001'
id_msgv1 = brfplus_function_id.
return.
ENDTRY.

IF is_ztrm_pt_brfplus-trace EQ 'X'.
*** save lean trace ***

TRY .
  lo_leantrace ?= lo_trace.
  IF lo_leantrace IS BOUND.
    lo_leantrace->if_fdt_trace->set_trace_db_table( iv_tracetable ).
    lo_leantrace->save( ).
  ENDIF.
CATCH cx_sy_move_cast_error.
  CALL METHOD go_msglist->add
    EXPORTING
    id_msgty = 'E'
    id_msgid = 'ZTRM_PT'
    id_msgno = '002'.
return.
CATCH cx_fdt_input.
  CALL METHOD go_msglist->add
    EXPORTING
    id_msgty = 'E'
    id_msgid = 'ZTRM_PT'
    id_msgno = '002'.
return.
CATCH cx_fdt_lean_trace.
  CALL METHOD go_msglist->add
    EXPORTING
    id_msgty = 'E'
    id_msgid = 'ZTRM_PT'
    id_msgno = '002'.
return.
ENDTRY.

**** start store lean trace ref data *****
data: wa_ZTRM_PT_LEANHIS type ZTRM_PT_LEANHIS.
move lo_leantrace->mv_uuid to wa_ZTRM_PT_LEANHIS-leantrace_id.
move brfplus_function_id to wa_ZTRM_PT_LEANHIS-function_id.
move lv_timestamp to wa_ZTRM_PT_LEANHIS-timestamp.
move ls_contract-BUKRS to wa_ZTRM_PT_LEANHIS-BUKRS.
move ls_contract-RECNNR to wa_ZTRM_PT_LEANHIS-RECNNR.
insert ZTRM_PT_LEANHIS from wa_ZTRM_PT_LEANHIS.
if sy-subrc <> 0.
  CALL METHOD go_msglist->add
    EXPORTING
    id_msgty = 'E'
    id_msgid = 'ZTRM_PT'
    id_msgno = '003'.
return.
endif.
endif.

**** calculate used objects plus prepare aggregation in case of many parcels ****
clear ct_CALC_USED_OBJECTS.
Loop at it_calc_value into is_calc_value.
if is_CALC_VALUE-parcel is not initial.
   move is_calc_value-parcel to is_CALC_USED_OBJECTS.
   append is_CALC_USED_OBJECTS to ct_CALC_USED_OBJECTS.
endif.

is_CALC_VALUE-parcel = '1'.
collect is_CALC_VALUE into ct_calc_value.
clear is_CALC_VALUE.
endloop.
sort ct_CALC_USED_OBJECTS.

delete adjacent duplicates from ct_CALC_USED_OBJECTS.
GV_BRFPLUS_EXECUTED = 'X'.
endmethod.

Method: CLEAR_ATTRIBUTES
method CLEAR_ATTRIBUTES.
   CALL METHOD SUPER->CLEAR_ATTRIBUTES.
   CLEAR GV_BRFPLUS_EXECUTED.
endmethod.

Method: CONDITION_VALUE
method CONDITION_VALUE.

* The following global attributes store the calculation results:
* GT_CALC_VALUE (list of calculation result)
* GT_CALC_USED_OBJECTS (list of parcel which are used)
* GT_CONDITION (condition list)

DATA:
   ls_msg TYPE recamsg,
   lv_parcel_intreno TYPE recaintreno,
   ls_parcel TYPE bapi_re_parcel_land_int,
   ls_calc_value_int TYPE fmca_pt_calc_value,
   lt_calc_value_int TYPE TABLE OF fmca_pt_calc_value,
   lv_calcrulepara1 TYPE reccalcrulepara,
   lv_calcrulepara2 TYPE reccalcrulepara,
   lv_valid_to TYPE ps_valid_to_date,
   ls_calc_value TYPE recd_calc_values_tab.

*If linked objects changed clear global attributes
   CALL METHOD me->check_linked_objects
   EXPORTING
      io_object = io_object.
* If calculation results exist do not recalculate:
  if GV_BRFPLUS_EXECUTED = '').

  go_msglist->clear( ).
  CALL METHOD calculates_values
    EXPORTING
      id_abs_from          = id_abs_from
      id_abs_to            = id_abs_to
      io_object            = io_object
      io_msglist           = go_msglist
    CHANGING
      ct_calc_used_objects = gt_calc_used_objects.
  endif.

  LOOP AT gt_calc_value INTO ls_calc_value_int
    WHERE condtype = is_condition-condtype
      AND valid_from >= id_abs_from
      AND valid_to   <= id_abs_to.

    ls_calc_value-CALCVALUE = ls_calc_value_int-calcvalue.
    ls_calc_value-valuevalidfrom = ls_calc_value_int-valid_from.
    ls_calc_value-valuevalidto   = ls_calc_value_int-valid_to.
    ls_calc_value-calcnrnumber   = sy-index.
    ls_calc_value-partnerobjnr   = ls_calc_value_int-contract.
    APPEND ls_calc_value TO ct_calc_values.
    CLEAR ls_calc_value.
  endloop.

  ct_calc_used_objects = gt_calc_used_objects.

  endmethod.
Method: CALCULATES_VALUES.

```
method CALCULATES_VALUES.

* The following global attributes store the calculation results:
  * GT_CALC_VALUE  (list of calculation result)
  * GT_CALC_USED_OBJECTS (list of parcel which are used)
  * GT_CONDITION ( condition list)

  types:
  BEGIN OF s_function,
    id TYPE if_fdt_types=>id,
    name TYPE if_fdt_types=>name,
    context TYPE if_fdt_types=>ts_id_name, "temporary, shall use ts_id_name
  END OF s_function .

  DATA:
  lv_start_mmd TYPE char4,
  lv_last_calc_date TYPE sydatum,
  lt_year TYPE zfmca_pt_t_year,
  lt_year_temp TYPE zfmca_pt_t_year,
  ls_year TYPE zfmca_pt_year,
  lv_end_date TYPE sydatum,
  lv_start_date TYPE sydatum,
  lv_cond_start_date TYPE sydatum,
  lv_end_year TYPE numc4,
  ls_contract TYPE bapi_re_contract_int,
  ls_contract_ci TYPE recn_contract_ci,
  ls_contract_type TYPE bapi_re_contract_type_int,
  lr_badi_fmca_pt_calc TYPE REF TO badi_fmca_pt_calc,
  lt_tax_rate TYPE fmca_pt_t_tax_rate,
  ls_condition TYPE bapi_re_condition_int,
  it_condition TYPE table of bapi_re_condition_int,
  lt_rhythm TYPE bapi_re_t_term_rh_int,
  ls_rhythm TYPE bapi_re_term_rh_int,
  ls_calc_value_ext TYPE recd_calc_values_tab,
  ls_calc_value TYPE zfmca_pt_calc_value,
  gs_calc_value TYPE fmca_pt_calc_value,
  ls_parcel TYPE bapi_re_parcel_land_int,
```
lv_parcel_intreno TYPE recaintreno,
lv_condamount TYPE boolean,
lt_MAIN_PARTNER TYPE BAPI_RE_T_PARTNER_INT,
lt_COND_CALC_OLD TYPE BAPI_RE_T_COND_CALC_INT,
wa_COND_CALC_OLD TYPE BAPI_RE_COND_CALC_INT,
lt_object_rel TYPE BAPI_RE_T_OBJECT_REL_INT,
is_ztrm_pt_brfplus TYPE ztrm_pt_brfplus.

DATA:
  lv_appl_name TYPE fdt_application_name,
  lv_appl_id TYPE if_fdt_types=>id,
  lo_brfp_manager TYPE REF TO cl_fmca_brfp_manager,
  lt_func TYPE cl_fmca_brfp_manager=>t_function,
  ls_func TYPE s_function,
  func_name TYPE if_fdt_types=>name,
  lv_func_id TYPE if_fdt_types=>id,
  brfplus_function_id TYPE if_fdt_types=>id.

*** NEW BRFplus variables ***
data: lt_calc_value TYPE table of zfmca_pt_calc_value.

data: lt_cond type RE_T_CONDITION_REL_GROUP,
  ls_cond type RECDC_CONDITION_REL_GROUP.

*read contract data
CALL FUNCTION 'API_RE_CN_GET_DETAIL'
  EXPORTING
    io_object = io_object
  IMPORTING
    es_contract = ls_contract
    es_contract_type = ls_contract_type
    es_ci_data = ls_contract_ci
    et_object_rel = lt_object_rel
    ET_MAIN_PARTNER = lt_MAIN_PARTNER
    ET_COND_CALC = lt_COND_CALC_old
    et_term_rhythm = lt_rhythm.

  select single * from ZTRM_PT_BRFPLUS
   into is_ztrm_pt_brfplus where RECNTYPE = ls_contract.RECNTYPE.
  if sy-subrc <> 0.
    CALL METHOD go_msglist->add
EXPORTING
   id_msgty = 'E'
   id_msgid = 'ZTRM_PT'
   id_msgno = '000'
   id_msgv1 = ls_contract-RECTYPE.
EXIT.
endif.

**** Delete ET_COND_CALC conditions which should be reused
if is_ztrm_pt_brfplus-RECALCSTARTDATE is not initial.
   delete lt_COND_CALC_old where VALUEVALIDfrom >= is_ztrm_pt_brfplus-RECALCSTARTDATE.
endif.

* read TAX_COND customizing TIVCDCONDREC / TIVCDCONDTYPE
CALL METHOD cl_recdc_condition_rel_group=>get_list_by_contracttype
EXPORTING
   id_contracttype = ls_contract-RECTYPE
IMPORTING
   et_list = lt_cond.

* calculate amount for each TAX_COND customizing record
LOOP AT lt_cond INTO ls_cond.
   CLEAR ls_condition.
   IF NOT is_condition IS INITIAL.
      MOVE-CORRESPONDING is_condition TO ls_condition.
      ls_condition-objnrcalc = is_condition-objnr.
   ELSE.
      ls_condition-objnrcalc = ls_contract-objnr.
      ls_condition-condtype = ls_cond-condtype.
      ls_condition-condpurposeext = ls_cond-condpurposeext.
      ls_condition-condcurr = ls_contract-recncncurr.
   ENDF.
   CALL BADI lr_badi_fmca_pt_calc->set_terms
      EXPORTING
         io_object = io_object
         is_condition = ls_condition
      CHANGING
         cv_condpurposeext = ls_condition-condpurposeext
cv_termnopy = ls_condition-termnopy
cv_termnorh = ls_condition-termnorh
cv_termnoaj = ls_condition-termnoaj
cv_termnooa = ls_condition-termnooa
cv_termnosr = ls_condition-termnosr
cv_termnomr = ls_condition-termnomr.

READ TABLE lt_rhythm INTO ls_rhythm
  WITH KEY termno = ls_condition-termnorh.
IF sy-subrc <> 0.
  CALL METHOD go_msglist->add
    EXPORTING
      id_msgty = 'E'
      id_msgid = 'FMCA_PT'
      id_msgno = '016'
      id_msgv1 = ls_condition-termnorh.
  EXIT.
ENDIF.

* Determine the month/day on which this condition starts:
  CALL METHOD get_rhythm_start_brfplus
    EXPORTING
      is_rhythm              = ls_rhythm
      iv_contract_valid_from = ls_contract-recnbeg
    IMPORTING
      ev_start_mmdd          = lv_start_mmdd
  EXCEPTIONS
    error                  = 1
    OTHERS                 = 2.
IF sy-subrc <> 0.
  go_msglist->add_symmsg( ).
  CONTINUE.
ENDIF.

* Determine earliest possible start for the condition:
  * - start of contract
  * - start of condition ***

  IF id_abs_from IS INITIAL OR ls_contract-recnbeg > id_abs_from.
    lv_start_date = ls_contract-recnbeg.
  ELSE.
lv_start_date = id_abs_from.
ENDIF.

*** Move start date to condition_start_date to preserve the value prior assignment to the conditions
lv_cond_start_date = lv_start_date.

IF is_ztrm_pt_brfplus-RECALCSTARTDATE is not initial.
   if is_ztrm_pt_brfplus-RECALCSTARTDATE > lv_start_date.
      lv_start_date = is_ztrm_pt_brfplus-RECALCSTARTDATE.
   endif.
endif.

* If the start date is after the frequency start MMDD, then bump up to next year:
  IF lv_start_date+4(4) > lv_start_mmdd.
     lv_start_date(4) = lv_start_date(4) + 1.
  ENDIF.

* Set start MMDD to frequency start MMDD:
  lv_start_date+4(4) = lv_start_mmdd.

* Determine end date for condition as earliest of:
  * - end of contract
  * - end of condition IS_CONDITION
  * - end of TAX_COND customizing record
    IF id_abs_to IS INITIAL.
       lv_end_date = '99991231'.
    ELSE.
       lv_end_date = id_abs_to.
    ENDIF.
    IF ls_contract-recnendabs IS NOT INITIAL AND ls_contract-recnendabs < lv_end_date.
       lv_end_date = ls_contract-recnendabs.
    ENDIF.

**** Set condition start date independent of the recalculation date *****

IF is_condition IS INITIAL.
   ls_condition-condvalidfrom = lv_cond_start_date.
   ls_condition-condvalidto = lv_end_date.
ENDIF.
* Determine end of calculations date based on current date and
  * number of years for cash flow forecasting into the future:
    CONCATENATE sy-datum(4) lv_start_mmdd INTO lv_last_calc_date.
    IF lv_last_calc_date < sy-datum.
      lv_last_calc_date(4) = lv_last_calc_date(4) + 1.
    ENDIF.
    lv_end_year = lv_last_calc_date(4) + ls_contract_type-cfperiod.
    CONCATENATE lv_end_year lv_start_mmdd INTO lv_last_calc_date.
    SUBTRACT 1 FROM lv_last_calc_date.

* If the condition only starts in the future after the current end of
  * cash flow forecasting, do not create the condition yet:
    IF lv_start_date > lv_last_calc_date.
      CONTINUE.
    ENDIF.

* Determine end date for calculations as earliest of:
  * - current tax year + number of years of cash flow forecasting for condition type
  * - end of condition
    IF lv_last_calc_date < lv_end_date OR lv_end_date = '99991231'.
      lv_end_date = lv_last_calc_date.
    ENDIF.

*build year table
  CALL METHOD set_year_values_brfplus
    EXPORTING
      iv_start_date = lv_start_date
      iv_end_date = lv_end_date
      is_rhythm = ls_rhythm
      is_condition = ls_condition
      iv_start_mmdd = lv_start_mmdd
      condtype = ls_cond-condtype
      condtypevalid_from = lv_cond_start_date
    CHANGING
      ct_year = lt_year_temp.

    append lines of lt_year_temp to lt_year.

    append ls_condition to it_condition.

endloop.
**** Retrieve BRFplus Function ID
move is_ztrm_pt_brfplus-application to lv_appl_name.
CREATE OBJECT lo_brfp_manager
  EXPORTING
    iv_application_name = lv_appl_name.
CALL METHOD lo_brfp_manager->get_function
  IMPORTING
    et_function = lt_func.
move is_ztrm_pt_brfplus-function to func_name.

Loop at lt_func into ls_func where name = func_name.
endloop.

**** error message
if sy-subrc = 4.
  l_bapiret2-type       = 'E'.
  l_bapiret2-id        = 'ZTRM_OBT'.
  l_bapiret2-number    = '002'.
  move func_name to l_bapiret2-message_v1.
  exit.
endif.
move ls_func-id to brfplus_function_id.

********** Start Call BRFplus once ****
* calculate amount for one condition
CALL METHOD calculate_amount_brfplus
  EXPORTING
    brfplus_function_id = brfplus_function_id
    it_year            = lt_year
    it_cond            = lt_cond
*    it_tax_rate       = lt_tax_rate
    iv_objnr           = is_condition-objnr
    io_object          = io_object
    io_msglist         = go_msglist
    IT_MAIN_PARTNER    =
    lt_MAIN_PARTNER    =
    IT_OBJECT_REL      =
    lt_OBJECT_REL      =
LS_CONTRACT = LS_CONTRACT
IT_COND_CALC_OLD = lt_COND_CALC_old
is_ztrm_pt_brfplus = is_ztrm_pt_brfplus

CHANGING
calc_value = lt_calc_value
calc_used_objects = ct_calc_used_objects.

IF io_msglist->has_messages_of_msgty( 'E' ) = abap_true.
  CONTINUE.
ENDIF.

**** Add values which have already been calculated
  clear ls_calc_value.

  if is_ztrm_pt_brfplus-RECALCSTARTDATE is not initial.
    Loop at lt_COND_CALC_old into wa_COND_CALC_old.
      move wa_COND_CALC_old-CONDTYPE to ls_calc_value-CONDTYPE.
      move wa_COND_CALC_old-CALCVALUE to ls_calc_value-CALCVALUE.
      move wa_COND_CALC_old-VALUEVALIDFROM to ls_calc_value-valid_from.
      move wa_COND_CALC_old-VALUEVALIDTO to ls_calc_value-valid_to.
      append ls_calc_value to Lt_calc_value.
    endloop.
  endif.

****** End Call BRFplus ******

LOOP AT lt_cond INTO ls_cond.
  lv_condamount = abap_false.

  Loop at Lt_calc_value into ls_calc_value where
    CONDTYPE = ls_cond-CONDTYPE.

  * add calculated values in global attribute gt_calc_value
    IF NOT ls_calc_value IS INITIAL
      AND ls_calc_value-calcvalue NE 0.
      move-corresponding ls_calc_value to gs_calc_value.
      APPEND gs_calc_value TO gt_calc_value.
    ENDIF.

  IF ls_calc_value-calcvalue NE 0.
    lv_condamount = abap_true.
  ENDIF.
  *
  ENDIF.
ENDLOOP.

* add current condition to condition list if at least one records exists
  IF lv_condamount = abap_true.

    clear ls_condition.
    read table it_condition into ls_condition with key condtype = ls_cond-CNDTYPE.

    CALL METHOD add_condition_value
      EXPORTING
        is_condition = ls_condition
      CHANGING
        ct_condition = ct_condition.
  ENDIF.
ENDLOOP.

SORT gt_calc_value BY condtype valid_from valid_to.

gt_calc_used_objects = ct_calc_used_objects.
endmethod.
Related Content

Flexible Real Estate

BRFplus

Tax and Revenue Management

For more information, visit the Business Rules Management homepage.
Calculating Property Tax with BRF+