Buchungsbelege erstellen

Folgend ein Report, der exemplarisch zeigt, wie FI-Belege gebucht werden können. Es werden Die Bausteine BAPI_ACC_DOCUMENT_CHECK und BAPI_ACC_DOCUMENT_POST verwendet.

Aktuell macht der Report gar nichts!

Die Kopf- und Positionsdaten müssen in LT_BKPF und LT_BSEG entsprechend eingefügt werden. Wie genau das geht, erfährst du in dieser ausführlichen Doku:

Dokumentation

FI Buchungen mittels BAPI_ACC_DOCUMENT_POST

Vorgaben

Ich gehe davon aus, das die zu buchenden Daten in der Form Kopf/Position vorliegen. Sollte dies nicht der Fall sein, so sollten diese vorher so aufbereitet werden, da dies den Umgang beim Programmieren wesentlich vereinfacht.

Das folgende Beispiel behandelt exemplarisch eine kreditorische Buchung,
läßt sich jedoch prinzipiell ebenso auf debitorische oder Sachkonten-
buchungen übertragen.
Als Schmankerl wird ein zusätzliches Feld, welches nicht in der Sachkontenzeile
enthalten ist (LZBKZ – Landeszentralbankkennzeichen) mitgegeben und die Steuer
mit einem Steuerschema gebucht, welches zwei Steuerzeilen enthält.

Prinzipiell muss man folgendes Wissen zur Buchung von FI-Belegen wissen:

1. Die erste Zeile des zu buchenden Beleges enthält den Betrag IMMER als Brutto-Wert! Unabhängig davon ob diese kreditorisch, debitorisch oder eine Sachkontenbuchung ist.
2. Bei kreditorischen oder debitorischen Buchungen tauchen diese als Zeile nur ein mal als erste Zeile auf!
3. Steuerkennzeichen MÜSSEN mitgeliefert werden sonst kann keine Steuer gebucht werden.

Kopfdaten

Wenn die Daten aus einem R3-System kommen, liegen die Daten normalerweise in der Form BUKRS/BELNR/GJAHR + Datenteil (für BKPF/BSEG) vor.
Sollte dies nicht der Fall, so erhält man normalerweise, wie in diesem Beispiel, eine laufende, eindeutige Nummer als Identifikator.

Für den Kopf auf jeden Fall müssen mitgegeben werden:

- BUDAT – Buchungsdatum
- BLDAT – Belegdatum
- BUKRS – Buchungskreis
- BLART – Belegart
- WAERS – Währung in der gebucht werden soll (Belegwährung)

Bei Fremdwährungsbuchungen müssen zusätzlich das Feld

- KURSF – Umrechnungskurs zur Buchungskreiswährung (Zeilentyp BAPIACCR09)
  ODER!!!!
- WWERT – Umrechnungsdatum (Zeilentyp BAPIACHE09)

mitgegeben werden – keinesfalls beide!

Bei kreditorischen Buchungen ist z.B. auch die Referenz zu füllen

- XBLNR – Referenznummer (Belegnummer des Lieferanten)

All diese Werte müssen bekannt sein und mitgegeben werden.

Das Buchungsdatum kann auch leer gelassen werden, dann wird aus dem Systemdatum die entsprechenden Werte für GJAHR und POPER errechnet (siehe Routine ADD_DATA_BKPF). Ansonsten wird das mitgegebene Buchungsdatum verwendet und zur Berechnung verwendet

- BKTXT – Kopftext (braucht man manchmal)

Positionsdaten

- Als Mussdaten sind notwendig:
- BUZEI fortlaufende Buchungszeilennummer.
  Kann man auch „on the fly“ erzeugen einfacher ist es, wenn sie bereits gefüllt ist. Die Buchungszeile stellt über den in jedem Segment vorhandenen Parameter ITEMNO_ACC die Verbindung bzw. die Sortierung der einzelnen Zeilen des Beleges untereinander sicher
- KOART Kontoart
  K – Kreditorische Buchungszeile, D – Debitorische Buchungszeile,
  S – Sachkontenzeile
- SHKZG Soll-/Haben Kennzeichen S/H
  Auch aus Fremdsystemen erhält man normalerweise dieses Kennzeichen. Es dient dazu, das Vorzeichen für die Beträge, welche dem BAPI übergeben werden, richtig zu ermitteln. Dies setzt voraus, das die Beträge (!!) immer als positive Werte übergeben werden – was im R/3 immer der Fall ist und bei Fremdsystembelegen zu 99% (SAP hat schließlich die Buchhaltung nicht erfunden)
  Habenwerte sind hierbei immmer mit -1 zu multiplizieren, Sollwerte behalten Ihr positives Vorzeichen
- GKONT Gegenkonto
  Im Fall einer kreditorischen bzw. debitorischen Zeile die Lieferanten- bzw. Kundennummer.
  Im Fall einer Sachkontenzeile die Kontonummer, auf welche gebucht werden soll
- MWSKZ Mehrwertsteuerkennzeichen
  MUSS, im Falle einer ggf. zu buchenden Steuer, mitgegeben werden. Hierüber werden die zu buchenden Steuerzeilen ermittelt

- BRUTTO Bruttowert der Buchungszeile. Wird immer benutzt bei „Kopfzeilen“ d.h. bei kreditorische, debitorischen oder der 1. Zeile eines Sachkontenbeleges
- NETTO Nettowert der Buchungszeile d.h. ohne Steuer
  Wird bei allen „Positionszeilen“, d.h. ab Zeile 2 des Beleges benutzt

Kontierungen

Im Beispiel habe ich Kontierungen auf Kostenstellen, Innenaufträge, SD-Belege und Anlagen vorgesehen.

- SGTXT Positionstext (braucht man manchmal)
- LANDL Lieferland
  Wird benötigt, wenn der Lieferant im Ausland ansässig ist – kann man aber auch durch nachlesen des Landes aus der Adresse des Lieferanten holen. Hier der Einfachheit halber in der Struktur

Schmankerl 1

- LZBKZ Landeszentralbankkennzeichen
  Die betrifft Rechnungen von ausländischen Lieferanten die eine Sonstige Leistung i.S. des Umsatzsteuergesetzes erbracht haben. Der §13b UStG regelt unter einzelnen Punkten in welchen Fällen der Leistungsempfänger für die Leistungen die Umsatzsteuer schuldet. Gleichzeitig darf der Leistungsempfänger sich diesen Betrag als Vorsteuer in Abzug bringen. Steht in der BSEG als Feld zur Verfügung, ist aber nicht in der kreditorischen Struktur BAPIACAP09 vorhanden!!! Muß über die Tabelle extension2 des BAPI’s übergeben werden Ausführungen dazu in der Unterroutine für die Kreditorenzeile

Schmankerl 2

- CO-PA Kontierung
  Bei Kontierung auf CO-PA Objekte (Ergebnisobjekte) kommt es oft genug vor, das vom Kunden nur der Vertriebsbeleg vorgegeben wird und alles andere soll ermittelt werden. Wir befinden uns nicht in der FB01 oder FB60 und können auf den Knopf „Ableitung“ drücken.
  Was also tun? Siehe hierzu die Unterroutine „ADD_COPA_LINE‘.

Coding

```
*&---------------------------------------------------------------------*
*& Report Z_POST_ACC_DOCUMENT
*&---------------------------------------------------------------------*
*&
*&---------------------------------------------------------------------*
REPORT z_post_acc_document.

******** KOPFDATEN **************
TYPES: BEGIN OF gtys_bkpf,
         id    TYPE numc10,
         bukrs TYPE bkpf-bukrs,
         gjahr TYPE bkpf-gjahr,
         poper TYPE poper,
         blart TYPE bkpf-blart,
         bldat TYPE bkpf-bldat,
```

```
         budat TYPE bkpf-budat,
         xblnr TYPE bkpf-xblnr,
         bktxt TYPE bkpf-bktxt,
         waers TYPE bkpf-waers,
         kursf TYPE bkpf-kursf,
         wwert TYPE bkpf-wwert,
         belnr TYPE bkpf-belnr,
       END OF gtys_bkpf,
       gtyt_bkpf TYPE STANDARD TABLE OF gtys_bkpf.

******** POSITIONSDATEN ***********
* Schmankerl 1
* LZBKZ  Landeszentralbankkennzeichen
*        Die betrifft Rechnungen von ausländischen Lieferanten die eine
Sonstige Leistung
*        i.S. des Umsatzsteuergesetzes erbracht haben. Der §13b UStG regelt
unter einzelnen
*        Punkten in welchen Fällen der Leistungsempfänger für die Leistungen
die Umsatzsteuer schuldet.
*        Gleichzeitig darf der Leistungsempfänger sich diesen Betrag als
Vorsteuer in Abzug bringen.
*        Steht in der BSEG als Feld zur Verfügung, ist aber nicht in der
kreditorischen
*        Struktur BAPIACAP09 vorhanden!!! Muß über die Tabelle extension2 des
BAPI's übergeben werden
*        Ausführungen dazu in der Unterroutine für die Kreditorenzeile
"ADD_CRED_LINE"
* Schmankerl 2
* VBELN   CO-PA Kontierung
*        Bei Kontierung auf CO-PA Objekte (Ergebnisobjekte) kommt es oft
genug vor,
*        das vom Kunden nur der Vertriebsbeleg vorgegeben wird und alles
andere soll
*        ermittelt werden. Wir befinden uns nicht in der FB01 oder FB60 und
können auf
*        den Knopf "Ableitung" drücken.
*        Was also tun?
*        Siehe hierzu die Unterroutine "ADD_COPA_LINE'.
* Schmankerl 3
* BWASL  Bewegungsartenschlüssel
*        Bei der direkten Buchung von Kreditor auf Anlage muß der
Bewegungsartenschlüssel
*        vorgegeben werden. Dieser ist zwingend notwendig, sonst kommt keine
Buchung zustande
*        bzw. wird abgelehnt.
*        Näheres hierzu in der Unterroutine "ADD_SACH_LINE'.
TYPES: BEGIN OF gtys_bseg,
         id     TYPE numc10,
         buzei  TYPE bseg-buzei,
         koart  TYPE bseg-koart,
         shkzg  TYPE bseg-shkzg,
```

```
         gkont  TYPE gkont,
         brutto TYPE bseg-wrbtr,
         netto  TYPE bseg-wrbtr,
         mwskz  TYPE bseg-mwskz,
         sgtxt  TYPE bseg-sgtxt,
         kostl  TYPE bseg-kostl,
         aufnr  TYPE bseg-aufnr,
         vbeln  TYPE bseg-vbeln,
         posnr  TYPE bseg-posn2,
         anln1  TYPE bseg-anln1,
         anln2  TYPE bseg-anln2,
         lzbkz  TYPE bseg-lzbkz,
         landl  TYPE bseg-landl,
         bwasl  TYPE bwasl,
       END OF gtys_bseg,
       gtyt_bseg TYPE STANDARD TABLE OF gtys_bseg.

* Struktur für den Aufbau der Steuerzeilen
TYPES: BEGIN OF gtys_tax,
         mwskz TYPE mwskz,
         wmwst TYPE wmwst,
         msatz TYPE msatz_f05l,
         ktosl TYPE ktosl,
         kawrt TYPE kawrt,
         hkont TYPE hkont,
         kschl TYPE kschl,
       END OF gtys_tax,
       gtyt_tax TYPE STANDARD TABLE OF gtys_tax.

* Org-Daten für CO-PA Ermittlung
TYPES: BEGIN OF gtys_org_copa,
         bukrs TYPE bukrs,
         kokrs TYPE kokrs,
         erkrs TYPE erkrs,
       END OF gtys_org_copa,
       gtyt_org_copa_hash TYPE HASHED TABLE OF gtys_org_copa
                          WITH UNIQUE KEY bukrs.
* Neue CO-PA Ableitung
TYPES: gtyt_copadata TYPE copadata_tab.

* Testparameter
PARAMETERS: p_xtest TYPE xfeld DEFAULT 'X'.

START-OF-SELECTION.
  PERFORM processing.
*&---------------------------------------------------------------------*
*&      Form  PROCESSING
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM processing .
```

```
  DATA: lt_bkpf  TYPE gtyt_bkpf.
  DATA: ls_bkpf  TYPE gtys_bkpf.
  DATA: lt_bseg  TYPE gtyt_bseg.
  DATA: ls_bseg  TYPE gtys_bseg.

* Die Datentabellen sind in diesem Beispiel nicht gefüllt
* Im beiliegenden PDF sind exemplarisch einige Buchungssätze
zusammengestellt,
* welche aber nur für das System gültig sind, in dem Sie verbucht wurden.
* Um das Beispiel nutzen zu können, müßt ihr hier die Daten entsprechend
* Eurem System von Hand aufbauen
* Das Buchungsdatum habe ich leer gelassen, da in diesem Beispiel immer mit
dem
* aktuellen Tagesdatum gebucht wird.
* Die Felder GJAHR und POPER werden während des Programmlaufes ermittelt
* und dann in die LS_BKPF übertragen - für den Fall, das man das ganze
* als Vorlage für eine Schnittstelle laufen lassen will 
*
* Für die Standardbuchung habe ich den Buchungssatz hier ausgesternt schon
* einmal vorbereitet:
*  CLEAR ls_bkpf.
*  ls_bkpf-id    = '1'.
*  ls_bkpf-budat = '00000000'.
*  ls_bkpf-bldat = '20180611'.
*  ls_bkpf-bukrs = '1000'.
*  ls_bkpf-gjahr = '0000'.
*  ls_bkpf-poper = '000'.
*  ls_bkpf-xblnr = '123'.
*  ls_bkpf-waers = 'EUR'.
*  APPEND ls_bkpf TO lt_bkpf.
*
*  CLEAR ls_bseg.
*  ls_bseg-id    = '1'.
*  ls_bseg-buzei = '001'
*  ls_bseg-koart = 'K'.
*  ls_bseg-shgkz = 'H'.
*  ls_bseg-gkont = '0000100205'.
*  ls_bseg-mwskz = abap_false.
*  ls_bseg-brutto = '1725.00'.
*  ls_bkpf-landl = 'DE'.
*  ls_bkpf-lzbkz = abap_false. "kein Nicht Deutscher Kreditor
*  APPEND ls_bseg TO lt_bseg.
*
*  CLEAR ls_bseg.
*  ls_bseg-id    = '1'.
*  ls_bseg-buzei = '002'.
*  ls_bseg-koart = 'S'.
*  ls_bseg-shgkz = 'S'.
*  ls_bseg-gkont = '0000479100'.
*  ls_bseg-mwskz = 'V3'.
*  ls_bseg-netto = '500.00'.
```

```
*  ls_bseg-kostl = '0000000100'.
*  ls_bseg-aufnr  = space.
*  ls_bseg-vbeln  = space.
*  ls_bseg-posnr  = '000000'.
*  ls_bseg-anln1  = space.
*  ls_bseg-anln2 = space.
*  ls_bseg-bwasl = space.
*  APPEND ls_besg TO lt_bseg.
*
*  CLEAR ls_bseg.
*  ls_bseg-id    = '1'.
*  ls_bseg-buzei = '003'.
*  ls_bseg-koart = 'S'.
*  ls_bseg-shgkz = 'S'.
*  ls_bseg-gkont = '0000479100'.
*  ls_bseg-mwskz = 'V2'.
*  ls_bseg-netto = '500.00'.
*  ls_bseg-kostl = '0000000100'.
*  ls_bseg-aufnr  = space.
*  ls_bseg-vbeln  = space.
*  ls_bseg-posnr  = '000000'.
*  ls_bseg-anln1  = space.
*  ls_bseg-anln2 = space.
*  ls_bseg-bwasl = space.
*  APPEND ls_besg TO lt_bseg.
*
*  CLEAR ls_bseg.
*  ls_bseg-id    = '1'.
*  ls_bseg-buzei = '004'.
*  ls_bseg-koart = 'S'.
*  ls_bseg-shgkz = 'S'.
*  ls_bseg-gkont = '0000479100'.
*  ls_bseg-mwskz = 'V3'.
*  ls_bseg-netto = '500.00'.
*  ls_bseg-kostl = '0000000140'.
*  ls_bseg-aufnr  = space.
*  ls_bseg-vbeln  = space.
*  ls_bseg-posnr  = '000000'.
*  ls_bseg-anln1  = space.
*  ls_bseg-anln2 = space.
*  ls_bseg-bwasl = space.
*  APPEND ls_besg TO lt_bseg.

  LOOP AT lt_bkpf INTO ls_bkpf.

    PERFORM post_acc_doc USING ls_bkpf
                               lt_bseg.
  ENDLOOP.

ENDFORM.
*&---------------------------------------------------------------------*
```

```
*&      Form  POST_ACC_DOC
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM post_acc_doc USING pis_bkpf  TYPE gtys_bkpf
                        pit_bseg  TYPE gtyt_bseg.

  DATA: ls_bkpf   TYPE gtys_bkpf.
  DATA: lt_bseg   TYPE gtyt_bseg.
  DATA: ls_bseg   TYPE gtys_bseg.
  DATA: lt_tax    TYPE gtyt_tax.

  DATA: BEGIN OF ls_awkey,
          belnr TYPE bkpf-belnr,
          bukrs TYPE bkpf-bukrs,
          gjahr TYPE bkpf-gjahr,
        END OF ls_awkey.

  DATA: lv_buzei TYPE bseg-buzei.
  DATA: lv_tabix TYPE sy-tabix.
  DATA: lv_subrc TYPE sy-subrc.
  DATA: lv_exit  TYPE xfeld.

* BAPI-Definitionen
* BKPF
  DATA ls_header TYPE bapiache09.
* BAPI-Protokoll
  DATA lt_return TYPE bapiret2_tab.
  DATA ls_return TYPE bapiret2.
* Sachkontenzeile
  DATA lt_acc_gl TYPE bapiacgl09_tab.
* Kreditorenzeile
  DATA lt_acc_py TYPE bapiacap09_tab.
* Debitorenzeile - Im Beispiel nicht ausgeführt
  DATA lt_acc_rv TYPE bapiacar09_tab               ##NEEDED.
* Wertzeile
  DATA lt_cur_am TYPE bapiaccr09_tab.
* Steuerzeile
  DATA lt_acc_tx TYPE bapiactx09_tab.
* Zusätzliche Parameter (siehe Routine add_cred_line)
  DATA lt_ext2   TYPE tt_bapiparex.
* CO-PA Merkmalstabelle
  DATA lt_acc_crit  TYPE bapiackec9_tab.
* CO-PA Wertetabelle
  DATA lt_acc_value TYPE bapiackev9_tab.

  ls_bkpf = pis_bkpf.

* Kopfdaten aufbereiten
  PERFORM add_head_line CHANGING ls_bkpf
                                 ls_header.
```

```
* Loop über alle Buchungszeilen des Beleges
  LOOP AT pit_bseg INTO ls_bseg
    WHERE id EQ ls_bkpf-id.
    lv_tabix = sy-tabix.
*   Kopieren der aktuellen Buchungszeile
*   Brauchen wir später, wenn wir die Steuerzeilen erzeugen wollen
    lv_buzei = ls_bseg-buzei.

*   Unterscheidung nach Kontoart
    CASE ls_bseg-koart.
*     Debitorenzeile
      WHEN 'D'.
*       Behandeln wir hier nicht - ist aber analog zum Kreditoren

*     Kreditorenzeile
      WHEN 'K'.
*       Kreditorenzeile aufbauen
        PERFORM add_cred_line USING    pis_bkpf
                                       ls_bseg
                              CHANGING lt_acc_py
                                       lt_ext2.

*       Betragszeile
        PERFORM add_curr_line USING    pis_bkpf
                                       ls_bseg
                                       lv_tabix
                              CHANGING lt_cur_am.
      WHEN 'S'.
*       Sachkontenzeile
        PERFORM add_sach_line USING    ls_bkpf
                                       ls_bseg
                              CHANGING lt_acc_gl
                                       lt_acc_crit
                                       lt_acc_value
                                       lt_ext2
                                       lv_subrc.
        IF lv_subrc NE 0.
          lv_exit = abap_true.
          EXIT.
        ENDIF.

*       Betragszeile
        PERFORM add_curr_line USING    pis_bkpf
                                       ls_bseg
                                       lv_tabix
                              CHANGING lt_cur_am.

*       Steuer für Sachkontenpositionszeilen ermitteln
        PERFORM get_tax USING    pis_bkpf
                                 ls_bseg
                                 lv_tabix
```

```
                        CHANGING lt_tax.

    ENDCASE.
  ENDLOOP.

* Fehler bei der CO-PA Ermittlung - und raus
  IF lv_exit EQ abap_true.
    RETURN.
  ENDIF.

* Nachdem wir auf den Sachkontenzeilen alle Steuern
* gesammelt haben, bauen wir daraus jetzt die Steuerzeilen auf
  PERFORM add_tax_lines USING    lt_tax
                                 pis_bkpf
                                 lv_buzei
                        CHANGING lt_acc_tx
                                 lt_cur_am.

* Beleg prüfen
  CALL FUNCTION 'BAPI_ACC_DOCUMENT_CHECK'
    EXPORTING
      documentheader = ls_header
    TABLES
      accountgl      = lt_acc_gl
      accountpayable = lt_acc_py
      accounttax     = lt_acc_tx
      currencyamount = lt_cur_am
      extension2     = lt_ext2
      return         = lt_return.
* Protokoll auswerten
  READ TABLE lt_return INTO ls_return
    WITH KEY type   = 'S'
             id     = 'RW'
             number = '614'.
  IF sy-subrc EQ 0.
*   Testflag abfragen
*   Wenn gesetzt, raus
    IF NOT p_xtest IS INITIAL.
      FORMAT COLOR COL_POSITIVE.
      WRITE:/ 'ID', pis_bkpf-id,
            'erfolgreich gebucht'                          ##NO_TEXT.
      FORMAT COLOR OFF.
      RETURN.
    ENDIF.
*   Kein Testflag. Buchen
    FREE lt_return.
    CALL FUNCTION 'BAPI_ACC_DOCUMENT_POST'
      EXPORTING
        documentheader = ls_header
      TABLES
        accountgl      = lt_acc_gl
```

```
        accountpayable = lt_acc_py
        accounttax     = lt_acc_tx
        currencyamount = lt_cur_am
        extension2     = lt_ext2
        return         = lt_return.
    READ TABLE lt_return INTO ls_return
      WITH KEY type   = 'S'
               id     = 'RW'
               number = '605'.
    IF sy-subrc EQ 0.
      MOVE ls_return-message_v2 TO ls_awkey.
*     Commit Work durchführen
      CALL FUNCTION 'BAPI_TRANSACTION_COMMIT'
        IMPORTING
          return = ls_return.
      IF NOT ls_return IS INITIAL.
      ELSE.
        FORMAT COLOR COL_POSITIVE.
        WRITE:/ 'ID', pis_bkpf-id,
                'erfolgreich gebucht'                 ##NO_TEXT.
        FORMAT COLOR OFF.
        WRITE:/ 'Beleg'                               ##NO_TEXT,
                pis_bkpf-id,
                ls_awkey-bukrs,
                ls_awkey-belnr,
                ls_awkey-gjahr,
                'erfolgreich gebucht'                 ##NO_TEXT.
      ENDIF.
    ELSE.
      FORMAT COLOR COL_NEGATIVE.
      WRITE: /'ID', pis_bkpf-id, 'fehlerhaft'         ##NO_TEXT.
      FORMAT COLOR OFF.
      LOOP AT lt_return INTO ls_return
      WHERE type NE 'S'
      AND   type NE 'I'.
        IF ls_return-id     EQ 'RW' AND
           ls_return-number EQ '609'.
          CONTINUE.
        ENDIF.
        WRITE: / pis_bkpf-id,
                 ls_return-row,
                 ls_return-type,
                 ls_return-id,
                 ls_return-number,
                 ls_return-message.
      ENDLOOP.
    ENDIF.
* Fehler bei der Belegprüfung
  ELSE.
    FORMAT COLOR COL_NEGATIVE.
    WRITE: /'ID', pis_bkpf-id, 'fehlerhaft'           ##NO_TEXT.
```

```
    FORMAT COLOR OFF.
    LOOP AT lt_return INTO ls_return
    WHERE type NE 'S'
    AND   type NE 'I'.
      IF ls_return-id EQ 'RW' AND
       ls_return-number EQ '609'.
        CONTINUE.
      ENDIF.
      WRITE:/ pis_bkpf-id,
              ls_return-row,
              ls_return-type,
              ls_return-id,
              ls_return-number,
              ls_return-message.
    ENDLOOP.
  ENDIF.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_HEAD_LINE
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_head_line  CHANGING pcs_bkpf   TYPE gtys_bkpf
                             pes_header TYPE bapiache09.

  CLEAR pes_header.

  DATA: ls_header TYPE bapiache09.

  ls_header-bus_act           = 'RFBU'.
"Betriebswirtschaftlicher Vorgang
  ls_header-username          = sy-uname.             "Name des Benutzers
  ls_header-header_txt        = pcs_bkpf-bktxt.       "Belegkopftext
  ls_header-comp_code         = pcs_bkpf-bukrs.       "Buchungskreis
  ls_header-doc_date          = pcs_bkpf-bldat.       "Belegdatum

* Ermitteln von Geschäftsjahr und Periode aus dem Buchungsdatum
* In unserem Fall ist das Buchungsdatum leer, also nehmen wir das
* Tagesdatum. Es kann aber auch ein bestimmtes Datum übergeben werden
* Dann aber darauf achten, das die entsprechenden Perioden zum
* Buchen offen sind
  PERFORM add_data_to_bkpf CHANGING pcs_bkpf.
  IF pcs_bkpf-budat IS INITIAL.
    ls_header-pstng_date      = sy-datum.             "Buchungsdatum
  ELSE.
    ls_header-pstng_date      = pcs_bkpf-budat.     "Buchungsdatum
  ENDIF.

  ls_header-trans_date        = sy-datum.             "Umrechnungsdatum
  ls_header-fisc_year         = pcs_bkpf-gjahr.       "Geschäftsjahr
```

```
  ls_header-fis_period        = pcs_bkpf-poper.       "Geschäftsmonat
  ls_header-doc_type          = pcs_bkpf-blart.       "Belegart
  ls_header-ref_doc_no        = pcs_bkpf-xblnr.       "Referenznummer

* Achtung !!!!
* Entweder Umrechnungsdatum oder Kurs - nicht beides mitgeben !!!!!
  IF NOT pcs_bkpf-wwert IS INITIAL.
    ls_header-trans_date    = pcs_bkpf-wwert.
  ENDIF.

* Sollten zusätzliche Werte benötigt werden, hier übergeben
* oder die Kopftabelle entsprechend "aufbohren".

  pes_header = ls_header.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_DATA_TO_BKPF
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_data_to_bkpf CHANGING pcs_bkpf TYPE gtys_bkpf.

  TYPES: BEGIN OF ltys_t001,
           bukrs TYPE t001-bukrs,
           periv TYPE t001-periv,
         END OF ltys_t001,
         ltyt_t001 TYPE HASHED TABLE OF ltys_t001 WITH UNIQUE KEY bukrs.

  STATICS: lv_first_call TYPE xfeld.
  STATICS: lt_t001       TYPE ltyt_t001.
  DATA: ls_t001 TYPE ltys_t001.
  DATA: lv_buper TYPE poper.
  DATA: lv_gjahr TYPE gjahr.
  DATA: lv_budat TYPE bkpf-budat.

* Ermittlung der Geschäftsjahresvariante für alle im System vorhandenen
* Buchungskreise
  IF lv_first_call IS INITIAL.
    SELECT bukrs periv
           FROM t001
           INTO TABLE lt_t001.
    MOVE abap_true TO lv_first_call.
  ENDIF.

  READ TABLE lt_t001 INTO ls_t001
    WITH TABLE KEY bukrs = pcs_bkpf-bukrs.
  IF sy-subrc EQ 0.
    IF pcs_bkpf-budat IS INITIAL.
      MOVE sy-datum       TO lv_budat.
    ELSE.
```

```
      MOVE pcs_bkpf-budat TO lv_budat.
    ENDIF.

*   Ermittlung der Periode und des Geschäftsjahres
*   anhand des Buchungsdatum und der Geschäftsjahresvariante
*   (Stichwort: Verschobenes Geschäftsjahr)
    CALL FUNCTION 'DATE_TO_PERIOD_CONVERT'
      EXPORTING
        i_date         = lv_budat
        i_periv        = ls_t001-periv
      IMPORTING
        e_buper        = lv_buper
        e_gjahr        = lv_gjahr
      EXCEPTIONS
        input_false    = 1
        t009_notfound  = 2
        t009b_notfound = 3
        OTHERS         = 4.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
    MOVE: lv_buper TO pcs_bkpf-poper,
          lv_gjahr TO pcs_bkpf-gjahr.
  ELSE.
*   Fehlermeldung - darf nicht vorkommen !!!!
*   Buchungskreis &1 ist nicht vorhanden
    MESSAGE a215(fagl_emu) WITH pcs_bkpf-bukrs.
  ENDIF.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_CRED_LINE
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_cred_line  USING    pis_bkpf TYPE gtys_bkpf
                             pis_bseg TYPE gtys_bseg
                    CHANGING pct_cred TYPE bapiacap09_tab
                             pct_ext2 TYPE tt_bapiparex.

  DATA: ls_struc_ext2 TYPE zsc_s_soawf_badi_acc_doc.
  DATA: ls_cred  LIKE LINE OF pct_cred.
  DATA: ls_ext2  LIKE LINE OF pct_ext2.

* Kreditorenzeile
  ls_cred-itemno_acc        = pis_bseg-buzei.       "Positionsnummer des
Rechnungswesenbeleges
  ls_cred-vendor_no         = pis_bseg-gkont.       "Lieferantennummer
* Das Sachkonto auf welches gebucht wird, ist immer das im Lieferantenstamm
* eingetragene Abstimmkonto
```

```
* Gilt übrigens auch für Debitoren. Dort ist es AKONT in der KNB1
  SELECT SINGLE akont
                FROM lfb1
                INTO ls_cred-gl_account
                WHERE lifnr EQ pis_bseg-gkont
                AND   bukrs EQ pis_bkpf-bukrs.
  ls_cred-item_text         = pis_bseg-sgtxt.       "Positionstext
  ls_cred-supcountry        = pis_bseg-landl.       "Lieferland wenn nicht DE

* Schmankerl 1
* Übergabe eines nicht in der Kreditorenstruktur vorhandenen Feldes - in
diesem
* Falle das Landeszentralbankkennzeichen
  IF NOT pis_bseg-lzbkz IS INITIAL.
*   Hierzu MUSS!! man sich im DDIC eine Struktur definieren, welche
*   all die Felder enthält, welche man zusätzlich übergeben will.
*   Diese wird im Feld STRUCTURE der Struktur BAPIPAREX übergeben.
*
*   Als Maximalmenge können die Felder der Struktur ACCIT übergeben werden.
*   Die Struktur MUSS!!! immer ein Feld vom Typen
*   POSNR TYPE POSNR_ACC
*   enthalten. Dieser stellt die Verbindung von der Übergabezeile (in diesem
Falle
*   der Kreditorenzeile) zur Extension2-Zeile her.
*
*   Als weitere Felder werden dann die zu zusätzlich benötigten Felder
übergeben.
*
*   Um nun die Felder auch ins BAPI zu bringen, muß man zum BADI ACC_DOCUMENT
*   mit der SE19 eine eigene Implementierung anlegen (oder, wenn Glück hat,
ist
*   bereits eine vorhanden).
*   Hierzu MUSS!!! man den Filterwert auf BKPFF (nicht BKPF) stellen, da
dieser
*   vom BAPI gefordert bzw. geschrieben wird.
*   Als Coding kann man das Coding der Beispielimplementierung der SAP
nutzen, welche man
*   aus der Beispielimplementierung in die eigene Implementierung kopiert.
*   ACHTUNG!!!!!!
*   Nicht den Fehler machen und die ACCIT als Strukturnamen und
Übergabestruktur
*   benutzen. Die Felder VALUEPART1 bis VALUEPART4 werden concateniert und
sind
*   insgesamt nur 960 Zeichen lang !!! Da kann man mit der ACCIT und Ihren
496 Feldern!
*   auch schon mal ganz schnell ins Klo greifen, wenn die Felder erst hinter
*   den 960 Zeichen aufgerufen werden.
*   Wenn man mehr als 240 Zeichen Übergabe benötigt, ein Charakterfeld mit
960 Zeichen
*   definieren, die Struktur darauf moven, dann zerhaken und als VALUEPART1
bis 4 übergeben.
```

```
*   ACHTUNG !!!!
*   Bei Übergabe von Zahlfeldern diese vorher in Charakterwerte umwandeln und
dann übergeben,
*   sonst geht es schief. Ggf. die Implentierung entsprechend anpassen.
*   ACHTUNG !!!
*   Da die Übergabefelder CHAR-Werte sind, kann Kleinschreibung nicht
übergeben werden.
*   Danke SAP!!!
    MOVE 'Z_BADI_ACC_DOC' TO ls_ext2-structure.
    ls_struc_ext2-posnr = pis_bseg-buzei.
    ls_struc_ext2-lzbkz = pis_bseg-lzbkz.
    MOVE ls_struc_ext2 TO ls_ext2-valuepart1.
    APPEND ls_ext2 TO pct_ext2.
  ENDIF.

  APPEND ls_cred TO pct_cred.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_CURR_LINE
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_curr_line  USING    pis_bkpf   TYPE gtys_bkpf
                             pis_bseg   TYPE gtys_bseg
                             piv_tabix  TYPE sy-tabix
                    CHANGING pet_curr   TYPE bapiaccr09_tab.

  DATA: ls_curr   LIKE LINE OF pet_curr.
  DATA: lv_amount TYPE bseg-wrbtr.

  CASE pis_bseg-koart.
*   Kreditorische/Debitorische Zeilen nur einmal im Beleg
*   als Kopfzeile - daher immer Brutto!!
    WHEN 'K' OR 'D'.
      MOVE pis_bseg-brutto TO lv_amount.
    WHEN 'S'.
*     Bei Sachkontenbuchungen Unterscheidung
*     nach Kopf- und Positionszeilen
      IF piv_tabix EQ 1.
        MOVE pis_bseg-brutto TO lv_amount.
      ELSE.
        MOVE pis_bseg-netto  TO lv_amount.
      ENDIF.
  ENDCASE.

* Habenwerte immer mit -1 multiplizieren (braucht das BAPI
* um zu erkennen, was es buchen soll)
  CASE pis_bseg-shkzg.
    WHEN 'H'.
      lv_amount = lv_amount * -1.
```

```
  ENDCASE.

  ls_curr-itemno_acc        = pis_bseg-buzei.
  ls_curr-currency          = pis_bkpf-waers.
* Wenn die Währung nicht stimmen sollte, hier umwandeln
  ls_curr-currency_iso      = pis_bkpf-waers.
  ls_curr-amt_doccur        = lv_amount.
* Währungstyp
* Hier steht die Belegwährung, was der Standardfall ist.
* Sollen zusätzlich andere Typen verwendet werden, müßen die Werte angepaßt
* werden. Aber das ist ein ganz eigenes Kapitel. Werde ich ggf. bei
* Zeiten ergänzen.
  ls_curr-curr_type         = '00'.
* Umrechnungskurs für Fremdwährung
* ACHTUNG!!!
* Entweder hier KURSF ODER!!!! im Kopf WWERT mitgeben
  ls_curr-exch_rate         = pis_bkpf-kursf.

  APPEND ls_curr TO pet_curr.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_SACH_LINE
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_sach_line  USING    pis_bkpf  TYPE gtys_bkpf
                             pis_bseg  TYPE gtys_bseg
                    CHANGING pct_sach  TYPE bapiacgl09_tab
                             pct_crit  TYPE bapiackec9_tab
                             pct_value TYPE bapiackev9_tab
                             pct_ext2  TYPE tt_bapiparex
                             pev_subrc TYPE sy-subrc.

  DATA: ls_struc_ext2 TYPE zsc_s_soawf_badi_acc_doc.
  DATA: ls_ext2       LIKE LINE OF pct_ext2.
  DATA: ls_acc_gl     LIKE LINE OF pct_sach.
  DATA: lv_subrc      TYPE sy-subrc.

  CLEAR pev_subrc.

  ls_acc_gl-itemno_acc        = pis_bseg-buzei.       "Positionsnummer des
Rechnungswesenbeleges
  ls_acc_gl-gl_account        = pis_bseg-gkont.       "Sachkonto der
Hauptbuchhaltung
  ls_acc_gl-item_text         = pis_bseg-sgtxt.       "Positionstext
  ls_acc_gl-acct_key          = space.                "Vorgangsschlüssel
* Bei Kontierung Kreditor auf Anlage muß die Kontoart auf 'A'
* wie Anlage umgestellt werden.
* Ferner brauchen wir das Abstimmkonto der Anlage
* und wir müssen, über die Externen Felder, die Bewegungsart
```

```
* mitgeben. Andernfalls bekommen wir keine Anlagenbuchung
  IF NOT pis_bseg-anln1 IS INITIAL.
    ls_acc_gl-acct_type         = 'A'.                "Kontoart
    ls_acc_gl-gl_account        = pis_bseg-gkont.     "Sachkonto der
Hauptbuchhaltung
    ls_acc_gl-acct_key          = 'ANL'.              "Vorgangsschlüssel für
Anlagenbuchungen (BSCHL 70/75)
    MOVE 'ZSC_S_SOAWF_BADI_ACC_DOC' TO ls_ext2-structure.
    ls_struc_ext2-posnr = pis_bseg-buzei.
    ls_struc_ext2-anbwa = '100'.
    MOVE ls_struc_ext2 TO ls_ext2-valuepart1.
    APPEND ls_ext2 TO pct_ext2.
  ELSE.
    ls_acc_gl-acct_type         = 'S'.                "Kontoart
    ls_acc_gl-gl_account        = pis_bseg-gkont.     "Sachkonto der
Hauptbuchhaltung
  ENDIF.

  ls_acc_gl-acct_type         = pis_bseg-koart.       "Kontoart
  ls_acc_gl-tax_code          = pis_bseg-mwskz.
"Mehrwertsteuerkennzeichen
* Standardkontierungen, ggf. ergänzen.
  ls_acc_gl-costcenter        = pis_bseg-kostl.       "Kostenstelle
  ls_acc_gl-orderid           = pis_bseg-aufnr.       "Innenauftrag
  ls_acc_gl-asset_no          = pis_bseg-anln1.       "Anlagenhauptnummer
  ls_acc_gl-sub_number        = pis_bseg-anln2.       "Anlagenunternummer

  IF NOT pis_bseg-vbeln IS INITIAL.
    PERFORM add_copa_line USING    pis_bkpf
                                   pis_bseg
                          CHANGING pct_crit
                                   pct_value
                                   lv_subrc.
    IF sy-subrc NE 0.
      pev_subrc = 4.
      RETURN.
    ENDIF.
  ENDIF.

  APPEND ls_acc_gl TO pct_sach.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_COPA_LINE
*&---------------------------------------------------------------------*
*       Schmankerl CO-PA Kontierung
*       Dem BAPI ACC_DOCUMENT_POST müssen die Merkmals- sowie die
*       Wertedaten für die CO-PA Kontierung mitgegeben werden
*       Hierbei kommt es vor, das von Kundenseite oder der Schnittstellt
*       nur der Kundenauftrag zur Verfügung gestellt wird und die
*       CO-PA Kontierungen dem Programmierer "überlassen" werden.
```

```
*       Danke für die Info 
*       Was tun?
*       Zum Glück für uns gibt es den Report RFBIBL00 bzw RFBBIBL01
*       der ebenfalls mit CO-PA Daten bestückt werden kann.
*       Hierfür gibt es den Baustein RKE_CONVERT_CRITERIA_PAOBJNR
*       der aus gegegbenen Daten eine neu CO-PA Ableitung fährt
*       aus der man dann die Merkmale nachlesen und dem BAPI unterschieben
kann.
******  ACHTUNG !!!!!! ***********
*       Das neue Kontierungsobjekt wird nur dann weggeschrieben und
*       kann nachgelesen werden, wenn ein COMMIT WORK erfolgt!!!!
*       Das ist zwar unschön aber ich kenne keine andere Methode um an eine
*       komplette Ableitung zu gelangen. Wenn jemand eine kennt, wäre
*       ich dankbar, wenn diese öffentlich gemacht würde.
*       Man muß also aufpassen, das man vorher keine Daten auf der
*       Datenbank bereits geändert oder angelegt hat. Diese würde
*       hier unweigerlich commited!!!!!
*
*       Das ist aber leider nur die Hälfte der Wahrheit
*       Wir brauchen leider nicht nur die Merkmale, sondern wir müssen
*       ja auch noch die WErtfelder bestücken.
*       Hierzu siehe Routine ADD_VALUE_LINE
*
*       Und als ob das noch nicht reichen würde, müssen wir die erhaltene
*       Ableitung auch noch durch eine Routine schicken, welche die
*       alle relevanten Konvertierungsexits durchläuft, weil der
*       Ableitungsbaustein nur die externe Darstellung (RFBIBL -> BI-
Programm)
*       zurückliefert aber nicht die interne, welche das BAPI fordert.
*       Hierzu siehe Routine 'CHANGE_DATA_CONV_EXIT
*       Diese Routine kann man in abgewandelter Form auch benutzen, um
*       z.B. Datenbankdaten für einen Batch-Input aufzubereiten (z.B.
*       Arbeitsplätze oder PSP-Elemente).
*---------------------------------------------------------------------*
FORM add_copa_line  USING    pis_bkpf  TYPE gtys_bkpf
                             pis_bseg  TYPE gtys_bseg
                    CHANGING pct_crit  TYPE bapiackec9_tab
                             pct_value TYPE bapiackev9_tab
                             pev_subrc TYPE sy-subrc.

  DATA: ls_copabbseg TYPE copabbseg.
  DATA: ls_crit      LIKE LINE OF pct_crit.
  DATA: lv_paobjnr   TYPE rkeobjnr.
  DATA: lv_subrc     TYPE sy-subrc.
  DATA: lt_copadata  TYPE gtyt_copadata.
  DATA: ls_copadata  LIKE LINE OF lt_copadata.
  DATA: ls_org_copa  TYPE gtys_org_copa.

  CLEAR pev_subrc.

  IF pis_bseg-vbeln IS INITIAL.
```

```
    RETURN.
  ENDIF.

* Zuordnung BURKS zu KOKRS/ERKRS ermitteln
  PERFORM get_orgdata_copa USING    pis_bkpf-bukrs
                           CHANGING ls_org_copa.

* Wertfeldtabelle füllen
  PERFORM add_value_line USING    pis_bkpf
                                  pis_bseg
                                  ls_org_copa-kokrs
                                  ls_org_copa-erkrs
                         CHANGING pct_value
                                  lv_subrc.
  IF lv_subrc NE 0.
    pev_subrc = 4.
    RETURN.
  ENDIF.

* Übergabe für Baustein füttern
* Buchungskreis MUSS zwingend gefüllt werden
  MOVE: pis_bkpf-bukrs TO ls_copabbseg-rke_bukrs.
  MOVE: pis_bseg-vbeln TO ls_copabbseg-rke_kaufn.
  MOVE: pis_bseg-posnr TO ls_copabbseg-rke_kdpos.

* Neue Ableitung
  CALL FUNCTION 'RKE_CONVERT_CRITERIA_PAOBJNR'
    EXPORTING
      is_copabbseg     = ls_copabbseg
*     i_date           =
    IMPORTING
      e_paobjnr        = lv_paobjnr
    EXCEPTIONS
      no_bukrs_found   = 1
      no_erkrs_found   = 2
      error_criterion  = 3
      error_derivation = 4
      OTHERS           = 5.
  IF sy-subrc <> 0.
*   Fehler bei Ableitung CO-PA Merkmale für ID & Buzei &.
    pev_subrc = 4.
    RETURN.
  ENDIF.

* Commit Work durchführen, damit die Ergebnisobjektnummer auf
* der Datenbank vorliegt
  COMMIT WORK  AND WAIT.
  IF sy-subrc NE 0.
*   Fehler bei der Fortschreibung PAOBJNR für ID & Buzei &.
    pev_subrc = 4.
    RETURN.
```

```
  ENDIF.

* Nachlesen der neuen Ableitung
  CALL FUNCTION 'RKE_CONVERT_PAOBJNR_COPADATA'
    EXPORTING
      bukrs          = pis_bkpf-bukrs
      kokrs          = ls_org_copa-kokrs
      paobjnr        = lv_paobjnr
    TABLES
      i_copadata     = lt_copadata
    EXCEPTIONS
      no_erkrs_found = 1
      paobjnr_wrong  = 2
      OTHERS         = 3.
  IF sy-subrc <> 0.
*   Fehler bei Ermittlung Wertfelder aus PAOBJNR für ID & Buzei &.
    pev_subrc = 4.
    RETURN.
  ENDIF.

* Umstellung der externen Darstellung der Merkmale auf
* interne Werte (durchlaufen der Konvertierungsexits)
  IF NOT lt_copadata IS INITIAL.
    PERFORM change_data_conv_exit USING    ls_org_copa-erkrs
                                  CHANGING lt_copadata.
  ENDIF.

* Anhängen der Merkmale an die Merkmalstabelle für das BAPI
  LOOP AT lt_copadata INTO ls_copadata.
    IF NOT ls_copadata-fval IS INITIAL.
      CLEAR ls_crit.

      MOVE: pis_bseg-buzei   TO ls_crit-itemno_acc,
            ls_copadata-fnam TO ls_crit-fieldname,
            ls_copadata-fval TO ls_crit-character.
      APPEND ls_crit TO pct_crit.
    ENDIF.
  ENDLOOP.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  GET_ORGDATA_COPA
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM get_orgdata_copa  USING    piv_bukrs    TYPE bukrs
                       CHANGING pes_org_copa TYPE gtys_org_copa.

  CLEAR pes_org_copa.

  DATA: ls_org_copa    LIKE pes_org_copa.
```

```
  DATA: lv_kokrs       TYPE kokrs.
  DATA: lv_erkrs       TYPE erkrs.
  STATICS: lt_org_copa TYPE gtyt_org_copa_hash.

  READ TABLE lt_org_copa INTO ls_org_copa
    WITH TABLE KEY bukrs = piv_bukrs.
  IF sy-subrc EQ 0.
    pes_org_copa = ls_org_copa.
    RETURN.
  ENDIF.

  CALL FUNCTION 'KOKRS_GET_FROM_BUKRS'
    EXPORTING
      i_bukrs        = piv_bukrs
    IMPORTING
      e_kokrs        = lv_kokrs
    EXCEPTIONS
      no_kokrs_found = 1
      OTHERS         = 2.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

  CALL FUNCTION 'COPA_ERKRS_FIND'
    EXPORTING
      bukrs              = piv_bukrs
    IMPORTING
      erkrs              = lv_erkrs
    EXCEPTIONS
      error_kokrs_find   = 1
      kokrs_wrong        = 2
      no_erkrs_defined   = 3
      no_erkrs_for_kokrs = 4
      OTHERS             = 5.
  IF sy-subrc <> 0.
    MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
            WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
  ENDIF.

  MOVE: piv_bukrs TO ls_org_copa-bukrs,
        lv_kokrs  TO ls_org_copa-kokrs,
        lv_erkrs  TO ls_org_copa-erkrs.
  INSERT ls_org_copa INTO TABLE lt_org_copa.
  pes_org_copa = ls_org_copa.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_VALUE_LINE
*&---------------------------------------------------------------------*
*       Das BAPI fordert von uns nicht nur die Merkmale sondern wir
```

```
*       müssen auch die Wertfelder bestücken - und zwar die richtigen!!!
*
*       Die erreichen wir, indem wir den Baustein
*       COPA_GET_SETTLEMENT_STRUCTURE benutzen. Dieser gibt uns
*       das Ergebnisschmema, die Zuordnung sowie den Inhalt der Tabelle
*       TKB9F zurück.
*       Hierzu muß man wissen, das für das Konto, auf welches gebucht
*       werden soll, im Customizing hinterlegt sein MUSS, auf welches
*       Wertfeld die Beträge laufen sollen.
*       Also lesen wir zunächst mit dem Baustein Schema und Zuordnung
*       und dann damit die Tabelle.
*       Nicht wundern, das hier bestimmte Werte hart verdrahtet sind.
*       Das ist so und muß auch so gecustomized sein.
*       Es MUSS ein Wertfeld sein (MWKZ = 1) und das Fix-/Variabel Kz.
*       MUSS auf Gesamtwert (FVKZ = 3) stehen. Ist das nicht der Fall,
*       ist dies ein Fehler!!!!
*       Hat man vor mehrere Belege für unterschiedliche Währungs-
*       sichten zu erzeugen, so muß man dies auch hier tun. Im diesem
*       Beispiel nehmen wir nur die Belegwährung
*---------------------------------------------------------------------*
FORM add_value_line USING    pis_bkpf  TYPE gtys_bkpf
                             pis_bseg  TYPE gtys_bseg
                             piv_kokrs TYPE kokrs
                             piv_erkrs TYPE erkrs
                    CHANGING pct_value TYPE bapiackev9_tab
                             pev_subrc TYPE sy-subrc.

  DATA: lt_9f TYPE STANDARD TABLE OF tkb9f.
  DATA: ls_9f LIKE LINE OF lt_9f.
  DATA: lv_ersch TYPE tkb9f-ersch.
  DATA: lv_erzuo TYPE tkb9f-erzuo.

  DATA: ls_value LIKE LINE OF pct_value.
  DATA: BEGIN OF ls_wertfeld,
          hkont  TYPE bseg-hkont,
          wrtfld TYPE fieldname,
        END OF ls_wertfeld.
  STATICS: lt_wertfeld LIKE HASHED TABLE OF ls_wertfeld
                       WITH UNIQUE KEY hkont.

  CLEAR pev_subrc.

* Statische Wertfeldtabelle lesen (damit es schneller geht)
  READ TABLE lt_wertfeld INTO ls_wertfeld
    WITH TABLE KEY hkont = pis_bseg-gkont.
  IF sy-subrc EQ 0.
    MOVE: pis_bseg-buzei     TO ls_value-itemno_acc,
          ls_wertfeld-wrtfld TO ls_value-fieldname,
          '00'               TO ls_value-curr_type,
          pis_bkpf-waers     TO ls_value-currency,
          pis_bkpf-waers     TO ls_value-currency_iso,
```

```
          pis_bseg-netto     TO ls_value-amt_valcom.
    APPEND ls_value TO pct_value.
    RETURN.
  ENDIF.

* Ansonsten Ergebnisschema und -zuordnung ermitteln
  CALL FUNCTION 'COPA_GET_SETTLEMENT_STRUCTURE'
    EXPORTING
      i_erkrs             = piv_erkrs
      i_vrgng             = 'RFBU'
      i_hkont             = pis_bseg-gkont
      i_kokrs             = piv_kokrs
    IMPORTING
      e_ersch             = lv_ersch
      e_erzuo             = lv_erzuo
    TABLES
      t_tkb9f             = lt_9f
    EXCEPTIONS
      incomplete_structure = 1
      error_structure      = 2
      OTHERS               = 3.
  IF sy-subrc <> 0.
*   Wertfeld für Bukrs & Konto & kann nicht ermittelt werden (&).
    pev_subrc = 4.
    RETURN.
  ENDIF.

* TKB9F mit den ermittelten Werten lesen
  READ TABLE lt_9f INTO ls_9f WITH KEY ersch = lv_ersch
                                       erzuo = lv_erzuo
                                       erkrs = piv_erkrs
                                       mwkz  = '1'
                                       fvkz  = '3'.
  IF sy-subrc NE 0.
*   Ist hier nichts zu finden, ist dies ein Fehler und der Beleg
*   würde sowieso aufbrummen - also Schluß
*   Wertfeld für Bukrs & Konto & kann nicht ermittelt werden (&)
    pev_subrc = 4.
    RETURN.
  ENDIF.

* Value Rückgabetabelle füllen
  MOVE: pis_bseg-buzei TO ls_value-itemno_acc,
        ls_9f-wrtfld   TO ls_value-fieldname,
        '00'           TO ls_value-curr_type,
        pis_bkpf-waers TO ls_value-currency,
        pis_bkpf-waers TO ls_value-currency_iso,
        pis_bseg-netto TO ls_value-amt_valcom.
  APPEND ls_value TO pct_value.

* Und zum Schluß noch die statische Wertfeldtabelle erweitern
```

```
  MOVE: pis_bseg-gkont TO ls_wertfeld-hkont,
        ls_9f-wrtfld   TO ls_wertfeld-wrtfld.
  INSERT ls_wertfeld INTO TABLE lt_wertfeld.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  CHANGE_DATA_CONV_EXIT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM change_data_conv_exit  USING    piv_erkrs TYPE erkrs
                            CHANGING pct_data  TYPE gtyt_copadata.

  DATA: BEGIN OF ls_ddic,
          tabname   TYPE dfies-tabname,
          fieldname TYPE dfies-fieldname,
          rollname  TYPE dfies-rollname,
          convexit  TYPE dfies-convexit,
          funcname  TYPE tfdir-funcname,
          reference TYPE REF TO data,
        END OF ls_ddic,
        lt_ddic LIKE HASHED TABLE OF ls_ddic
                WITH UNIQUE KEY tabname fieldname.
  DATA: lt_ddic_info TYPE ddfields.
  DATA: ls_ddic_info LIKE LINE OF lt_ddic_info.
  DATA: lv_tabname   TYPE tabname.
  DATA: lv_fieldname TYPE fieldname.
  DATA: lv_value     TYPE REF TO data.

  FIELD-SYMBOLS: <ls_data>  LIKE LINE OF pct_data,
                 <lv_value> TYPE any.


  CONCATENATE 'CE0' piv_erkrs INTO lv_tabname.

  LOOP AT pct_data ASSIGNING <ls_data>.
    CHECK NOT <ls_data>-fval IS INITIAL.
    CLEAR: lv_tabname, ls_ddic.

    CONCATENATE 'CE0' piv_erkrs INTO lv_tabname.
    CONDENSE lv_tabname.

    MOVE <ls_data>-fnam TO lv_fieldname.
    READ TABLE lt_ddic INTO ls_ddic
      WITH TABLE KEY tabname   = lv_tabname
                     fieldname = lv_fieldname.
    IF sy-subrc EQ 0.
      IF ls_ddic-funcname IS INITIAL.
        CONTINUE.
      ENDIF.
    ELSE.
```

```
      CALL FUNCTION 'DDIF_FIELDINFO_GET'
        EXPORTING
          tabname        = lv_tabname
          fieldname      = lv_fieldname
        TABLES
          dfies_tab      = lt_ddic_info
        EXCEPTIONS
          not_found      = 1
          internal_error = 2
          OTHERS         = 3.
      IF sy-subrc <> 0.
        MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
                WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
      ENDIF.
      CLEAR ls_ddic.
      READ TABLE lt_ddic_info INTO ls_ddic_info INDEX 1.
      MOVE-CORRESPONDING ls_ddic_info TO ls_ddic.
      IF NOT ls_ddic-convexit IS INITIAL.
        CONCATENATE 'CONVERSION_EXIT_' ls_ddic_info-convexit '_INPUT'
                    INTO ls_ddic-funcname.
        CONDENSE ls_ddic-funcname.
        CREATE DATA lv_value TYPE (ls_ddic-rollname).
        MOVE lv_value TO ls_ddic-reference.
      ENDIF.
      INSERT ls_ddic INTO TABLE lt_ddic.
    ENDIF.
    IF NOT ls_ddic-funcname IS INITIAL.
      ASSIGN ls_ddic-reference->* TO <lv_value>.
      MOVE <ls_data>-fval TO <lv_value>.
      CALL FUNCTION ls_ddic-funcname
        EXPORTING
          input  = <lv_value>
        IMPORTING
          output = <lv_value>
        EXCEPTIONS
          OTHERS = 0.
      CLEAR <ls_data>-fval.
      MOVE <lv_value> TO <ls_data>-fval.
    ENDIF.
  ENDLOOP.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  GET_TAX
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM get_tax  USING    pis_bkpf  TYPE gtys_bkpf
                       pis_bseg  TYPE gtys_bseg
                       piv_tabix TYPE sy-tabix
              CHANGING pct_tax   TYPE gtyt_tax.
```

```
  DATA: lv_amount TYPE bseg-wrbtr.
  DATA: lt_mwdat  TYPE STANDARD TABLE OF rtax1u15.
  DATA: ls_mwdat  LIKE LINE OF lt_mwdat.
  DATA: ls_tax    TYPE gtys_tax.
  DATA: lt_tax    TYPE gtyt_tax.

* Nur Steuer auf Positions-, nicht auf Kopfzeilen
  IF piv_tabix EQ 1.
    RETURN.
  ENDIF.

* Steuerkennzeichen muß mitgegeben sein
  IF NOT pis_bseg-mwskz IS INITIAL.
*   ACHTUNG
*   Der Baustein ermittelt die Steuer abhängig vom
*   übergebenen Wert. Da wir mit positiven Werten arbeiten
*   müssen wir hier natürlich auch ggf. das Vorzeichen drehen
*   sonst geht es schief.
    lv_amount = pis_bseg-netto.
    IF pis_bseg-shkzg EQ 'H'.
      lv_amount = pis_bseg-netto * -1.
    ENDIF.

*   Baustein für Ermittlung der Steuer aus dem Nettowert
*   Anmerkung: Gibt es auch für Bruttowerte
*              CALCULATE_TAX_FROM_GROSSAMOUNT
    CALL FUNCTION 'CALCULATE_TAX_FROM_NET_AMOUNT'
      EXPORTING
        i_bukrs           = pis_bkpf-bukrs
        i_mwskz           = pis_bseg-mwskz
        i_waers           = pis_bkpf-waers
        i_wrbtr           = lv_amount
      TABLES
        t_mwdat           = lt_mwdat
      EXCEPTIONS
        bukrs_not_found   = 1
        country_not_found = 2
        mwskz_not_defined = 3
        mwskz_not_valid   = 4
        ktosl_not_found   = 5
        kalsm_not_found   = 6
        parameter_error   = 7
        knumh_not_found   = 8
        kschl_not_found   = 9
        unknown_error     = 10
        account_not_found = 11
        txjcd_not_valid   = 12
        OTHERS            = 13.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
              WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
```

```
    ENDIF.
    LOOP AT lt_mwdat INTO ls_mwdat.
      MOVE-CORRESPONDING ls_mwdat TO ls_tax                ##ENH_OK.
*     Wir brauchen für die Steuerzeilen aber auch noch das Steuerkenn-
*     zeichen. Daher umkopieren auf eigene Struktur
      MOVE: pis_bseg-mwskz TO ls_tax-mwskz.
      APPEND ls_tax TO lt_tax.
    ENDLOOP.
    IF NOT lt_mwdat IS INITIAL.
      APPEND LINES OF lt_tax TO pct_tax.
    ENDIF.
  ENDIF.

ENDFORM.
*&---------------------------------------------------------------------*
*&      Form  ADD_TAX_LINES
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
FORM add_tax_lines  USING    pit_tax    TYPE gtyt_tax
                             pis_bkpf   TYPE gtys_bkpf
                             piv_buzei  TYPE bseg-buzei
                    CHANGING pct_acc_tx TYPE bapiactx09_tab
                             pct_cur_am TYPE bapiaccr09_tab.

  DATA: ls_tax    LIKE LINE OF pit_tax.
  DATA: ls_acc_tx LIKE LINE OF pct_acc_tx.
  DATA: ls_cur_am LIKE LINE OF pct_cur_am.
  DATA: lv_buzei  TYPE bseg-buzei.

  DATA: BEGIN OF ls_collect,
          mwskz TYPE mwskz,
          kschl TYPE kschl,
          ktosl TYPE ktosl,
          hkont TYPE hkont,
          wmwst TYPE wmwst,
          kawrt TYPE kawrt,
        END OF ls_collect.
  DATA: lt_collect LIKE HASHED TABLE OF ls_collect
                   WITH UNIQUE KEY mwskz ktosl kschl hkont.

* Steuern müssen wir schon ermittelt haben
  IF NOT pit_tax IS INITIAL.

*   Über den Collect werden Steuer und Steuerbasis summiert
*   und ggf. verdichtet
    LOOP AT pit_tax INTO ls_tax.
      MOVE-CORRESPONDING ls_tax TO ls_collect.
      COLLECT ls_collect INTO lt_collect.
    ENDLOOP.
```

```
*   Wir kopieren die letzte Zeilennummer
    lv_buzei = piv_buzei.

*   Und los
    LOOP AT lt_collect INTO ls_collect.
*     Eins auf die Buchungszeile da wir eine neue
*     Zeile erzuegen
      ADD 1 TO lv_buzei.
      CLEAR ls_acc_tx.
*     Übertragung der Steuerdaten
      MOVE: lv_buzei         TO ls_acc_tx-itemno_acc,
            ls_collect-hkont TO ls_acc_tx-gl_account,
            ls_collect-kschl TO ls_acc_tx-cond_key,
            ls_collect-ktosl TO ls_acc_tx-acct_key,
            ls_collect-mwskz TO ls_acc_tx-tax_code.
*     Und anhängen
      APPEND ls_acc_tx TO pct_acc_tx.

*     Da dies nur die Parameter für die Steuerzeile sind,
*     müssen wir auch noch eine Wertezeile erzeugen
      CLEAR ls_cur_am.
*     Gleiche Zeile wie Steuerzeile (Bezug!!!)
      MOVE: lv_buzei         TO ls_cur_am-itemno_acc,
            pis_bkpf-waers   TO ls_cur_am-currency,
            pis_bkpf-waers   TO ls_cur_am-currency_iso,
*           Ermittelter, summierter Steuerwert für Steuerkennzeichen
            ls_collect-wmwst TO ls_cur_am-amt_doccur,
*           Ermittelter, summierter Basiswert für Steuerkennzeichen
            ls_collect-kawrt TO ls_cur_am-amt_base,
            '00'             TO ls_cur_am-curr_type,
*           Umrechnungskurs für Fremdwährung
*           ACHTUNG!!!
*           Entweder hier KURSF ODER!!!! im Kopf WWERT mitgeben!!!
            pis_bkpf-kursf   TO ls_cur_am-exch_rate.
      APPEND ls_cur_am TO pct_cur_am.
    ENDLOOP.
  ENDIF.

ENDFORM.
```