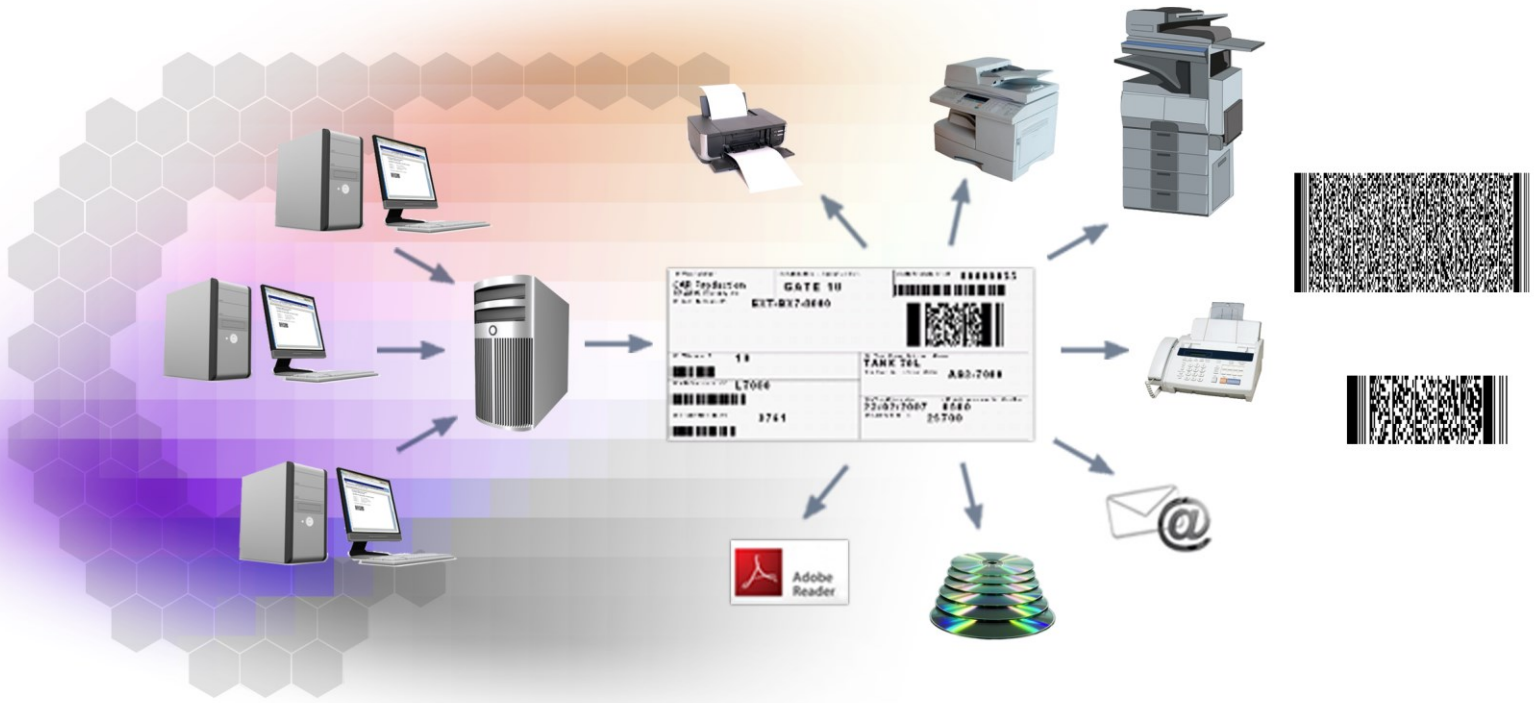
Suchy MIPS

www.suchymips.de



RBarc/PDF417

For SAP® Systems

Version 12

Reference Manual

Contents

1. Notice

The information contained in this document is subject to change without notice.

Suchy mips makes no representations of warranties either express or implied, with respect to this publication and accompanying software and specifically disclaims any implied warranties with regard to its sales potential or fitness for any particular purpose.

Suchy MIPS
Schichtlstrs. 32A

81929 Muenchen
Germany

Phone: +49 (0) 89 - 944 19 77 - 0
Fax: +49 (0) 89 - 944 19 77 - 13
E-mail: info@suchymips.de
Internet: www.suchymips.de

2. Restrictions to the DEMO version

This manual describes as well the DEMO version as the FULL version of RBarc/PDF417©
The following restrictions must be considered if you have the DEMO version of RBarc/PDF417©:

Error Correction Levels 0, and 1 only are supported.
The text " PDF417 from Suchy mips" will be encoded whith each label.

Example of a DEMO label:

3. Introduction

RBarc/PDF417© is an ABAP generator for the 2D bar code PDF417. Barcodes generated with RBarc/ PDF417© are device independent bitamps, thus, they can be outputed on any device with graphic capability like printers and faxes. Bar codes generated with RBarc/ PDF417© are visible in the preview and will remain during archiving, mailing or converting SAP documents into PDF documents.

4. Requirements

For using RBarc/PDF417© an installed and working SAP® R/3 ver 3.x or higher is required.

Beginning with the version 2006, Rbarc/PDF417© doesn't requires PLC Printers. Barcodes generated by Rbarc/PDF417© can be printed on any printer with graphic capabilities as well as faxed, mailed or archived.

Note: RBarc/PDF417© is dedicated to System Administrators and Developers only. Consequently good knowledge in R/3 Basis and ABAP/4 Programming is requirement. If SAPscript, Smart Forms or AdobeForms forms have to be adapted, a basic knowledge about SAPscript and/or Smarforms is also required.
After RBarc/PDF417© has been installed in R/3 and appropriate forms and reports completed for printing bar codes with RBarc/PDF417© user starting printing forms or reports will not be confronted with topics of this manual in any way.

Note: User, who print documents including barcodes generated by RBarc+ must have ollowing permissions for the object **S_BDS_DS:**

ACTVT = 01,02,03 and 06
CLASSNAME = DEVC_STXD_BITMAP
CLASSTYPE = OT

The appropriate settings may be performed with the transaction "**PFCG**".
The User Role must include the transaction **SE78** and the Authorisation for **BC-SRV-KPR-BDS** (Technical Name **S_BDS_DS**).
S_BDS_DS is assigned to class "**Basis-Central functions**".

If the permission is missing, no barcode will appear on the output or generated barcodes will not be deleted from the system.

---

5. Compatibility with earlier Versions

RBarc/PDF417© v12 is compatibel with the earlier versions 2007 of RBarc/PDF417© .

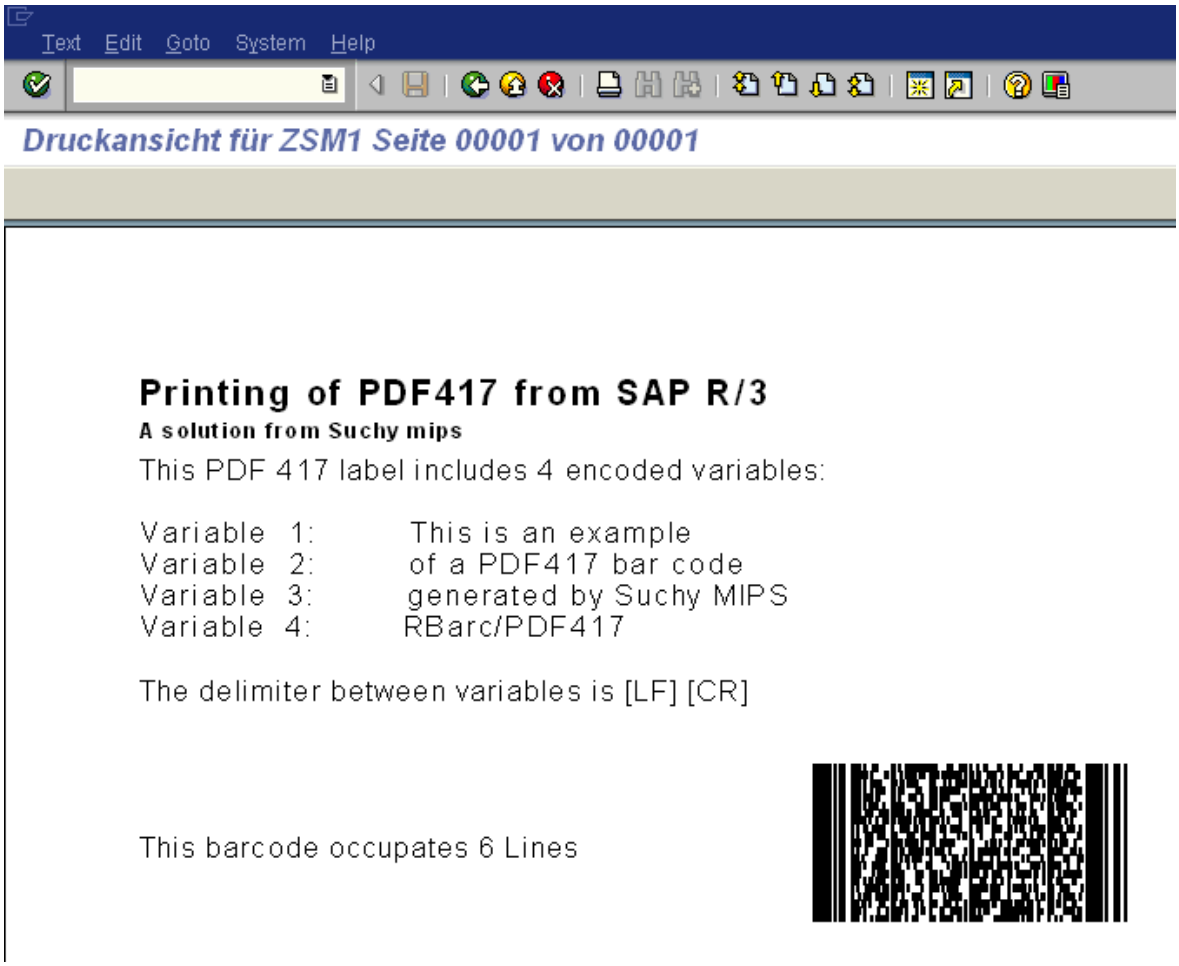
User may use new features and parameters without changing older settings in forms or bar code setting programs.

6. General information about 2D Bar Code PDF 417

PDF 417 is a 2D (2-dimensional) high capacity bar code, which can encode up to about 2000 characters in one label. For SAP R/3 users it means, that they can encode many of data base fields in only one label instead of using more labels of 1D bar codes (like bar code 39 or 2/5 interleaved). Above that, this bar code provides the user with a high security level, which allows to read even partially damaged labels. The bar code PDF417 displayed below encodes encodes following values:

[this is an exaple]
[of a PDF417 bar code]
[generated by Suchy MIPS]
[RBarc/PDF417]

The delimiter <LF><CR> between the variables has also been encoded:



7. Installation

RBarc/PDF417© includes different routines for printing bar code either with **SapScript** or **SmartForms**. Both routines can be installed on the system independently of each other.

7.1. Installation steps related to printing PDF417 with SAPScript

Installation steps:

1. Login to R/3 system. You must be registered as a developer in OSS or MLP.

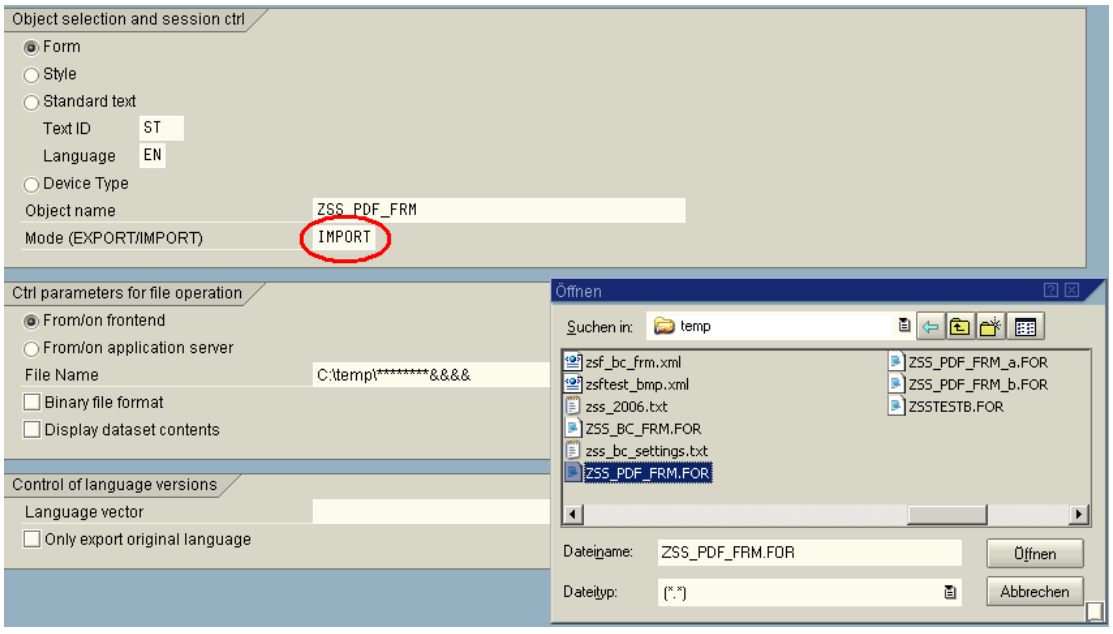
2. Create a package (development class) which will be used to store programs and includes for RBarc/PDF417© (e.g. ZPDF).

3. Create 2 new programs in the RBarc/PDF417© development class. Call them "ZSS_PDF_PRINT" and "ZSS_PDF_SET". Insert the source code for these programs from the appropriate files from the CD. These files have the extention .PRG, e.g. ZSS_PDF_SET.PRG for the program ZSS_PDF_SET. The source code can be inserted by using of "copy and paste" as well as by using of the function "upload".

4. Create 1 new include in the RBarc/PDF417© development class. Call it "ZSS_RBARCPDF". Insert the source code for this program from the installation directory. The name of the file is the "ZSS_RBARCPDF.INC".

5. Start the SAP standard program RSTXSCRP to upload the example form ZSS_PDF_FRM from the installation CD. The File name for this form is ZSS_PDF_FRM.FOR.

Note: The program RSTXSCRP must be executed in the modus "IMPORT" to upload the form to the R/3 server.



7.2. Installation steps related to printing PDF417 with SmartForms

Installation steps:

1. Login to R/3 system. You must be registered as a developer in OSS or MLP.

2. If not yet done, create a development class which will be used to store programs and includes for RBarc/PDF417© (e.g. ZPDF).

3. Create a new program in the RBarc/PDF417© development class called “ZSF_PDF_SET". Insert the source code for this program from the file “ZSF_PDF_SET.PRG” from the installation directory.

4. Create 1 new include in the RBarc/PDF417© development class. Call it “ZSF_RBARCPDF”. Insert the source code for this program from the installation installation directory. The name of the file is the “ZSF_RBARCPDF.INC”. The source code can be inserted by using of “copy and paste” as well as by using of the function “upload”.

5. Upload the Smartforms form ZSF_PDF_FORM from the installation directory: start the transaction “smartforms” and select from the menu bar “tools/upload form”. Enter the form name “ZSF_PDF_FRM” and follow the instructions to upload the form “ZSF_PDF_FORM.XML” from the Installation directory.

7.3. Installation steps related to printing PDF417 with AdobeForms

Installation steps:

1. Login to R/3 system. You must be registered as a developer in OSS or MLP.

2. If not yet done, create a development class which will be used to store programs and includes for RBarc/PDF417© (e.g. ZPDF).

3. Create a new program in the RBarc/PDF417© development class called “ZAF_PDF_SET". Insert the source code for this program from the file “ZAF_PDF_SET.PRG” from the installation directory.

4. Create 1 new include in the RBarc/PDF417© development class. Call it “ZAF_RBARCPDF”. Insert the source code for this program from the installation CD. The name of the file is the “ZAF_RBARCPDF.INC”. The source code can be inserted by using of “copy and paste” as well as by using of the function “upload”.

5. Upload theAdobeForms interface ZAF_PDF_INTERFACE from the installation directory: start the transaction “SFP” and select from the menu bar “tools/upload form”. Enter the interface name “ZAF_PDF_INTERFACE” and follow the instructions to upload the interface “ZAF_PDF_INTERFACE.XML” from the Installation directory.

6. Upload the AdobeForms form ZAF_PDF_FORM from the installation directory: start the transaction “SFP” and select from the menu bar “tools/upload form”. Enter the form name “ZAF_PDF_FORM” and follow the instructions to upload the form “ZAF_PDF_FORM.XML” from the Installation directory.

8. Testing the installation of RBarc/PDF417©

8.1. Test with SAPScript

Execute the program ZSS_PDF_PRINT. This program prints the form ZSS_PDF_FRM with an example of barcode PDF 417.

8.2. Test with SmartForms

Load the Smartforms form ZSF_PDF_FORM and proceede the forms test.

8.3. Test with AdobeForms

Execute the program ZAF_PDF_PRINT. This program prints the form ZAF_PDF_FORM with an example of barcode PDF 417.

9. Printing PDF417 labels with RBarc/PDF417©

9.1. Collecting main information about the estimated PDF417 label

Before you will print PDF417 from SAP R/3 you have to collect some information. Due to this information you should program the procedure call and parameter settings for the PDF417 label. To collect information needed for printing PDF417, try to answer following questions:

- how many and which variables do you want to encode?
- do you want delimiters between variables? If yes: will it always be the same delimiter or are you going to use different delimiters?
- which security level do you want to use?
- how large should the label be (approximately) ?
- which shape should the label have (relation between height and width)?

Due to this information you will set main parameters for PDF417 label printing, which is described below.

9.2. Preparation of the print environment

Before you try to print a PDF417 label with RBarc/PDF417©, prepare all that you need to print a given form. Usually it is an ABAP report collecting data from database and using a SAPScript or SmartForms form to print it. Try to print the form before you insert a PDF417 label into it. If the form was printed correctly, you can go to the next step.

Inserting a PDF417 label requires 2 additional steps:

Step 1: Inserting statements into the SAPScript or SmartForms form, which pass all variables for encoding to an additional ABAP program.

Step 2: Creating of an additional ABAP program which:
- collects all variables and delimiters between variables in an internal table
- calls the RBarc/PDF417© procedure which generates the bar code label

Step 3: Inserting statements into the SAPScript or SmartForms form, which includes the dynamically generated bar code into the form and deletes it from the system after it was used.

9.3. Printing PDF417 with SAPScript.

Note: for the following examples we suppose 4 variables to be encoded: ***bseg-kunnr***, ***bseg-blart***, ***bseg-lifnr*** and ***bseg-saknr,*** but up to recent requirement much more variables may be encoded.

Line	SAPScript
1	/: DEFINE &SMPDFVAR1& = &BSEG-KUNNR&
2	/: DEFINE &SMPDFVAR2& = &BSEG-BLART&
3	/: DEFINE &SMPDFVAR3& = &BSEG-LIFNR&
4	/: DEFINE &SMPDFVAR4& = &BSEG-SAKNR&
5	/: DEFINE &SMPDFIDENT& = 'PDF01'
7	/: DEFINE &BC_METHOD& = 'OTF'
8	/: DEFINE &XPOS& = '100'
9	/: DEFINE &YPOS& = '99.00'
10	/: PERFORM GEN_PDF417 IN PROGRAM ZSS_PDF_SET
11	/: USING &SMPDFVAR1&
12	/: USING &SMPDFVAR2&
13	/: USING &SMPDFVAR3&
14	/: USING &SMPDFVAR4&
15	/: USING &SMPDFIDENT&
16	/: USING &BC_METHOD&
17	/: USING &XPOS&
18	/: USING &YPOS&
19	/: CHANGING &SMPDFNAME&
20	/: CHANGING &LINES&
21	/: ENDPERFORM
22	/: INCLUDE &SMPDFNAME& OBJECT TEXT ID ADRS LANGUAGE &SY-LANGU&
23	/: PERFORM DEL_PDF IN PROGRAM ZSS_PDF_SET
24	/: USING &SMPDFNAME&
25	/: USING &BC_METHOD&
26	/: ENDPERFORM

Comments:

Line 1-4: Define variables for encoding with PDF417. Names of variables SMPDFVAR1... are examples only. You can use your own variable names. The only condition is, the name should correspond to the names used in the program with parameter settings (ZPDF_SET in our example**). The length of each variable is limited by SAPScript to 80 characters**

Line 5: Defines an identifier for this bar code label. If you print more than one bar code on the same form, be sure, the identifier is different for each bar code.

Line 6: Performs the procedure (form) SMPDF in the ABAP program ZPDF_SET. The name for the program and procedure are examples only, you can use your own program name and procedure (form) name.

Line 7: Defines the method used for generating the bar code. “OTF” generates the bar code as text which can be moved in horizontal direction inside the window with the using parameter XPOS.

Line 8: Defines the horizontal position of the bar code label (relative to the left border of the windows. The units for this value shell be defined in the program ZSS_PDF_SET in the parameter “UNIT” and can be “MM”, “CM” or “INCH”.

Line 9:	Defines the vertical position of the bar code on the paper. This is an absolut position and not a relative one, thus the bar code will appear always on the same place of the document.
Line 10-14:	pass defined variables to the program with PDF417 parameter settings (ZSS_PDF_SET).
Line 15:	Passes the bar code identifier to the program with PDF417 parameter settings (ZSS_PDF_SET).
Line 16:	Passes the method for bar code generating to the program with PDF417 parameter settings (ZSS_PDF_SET).
Line 17:	Passes the horizontal position of the bar code (relative to the left border of the used window) to the program with PDF417 parameter settings (ZSS_PDF_SET).
Line 18:	Passes the vertical position of the bar code (absolut position on the document) to the program with PDF417 parameter settings (ZSS_PDF_SET).
Line 19:	RBarc/PDF417© generates the bar code label and stores it as a standard text or bitmap. The name of the generated object will be passed back to the form in this changing variable.
Line 20:	This value returns the number of lines occupated by the bar code.
Line 21:	End of the routine perform.
Line 22:	Includes the generated bar code bitmap.
Line 23 – 26:	Deletes the bar code from the system.

9.4. Creating an ABAP program for PDF417 parameter settings (SAPScript)

The form example above has defined variables for encoding with PDF417 and passed this information to the procedure (form) SMPDF in the ABAP Program ZPDF_SET. Here is an example of such a program with comments:

Report ZSS_PDF_SET

Data declaration for this program:

```
DATA: PDF(256), ECLEVEL TYPE I, ROWS TYPE I, COLUMNS TYPE I,
      ASPRATIO TYPE F, X_DIM TYPE F, Y_DIM TYPE F, TRUNCATE TYPE I,
      XPOS TYPE F, YPOS TYPE F, ROT TYPE I, POSTYP TYPE I,
      TEXT_ID(4), BCVALUE01(40), BARIDENT(10), UNIT(4),
      SMVAR1(31), SMVAR2(31), SMVAR3(31), SMVAR_N(70),
      SMCHG1type string, SMCHG2(31), SMCHG3(31), NO_ERR_PRINT TYPE I,
     PDFVALUE01(80), PDFIDENT(10), GRAPHIC TYPE I.
```

Begin of the form SMPDF. This form uses a table in_tab which includes the structure ITCSY, which is a standard interface between SAPScript and and a ABAP report.

```
FORM SMPDF TABLES IN_TAB STRUCTURE ITCSY
                  OUT_TAB STRUCTURE ITCSY.
```

Next statement calls a procedure from RBarc/PDF417© which initializes all tables and variables and is obligatory.

```
   PERFORM INIT_PDFDATA.
```

The following part reads all variables for encoding from the table IN_TAB.
If you want delimiters between variables, you can define up to 5 delimiters. Each character you can insert via keyboard is allowed as a delimiter. Also following symbols are allowed: 'NUL', 'SOH', 'STX', 'ETX', 'EOT', 'ENQ', 'ACK', 'BEL', 'BS', 'HT', 'LF', 'VT', 'FF', 'CR', 'SO', 'SI', 'DLE', 'DC1', 'DC2', 'DC3', 'DC4', 'NAK', 'SYN', 'ETB', 'CAN', 'EM', 'SUB', 'ESC', 'FS', 'GS', 'RS', 'US' *(ASCII characters 0 - 31) and ASCII Char 39 (signle quote). If you don't want to have delimiters between variables, leave them empty.*

Note: *only one character or symbol per delimiter is allowed. The definition like* ***PERFORM COLLECT_PDF USING PDFVALUE01 '}}>' " " " "*** *is invalid!*

Note: *this example encodes 4 variables but there is no limitation in the number of allowed variables. However, if you try to encode to much data (PDF417 is limited up to 928 codeword - including error detection and correction codewords, which is 1850 ASCII characters or 1108 bytes or 2710 digits at ECC-Level 0), the program generates and prints an error.*

```
     READ IN_TAB WITH KEY 'SMPDFVAR1.
     PDFVALUE01 = IN_TAB-VALUE.
     PERFORM COLLECT_PDF USING PDFVALUE01 ';' '' '' '' ''.

     READ IN_TAB WITH KEY 'SMPDFVAR2.
     PDFVALUE01 = IN_TAB-VALUE.
     PERFORM COLLECT_PDF USING PDFVALUE01 '<' '>' '' '' ''.

     READ IN_TAB WITH KEY 'SMPDFVAR3.
     PDFVALUE01 = IN_TAB-VALUE.
     PERFORM COLLECT_PDF USING PDFVALUE01 '' '' '' '' ''.

     READ IN_TAB WITH KEY 'SMPDFVAR4.
     PDFVALUE01 = IN_TAB-VALUE.
     PERFORM COLLECT_PDF USING PDFVALUE01 '' '' '' '' ''.
```

The following part reads other values passed from the SAPscript form to decide, which bar code generating method (PCL, BMP or OTF) shell be used. The default value is “BMP”.

```
READ TABLE IN_TAB WITH KEY 'SMPDFIDENT'.
PDFIDENT = IN_TAB-VALUE.
READ TABLE IN_TAB WITH KEY 'BC_METHOD'.
if SY-SUBRC = 0.
  if in_tab-value = 'BMP'.
     GRAPHIC = 3.
  elseif in_tab-value = 'OTF'.
     GRAPHIC = 4.
  elseif in_tab-value = 'PCL'.
     GRAPHIC = 1.
  else.
     GRAPHIC = 3. "default
  endif.
else.
   GRAPHIC = 3. "default
endif.

READ TABLE IN_TAB WITH KEY 'XPOS'.
IF SY-SUBRC = 0.
   XPOS = IN_TAB-VALUE.
ENDIF.

READ TABLE IN_TAB WITH KEY 'YPOS'.
IF SY-SUBRC = 0.
   YPOS = IN_TAB-VALUE.
ENDIF.
```

Since PDFIDENT identifiers the PDF417 label on the form, you can set different parameter for each label on the form. Due to this the following statements call a form which sets parameters for the barcode which was just identified via variable PDFIDENT. In our example it is the label identified as "PDF01", thus the form "PDF01" will be called.

```
CASE PDFIDENT.
      WHEN 'PDF01'.
         PERFORM PDF01.
ENDCASE.
```

After executing the form called above the PDF417 has been generated and stored as standard text, which should be included into the SAPScript form. We use the standard interface between SAPScript and ABAP reports, the table OUT_TAB with the structure ITCSY to pass the name of the text back to the SAPScript form.

```
READ TABLE OUT_TAB WITH KEY 'SMPDFNAME'.
   OUT_TAB-VALUE = SMCHG1.
   MODIFY OUT_TAB INDEX SY-TABIX.
   READ TABLE OUT_TAB WITH KEY 'LINES'.
   IF SY-SUBRC = 0.
      SHIFT SMCHG2 LEFT DELETING LEADING '0'.
      SHIFT SMCHG2 LEFT DELETING LEADING SPACE.
      OUT_TAB-VALUE = SMCHG2.
      MODIFY OUT_TAB INDEX SY-TABIX.
   ENDIF.

ENDFORM.
```

The following form sets parameters for the PDF417 label and calls the procedure, which generates this bar code.

FORM PDF01

RBarc/PDF417© generates PCL data either as PCL bitmap graphic, as PCL vectors graphic or as SAP bitmap. Parameter GRAPHIC defines which method should be used.
GRAPHIC = 1 prints the PDF as a PCL-Bitmap (needs more CPU time but produces less data)
GRAPHIC = 2 prints the PDF as vector graphic (2 works faster but produces more data)
GRAPHIC = 3 generates the PDF as device independet SAP bitmap
GRAPHIC = 4 generates the PDF as device independet SAP OTF-bitmap
As GRAPHIC was defined above, we repeat here the definition for compatibilty reason with older versions.

```
    GRAPHIC = GRAPHIC.
```

Parameter UNIT. Valid units are 'MM', 'CM' or 'INCH'. This unit is valid for defining the horizontal position of the label (parameter XPOS), vertical position of the label (parameter YPOS), the horizontal size of the smallest element (parameter X_DIM) and the vertical size of the smallest bar code element (Y_DIM)

```
    UNIT = 'MM'.
```

Parameter POSTYP. Valid values are 0 and 1.
Value 0 means "absolute position". Only positive values for XPOS and YPOS are allowed. The PDF417 label appears always at the same position on the paper irrespective of the used SAPScript form.
Value 1 means "relative position". Positive and negative values for XPOS and YPOS are allowed. The PDF417 label appears relative to the window defined in the SAPScript form. Negative values move the label left and up, positive values move the label right and down.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.

```
    POSTYP = 1.
```

The next parameter is - depending on the parameter POSTYP - either the absolute position of the label on the paper or the movement relative to the windows defined in the SAPScript form.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.
If GRAPHIC = 4 (OTF Bitmap) and POSTYP = 0 XPOS and YPOS are absolut positions on the document.
If GRAPHIC = 4 (OTF Bitmap) and POSTYP = 1 XPOS is a realitve position to the left border of the used window and YPOS doesn't apply.
As XPOS and YPOS were defined above, we repeat here the definition for compatibilty reason with older versions.

```
    XPOS = XPOS.  YPOS = YPOS.
```

Parameter ECLEVEL defines the error detection and correction level. There are 9 error correction levels in PDF417 (0 - 8). Level 0 generates 2 check sums, level 8 generate 512 check sum. If you are not sure, which correction level to use, define the level 10. This value causes an automatic error detection and correction level calculating. This Level will be calculated by RBarc/PDF417© depending on the amount of encoded data.

```
  ECLEVEL = 10.
```

PDF417 encodes data via "codewords" and prints them in rows and columns between start and stop patterns. Given the case, your data need 30 codewords for encoding, they can be put in e.g. 10 rows and 3 columns, 5 rows and 6 columns and so on. If there is more data needed for the defined rows and columns, pad codewords will be added to the generated codewords. If the program generates more codewords than have place in defined rows and columns, RBarc/PDF417© increases the number of rows and columns or generates an error, up to the parameter NO_ERROR_PRINT. If the parameter NO_ERROR_PRINT was set to 1, RBarc/PDF417© will print the label by default values, if NO_ERROR_PRINT was set to 0, RBarc/PDF417© will generate an error.
3 - 90 rows and 1 - 30 columns are allowed. Setting the Parameter ROWS and COLUMNS to "0" causes an automatic calculation of number of rows and columns for the generated label. Other values generate an error.

```
  ROWS = 0.
  COLUMNS = 0.
```

Parameter ASPRATIO defined the relation between the label height and width. This value will be used only, if values for Rows or/and columns were set to 0.

```
  ASPRATIO = '0.5'.
```

X_DIM and Y_DIM are parameters, which define the size of the smallest element in PDF417. Usually, the relation X/Y should be 1/3, but you can use another values. Since RBarc/PDF417© doesn't check this values for plausibility, you have to check the generated label with your reader if it is readable. X_DIM and Y_DIM have to be set in measure units defined in the parameter UNIT.

Note: *using this values you can control the size of the generated label. Making X_DIM and Y_DIM lager or smaller causes a label lager or smaller in the same relation.*

```
X_DIM = '0.254'. Y_DIM = '0.762'.
```

All data codeword from a PDF417 label will be printed between Left Row Indicator (Start pattern) and Right Row Indicator (Stop patter). In a relatively "clean" closed environments it is possible to print the label without the Right row indicator (a "truncated" label). This saves place you on the paper you need for the label. Valid values are:
0: label with Right Row Indicator
1: label without Right Row indicator.

```
TRUNCATE = 0.
```

Parameter TEXT_ID allows RBarc/PDF417© to identify the just generated bar code. Up to 4 Characters can be used.

```
TEXT_ID = 'P1'.
```

Parameter NO_ERR_PRINT defines if errors detected by RBarc/PDF417© should be printed or not.
NO_ERR_PRINT = 0 causes error printing.
NO_ERR_PRINT = 1 suppresses the error printing. In this case RBarc/PDF417© sets all wrong values to default.

```
NO_ERR_PRINT = 0.
```

With the parameter ROT the PDF417 label can be rotated in 90 deg. steps.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). SAP Bitmap can be printed only at 0 Deg.

```
ROT = 0.
```

After all parameters for printing the PDF417 label have been defined, the next statement performs the procedure which passes all values to RBarc/PDF417© which generates the PDF417 label.

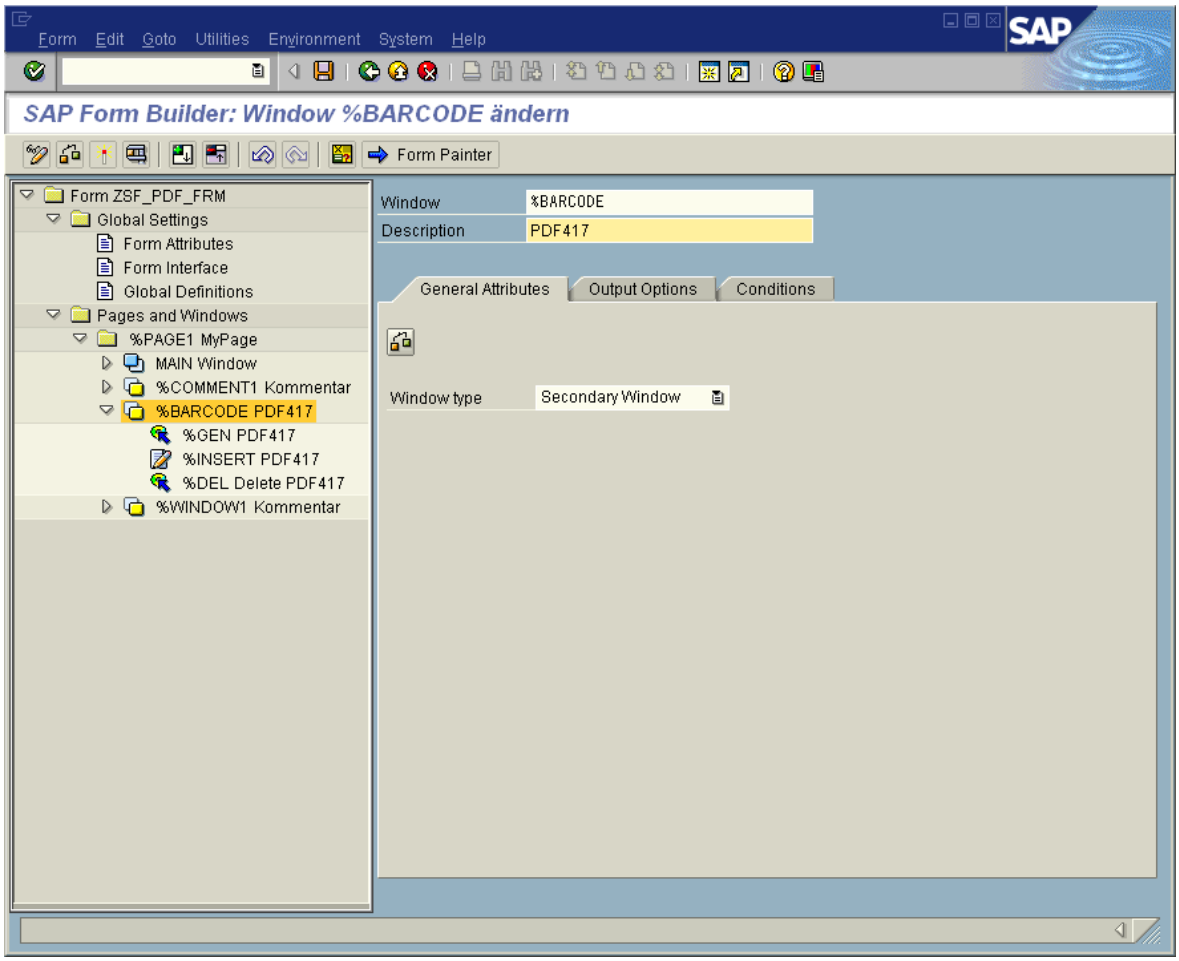
```
PERFORM GEN_PDF.

ENDFORM.
```

The next form deletes standard text created by RBarc/PDF417©. You can use the same procedure in your own programs. The structure of the text name is:
'ZPDF' + TEXT_ID(4) USER-NAME(4) SY-UZEIT + Counter
Note: *if you change names of variables used in our examples, check this procedure to be sure, bar codes generated by RBarc/PDF417© will be deleted in fact, otherwise you will have plenty of texts in your system. We use the changing variable SMPDFNAME to store the text name. If you change this name in SAPScript, you also should change this name in this procedure. If you are not sure, if your texts will be deleted, search in your system for texts beginning with 'ZPDF*'.*

```
FORM DEL_PDF TABLES IN_TAB STRUCTURE ITCSY
                    OUT_TAB STRUCTURE ITCSY.

DATA: LANG, SMVAR_N(70), Text_ID like stxh-tdid.
TABLES: STXH.

   READ TABLE IN_TAB WITH KEY 'SMPDFNAME'.
   SMVAR_N = IN_TAB-VALUE.

  READ TABLE IN_TAB WITH KEY 'BC_METHOD'.
  IF SY-SUBRC = 0.
     IF IN_TAB-VALUE = 'OTF'.
        Text_ID = 'ADRS'.
     ENDIF.
  ELSE.
     Text_ID = 'ST'. "default
  ENDIF.


    SELECT SINGLE * FROM STXH WHERE TDOBJECT = 'TEXT'
                              AND TDID = Text_ID
                      AND TDNAME = SMVAR_N.
    IF SY-SUBRC = 0.
      LANG = STXH-TDSPRAS.
          CALL FUNCTION 'DELETE_TEXT'
               EXPORTING
                    ID        = 'ST'
                    LANGUAGE  = LANG
                    NAME      = SMVAR_N
                    OBJECT    = 'TEXT'
               EXCEPTIONS
                    NOT_FOUND = 1
                    OTHERS    = 2.
      ENDIF.
ENDFORM.
```

9.5. Printing PDF417 with SmartForms.

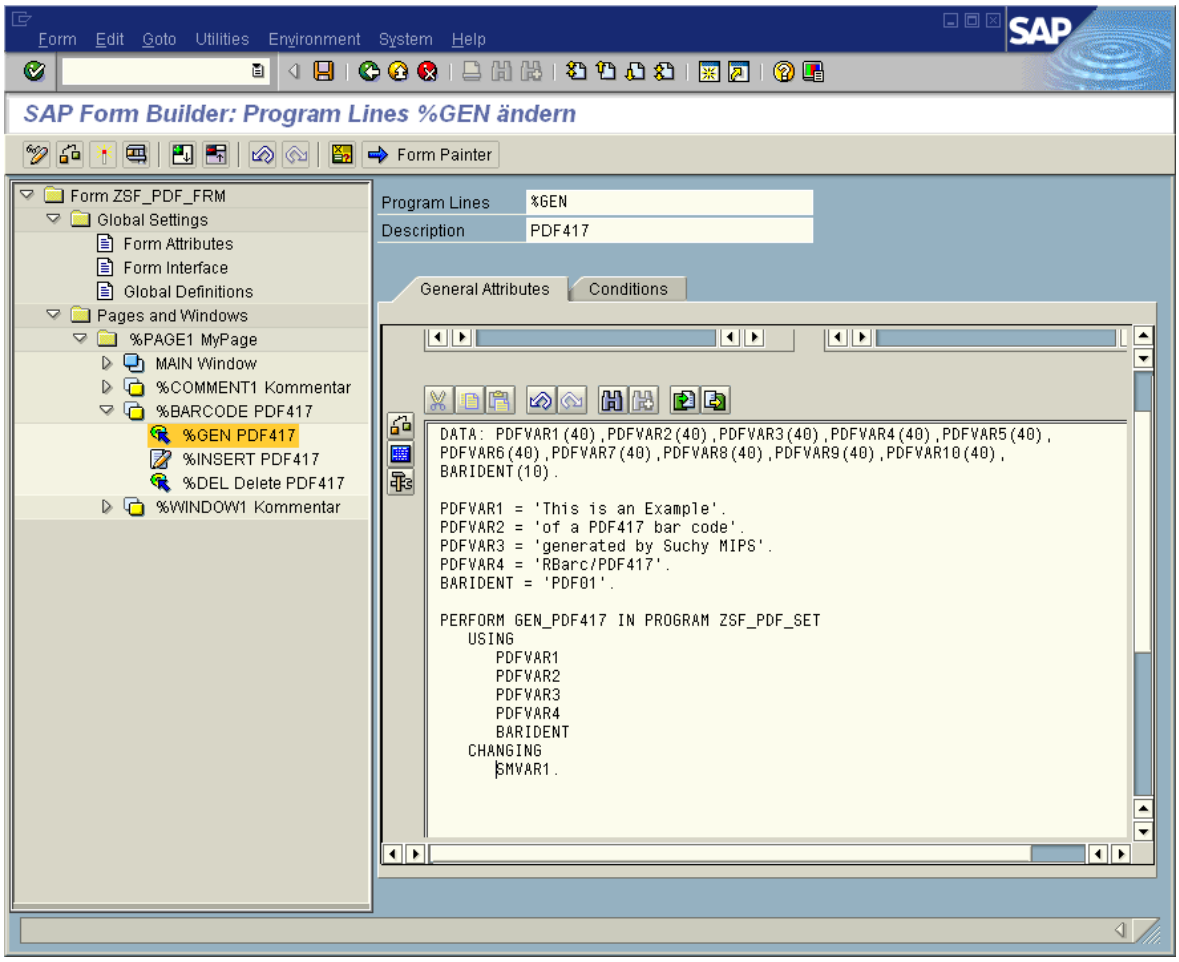
For each bar code which have to be printed in a Form a special node with 3 subnodes have to be created



In the example above the node %BARCODE is a window for the PDF417 bar code.

Each bar code window consists of 3 nodes.

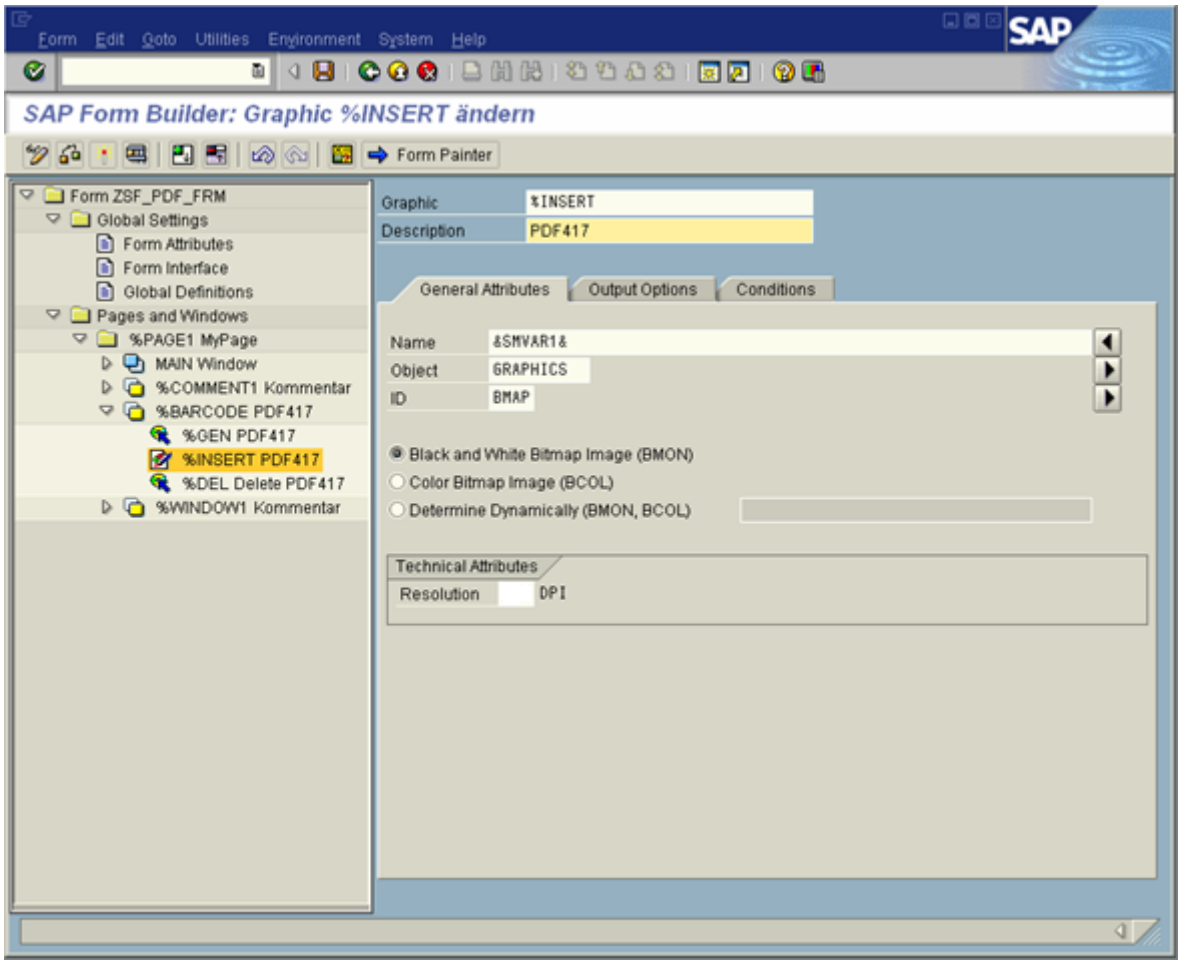
The 1st node (%GEN in the example) is a node of type “Program Lines”.



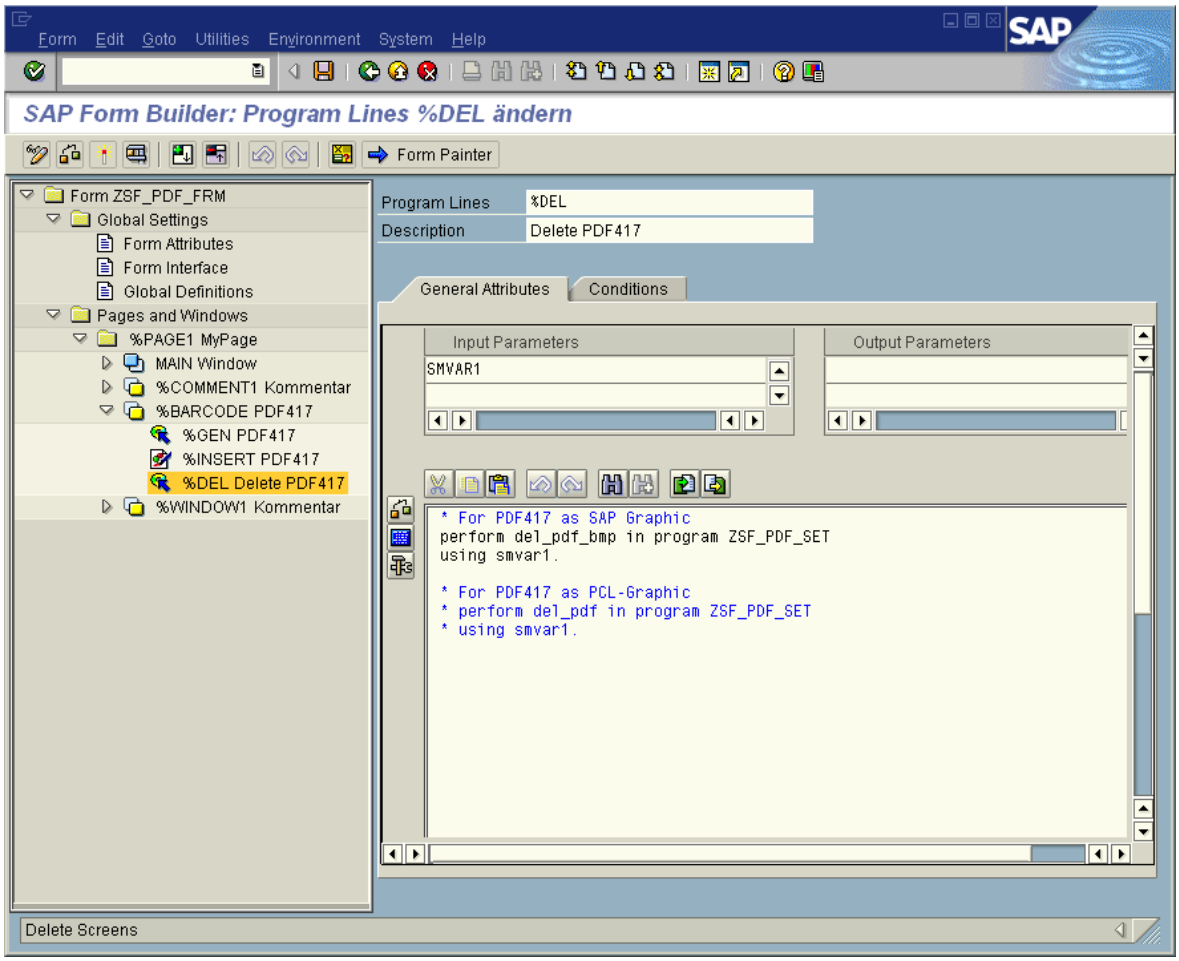
The program in this node calls an RBarc/PDF417© procedure which generates a bar code from the encoding values. The encoding value in this example are PDFVAR1,. PDFVAR2, PDFVAR3 and PDFVAR4.

The called program (ZSF_PDF_SET in this example) is a program including all bar code parameters (like hight, widht and so on) and generating a bar code. The bar code will be stored in the system as standard text and the unique name of this text (generated dynamically) will be returned as SMVAR1. SMVAR1 has to be declared as Output Parameter in this node and also in global definitions. BARIDENT is a value which allow the system to differntiate between each bar code generated inside the same form. **Remember to use uniqe names of BARIDENT for each defined bar code**.

The 2nd node is of type. This text includes the bar code generated by RBarc/PDF417©. The bitmap name is given in the variable SMVAR1.



The 3th node is of type “Program Lines”. This Program lines delete the standard text generated by RBarc/PDF417© from the system.



Note:	The name of the program ZSF_PDF_SET is an example only. This name can be different.

Follow the steps in the next section to create a report which defines values for Bar Code properties and performs the form PRINT_PDF in the include ZPDFSMARTFORMS.

9.6. Creating an ABAP program for PDF417 parameter settings (SmartForms)

The form example above has defined variables for encoding with PDF417 and passed this information to the procedure (form) SMPDF in the ABAP Program ZPDF_SET_SF. Here is an example of such a program with comments:

Report ZSF_PDF_SET

Data declaration for this program:

```
DATA:  PDF(256), ECLEVEL TYPE I, ROWS TYPE I, COLUMNS TYPE I,
       ASPRATIO TYPE F, X_DIM TYPE F, Y_DIM TYPE F, TRUNCATE TYPE I,
       XPOS TYPE F, YPOS TYPE F, ROT TYPE I, POSTYP TYPE I,
       TEXT_ID(4), BCVALUE01(40), BARIDENT(10), UNIT(4),
       SMVAR1(31), SMVAR2(31), SMVAR3(31), SMVAR_N(70),
       SMCHG1(40), SMCHG2(31), SMCHG3(31), NO_ERR_PRINT TYPE I,
       PDFVALUE01(80), PDFIDENT(10), GRAPHIC TYPE I,
       ESC(6) value '<0365>'.
```

Begin of the form SMPDF. This form receives encoding data from the SmartForms form.

```
FORM GENPDF USING PDFVALUE01 PDFVALUE02 PDFVALUE03 PDFVALUE04
                  PDFIDENT
            CHANGING TEXTNAME.
```

Next statement calls a procedure from RBarc/PDF417© which initializes all tables and variables and is obligatory.

```
  PERFORM INIT_PDFDATA.
```

The following part Collects all data for encoding in a table..
If you want delimiters between variables, you can define up to 5 delimiters. Each character you can insert via keyboard is allowed as a delimiter. Also following symbols are allowed: 'NUL', 'SOH', 'STX', 'ETX', 'EOT', 'ENQ', 'ACK', 'BEL', 'BS', 'HT', 'LF', 'VT', 'FF', 'CR', 'SO', 'SI', 'DLE', 'DC1', 'DC2', 'DC3', 'DC4', 'NAK', 'SYN', 'ETB', 'CAN', 'EM', 'SUB', 'ESC', 'FS', 'GS', 'RS', 'US' *(ASCII characters 0 - 31). If you don't want to have delimiters between variables, leave them empty.*

Note: *only one character or symbol per delimiter is allowed. The definition like* ***PERFORM COLLECT_PDF USING PDFVALUE01 '}}>' " " " " is invalid!***

Note*: this example encodes 4 variables but there is no limitation in the number of allowed variables. However, if you try to encode to much data (PDF417 is limited up to 928 codeword - including error detection and correction codewords, which is 1850 ASCII characters or 1108 bytes or 2710 digits at ECC-Level 0), the program generates and prints an error.*

```
PERFORM COLLECT_PDF USING PDFVALUE01 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE02 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE03 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE04 'LF' 'CR' " " ".
```

Since PDFIDENT identifiers the PDF417 label on the form, you can set different parameter for each label on the form. Due to this the following statements call a form which sets parameters for the barcode which was just identified via variable PDFIDENT. In our example it is the label identified as "PDF01", thus the form "PDF01" will be called.

```
CASE PDFIDENT.
    WHEN 'PDF01'.
      PERFORM PDF01.
ENDCASE.
```

After executing the form called above the PDF417 has been generated and stored as standard text, which should be included into the SmartFormst form. This name has to passed back to the SmartForms form.

```
TEXTNAME = SMCHG1.


ENDFORM.
```

The following form sets parameters for the PDF417 label and calls the procedure, which generates this bar code.

```
FORM PDF01
```

RBarc/PDF417© generates PCL data either as graphic or as rectangle. Parameter GRAPHIC defines which method should be used.
GRAPHIC = 1 prints the PDF as a Bitmap (needs more CPU time but produces less data)
GRAPHIC = 2 prints the PDF as vector graphic (2 works faster but produces more data)
GRaPIC = 3 generates the PDF as SAP bitmap

```
  GRAPHIC = 2.
```

Parameter UNIT. Valid units are 'MM', 'CM' or 'INCH'. This unit is valid for defining the horizontal position of the label (parameter XPOS), vertical position of the label (parameter YPOS), the horizontal size of the smallest element (parameter X_DIM) and the vertical size of the smallest bar code element (Y_DIM)

```
  UNIT = 'MM'.
```

Parameter POSTYP. Valid values are 0 and 1.
Value 0 means "absolute position". Only positive values for XPOS and YPOS are allowed. The PDF417 label appears always at the same position on the paper irrespective of the used SAPScript form.
Value 1 means "relative position". Positive and negative values for XPOS and YPOS are allowed. The PDF417 label appears relative to the window defined in the SAPScript form. Negative values move the label left and up, positive values move the label right and down.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.

```
  POSTYP = 1.
```

The next parameter is - depending on the parameter POSTYP - either the absolute position of the label on the paper or the movement relative to the windows defined in the SAPScript form.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.

```
  XPOS = 10. YPOS = 10.
```

Parameter ECLEVEL defines the error detection and correction level. There are 9 error correction levels in PDF417 (0 - 8). Level 0 generates 2 check sums, level 8 generate 512 check sum. If you are not sure, which correction level to use, define the level 10. This value causes an automatic error detection and correction level calculating. This Level will be calculated by RBarc/PDF417© depending on the amount of encoded data.

```
  ECLEVEL = 10.
```

PDF417 encodes data via "codewords" and prints them in rows and columns between start and stop patterns. Given the case, your data need 30 codewords for encoding, they can be put in e.g. 10 rows and 3 columns, 5 rows and 6 columns and so on. If there is more data needed for the defined rows and columns, pad codewords will be added to the generated codewords. If the program generates more codewords than have place in defined rows and columns, RBarc/PDF417© increases the number of rows and columns or generates an error, up to the parameter NO_ERROR_PRINT. If the parameter NO_ERROR_PRINT was set to 1, RBarc/PDF417© will print the label by default values, if NO_ERROR_PRINT was set to 0, RBarc/PDF417© will generate an error.
3 - 90 rows and 1 - 30 columns are allowed. Setting the Parameter ROWS and COLUMNS to "0" causes an automatic calculation of number of rows and columns for the generated label. Other values generate an error.

ROWS = 0.
COLUMNS = 0.

Parameter ASPRATIO defined the relation between the label height and width. This value will be used only, if values for Rows or/and columns were set to 0.

ASPRATIO = '0.5'.

X_DIM and Y_DIM are parameters, which define the size of the smallest element in PDF417. Usually, the relation X/Y should be 1/3, but you can use another values. Since RBarc/PDF417© doesn't check this values for plausibility, you have to check the generated label with your reader if it is readable. X_DIM and Y_DIM have to be set in measure units defined in the parameter UNIT.

***Note**: using this values you can control the size of the generated label. Making X_DIM and Y_DIM lager or smaller causes a label lager or smaller in the same relation.*

X_DIM = '0.254'. Y_DIM = '0.762'.

All data codeword from a PDF417 label will be printed between Left Row Indicator (Start pattern) and Right Row Indicator (Stop patter). In a relatively "clean" closed environments it is possible to print the label without the Right row indicator (a "truncated" label). This saves place you on the paper you need for the label. Valid values are:
0: label with Right Row Indicator
1: label without Right Row indicator.

TRUNCATE = 0.

Parameter TEXT_ID allows RBarc/PDF417© to identify the just generated bar code. Up to 4 Characters can be used.

TEXT_ID = 'P1'.

Parameter NO_ERR_PRINT defines if errors detected by RBarc/PDF417© should be printed or not.
NO_ERR_PRINT = 0 causes error printing.
NO_ERR_PRINT = 1 suppresses the error printing. In this case RBarc/PDF417© sets all wrong values to default.

NO_ERR_PRINT = 0.

With the parameter ROT the PDF417 label can be rotated in 90 deg. steps.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). SAP Bitmap can be printed only at 0 Deg.

ROT = 0.

After all parameters for printing the PDF417 label have been defined, the next statement performs the procedure which passes all values to RBarc/PDF417© which generates the PDF417 label.

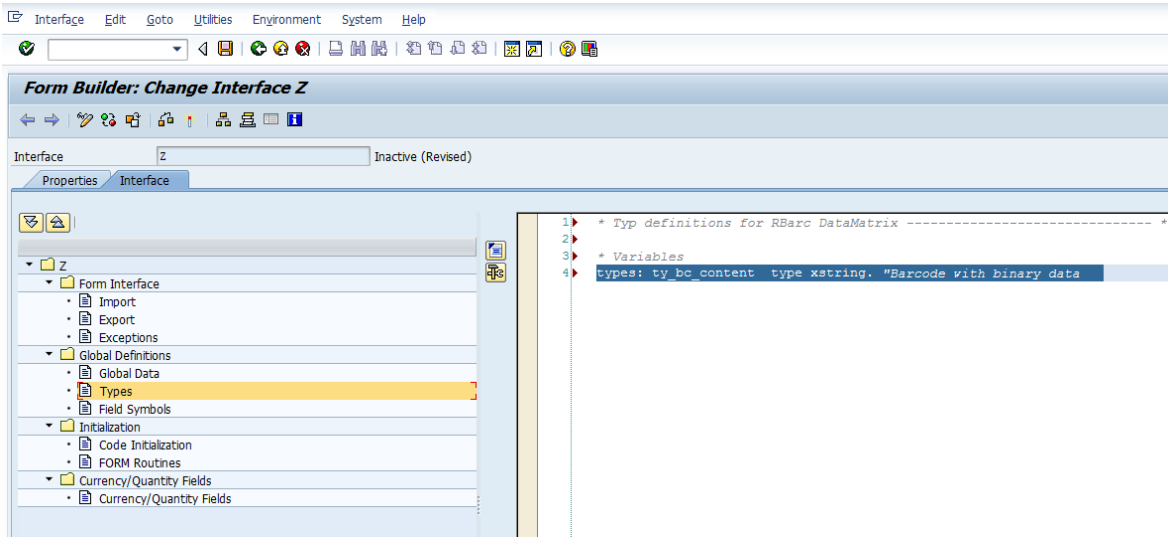
PERFORM GEN_PDF.

ENDFORM.

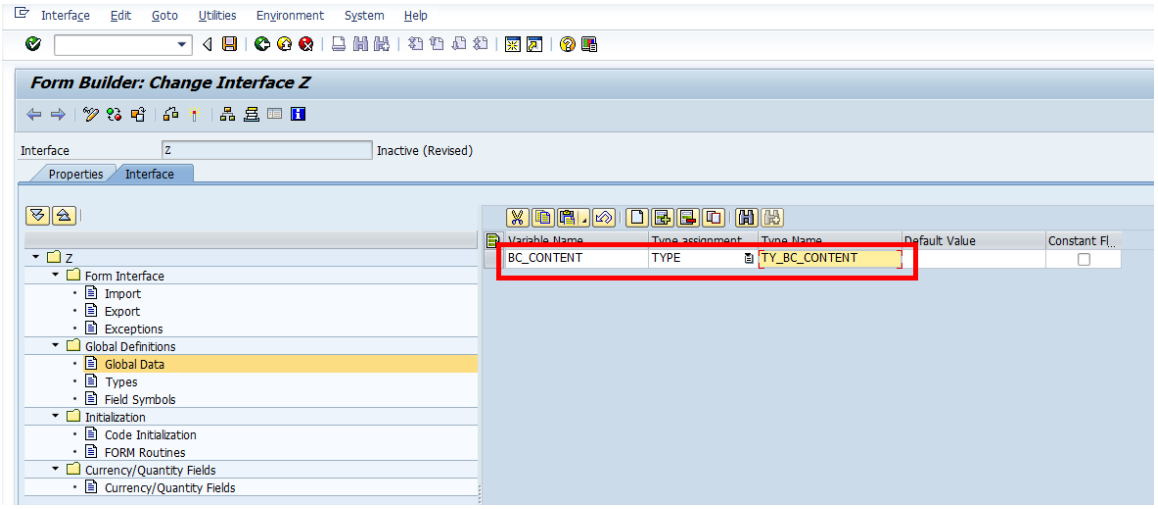
9.7. Printing PDF417 with AdobeForms

Expanding the AdobeForms Interface

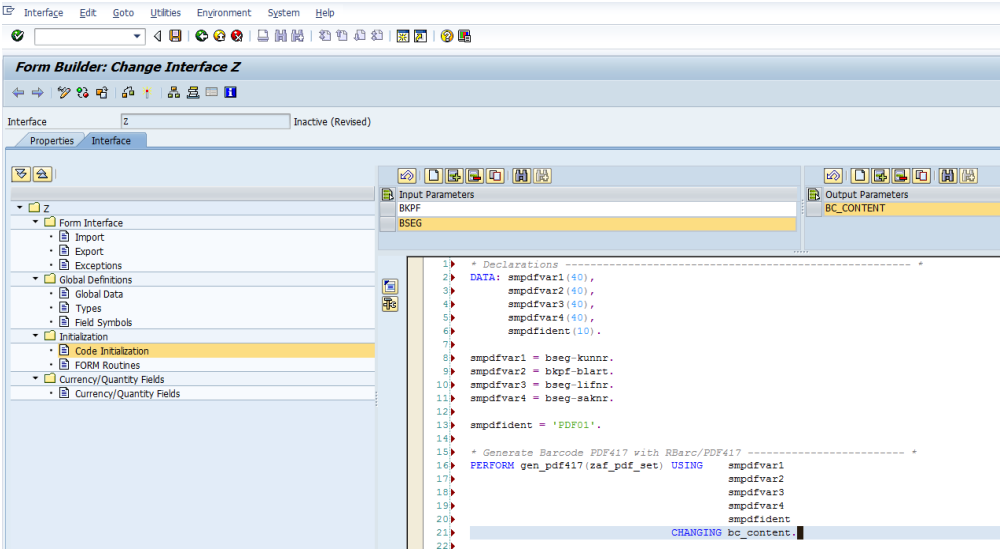
1. At first, create a global data type of the type XSTRING in the node „Global Definition→ Type“.



2. Now, create a global variable in the nod „Global Definition→ Global Data“.



3. Then, inster the following example coding in the knot „Initialization→Code Initialization“.




```
* Declarations ---------------------------------------------------------- *
DATA: smpdfvar1(40),
      smpdfvar2(40),
      smpdfvar3(40),
      smpdfvar4(40),
      smpdfident(10).

smpdfvar1 = bseg-kunnr.
smpdfvar2 = bkpf-blart.
smpdfvar3 = bseg-lifnr.
smpdfvar4 = bseg-saknr.

smpdfident = 'PDF01'.

* Generate Barcode PDF417 with RBarc/PDF417 ------------------------- *
PERFORM gen_pdf417(zaf_pdf_set) USING    smpdfvar1
                                         smpdfvar2
                                         smpdfvar3
                                         smpdfvar4
                                         smpdfident
                                CHANGING bc_content.
```

In the above example, 4 variables have to be encoded: **SUMPDFDMVAR1, SUMPDFVAR2, SUMPDFVAR3 und SUMPDFVAR4**. The amount of variables is arbitrary, so that, if needed, further variables can be added. Of course, you have to make sure that these variables can be allocated with the desired values from the form's flow logic.

With the variable **SUMPDFIDENT** (in the example it carries the value PDF01) a sub program is called in program **Z_AF_PDF_SET**, where the specific paramterizations for formatting the PDF417 barcodes are stored.

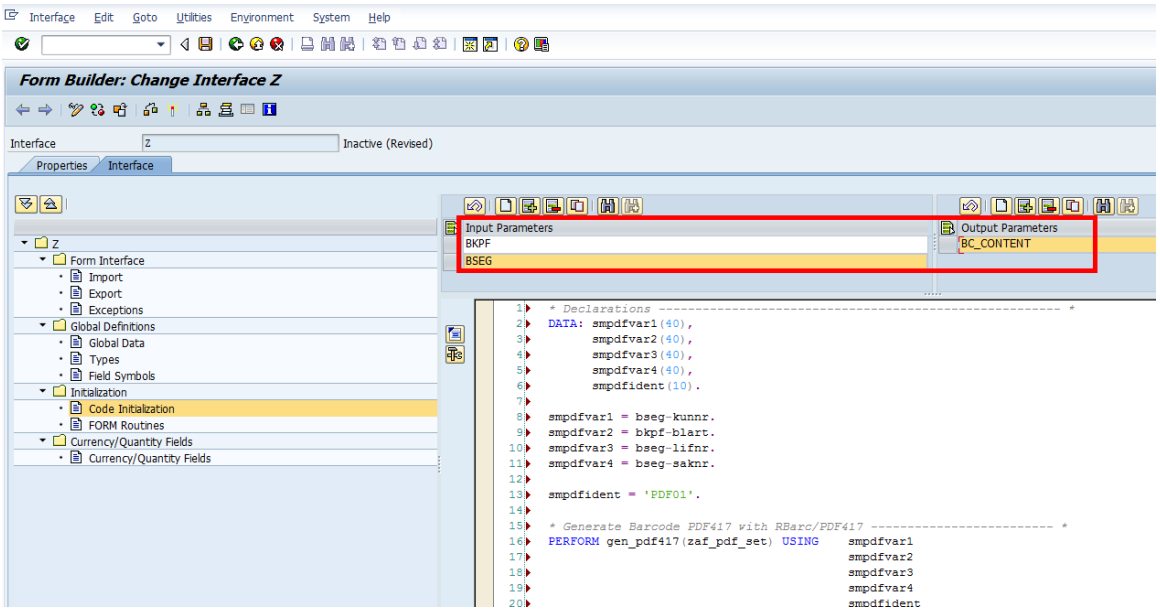
The variables are transferred to the form routine **GEN_PDF417** in program **ZAF_PDF_SET** with the statement

perform GEN_PDF417 IN PROGRAM ZAF_PDF_SET

Should you increase or decrease the number of variables in the form, remember to do this also in the form routine **GEN_PDF417** in program **ZAF_PDF_SET**.

Important:

Please keep in mind that the used variables have to be included in the node „Initialization", meaning that you also have to define them as input parameters and the barcode contents (in the example: BC_CONTENT) as output parameters.

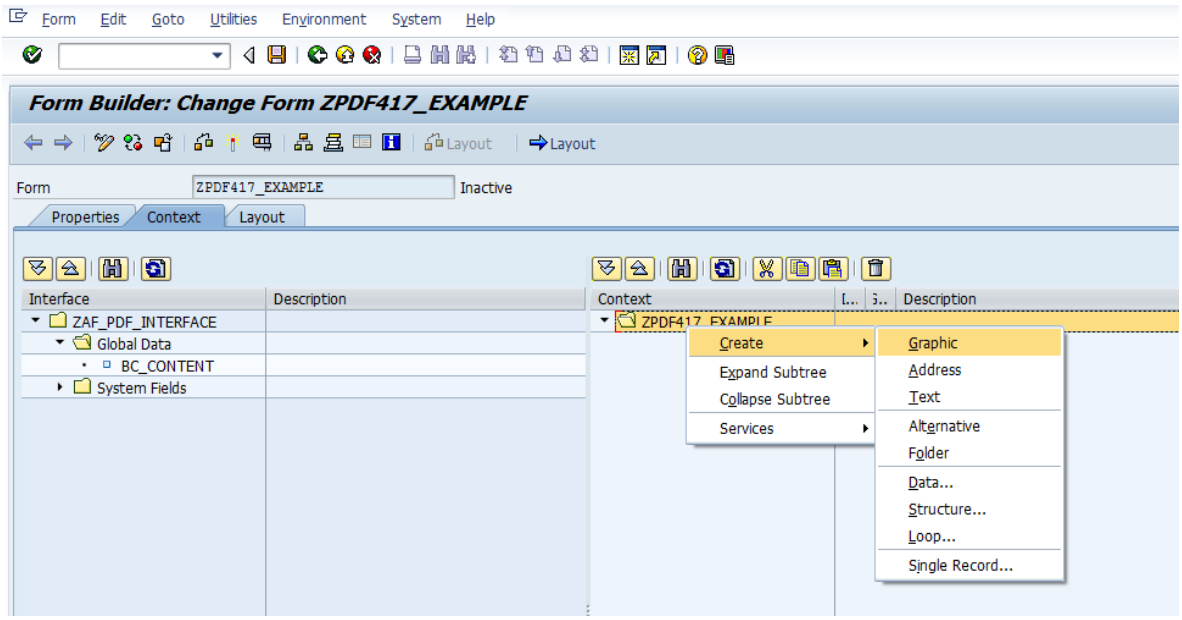


Expanding the AdobeForms Form

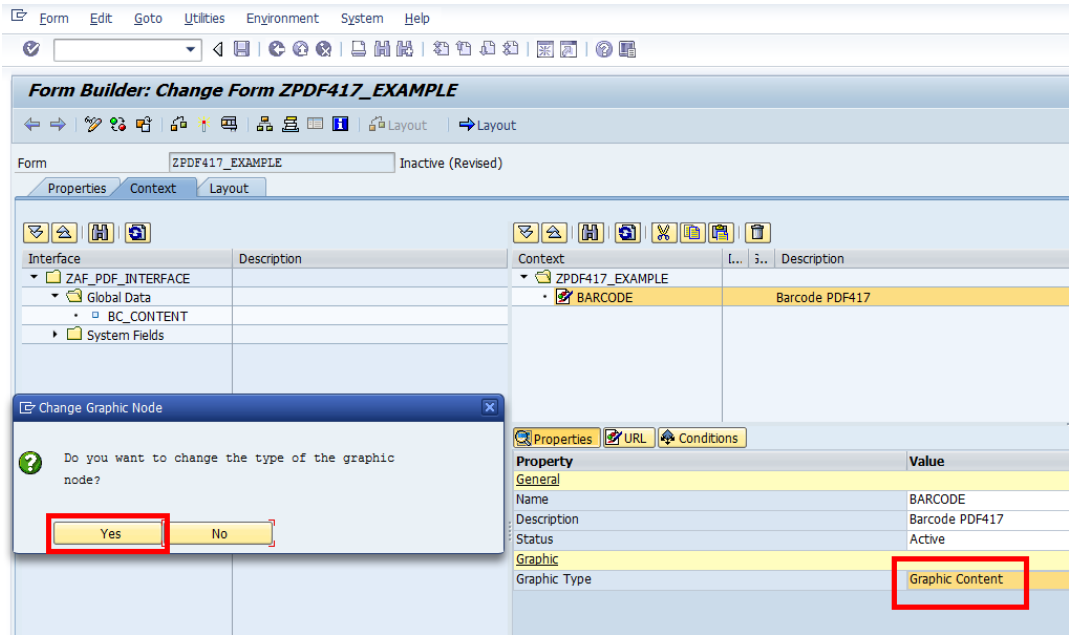
In the AdobeForms form, the barcode created with RBarc/PDF417 is shown as an image object.

Therefore, at first the context has to be expanded by the necessary elements for a barcode.

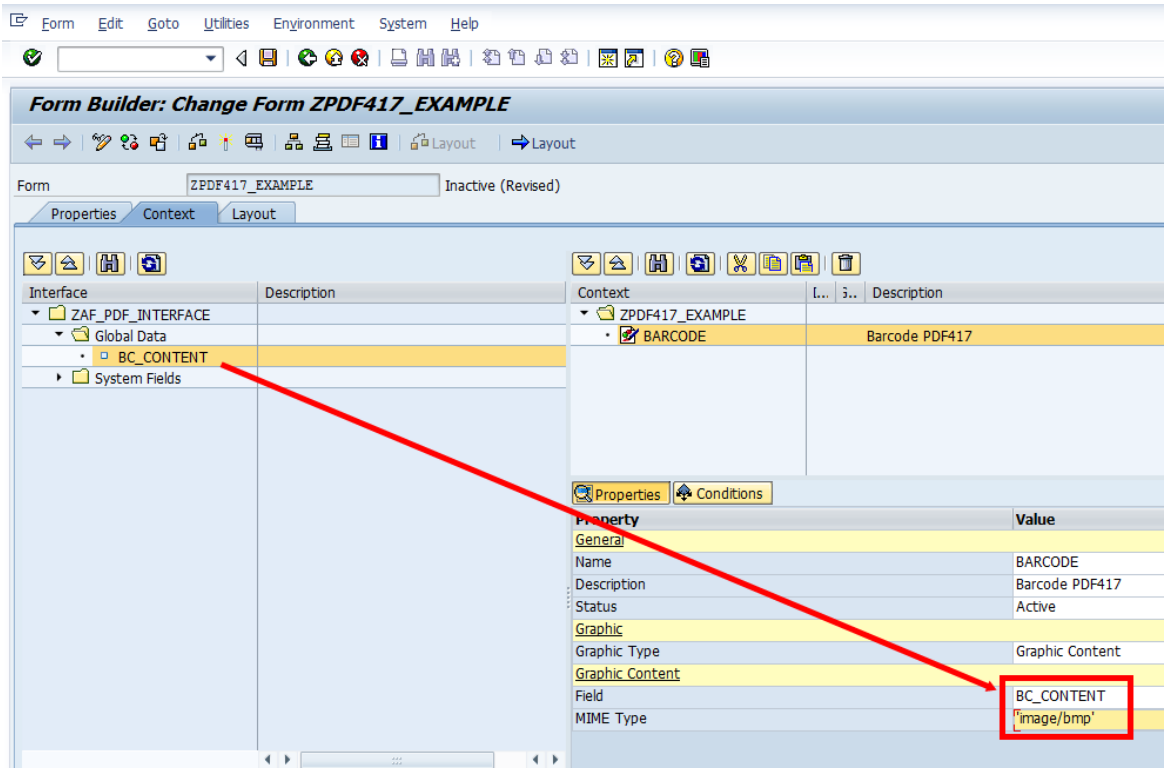
1. In the form (Transaction SFP), go into the context and create a graphic.
   Name the graphic and give it a description. In the example, the graphic is called BARCODE.



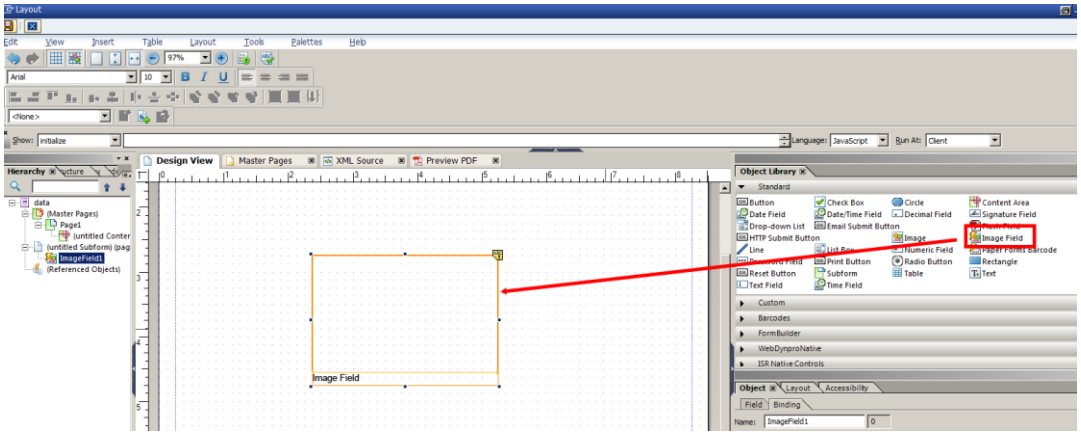
2. Now, change the graphic type to „graphic content“ and confirm the popup with „Yes“.



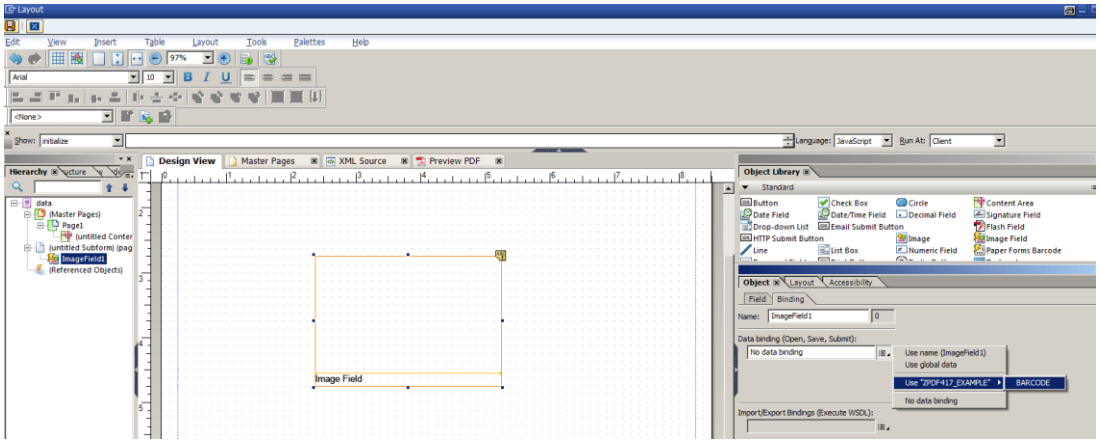
3. Now, link the graphic BARCODE to the data field that contains the barcode content by drawing the field BC_CONTENT, with the left mouse button pressed down, from the interface to the graphic BARCODE. As a MIME-Typ, please add ‚image/bmp‘.



4. Now, change into the layout view of the form Builders and create an element of the type „image field“.



5. In the image field enter, as a data binding, the graphic knot defined n the context (in the example: BARCODE).



9.8. Creating an ABAP program for PDF417 parameter settings (AdobeForms)

The form example above has defined variables for encoding with PDF417 and passed this information to the procedure (form) GEN_PDF417 in the ABAP Program ZAF_PDF_SET. Here is an example of such a program with comments:

Report ZAF_PDF_SET

Data declaration for this program:

```
DATA:  PDF(256), ECLEVEL TYPE I, ROWS TYPE I, COLUMNS TYPE I,
       ASPRATIO TYPE F, X_DIM TYPE F, Y_DIM TYPE F, TRUNCATE TYPE I,
       XPOS TYPE F, YPOS TYPE F, ROT TYPE I, POSTYP TYPE I,
       TEXT_ID(4), BCVALUE01(40), BARIDENT(10), UNIT(4),
       SMVAR1(31), SMVAR2(31), SMVAR3(31), SMVAR_N(70),
       SMCHG1(40), SMCHG2(31), SMCHG3(31), NO_ERR_PRINT TYPE I,
       PDFVALUE01(80), PDFIDENT(10), GRAPHIC TYPE I,
       ESC(6) value '<0365>',
      BITMAP_NAME TYPE TDOBNAME,
      BC_CONTENT TYPE XSTRING.
```

Begin of the form SMPDF. This form receives encoding data from the SmartForms form.

```
FORM GEN_PDF417 USING PDFVALUE01 PDFVALUE02 PDFVALUE03 PDFVALUE04
                      PDFIDENT
             CHANGING $BC_CONTENT.
```

Next statement calls a procedure from RBarc/PDF417© which initializes all tables and variables and is obligatory.

```
 PERFORM INIT_PDFDATA.
```

The following part Collects all data for encoding in a table..
If you want delimiters between variables, you can define up to 5 delimiters. Each character you can insert via keyboard is allowed as a delimiter. Also following symbols are allowed: 'NUL', 'SOH', 'STX', 'ETX', 'EOT', 'ENQ', 'ACK', 'BEL', 'BS', 'HT', 'LF', 'VT', 'FF', 'CR', 'SO', 'SI', 'DLE', 'DC1', 'DC2', 'DC3', 'DC4', 'NAK', 'SYN', 'ETB', 'CAN', 'EM', 'SUB', 'ESC', 'FS', 'GS', 'RS', 'US' *(ASCII characters 0 - 31). If you don't want to have delimiters between variables, leave them empty.*

Note: *only one character or symbol per delimiter is allowed. The definition like* ***PERFORM COLLECT_PDF USING PDFVALUE01 '}}>' " " " " is*** *invalid!*

Note*: this example encodes 4 variables but there is no limitation in the number of allowed variables. However, if you try to encode to much data (PDF417 is limited up to 928 codeword - including error detection and correction codewords, which is 1850 ASCII characters or 1108 bytes or 2710 digits at ECC-Level 0), the program generates and prints an error.*

```
PERFORM COLLECT_PDF USING PDFVALUE01 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE02 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE03 'LF' 'CR' " " ".
PERFORM COLLECT_PDF USING PDFVALUE04 'LF' 'CR' " " ".
```

Since PDFIDENT identifiers the PDF417 label on the form, you can set different parameter for each label on the form. Due to this the following statements call a form which sets parameters for the barcode which was just identified via variable PDFIDENT. In our example it is the label identified as "PDF01", thus the form "PDF01" will be called.

```
CASE PDFIDENT.
    WHEN 'PDF01'.
      PERFORM PDF01.
ENDCASE.
```

After executing the form called above the PDF417 has been generated and stored as bitmap in the SAP database.
Das Bitmap muss jetzt noch als binaries Objekt an die Schnittstelle des AdobeForms zurückgegeben werden.

```
  bitmap_name = smchg1.

 CALL METHOD cl_ssf_xsf_utilities=>get_bds_graphic_as_bmp
  EXPORTING
   p_object      = 'GRAPHICS'
   p_name        = bitmap_name
   p_id          = 'BMAP'
   p_btype       = 'BMON'
  RECEIVING
   p_bmp         = $bc_content
  EXCEPTIONS
   not_found     = 1
   internal_error = 2
   OTHERS        = 3.
```

Zum Schluss wird das Bitmap in der SAP Datenbank gelöscht.

```
  PERFORM del_pdf_bmp USING lv_name.

ENDFORM.
```

The following form sets parameters for the PDF417 label and calls the procedure, which generates this bar code.

FORM PDF01

RBarc/PDF417© generates PCL data either as graphic or as rectangle. Parameter GRAPHIC defines which method should be used.
GRAPHIC = 1 prints the PDF as a Bitmap (needs more CPU time but produces less data)
GRAPHIC = 2 prints the PDF as vector graphic (2 works faster but produces more data)
GRaPIC = 3 generates the PDF as SAP bitmap

```
  GRAPHIC = 2.
```

Parameter UNIT. Valid units are 'MM', 'CM' or 'INCH'. This unit is valid for defining the horizontal position of the label (parameter XPOS), vertical position of the label (parameter YPOS), the horizontal size of the smallest element (parameter X_DIM) and the vertical size of the smallest bar code element (Y_DIM)

```
  UNIT = 'MM'.
```

Parameter POSTYP. Valid values are 0 and 1.
Value 0 means "absolute position". Only positive values for XPOS and YPOS are allowed. The PDF417 label appears always at the same position on the paper irrespective of the used SAPScript form.

Value 1 means "relative position". Positive and negative values for XPOS and YPOS are allowed. The PDF417 label appears relative to the window defined in the SAPScript form. Negative values move the label left and up, positive values move the label right and down.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.

POSTYP = 1.

The next parameter is - depending on the parameter POSTYP - either the absolute position of the label on the paper or the movement relative to the windows defined in the SAPScript form.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). In this case the bar code should be positioned in the form.

XPOS = 10. YPOS = 10.

Parameter ECLEVEL defines the error detection and correction level. There are 9 error correction levels in PDF417 (0 - 8). Level 0 generates 2 check sums, level 8 generate 512 check sum. If you are not sure, which correction level to use, define the level 10. This value causes an automatic error detection and correction level calculating. This Level will be calculated by RBarc/PDF417© depending on the amount of encoded data.

ECLEVEL = 10.

PDF417 encodes data via "codewords" and prints them in rows and columns between start and stop patterns. Given the case, your data need 30 codewords for encoding, they can be put in e.g. 10 rows and 3 columns, 5 rows and 6 columns and so on. If there is more data needed for the defined rows and columns, pad codewords will be added to the generated codewords. If the program generates more codewords than have place in defined rows and columns, RBarc/PDF417© increases the number of rows and columns or generates an error, up to the parameter NO_ERROR_PRINT. If the parameter NO_ERROR_PRINT was set to 1, RBarc/PDF417© will print the label by default values, if NO_ERROR_PRINT was set to 0, RBarc/PDF417© will generate an error.
3 - 90 rows and 1 - 30 columns are allowed. Setting the Parameter ROWS and COLUMNS to "0" causes an automatic calculation of number of rows and columns for the generated label. Other values generate an error.

ROWS = 0.
COLUMNS = 0.

Parameter ASPRATIO defined the relation between the label height and width. This value will be used only, if values for Rows or/and columns were set to 0.

ASPRATIO = '0.5'.

X_DIM and Y_DIM are parameters, which define the size of the smallest element in PDF417. Usually, the relation X/Y should be 1/3, but you can use another values. Since RBarc/PDF417© doesn't check this values for plausibility, you have to check the generated label with your reader if it is readable. X_DIM and Y_DIM have to be set in measure units defined in the parameter UNIT.

***Note**: using this values you can control the size of the generated label. Making X_DIM and Y_DIM lager or smaller causes a label lager or smaller in the same relation.*

X_DIM = '0.254'. Y_DIM = '0.762'.

All data codeword from a PDF417 label will be printed between Left Row Indicator (Start pattern) and Right Row Indicator (Stop patter). In a relatively "clean" closed environments it is possible to print the label without the Right row indicator (a "truncated" label). This saves place you on the paper you need for the label. Valid values are:
0: label with Right Row Indicator
1: label without Right Row indicator.

TRUNCATE = 0.

Parameter TEXT_ID allows RBarc/PDF417© to identify the just generated bar code. Up to 4 Characters can be used.

TEXT_ID = 'P1'.

Parameter NO_ERR_PRINT defines if errors detected by RBarc/PDF417© should be printed or not.
NO_ERR_PRINT = 0 causes error printing.
NO_ERR_PRINT = 1 suppresses the error printing. In this case RBarc/PDF417© sets all wrong values to default.

NO_ERR_PRINT = 0.

With the parameter ROT the PDF417 label can be rotated in 90 deg. steps.
This parameter doesn't apply, if GRAPHIC = 3 (SAP bitmap). SAP Bitmap can be printed only at 0 Deg.
ROT = 0.

After all parameters for printing the PDF417 label have been defined, the next statement performs the procedure which passes all values to RBarc/PDF417© which generates the PDF417 label.
PERFORM GEN_PDF.

ENDFORM.

10. Parameter list for the main function PRINT_PDF.

Parameter	Type	Range/Limitations
ASPRATIO	F	> 0
COLUMNS	I	0, 1 - 30
ECLEVEL	I	0 - 8, 10
GRAPHIC	I	1, 2,3
NO_ERROR_PRINT	I	0, 1
POSTYP	I	0, 1
ROT	I	0, 90, 180, 270
ROWS	I	0, 3 - 90
SMCHG1	40 characters	
SMCHG2	31 characters	
SMCHG3	31 characters	
SMVAR1	31 characters	Graphic resolution (default = 150 dpi)
SMVAR2	31 characters	
SMVAR3	31 characters	
TEXT_ID	4 characters	
TRUNCATE	I	0, 1
UNIT	4 characters	MM, CM, INCH
X_DIM	F	> 0
XPOS	F	=>0 if POSTYP = 0, unlimited if POSTYP = 1
Y_DIM	F	> 0
YPOS	F	=>0 if POSTYP = 0, unlimited if POSTYP = 1

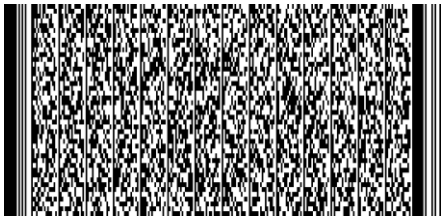
11. Examples of PDF417 labels with varying parameters.

11.1. ECLEVEL

PDF417 includes an error detection and correction mechanism. This mechanism uses special check sums for errors detection and correction. There are 9 Levels (0 - 8) of an error correction level. The higher the level, the more security and protection against damage the label offers. The example below shows the same data "PDF417 from Suchy mips" encoded with level 0, 4 and 8.


ECC Level 0


ECC Level 4


ECC Level 8

Of course, the higher the ECC Level, the more check sums the label needs and the larger the label will be. PDF417 generates 512 check sums for ECLEVEL 8, otherwise, the higher the level, the more protection against damages the label offers. The last label (level 8) is even readable, with a damage like this:

11.2. Rows and Columns

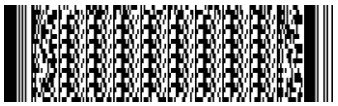
The same information can be encoded in a label of different shape, up the number of defined rows and columns. The example below shows the same information encoded in 3 different labels:


3 rows and 8 columns


9 rows and 3 columns


25 rows and 1 column

Note: using rows and columns requires a knowledge about the number of generated codewords. If you get the error "To much data for # rows and # columns" you must increase the number of rows or columns in this way, that the label can contain all codewords. The label with the string "PDF417 from Suchy mips" generates 16 codewords in level 0. You can encode this label using e.g. 2 rows and 5 columns (this is place enough for 15 codewords). But if you want, you can define much more rows and columns as you need for encoding data. Below we present a label with the string "PDF417 from Suchy mips" encoded with 20 rows and 8 columns, which gives place enough for 160 codewords. The rest of data will be filled out with s.c. pad-codewords.



11.3. Aspect Ratio

If you are not sure, how many rows and columns you should set for your PDF417 label, you can use AspectRatio for defining the shape of the label. Aspect ratio is the relation between the height and width of the label. Examples below show the same data "PDF417 from Suchy mips" encoded with different Aspect Ratio.
Note: you must set Rows and Columns to 0 for using AspectRatio.

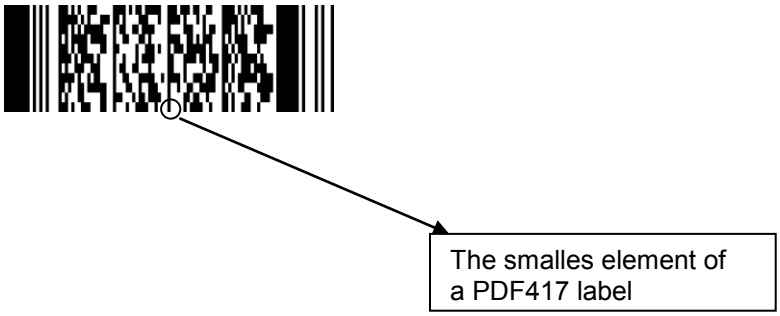

AspectRatio = 1:5 (0.2)


AspectRatio = 1:3 (0.33)


AspectRatio = 1:1 (1.0)

11.4. X_DIM and Y_DIM

X_DIM and Y_DIM define the size of the smallest element in a PDF417 label.



Via X_DIM and Y_DIM you can control the size of the label. The width and height of the module depend on the used print- and scanning technology. Using laser printers for printing PDF417 we recommend don't to use values for X_DIM less then 0.25 mm.
For Y_DIM follow the rules below:

if the scanning is certain to be done using perfect registration scanning systems or 2 dimensional CCD scanners:

Y_DIM = X_DIM

For symbols with at least the recommended minimum level of error correction, for all other scanning systems:

Y_DIM = 3 x X_DIM

For symbols with less than the recommended minimum level or error correction, for all other scanning systems:

Y_DIM = 4 x X_DIM.

Note: if you are not sure about the quality of your print- and scanning environment, we recommend to use the rule: Y_DIM = 3 x X_DIM

Here are some examples with the same encoding "PDF417 from Suchy mips" with different values for X_DIM and Y_DIM:


X_DIM = 0.212 mm, Y_DIM = 0.212 mm


X_DIM = 0.212 mm, Y_DIM = 0.423 mm


X_DIM = 0.212 mm, Y_DIM = 0.635 mm


X_DIM = 0.423 mm, Y_DIM = 1.693 mm

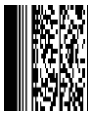
11.5. Truncated PDF417

To reduce the non-data overhead from a PDF417 label (4 codewords per row to 2 codewords per row), the right row indicator may be omitted.

Note: this procedure is possible in a relatively "clean" environment only (e.g. in an office).

Here is an example of the same encoding "PDF417 from Suchy mips" in a full- and truncated label:

Full label



Truncated label.

12. Description of possible errors

If you use wrong settings, RBarc/PDF417© generates errors. If the parameter NO_ERR_PRINT is set to 0, errors will be printed. If NO_ERR_PRINT is set to 1, RBarc/PDF417© will set wrong parameters to the default and try to print the label anyway.

Note: Message 16 and 17 appears always, even if NO_ERR_PRINT is set to 1.

```
1.      Wrong value for NO_ERR_PRINT:
            Value must be: 0 or 1
2.      Wrong value for ECLEVEL:
            Value must be in range 0 - 10
3.      Wrong value for ROWS:
            Value must be between 3 and 90
4.      Wrong value for COLUMNS:
            Value must be between 1 and 30
5.      Wrong value for ASPRATIO:
            Value must be > 0
6.      Wrong value for X_DIM:
            Value must be >0
7.      Wrong value for Y_DIM:
            Value must be >0
8.      Wrong value for TRUNCATE:
            Value must be: 0 or 1
9.      Wrong value for POSTYP:
            Value must be: 0 or 1
10.     Wrong value for XPOS:
            XPOS can`t be negative if POSTYP = 0
11.     Wrong value for YPOS:
            YPOS can`t be negative if POSTYP = 0
12.     Wrong Unit:
            Valid values are: MM, CM and INCH
13.     Wrong value for GRAPHIC:
            Value must be: 1 or 2
14.     Wrong value for ROT:
            Valid values are: 0, 90, 180 and 270
15.     To much data for # rows and # columns
            Automatic calculation of rows and columns proceeded
16.     Nothing to encode!!! <--- check your variables!
17.     To much data for PDF417:
    # Codwords were generated but only 928 are allowed
```

13. Programming hints

13.1. Encoding variables longer then 80 characters.

The standard interface between a SAPScript Form and a report transfers only max. 80 characters per USING-parameter. If you have variables longer then 80 characters, you have to cat them before in more variables.

Example:
Given an example with variable VAR, 160 characters long, you can use following statements in the SAPScript Form:

```
/:DEFINE &VAR1& := &VAR(80)&
/:DEFINE &VAR2& := &VAR+80(80)
/:PERFORM SMPDF IN PROGRAM Z_BARCODE_PDF
/:   USING &VAR1&
/:   USING &VAR2&
/:   USING &SMPDFIDENT&
/:   CHANGING &SMPDFNAME&
/:ENDPERFORM
```

Don't use any delimiters calling the function "COLLECT_PDF' to get values for VAR1 and VAR2 together at the reading time.

Note: if the variable VAR is longer then 255 characters, SAPScript can't read it above position 255. In this case you have to cat it in the report which handles the form printing.

13.2. Spaces at the end of one variable.

If you want to encode a variable with spaces at the and (e.g. "BC 123 "), you will loose the last spaces during reading the variable from the in_tab table. To omit this problem, we have implemented another space in RBarc/PDF417©. This is the character "÷" (ASCII: 247). If you don't want to loose spaces at the end of a variable, just change the original space to the character "÷". RBarc/PDF417© will encode it as a space.

Example:

```
IF PDFVALUE01+100 = ' '. PDFVALUE01+10 = '÷'. ENDIF.
PERFORM COLLECT_PDF USING PDFVALUE01 " " " " ".
```

13.3. No delimiters between variables

If you don't need special delimiters between variables, set it to be empty:
PERFORM COLLECT_PDF USING PDFVALUE01 " " " " ".

13.4. Using the same delimiter between all variables

If you use the same delimiter between all variables, you can read the table in_tab in a loop and call the function COLLECT_PDF to save programming space.

```
LOOP AT IN_TAB.
    IF SY-TABIX < SY-TFILL.
      PDFVALUE01 = IN_TAB-VALUE.
      PERFORM COLLECT_PDF USING PDFVALUE01 'LF' 'CR' " " ".
    ELSE.
      PDFIDENT = IN_TAB-VALUE.
    ENDIF.
  ENDLOOP.
```

13.5. Programming the 2D bar code for the global label template defined by General Motors

The global label defined by GM uses PDF417 for encoding important shipment data. This data can not be encode just like it is, it must be encapsulated in a special format using a Message Header, Format Header, Format Trailer and Message Trailer.

Message Header contains **[)>$^{R}_{S}$**. To print it, call in the program with PDF417 settings the form COLLECT_PDF using '[)>' as encoding data end <RS> as a delimieter.
COLLECT_PDF USING '[)>' 'RS' " " " ".

Format Header contains **06$^{G}_{S}$** To print it, call in the program with PDF417 settings the form COLLECT_PDF using '06' as encoding data end <GS> as a delimieter
COLLECT_PDF USING '06' 'GS' " " " ".

Then collect encoding data from the SAPscript form using <GS> as delimiter:
COLLECT_PDF USING PDFVALUE01 'GS' " " " ".

Note: last data don't use the message separator <GS>, but Format Trailer (**06$^{G}_{S}$)** and Message Trailer <EOT> follow, so you can use them as delimiter
COLLECT_PDF USING PDFVALUE01 '0' '6' 'GS' 'EOT' ".

14. INDEX