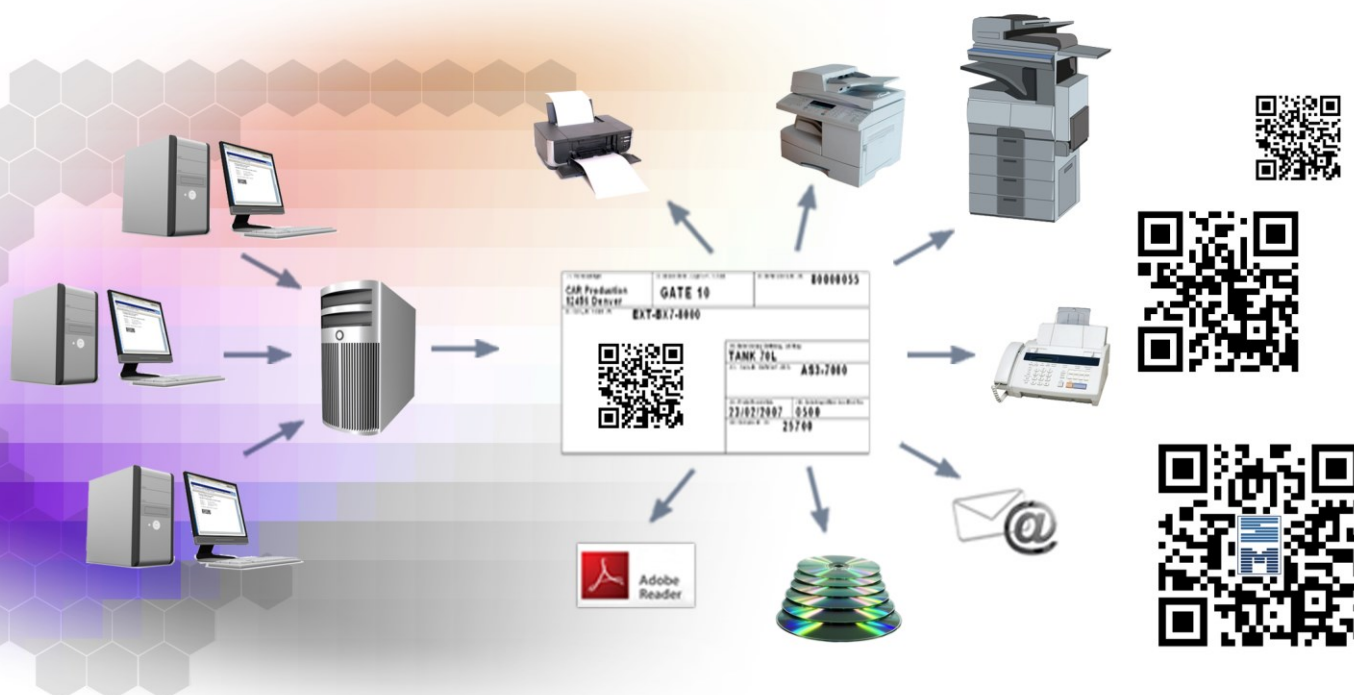




SUCHY MIPS



RBarc/QRCode For SAP® Systems Version 2 Reference Manual

Release: December 2021

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.

© Copyright Suchy MIPS 2013 - 2021
All rights reserved.

No parts of this publication may be reproduced, transmitted, stored in a retrieval system, or translated into any human or computer language, in any form by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of the copyright owner, Suchy MIPS in Munich/Germany. Copyright infringement carries with it serious civil and criminal penalties under international copyright laws.

This manual describes the installation and use of copyrighted software. Said software is licensed to the End User for use only in strict accordance with the End User License Agreement. Licensee is advised to read this End User License Agreement carefully before commencing use of product.

R/3, SAP, mySAP and Netweaver are trademarks of SAP® AG.
MS-Windows is a trademark of Microsoft® Corporation, Inc.
All other product names are trademarks of their respective holders.

Suchy MIPS GmbH
Prinzregentenstrasse 128

81677 München
Germany

Phone: +49 (0) 89 - 944 19 77 - 0
Fax: +49 (0) 89 - 944 19 7 7 - 13
e-mail: info@suchymips.de
Internet: www.suchymips.de

Contents

No.	Title	Page
1	Introduction	3
2	System requirements	4
3	Installation	5
4	Differences between the full and the demo version of RBarc/QRCode	6
5	Testing the installation of RBarc/QRCode	7
5.1	Testing with SAPscript	7
5.2	Test with Smart Forms	8
5.3	Testing with Interactive Forms	9
6	Embedding the QR-Code bar code in forms	10
6.1	The functional principle of RBarc/QRCode.	10
6.2	Categorization of the variables	11
6.3	Categorization of the variables	12
6.4	QR-Code printing from a SAPscript form	15
6.4.1	Bar code definition in a SAPscript form	15
6.5	QR-Code printing with a Smart Forms form	19
6.5.1	Bar code definition in a Smart Forms form	19
6.6	QR-Code printing with Interactive Forms	24
6.6.1	Customizing the Interactive Forms Interface	24
6.6.2	Customizing the Interactive Forms Form	29
6.7	Definition of bar code properties	32
7	Example of a job definition for a SAPscript Form.	47
7.1	Job	47
8	Example of a job definition for a Smart Forms Form.	49
8.1	Job.	49
9	Annex 1: Creating a new package (development class).	52
10	Annex 2: Creating an include in the ABAP Workbench	54
11	Annex 3: Creating a program in the ABAP workbench.	57
12	Annex 4: inserting content of installation files into an ABAP Program.	60
13	Annex 5: Activating of programs and includes	61
14	Anhang 6: Executing of an ABAP program.	63
15	Annex 7: Importing of a SAPscript form	64
16	Annex 8: Uploading of a Smart Forms Form.	66
17	Annex 9: Testing a Smart Forms form	68
18	Annex 10: Upload an Interactive Forms Interface.	70
19	Annex 11: Upload an Interactive Forms Form	72
20	Annex 12: Test an Interactive Forms Form	73
21	Annex 13: Swiss QRCode	75

1 Introduction

RBarc/QRCode is an ABAP Program for generating the 2D bar code QR-Code on SAP Systems (R/3, MySAP, ERP, Netweaver ...). This functionality is available for **SAPscript** and **Smart Forms** and **Interactive Forms** Forms.

RBarc/QRCode is compliant with the ISO QR-CODE Model 2 specification.

RBarc/QRCode generates the 2D bar code in such a way, that it becomes an integral part of the document. Thus, the bar code appears on the document not only if it will be printed on special printers with bar code capabilities, but also, if the document will be faxed, mailed, archived or converted to PDF. Furthermore, the document with the bar code can be printed on any printer (PCL, PostScript, Prescribe, Inkjet and so on). No hardware extensions are required.

Important:

RBarc/QRCode does NOT change the SAP standard. **RBarc/QRCode** programs and includes begin with the letter "Z" and will be saved in the customers area. Thus, updates will not overwrite any parts of **RBarc/QRCode**.

2 System requirements

RBarc/QRCode requires an installed SAP® System R/3 vers. 3.x or higher (also MySAP ERP and NetWeaver).

Important:

RBarc/QRCode is dedicated to system administrators (installation) and developers (bar code configuration and embedding in the form). Thus, a good knowledge in SAP basis is required for installation and a good knowledge in **SAPscript** or **Smart Forms Interactive Forms** (up to the used technology) is required for embedding bar codes in the appropriate forms. A good knowledge in ABAP will be helpful for the bar code configuration.

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

ACTVT = 01, 02, 03 und 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.

3 Installation

The installation steps will be described here in a short form. If you are not familiar with some operations, (e.g. creating of a new program), click on the blue link next to the description. This will bring you to the page with the more detailed description of the operation.

The program consists of an ABAP program (report) and an ABAP include. Both files are located in the "Install" directory.

The test forms for **Smart Forms**, **SAPscript** and **Interactive Forms** are located in corresponding subdirectories: "Test-Smartforms", "Test-SAPscript", "Test-Interactiveforms".

1. Start the **SAP GUI** and logon at the SAP System as Administrator.
2. Start the **Object Navigator** (transaction **SE80**) and create the package (development class) **ZQRCODE** (recommended).
(go to page 52: [Annex 1: Creating a new package \(development class\)](#))
3. Create an include **ZQRCODE** in the package (development class) **ZQRCODE** and delete its content, which was generated automatically.
(go to page 54: [Annex 2: Creating an include in the ABAP Workbench](#))
4. Copy the content from the file **Install ZQRCODE.INC** and paste it into the include **ZQRCODE**. Save the include and activate it.
(go to page 60: [Annex 4: inserting content of installation files into an ABAP Program](#))
5. Create a program (report) **Z_SET_QRCODE** in the package **ZQRCODE** and delete its content, which was generated automatically.
(go to page 52: [Annex 3: Creating a program in the ABAP workbench](#))
6. Copy the content from the file **Z_SET_QRCODE.PRG** and paste it into the report **Z_SET_QRCODE**. Save the report and activate it.

This completes the installation of RBarc/QRCode.

Now test objects can be installed to print one Smartforms, Interactiveforms and Sapscript form each with a QRCode barcode.

7. Create a program (report) **ZSF_QR_PRINT** in the package **ZQRCODE** and delete its content, which was generated automatically.
8. Copy the content from the file **ZSF_QR_PRINT.PRG** and paste it into the report **ZSF_QR_PRINT**. Save the report and activate it.
9. Create a program (report) **ZIF_QR_PRINT** in the package **ZQRCODE** and delete its content, which was generated automatically.
10. Copy the content from the file **ZIF_QR_PRINT.PRG** and paste it into the report **ZIF_QR_PRINT**. Save the report and activate it.
11. Create a program (report) **ZSS_QR_PRINT** in the package **ZQRCODE** and delete its content, which was generated automatically.
12. Copy the content from the file **ZSS_QR_PRINT.PRG** and paste it into the report **ZSS_QR_PRINT**. Save the report and activate it.

(go to page 61: [Annex 5: Activating of programs and includes](#))

4 Differences between the full and the demo version of RBarc/QRCode

- Only QRVersion 1 is supported, which encodes 25 alphanumeric, 41 numeric characters or 17 bytes. In mixed mode, or when using special characters, such as GS (Group separator) the amount of encryptable data is not easily predictable, since the change from one alphabet to another alphabet creates a certain overhead data the amount of which depends not only on the alphabets used but also on the distribution of characters.
- The number „1“ will be exchanged against the „0“ and vice versa.

Important:

If you want to test with real data, contact us in order to negotiate a test phase with the full product version. In the chargeable test phase user can test for 4 weeks with the full product version having an appropriate support.

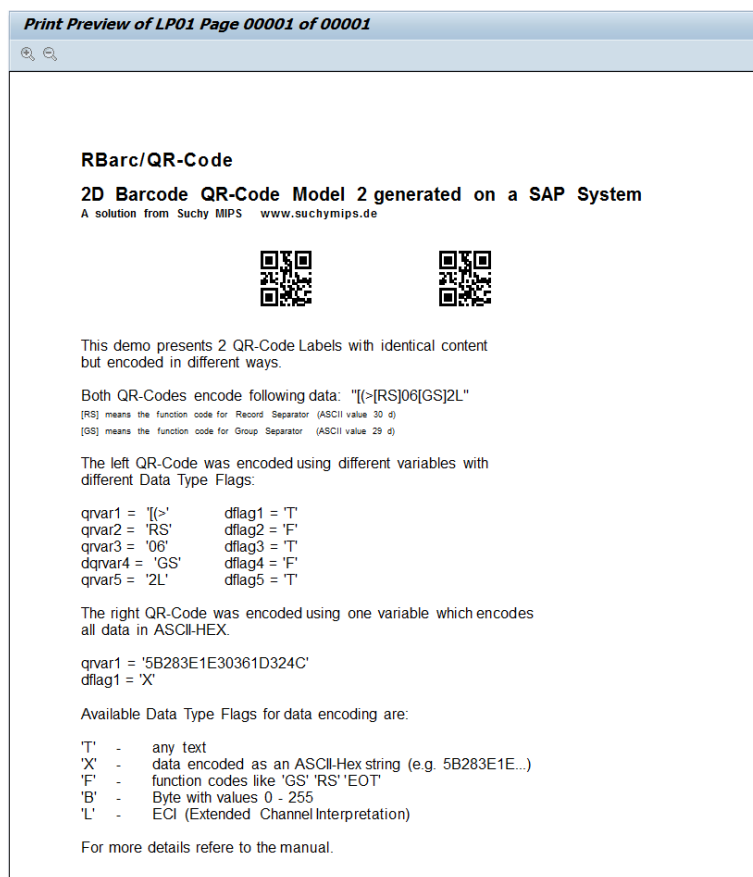
5 Testing the installation of RBarc/QRCode

The subdirectory “TestForms” of the CD contains one **SAPscript**-, one **Smart Forms**- and one **Interactive Forms** Form.

5.1 Testing with SAPscript

1. Execute the standard program **RSTXSCR**P and import the **SAPscript** Form **ZSS_QR_FORM** from the file **ZSS_QR_FORM.FOR**.
(go to page 64: [Annex 7: Importing of a SAPscript form](#))
2. Execute the report **ZSS_QR_PRINT** to print the **SAPscript** test form **ZSS_QR_FORM**.
(go to page 63: [Anhang 6: Executing of an ABAP program](#)).

The program **ZSS_PRINT** prints the **SAPscript** Form **ZSS_QR_FORM**. You can see the QR-Code bar code already in the print preview. It should look similar to this:



To embed a bar code in your own SAPscript Form, go to page 15: [Bar code definition in a SAPscript form](#))

5.2 Test with Smart Forms

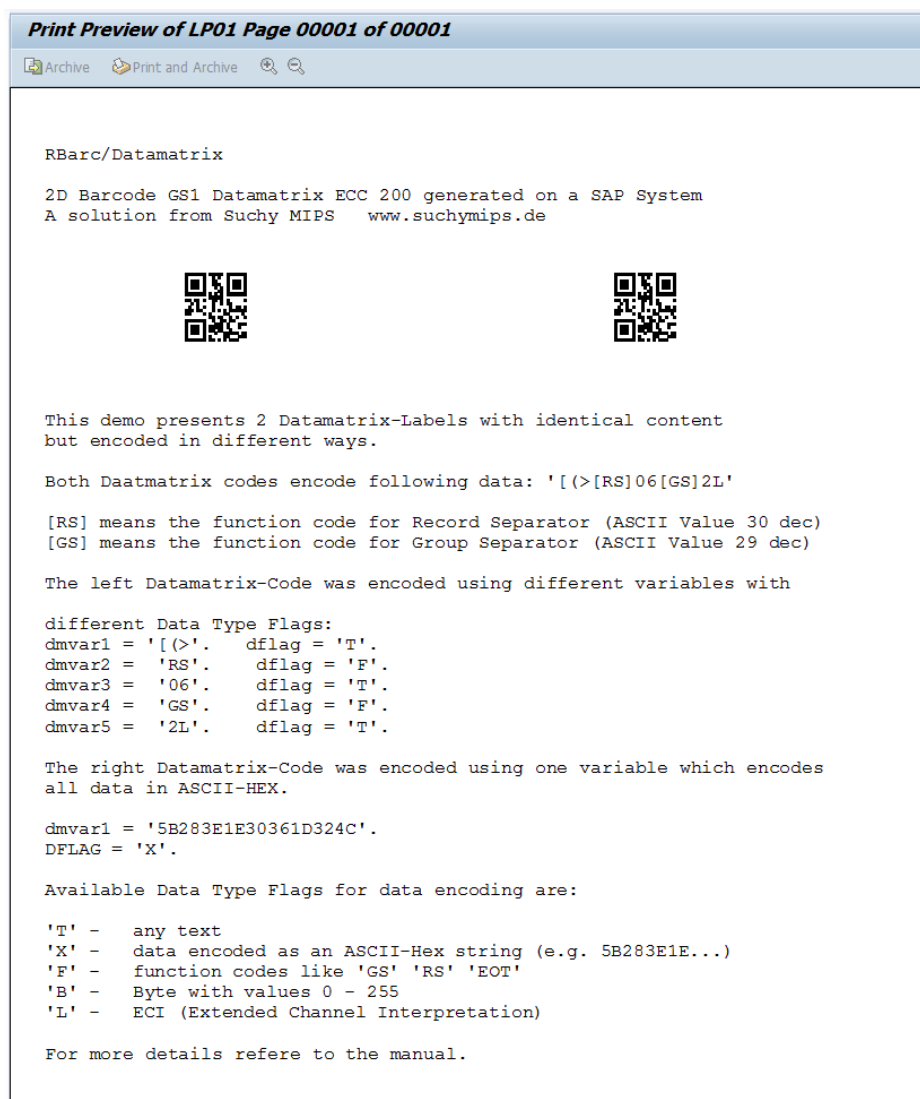
1. Start the transaction „**Smartforms**“ and upload the **Smart Forms** Form **ZSF_QR_FORM** from the file **zsf_qr_form.xml**,
(go to page 66: [Annex 8: Uploading of a Smart Forms Form](#))

Notice:

This function is available beginning with R/3 vers. 4.7. User of R/3 vers. 4.6x must create a **Smart Forms** Form manually. Go to page19: [QR-Code printing with a Smart Forms form](#)

2. To print the **Smart Forms** test form **ZSF_QR_FORM** open the form and select from the menu „Form / test“.
(go to page 68: [Annex 9: Testing a Smart Forms form](#)).

You can see the QR-Code bar code already in the print preview. It should look similar to this:

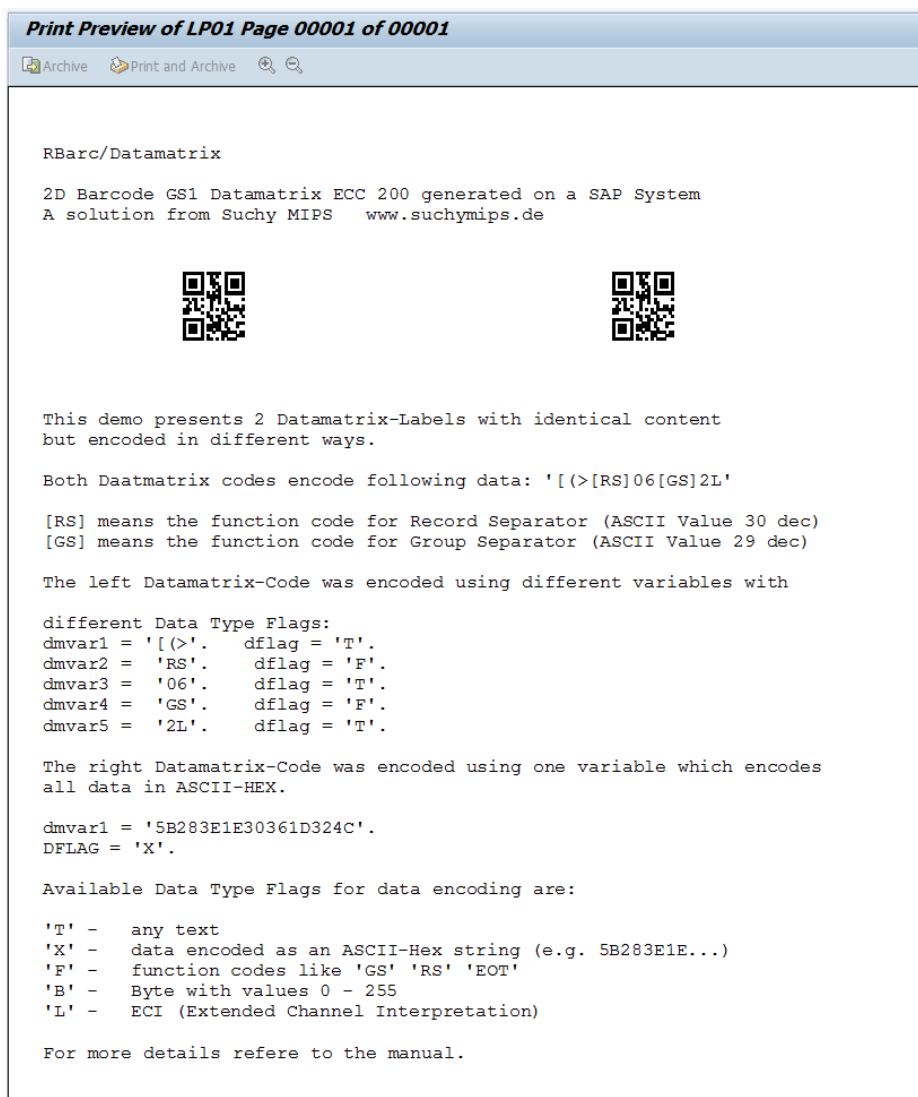


To embed a QR-Code bar code in your own **Smart Forms** form,
go to page 19: [QR-Code printing with a Smart Forms form](#)

5.3 Testing with Interactive Forms

1. Start the transaction „SFP“ and upload the **Interactive Forms** interface **ZIF_QR_INTERFACE** from file **zif_qr_interface.xml**, to be found in directory **TestForms**.
(go to page 70: [Annex 10: Upload an Interactive Forms Interface](#))
2. Start the transaction „SFP“ and upload the **Interactive Forms** form **ZIF_QR_FORM** from the file **zif_qr_form.xml**, located in directory **TestForms**.
(go to page 72: [Annex 11: Upload an Interactive Forms Form](#))
3. In order to print the **Interactive Form** test form **ZIF_QR_FORM**, open the form and select from the menu „Form / Test“.
(go to page 73: [Annex 12: Test an Interactive Forms Form](#)).

You can print out the form on the printer or look at it in the print preview. The result should present itself approximately as follows:



In order to integrate a barcode into its own **Interactive Forms** form, please read further down under: „Printing Barcodes from **Interactive Forms**“ (go to page 24: [QR-Code printing with Interactive Forms](#)).

6 Embedding the QR-Code bar code in forms

QR-Code is a 2D bar code, which can encode lots of data (up to 7089 numeric, 4296 alphanumeric or 2953 byte data). The requirement of ERP applications is not to encode continuous text but much more specific data from data fields of the data base. These fields shall mostly not to be strictly encoded, but also separated by field separators. Often the data container to be encoded gets a header and a trailer. Below is the working principle of RBarc /QR-Code as well as the rules for the implementation of QR-Code bar codes in the various SAP form technologies: SAPscript SmartoFrms and Adobe Forms.

Useful examples of tasks from the field and appropriate implementations see the chapter:

[Example of a job definition for a SAPscript Form](#)

[Example of a job definition for a Smart Forms Form](#)

6.1 The functional principle of RBarc/QRCode.

RBarc/QRCode functions as follows:

- All variables that shall to be encoded in the QR-Code bar code, must be categorised and written into an interface table in pairs (variable / type). The types of the variables are described in the next section.
- Optional parameters for bar code properties such as **module size**, **version number** or **graphic resolution** can be also written into the interface table. The Optional parameters are described in the next section. If optional parameters are not defined, then **RBarc/QRCode** will automatically set default values for these parameters.
- The interface table with encoding variables and their corresponding variable type as well as possible optional parameters will be passed to a form routine in the report **Z_SET_QRCODE**. The name of the form routine is different for each SAP form technologie:

for SAPscript :	GEN_QRCODE_SS
for Smart Forms:	GEN_QRCODE_SF
for Interactive Forms:	GEN_QRCODE_IF

- At runtime the program **Z_SET_QRCODE** generates the bar code dynamically and passes the unique name of the created bar code back to the form or the bar code as a binary object in case of Interactive Forms.
- The bar code will be included in the form dynamically.
- Finally the bar code will be deleted from the system.

Notice

The transfer of all parameters via an interface table that allows encoding of any number of variables, without each time the interface should be adapted.

The interface between the form and the report **Z_SET_QRCODE** is, in principle, for all SAP form technologies the same. However, since form technologies are different, there are differences between SAPscript, Smart Forms and Interactive Forms in implementation of the interface. Therefore, the implementation details are described in the appropriate chapters for each form technology.

6.2 Categorization of the variables

The bar code QR-Code can encode all ASCII characters in the range 0 – 255 (dec). Because not all of these characters can be processed directly by a SAP system, we have developed a special procedure, which allows to generate each character of the full ASCII range. For this reason each variable must be categorized, using the parameter "**dflag**" before it will be saved in the interface table record. The following categories of variables (values for "**dflag**") are valid:

'T' each alphanumeric value, eg. 'ABCDabcd12345AbCd'.

'X' variable in ASCII HEX format 0x00 – 0xFF. Example: the variable '**123ABC**' with **dflag** = '**T**' could also be set to the value '**303132414243**' using dflag = '**X**'. Using this method all values 00 – 255 can be encoded, even non printable characters like the binary value 0 as ASCII-HEX '00'.

variable is a function code. Only one such variable can be defined per step. If more then one such variable should be encoded, each one has to be saved separately together with its category (dflag) as a record in the interface table. Following function codes are valid:

- ASCII characters 0 - 31:
'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'.
- The EAN function character: '**FNC1**'
- The EAN function character: '**FNC2**'. The identifier 'FNC2' must immediately follow the alphanumeric, 2 character long Application Number, e.g. 'FNC2**A7**'.
- The "Extended Channel Interpretation" character: '**ECI**'

'B' – variable is a decimal binary value. Values 0 – 255 are valid. Only one value per step is allowed. If more then one such variable should be encoded, each has to be saved separately together with its category (dflag) as a record in the interface table. This method also can be - similar to type 'X' use for encoding all ASCII characters.

'L' – the variable is an "Extended Channel Interpretation" value. Each variable has to begin with 'ECI', followed by a 6 digit number. Valid ECI numbers are 0 - 999999. The default value is 000003 (ASCII for chars 0 - 127 and ISO 8859-1 for chars 128 – 255). ECI forces a special character interpretation (default = western Europe). Please remember, that ECI can be interpreted only by special scanners. A detailed description of the ECI protocol can be found in the document "*Extended Channel Interpretation (ECI) Assignments*". Example of an ECI: ECI000978.

6.3 Categorization of the variables

Optional parameters are used to control external bar code properties. These include the **size of the bar code**, the **version number**, the **resolution of the graphics**, the **ecc_level** and the **treatment of the place for the graphic**. If a parameter is not defined, then it is automatically set to the default value.

qrversion Version Number of the QR-Code in accordance with ISO specification. The ISO standard specifies 40 occurrences of the QR-Code that can be selected with **qrversion** here. Normally we try to bring the data to be encoded in a matrix as small as possible. In some applications, however, it is necessary to obtain a constant bar code size. In this case, a value for **qrversion** must be specified. The value should be chosen so that all eligible data have space in the chosen version. If during the run time it turns out, that the chosen matrix size is too small, an error message is issued and the bar code is not generated

Permissible values: 0 - 40.

Default value: 0 (automatical calculation of smallest possible matrix).



qrversion = 1



qrversion = 4

Both bar code encode the same text "Suchy MIPS"

A complete list of all matrix sizes, see the chapter [Definition of bar code properties](#)

RES Specifies the resolution at which the bar code graphic is generated. The smaller the resolution, the faster the bar code is generated and the less data are generated. Doubling the resolution with the same bar code size represents a quadrupling of the data quantity or a decrease in the bar code size by four times at constant **X_DIM**. Since the modules of a data matrix bar codes are squares, the resolution has no decisive influence on his quality. Higher resolutions should only be selected if desired bar code size can not be reached with the default resolution. If **RES** is increased without **X_DIM** is changed, then the bar code and its size can be precisely controlled with reduced **X_DIM**. If a higher resolution for the same bar code size is desired, then the parameter **X_DIM** must be increased by the same factor as the resolution.

Permissible values: >0 (Dots per inch)

Default value: 150



RES = 150 dpi



RES = 300 dpi

Both bar code encode the same text "Suchy MIPS"

X_DIM Horizontal size of the smallest module. The larger **X_DIM** the greater the bar code. The unit is 1/RES inches. The higher the resolution the smaller the **X_DIM** unit and the finer the bar code size can be determined.

Permissible values: 1 to 100.

Default value: 4



X_DIM = 5 at 150 dpi



X_DIM = 10 at 150 dpi

Both bar code encode the same text "Suchy MIPS"

A detailed description of X_DIM, see the chapter [Definition of bar code properties](#)

To print the Swiss QRCode, the X_DIM parameter must have the value **1046**. At this value the program recognizes that instead of the "normal" QR code the Swiss QR code should be generated. The Swiss QR Code is always printed in the size 46 x 46 mm and the module size is calculated automatically.

Y_DIM Should be defined only in exceptional cases.

Permissible values: 1 to 100

Default value: X_DIM

XPOS **XPOS** applies only to SAPscript. The parameter determines the horizontal position of the bar code graphic relative to the left border of the window in which the bar code is included. The value must not be negative.

Permissible values: 0 to width of page

Default value: 0

YPOS **YPOS** applies only to SAPscript. The parameter determines the vertical position of the bar code graphic relative to the row where the barcode is included. The value must not be negative.

Permissible values: 1 to length of page

Default value: 0

ECC_LEVEL Error Correction Level. The ISO-Norm specifies 4 ECC levels: '**L**', '**M**', '**Q**' and '**H**'. For each level special error detection and correction codes, so called Reed Solomon codes are calculated and appended to the bar codes. These codes guarantee a readability of partially destroyed bar codes according to following rules:

L ~ 7% of the bar code may be destroyed

M ~15% of the bar code may be destroyed

H ~ 25% of the bar code may be destroyed

Q ~ 30% of the bar code may be destroyed.

The higher the Error Correction Level the more RSS codewords are calculated and a few more data can be encoded in a given version of QRCode.

Permissible values: 'L', 'M', 'Q', 'H'

Default value: 'L'

Autoheight Type of place reservation for the barcode graphic in **SAPscript** forms. If Autoheight is set to '**Y**', then the whole space is required on the left and right of it for the graphics and no text can be placed next to it. If the value is set to '**N**', then the graphics claims no place.

Permissible values: 'Y' 'N'

Default value: 'N'

6.4 QR-Code printing from a SAPscript form

6.4.1 Bar code definition in a SAPscript form

The following example shows the implementation of a data matrix Label with 5 different variables in a SAPscript form:

```
/E ITEM_LINE
* <HL>RBarc/QR-Code
*
* <HL>2D Barcode QR-Code Model 2 generated on a SAP System
* <HN>A solution from Suchy MIPS www.suchymips.de
*
/* ***** RBarc/QR-Code 1 *****
/* ***** definition of encoding data and data types *****
/: DEFINE &QVRAR1% = '(>'
/: DEFINE &DFLAG1% = 'T'
/: DEFINE &QVRAR2% = 'RS'
/: DEFINE &DFLAG2% = 'F'
/: DEFINE &QVRAR3% = '06'
/: DEFINE &DFLAG3% = 'T'
/: DEFINE &QVRAR4% = 'GS'
/: DEFINE &DFLAG4% = 'F'
/: DEFINE &QVRAR5% = '2L'
/: DEFINE &DFLAG5% = 'T'
/*
/* ***** RBarc/QR-Code 1 *****
/* ***** definition of optional barcode parameters *****
/: DEFINE &RES% = 150
/: DEFINE &QVERSION% = 0
/: DEFINE &ECC_LEVEL% = 'L'
/: DEFINE &X_DIM% = 4
/: DEFINE &XPOS% = '50.00'
/: DEFINE &YPOS% = '0.00'
/: DEFINE &AUTOHEIGHT% = 'N'
/*
/* ***** RBarc/QR-Code 1 *****
/* ***** performing barcode generation *****
/: PERFORM GEN_QRCODE_SS IN PROGRAM Z_SET_QRCODE
/: USING &QVRAR1%
/: USING &DFLAG1%
/: USING &QVRAR2%
/: USING &DFLAG2%
/: USING &QVRAR3%
/: USING &DFLAG3%
/: USING &QVRAR4%
/: USING &DFLAG4%
/: USING &QVRAR5%
/: USING &DFLAG5%
/: USING &RES%
/: USING &QVERSION%
/: USING &ECC_LEVEL%
/: USING &X_DIM%
/: USING &XPOS%
/: USING &YPOS%
/: USING &AUTOHEIGHT%
/: CHANGING &QRNAME%
/: CHANGING &QRWIDTH%
/: CHANGING &RESULT%
/: ENDPERFORM
/*
/* ***** RBarc/QR-Code 1 *****
/* ***** including barcode bitmap *****
/: BITMAP &QRNAME% OBJECT GRAPHICS ID BMAP TYPE BMON XPOS &XPOS% MM
/*
/* ***** RBarc/QR-Code 1 *****
/* ***** deleting barcode bitmap *****
/: PERFORM DEL_QR_SS IN PROGRAM Z_SET_QRCODE
/: USING &QRNAME%
/: CHANGING &RESULT%
/: ENDPERFORM
/* ***** END of QR-Code 1 *****
```


Explanations:

DEFINE &QVAR1& = '(>Ä'

DEFINE &DFLAG1& = 'T'

...

Definition of variables for encoding. There is no limitation in the number of variables. The names of variables are preset and can not be changed. The data to be encrypted must be stored in **QVARx**, where **x** is the number of variable starting with **1** in ascending order.

DEFINE &RES& = 150

Definition of the bitmap resolution, as described in 6.3. If the parameter is omitted, then the corresponding **USING** variable in the **PERFORM GEN_QRCODE_SS** command must be removed.

DEFINE &QRVERSION& = 0

Definition of the QR Version Number like described in 6.3. If the parameter is omitted, then the corresponding **USING** variable in the **USING PERFORM GEN_QRCODE_SS** command must be removed

DEFINE &ECC_LEVEL& = 0

Definition of the Error Correction Level as described in 6.3. If the parameter is omitted, then the corresponding **USING** variable in the **USING PERFORM GEN_QRCODE_SS** command must be removed

DEFINE &X_DIM& = 4

Definition of the horizontal size of the smallest module as described under 6.3. If the parameter is omitted, then the corresponding **USING** variable in the **PERFORM GEN_QRCODE_SS** command must be removed.

DEFINE &XPOS& = '50.00'

Definition of the horizontal position of the bar code graphic relatively to the left window border. The unit is millimeters. The value must be single quoted and be higher than or equal zero. If the parameter is omitted, then the corresponding **USING** variable in the **PERFORM GEN_QRCODE_SS** command must be removed.

DEFINE &YPOS& = '0.00'

Definition of the vertical position of the bar code graphic, relatively to the recent line. The unit is millimeters. The value must be single quoted and be higher than or equal zero. If the parameter is omitted, then the corresponding **USING** variable in the **PERFORM GEN_QRCODE_SS** command must be removed.

DEFINE &AUTOHEIGHT& = 0

Definition of the bar code space treatment, like described in 6.3. If the parameter is omitted, then the corresponding **USING** variable in the **PERFORM GEN_QRCODE_SS** command must be removed.

PERFORM GEN_QRCODE_SS IN PROGRAM Z_SET_QRCODE

```
USING &QVAR1&
USING &DFLAG1&
USING &QVAR2&
USING &DFLAG2&
USING &QVAR3&
USING &DFLAG3&
USING &QVAR4&
USING &DFLAG4&
USING &QVAR5&
USING &DFLAG5&
USING &RES&
USING &QVERSION&
USING &ECC_LEVEL&
USING &X_DIM&
USING &XPOS&
USING &YPOS&
USING &AUTOHEIGHT&
CHANGING &QRNAME&
CHANGING &QRWIDTH&
CHANGING &RESULT&
```

ENDPERFORM

Passing the encoding variables "**QVAR...**" with their variable category "**DFLAG...**" and optional bar code parameters "**RES**" (Resolution), "**QVERSION**" (Version Number) **ECC_LEVEL** (Error Correction Level), "**X_DIM**" (dimension of the smallest module) **XPOS** (horizontal position relatively to the left windows borders) **YPOS** (vertical position relatively to the recent line) and **AUTOHEIGHT** (space treatment) to the form routine **GEN_QRCODE_SS** in the program **Z_SET_QRCODE**. The number of variables is not limited and can be extended taking into account the rule of variable numbering as described above. The program returns the name of the generated bar code (parameter "**QRNAME**"), the bar code width in dots of recent resolution ("**QRWIDTH**") and the result ("**RESULT**"). If no error occurs, then **RESULT = 'No errors'**.

BITMAP &QRNAME& OBJECT GRAPHICS ID BMAP TYPE BMON XPOS &XPOS& MM

The bar code will be embedded into the form dynamically. The parameter "**XPOS**" moves the bar code horizontally inside the window in which it was embedded.

PERFORM DEL_QR_SS IN PROGRAM Z_SET_QRCODE

```
USING &QRNAME&
USING &RESULT&
```

ENDPERFORM

After including the bar code into the form, it will be deleted from the system. The parameter **RESULT** returns the result of the operation. If no errors occurs, then **RESULT = 'No errors'**.

It is very important to delete the bar. Otherwise, it remains in the system must be maintained and subsequently deleted. Check with the transaction "**SE78**" whether graphics that begin with "**ZQR**" and generated during testing are still in the system. Graphics can be deleted with the program "**ZDELBMP**", which can be found in the "**UTILITY**" subdirectory.

Remember:

- The number of encoding variables (**QVAR...**) is not limited.
- The corresponding variable type (**DFLAG...**) must be assigned to each variable.
- The encoding variables must be numbered without gaps, starting with 1.
- The type **DFLAGx** must have the same number as the corresponding variable **QVARx**.
- Each variable **QVARx** and the corresponding type **DFLAGx** must follow each other as **USING** Parameter.
- Only actually present or explicitly with the **DEFINE** declared variables may be passed as **USING** parameters. If an optional parameter is not defined, it cannot be passed as a **USING** parameter, otherwise errors will occur.
- The parameters "**RES**", "**QVERSION**", "**ECC_LEVEL**", "**X_DIM**", "**XPOS**", "**YPOS**" and **AUTOHEIGHT** are optional. If not used, then default values are used automatically.
- **XPOS** and **YPOS** may not be negative values.

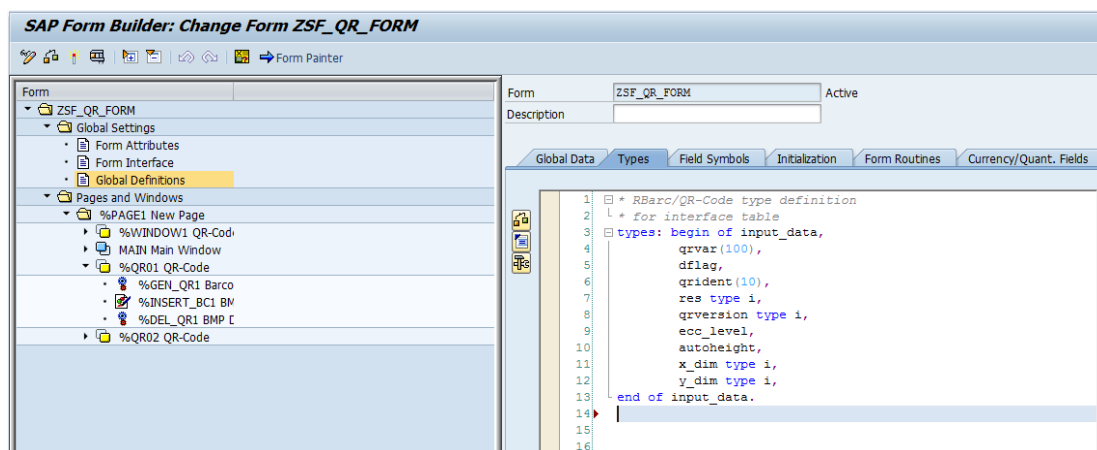
6.5 QR-Code printing with a Smart Forms form

6.5.1 Bar code definition in a Smart Forms form

An existing **Smart Forms** form shall be modified as described below:

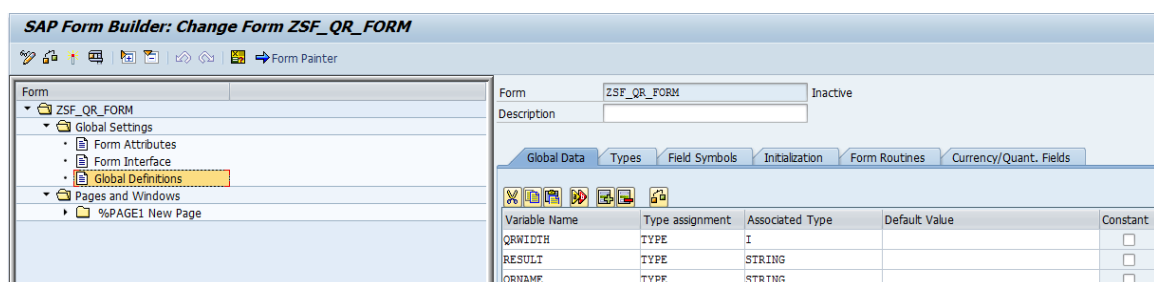
1. Define under tabpaint "Types" in the tree node "**Global Definitions**" the type input_data:

```
types: begin of input_data,  
      qrvar(100),  
      dflag,  
      qrident(10),  
      res type i,  
      qrversion type i,  
      ecc_level,  
      autoheight,  
      x_dim type i,  
      y_dim type i,  
end of input_data.
```

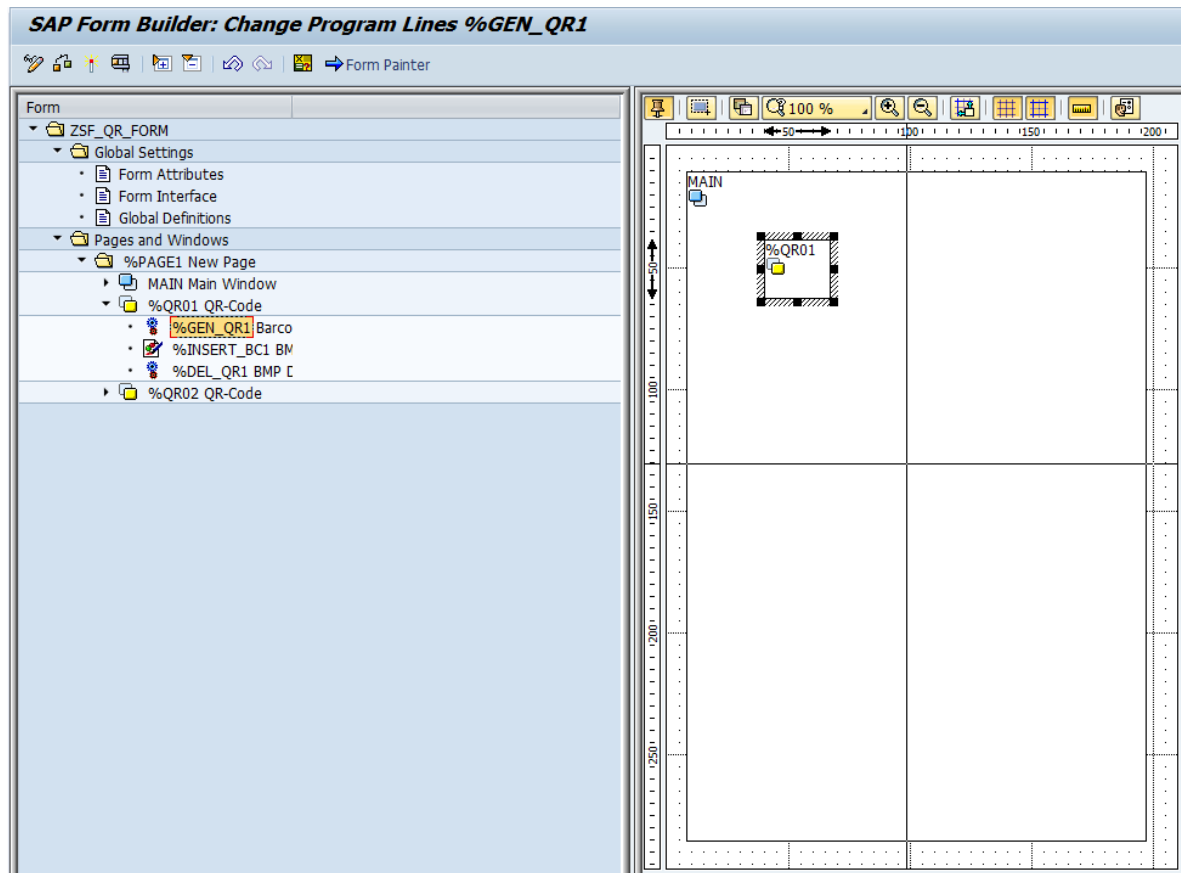


2. Define in the tabpaint "**Grobal Data**" from the tree node "**Global Definitions**" the following variables:

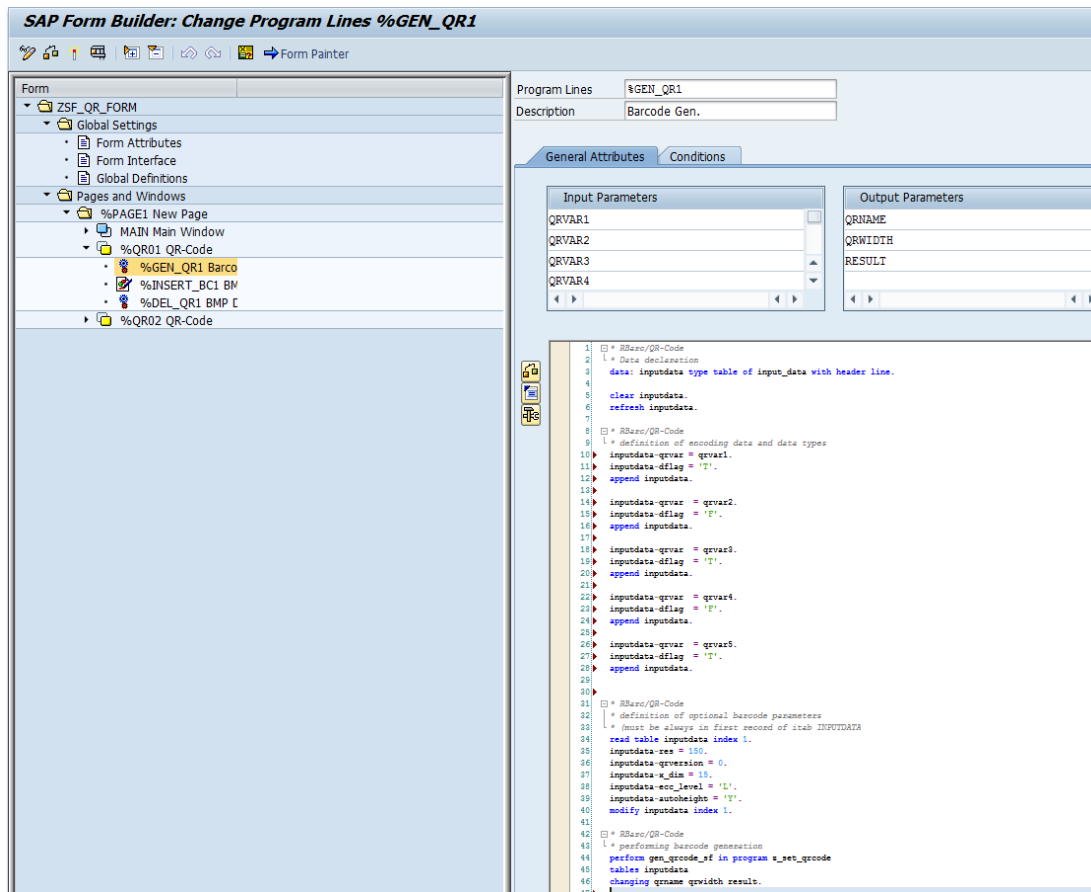
QRNAME	Type	String
QRWIDTH	Type	I
RESULT	Type	String



3. Create window, which is large enough to contain the bar code label.
This is the windows "QR01 QR-Code" in the example below.



4. Create in the new window a nod of type "Program Lines".



- Enter all necessary encoding variables as **Input Parameters**.
- Enter the variable **QRNAME** and optionally **QRWIDTH** and **RESULT** as **Output Paramter**. **QRWDITH** and **RESULT** is for your own use.
- Declare the **internal Table INPUTDATA**
`data: inputdata type table of input_data with header line.`
- Delete the content of **INPUTDATA** (important, for example if the form has several pages where the same routine is used).
`clear inputdata.`
`refresh inputdata.`
- Write to each encoding variable into the table field **QRVAR** and the corresponding data type in the table field **DFLAG** and append each record to the table **INPUTDATA**:
`inputdata-qrvar = qrvar1.`
`inputdata-dflag = 'T'.`
`append inputdata.`
... and so on for each encoding variable.

The number of encodable variables is unlimited. The variables with their corresponding types must be contiguously written into the table. Thus, each variable with its corresponding data type is a data set (= record) in the table.

- After all variables to be encoded were append to the internal table **INPUTDATA**, read the first record of this table and enter there the desired values for "**RES**" (Resolution), "**QRVERSION**" (Version Number) **ECC_LEVEL** (Error Correction Level) and "**X_DIM**" (dimension of the smallest module) and modify the record accordingly

```
read table inputdata index 1.
inputdata-res = 150.
inputdata-qrversion = 0.
inputdata-x_dim = 15.
inputdata-ecc_level = 'L'.
modify inputdata index 1.
```

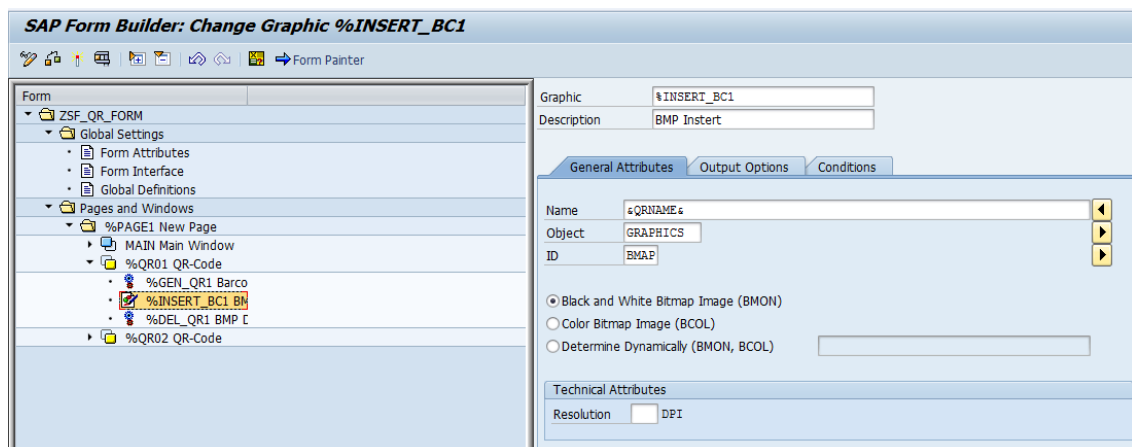
This part is optional. If it is not used, then default values are used for the parameters **RES**, **QRVERSION**, **ECC_LEVEL** and **X_DIM**.

- Pass the table **INPUTDATA** to the form routine **GEN_QRCODE_SF** in the program **Z_SET_QRCODE**

```
perform gen_qrcode_sf in program z_set_qrcode
tables inputdata
changing qrcode qwidth result
```

- The program returns the name of the generated bar codes (parameter "**QRNAME**"), the bar code width in dots of recent resolution ("**QRWIDTH**") and the result ("**RESULT**"). If no error occurs, then **RESULT = 'No errors'**.

- Create (still in the same window) a second node of type "**Graphic**" and fill the fields "**Name**", "**Object**", and "**ID**" like presented in the example below:



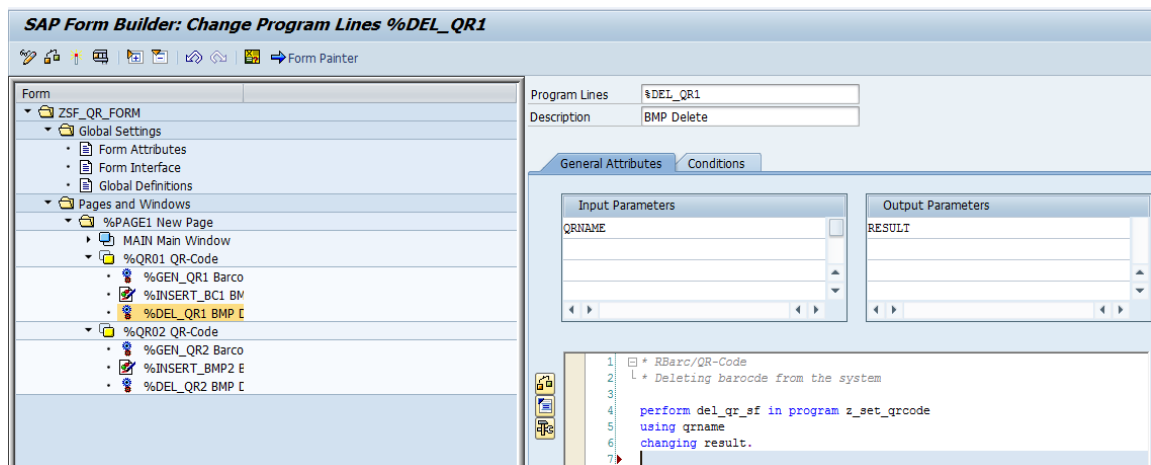
Name = &QRNAME&
Object = GRAPHICS
ID = BMAP
Resolution = leave it empty

6. Create (still in the same window) a third node of type "**Program Lines**" and insert following program lines::

```
perform del_qr_sf in program z_set_qrcode  
using qname  
changing result.
```

Enter as **Input Parameter QRNAME**.

Enter as **Output Parameter RESULT** (optionally, if you want to use it).

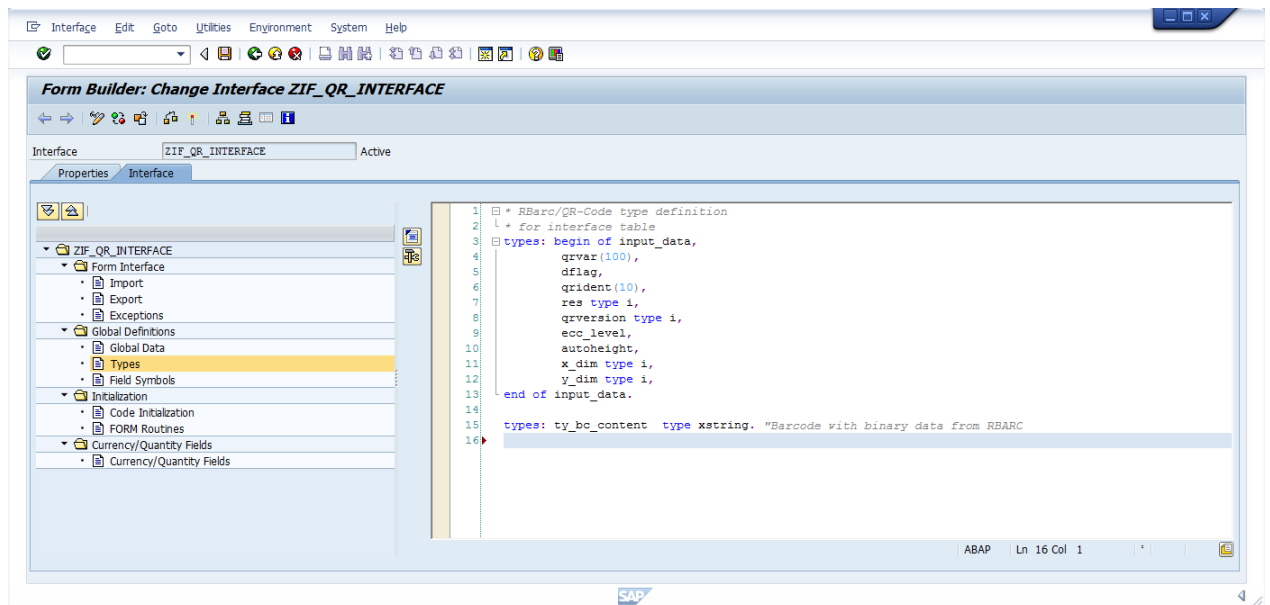


6.6 QR-Code printing with Interactive Forms

6.6.1 Customizing the Interactive Forms Interface

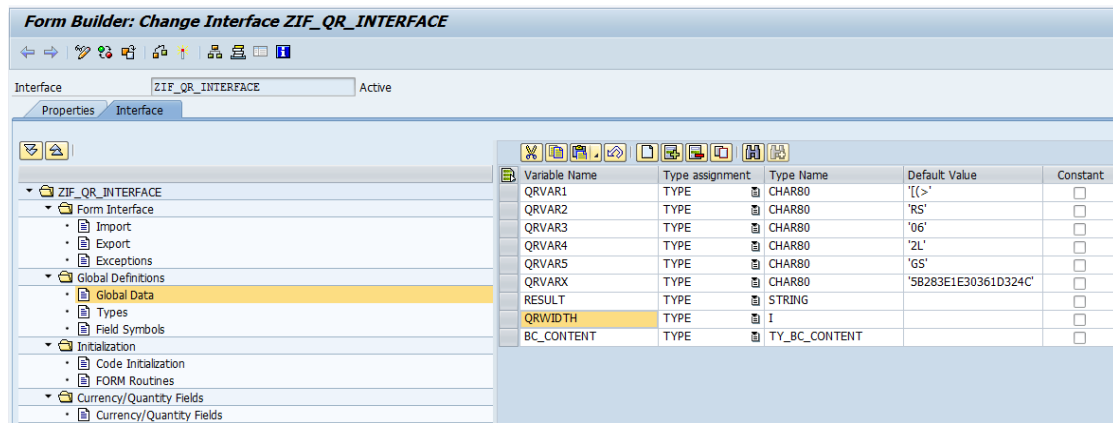
1. Define the types **input_data** and **ty_bc_content** in the node „Global Definitions→ Types“ as follows:

```
types: begin of input_data,  
      qrvar(100),  
      dflag,  
      qrident(10),  
      res type i,  
      qrversion type i,  
      ecc_level,  
      autoheight,  
      x_dim type i,  
      y_dim type i,  
end of input_data.  
  
types: ty_bc_content type xstring.
```



- Define following global variables in the node „**Global Definitions**→ **Global Data**“.

QRWIDTH	TYPE	I
RESULT	TYPE	STRING
BC_CONTENT	TYPE	TY_BC_CONTENT



Notice:

In the example above the encoding variables **QRVAR1** to **QRVAR5** were also defined as global variables. In general, however, the variables will come from the workflow of data processing. You then need to ensure by appropriate means that the required variables are also visible in the interface. Usual, it is then not necessary to define such variables as global variables, as in the case of the field **MATNR** from the table **MARD**.

For multiple bar codes in a form more global variable of Type **TY_BC_CONTENT** can be defined.

3. Insert following program lines into the node „Global Definition→ Code Initialization“.

```
* RBarc/QRCode
* Data declaration
data: inputdata type table of input_data with header line.

clear inputdata.
refresh inputdata.

*RBarc/QRCode
*definition of encoding data and data types
inputdata-qrvar = qrvar1.
inputdata-dflag = 'T'.
append inputdata.

inputdata-qrvar = qrvar2.
inputdata-dflag = 'F'.
append inputdata.

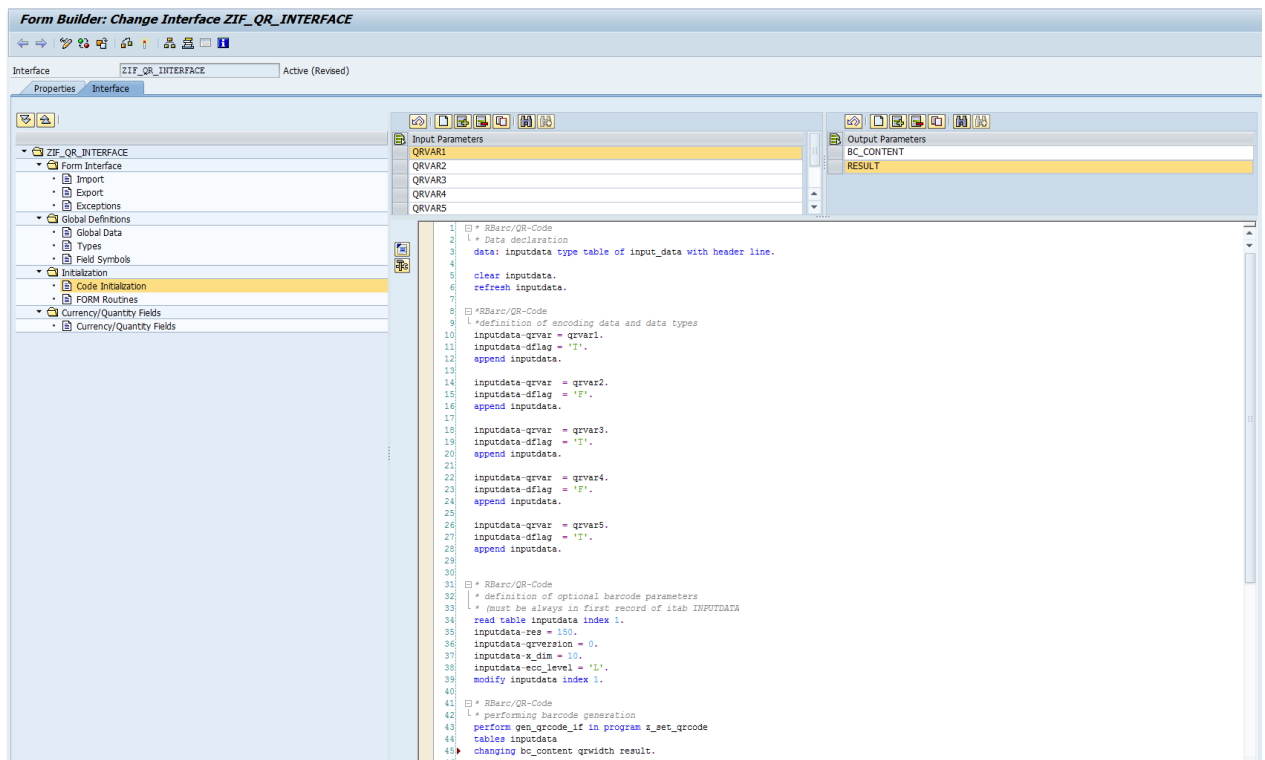
inputdata-qrvar = qrvar3.
inputdata-dflag = 'T'.
append inputdata.

inputdata-qrvar = qrvar4.
inputdata-dflag = 'F'.
append inputdata.

inputdata-qrvar = qrvar5.
inputdata-dflag = 'T'.
append inputdata.

*RBarc/QRCode
* definition of optional barcode parameters
* (must be always in first record of itab INPUTDATA
read table inputdata index 1.
inputdata-res = 150.
inputdata-qrversion = 0.
inputdata-x_dim = 10.
inputdata-ecc_level = 'L'.
modify inputdata index 1.

*RBarc/QRCode
* performing barcode generation
perform gen_qrcode_if in program z_set_qrcode
tables inputdata
changing bc_content qrwidth result.
```



Explanations:

- Enter as **Input Parameter** all encoding variables..
- Enter as **Output Parameter** the variables **BC_CONTENT** and **RESULT**.
- declare the internal table **INPUTDATA**

`data: inputdata type table of input_data with header line.`

- Delete the content of **INPUTDATA** (important, for example if the form has several pages where the same routine is used).

`clear inputdata.`
`refresh inputdata.`

- Write each encoding variable into the table field **QRVAR** and the corresponding data type in the table field **DFLAG** and append each record to the table **INPUTDATA**:

`inputdata-qrvar = qrvar1.`
`inputdata-dflag = 'T'.`
`append inputdata.`

... and so on for each encoding variable.

The number of encodable variables is unlimited. The variables with their corresponding types must be contiguously written into the table. Thus, each variable with its corresponding data type is a data set (= record) in the table.

- After all variables to be encoded were append to the internal table **INPUTDATA**, read the first record of this table and enter there the desired values for **RES** (resolution), **QRVERSION** (matrix size) **ECC_LEVEL** (Error Correction Level) **X_DIM** (size of the smallest module) and modify the record accordingly

```
read table inputdata index 1.
inputdata-res = 150.
inputdata-qrversion = 0.
inputdata-ecc_level = 'L'.
inputdata-x_dim = 4.
modify inputdata index 1.
```

This part is optional. If it is not used, then default values are used for the parameters **RES**, **QRVERSION**, **ECC_LEVEL** and **X_DIM**.

- Pass the table **INPUTDATA** to the form routine **GEN_QRCODE_IF** in the program **Z_SET_QRCODE**

```
perform gen_qrcode_if in program z_set_qrcode
tables inputdata
changing bc_content qrwidth result.
```

Return paramaters are: **bc_content** (this is he bar code graphic), **qrwidth** (the bar code bitmap width in dots oft he current resolution) und **result** (the result). If the bar code generation was proceeded without errors, the result contains the message: **result = 'No errors'**.

Important:

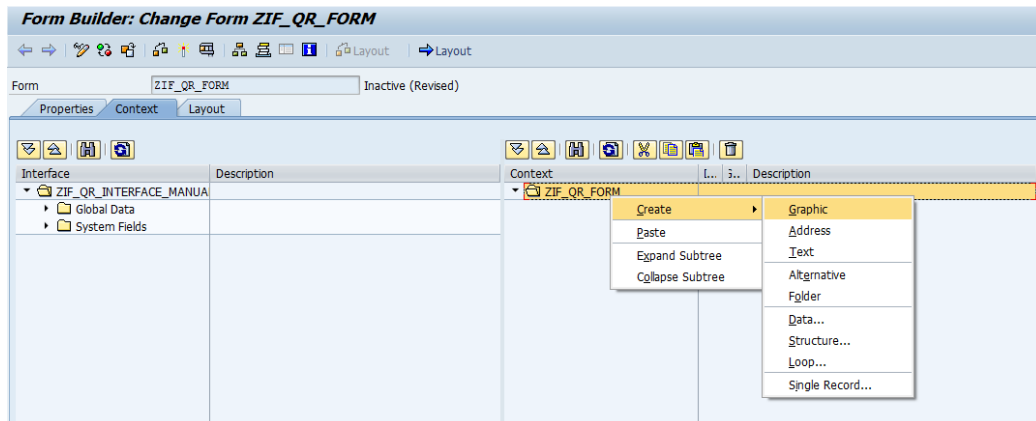
Please remember that all used encoding variables have to be also advertised in the node „Initialization“, meaning you have to define them as input parameters together with other parameters..

6.6.2 Customizing the Interactive Forms Form

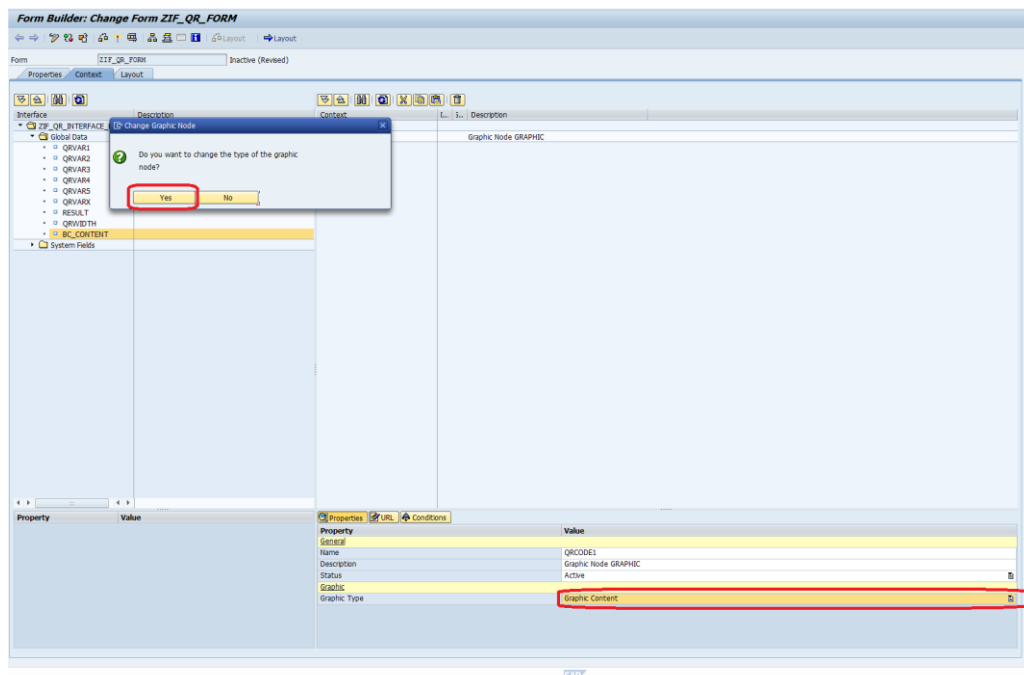
In the Interactive Forms form the bar code created with **RBarc/QRCode** is displayed as an image object. Therefore, at first the context has to be expanded by the necessary elements for a bar code.

1. Select in the form (Transaktion **SFP**) in the context menu (right windows side) the form name, click on the right mouse button and select from context menu **Create→Graphik** in order to create a graphic object.

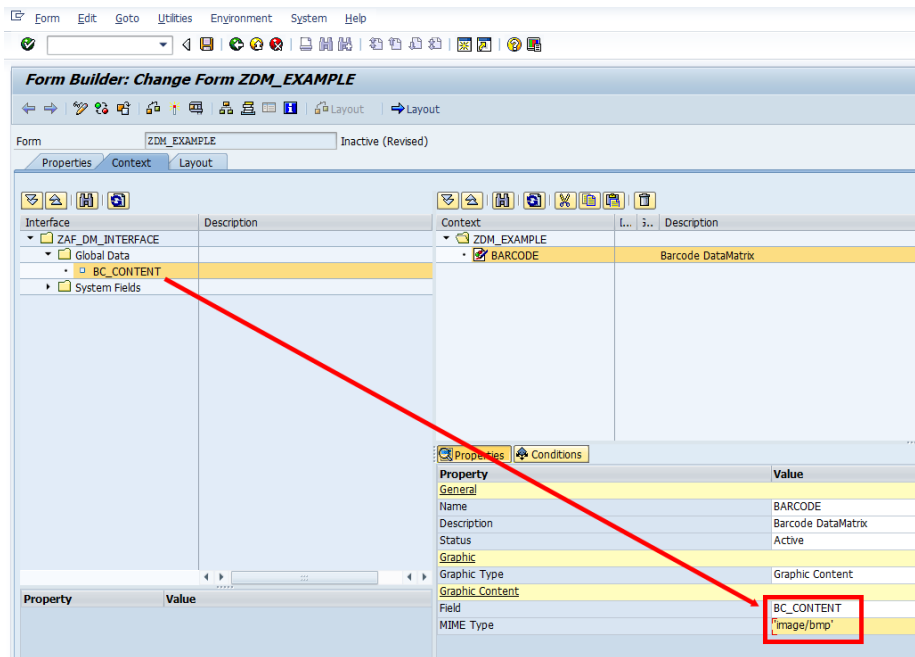
Assign a name and description to the created graphic object. In the example below it is **QRCODE1**.



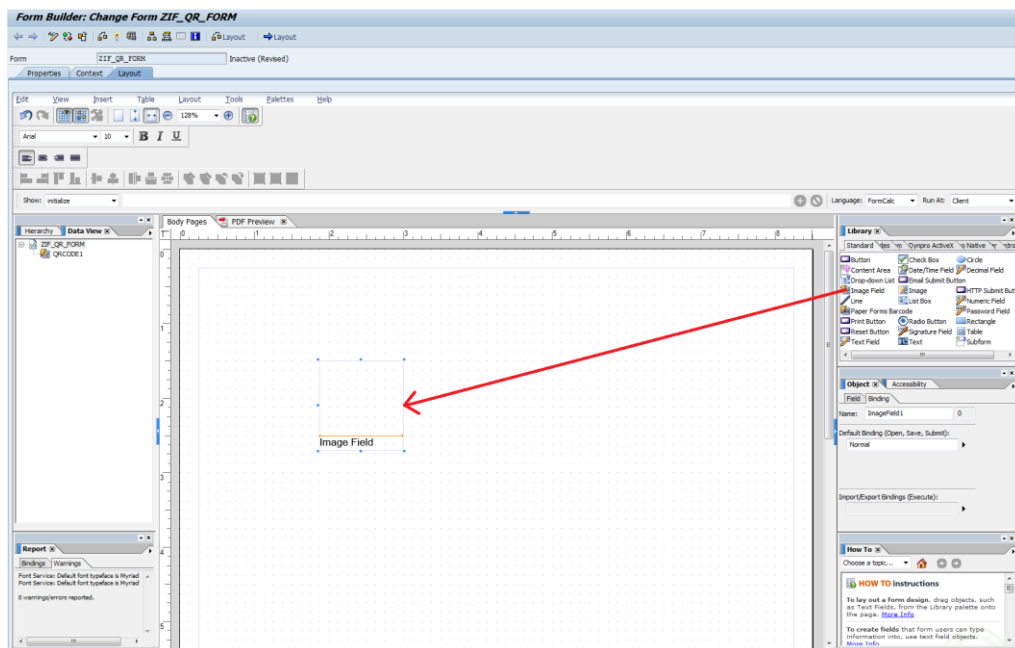
2. Now, change the graphic type to „graphic content“ and confirm the popup with „Yes“.



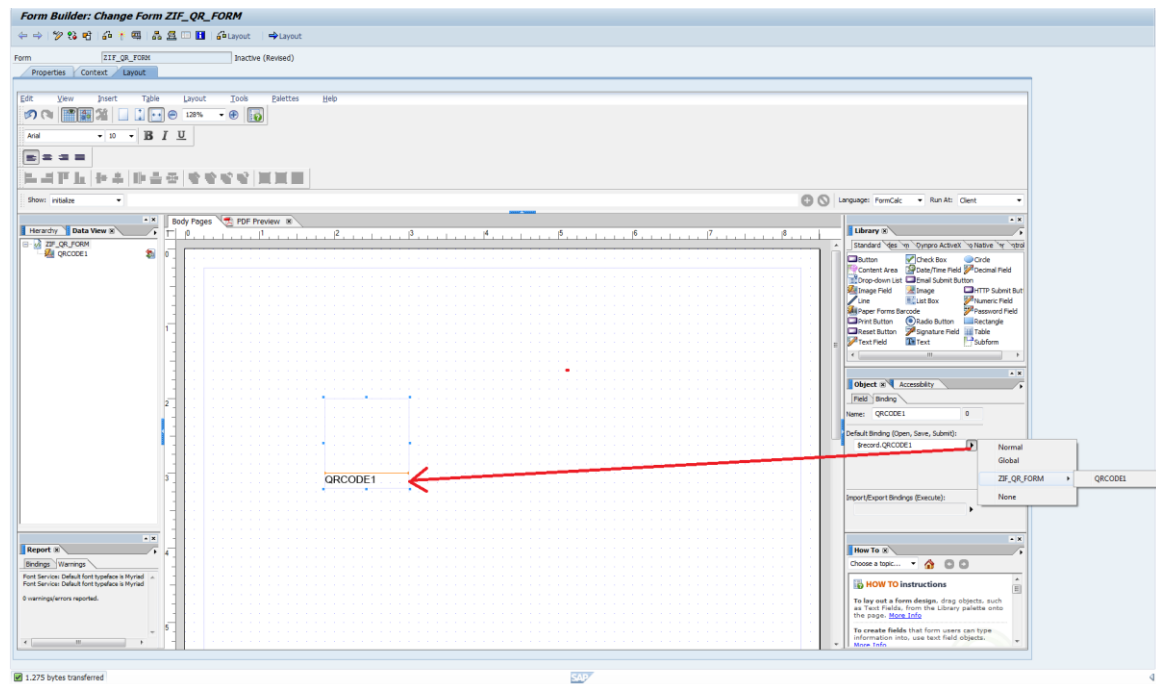
- Now, assign the graphic **QRCODE1** to the data field that contains the content of the barcode by dragging with the left mouse button pressed from the field **BC_CONTENT** from the interface into the appropriate field on the right. Provide it with **,Image/bmp'** as a **MIME-Type**.



- Change to the layout view of the Form Builders and create an element of the type „image field“.



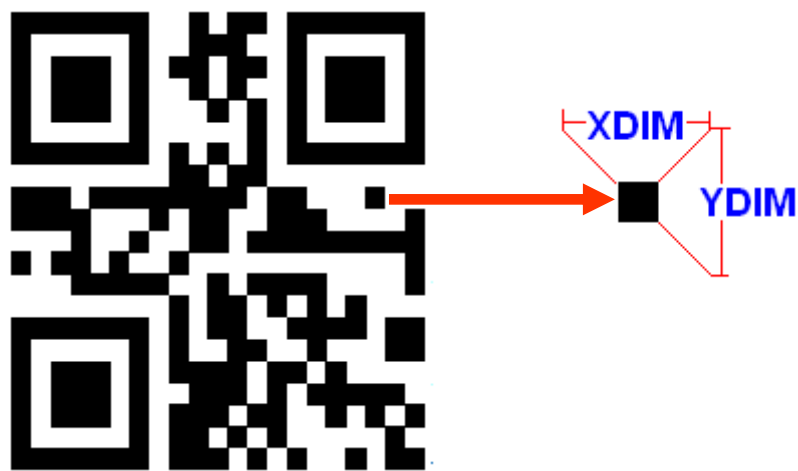
5. As a data connection provide the image field with the graphic node defined in the context (in the example: **QRCODE1**).



Ready!

The newly generated binding of the image field to the graphic **QRCODE1** will cause, that during the run time the QR-Code bar code generated by **RBarc/QRCode** will be inserted ON-THE-FLY into the form at the selected place.

6.7 Definition of bar code properties.



A QR bar code consists of black and white modules whose size is variable.

Note:

If Swiss QR Code is printed, the X_DIM parameter has a different meaning. To print the Swiss QRCode, the X_DIM parameter must have the value 1046. At this value the program recognizes that instead of the "normal" QR code the Swiss QR code should be generated. The Swiss QR Code is always printed in the size 46 x 46 mm and the module size is calculated automatically.








Following matrix size for a QR-Code label may be defined:






QRVersion	Anzahl Module	Datencodewords
0	auto	Je nach Datenmenge
1	21x21	26
2	25x25	44
3	29x29	70
4	33x33	100
5	37x37	134
6	41x41	172
7	45x45	196
8	49x49	242
9	53x53	192
10	57x57	346
11	61x61	404
12	65x65	466
13	69x69	632
14	73x73	581
15	77x77	655
16	81x81	733
17	85x85	815
18	89x89	901
19	93x93	991
20	97x97	1085
21	101x101	1156
22	105x105	1258
23	109x109	1364
24	113x113	1474
25	117x117	1588
26	121x121	1706
27	125x125	1828
28	129x129	1921
29	133x133	2051
30	137x137	2185
31	141x141	2323
32	145x145	2465
33	149x149	2611
34	153x153	2761
35	157x157	2876
36	161x161	3034
37	165x165	3196
38	169x169	3362
39	173x173	3532
40	177x177	3706





The table above shows the data codeword capacity of a QR-Code label depending on the selected version number (parameter **qversion**). Note, that this value does not reflect the capacity of real data to be encoded. QR-Code uses additional codewords to change automatically the encoding scheme, so that sometimes the capacity of real data is higher than the capacity of codewords, but sometimes it also could be lower. For more information refer to the official specification of the QR-Code bar code available at AIM.





Notice:

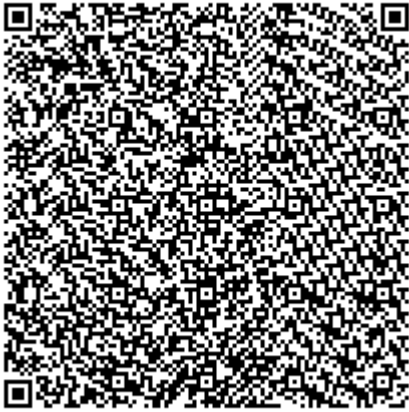
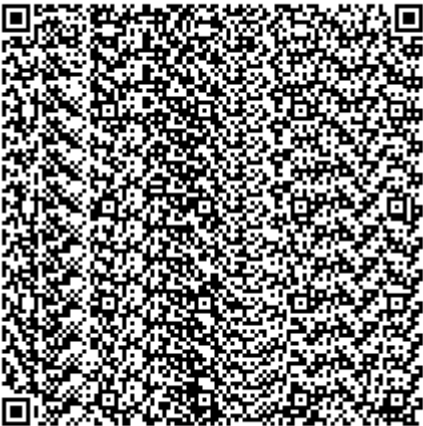
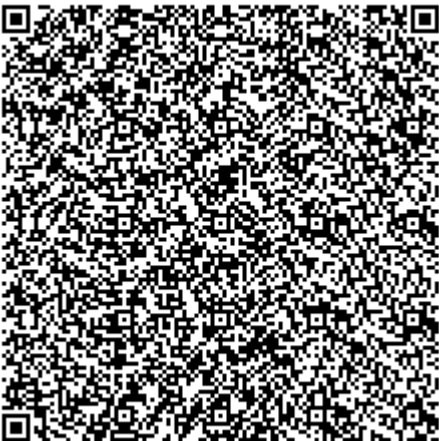
qversion = 0 causes, that the version number will be calculated automatically with the smallest possible value. The advantage here is, that the programmer doesn't have to care about the data size send to **RBarc/QRCode**. The disadvantage is, that the labels on different pages may be of different size, due to amount of data, which was encoded. If a fix bar code size is required, the programmer should calculate the maximum possible data size and then to define a bar code label of a fix size (e.g. **qversion = 5**). The disadvantage of this procedure is, that if the amount of calculated data codewords will exceed the defined bar code capacity, an error will occur.

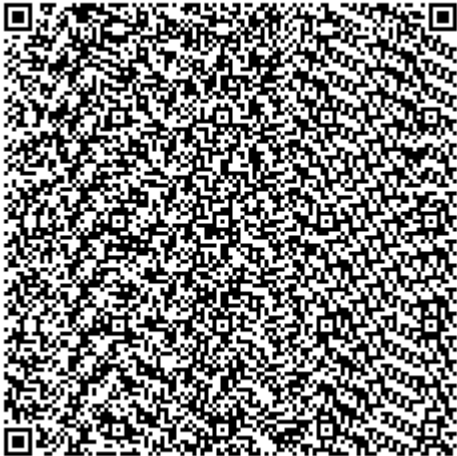
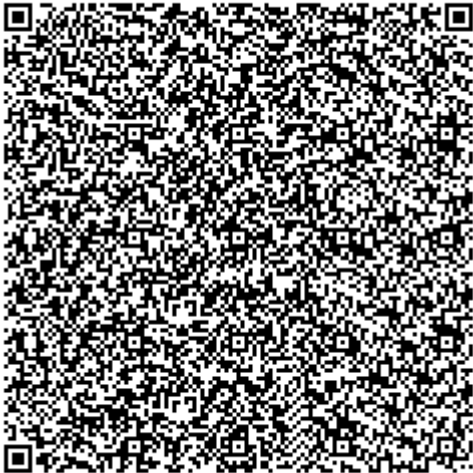
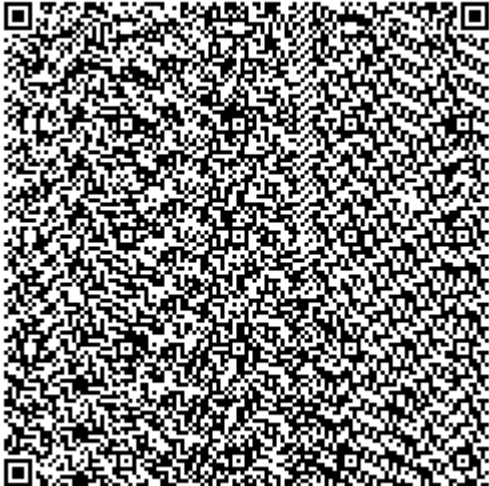
Version	Modules	Bar Code Example
1	21x21	
2	25x25	
3	29x29	
4	33x33	
5	37x37	
6	41x41	
7	45x45	

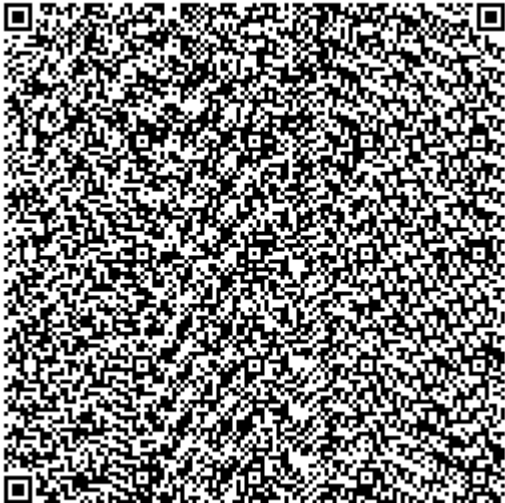
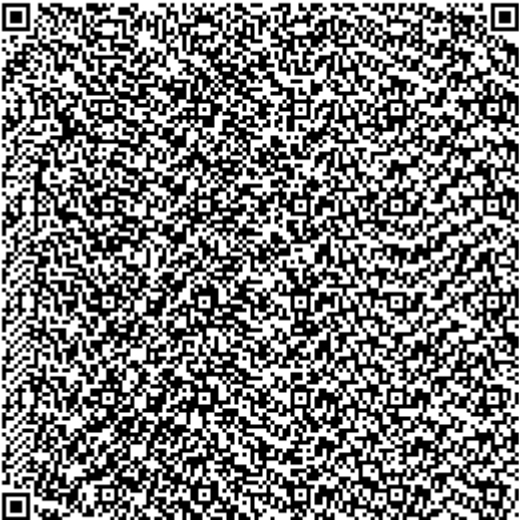
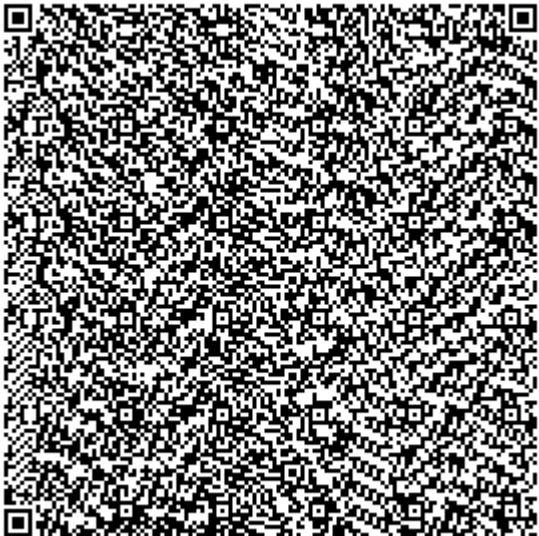
8	49x49	
9	53x53	
10	57x57	
11	61x61	
12	65x65	

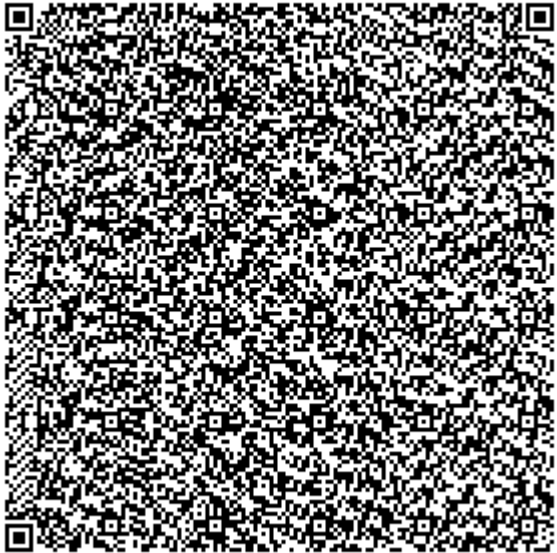
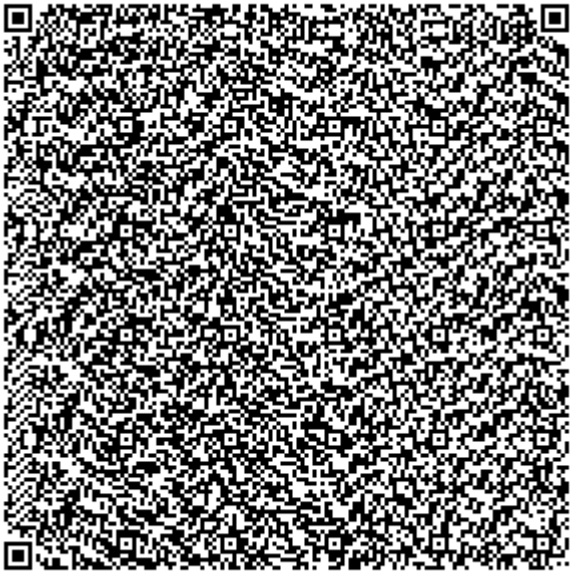
13	69x69	
14	73x73	
15	77x77	
16	81x81	

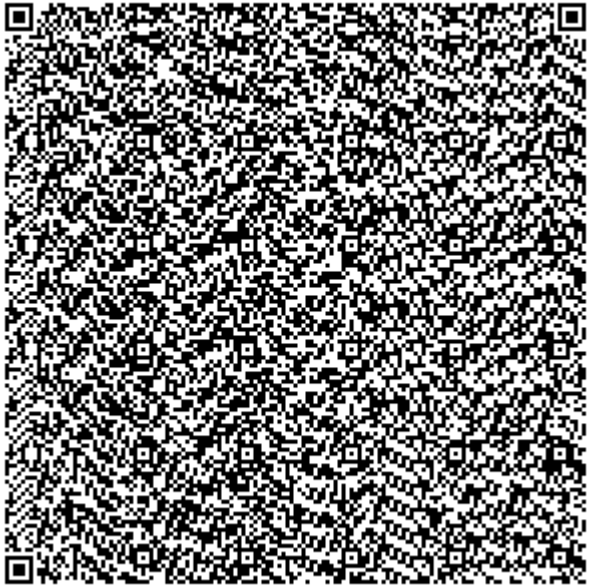
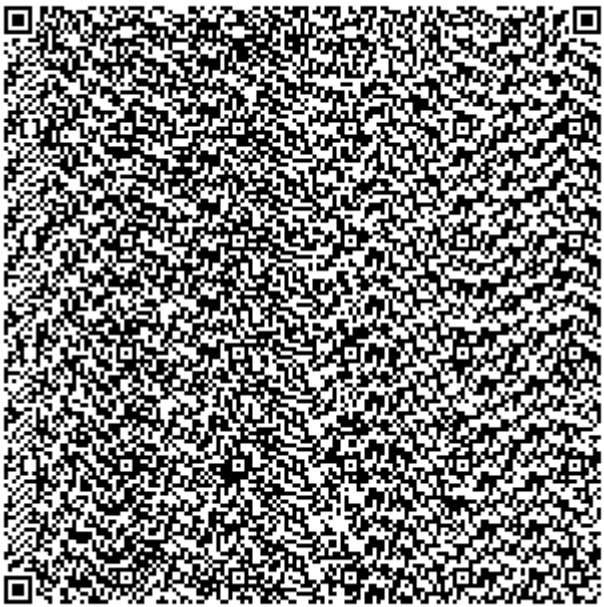
17	85x85	
18	89x89	
19	93x93	
20	97x97	

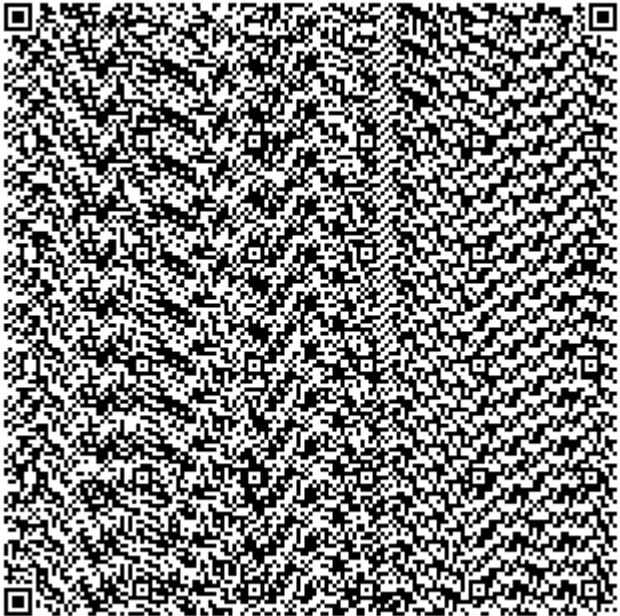
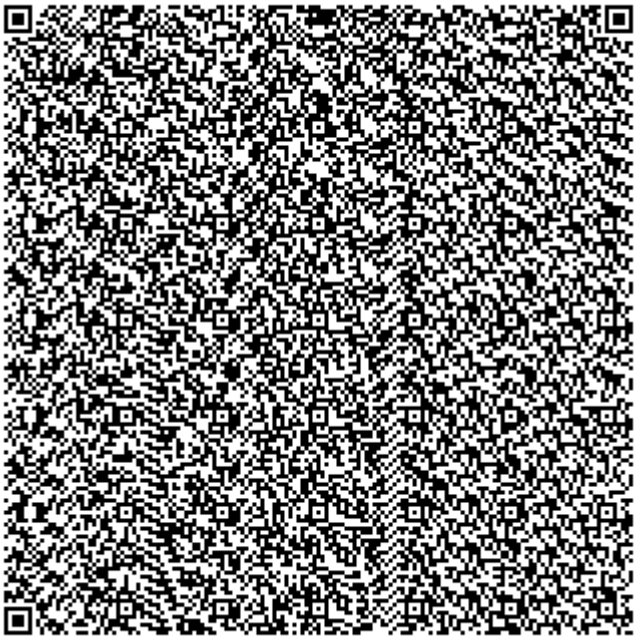
21	101x101	
22	104x104	
23	109x109	

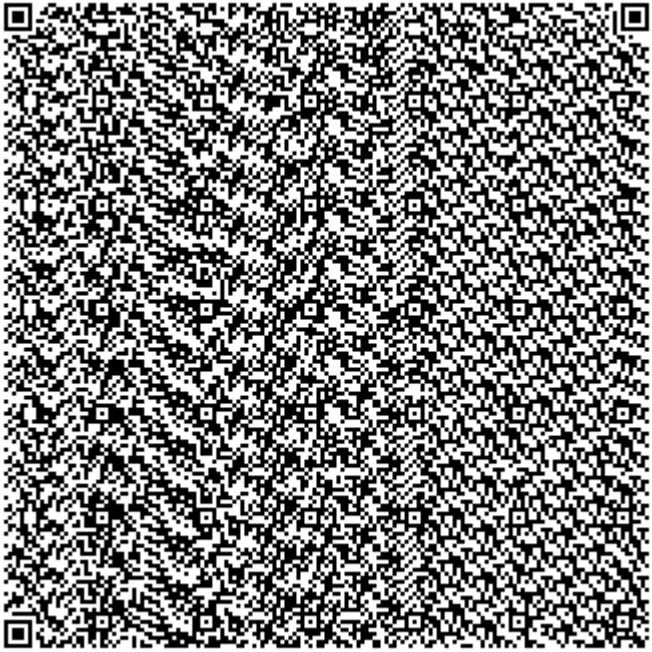
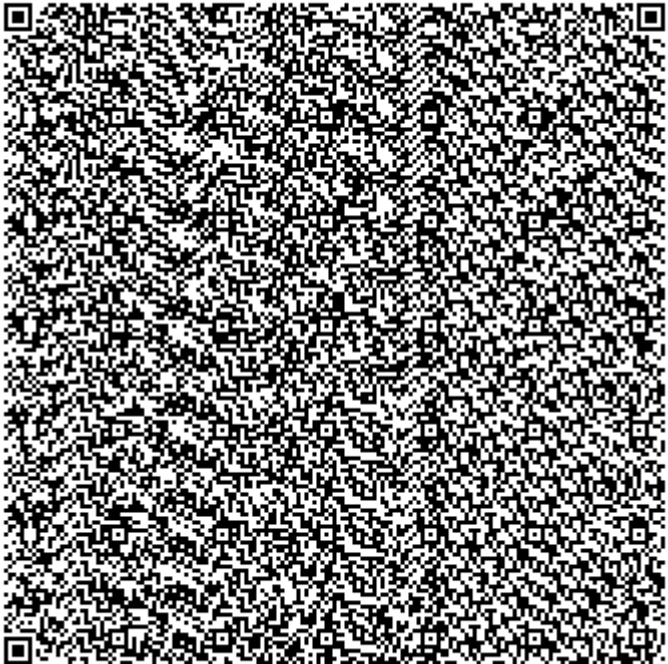
24	113x113	
25	117x117	
26	121x121	

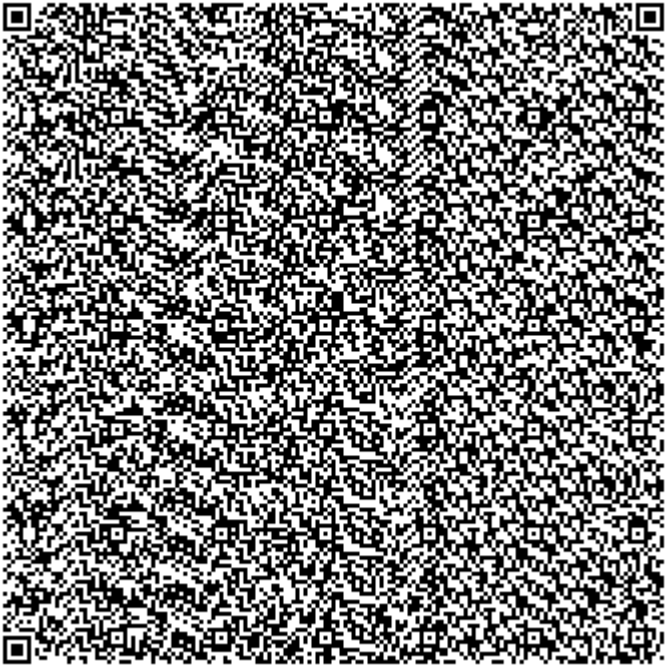
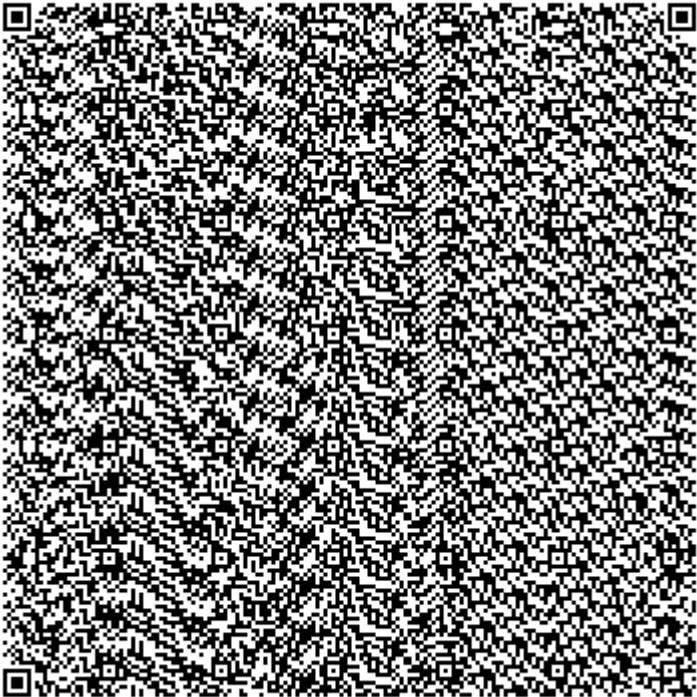
27	125x125	
28	129x129	
29	133x133	

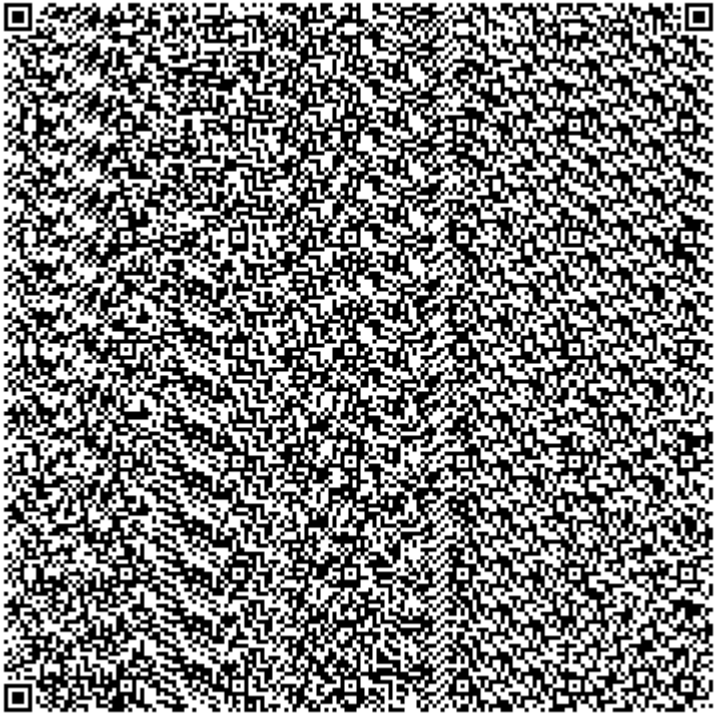
30	.137x137	
31	141x141	

32	145x145	
33	149x149	

34	153x153	
35	157x157	

36	161x161	
37	165x165	

38	169x169	
39	173x173	

40	177x177	
----	---------	--

7 Example of a job definition for a SAPscript Form.

7.1 Job

The following fields shall be encoded in a QR-Code bar code label:

VBELN from the table **LIKP**,
VBELN from the table **VBRK** und
ADDRNUMBER from the table **ADRT**.

The variables shall be separated by a **GS** (Group Separator). The bar code should be printed **50 Mm** right from the left border of the window, it was defined in. The module size shall be **1 Mm** wide and **1 Mm** heigh. The size of Matrix shall be calculated automatically

Solution:

1. Define the 3 variables for encoding as well as a bar code identifier in the **SAPscript** Form and pass their values in one step to the form routine **GEN_QR-CODE_SS** in the report **Z_SET_QRCODE**. Then define an include command line for the generated bar code and finally a delete command for the bar code.

```
/: DEFINE &QVAR1& = &LIKP-VBELN&  
/: DEFINE &DFLAG1& = 'T'  
/: DEFINE &QVAR2& = 'GS'  
/: DEFINE &DFLAG2& = 'F'  
/: DEFINE &QVAR3& = &VBRK-VBELN&  
/: DEFINE &DFLAG3& = 'T'  
/: DEFINE &QVAR4& = 'GS'  
/: DEFINE &DFLAG4& = 'F'  
/: DEFINE &QVAR5& = &ADRT-ADDRNUMBER&  
/: DEFINE &DFLAG5& = 'T'  
/: DEFINE &RES& = 150  
/: DEFINE &QVERSION& = 0  
/: DEFINE &ECC_LEVEL& = 'L'  
/: DEFINE &X_DIM& = 6  
/: DEFINE &AUTOHEIGHT& = 'N'
```



```
/: PERFORM GEN_QRCODE_SS IN PROGRAM Z_SET_QRCODE
```

```
/: USING &QVAR1&
```

```
/: USING &DFLAG1&
```

```
/: USING &QVAR2&
```

```
/: USING &DFLAG2&
```

```
/: USING &QVAR3&
```

```
/: USING &DFLAG3&
```

```
/: USING &QVAR4&
```

```
/: USING &DFLAG4&
```

```
/: USING &QVAR5&
```

```
/: USING &DFLAG5&
```

```
/: USING &RES&
```

```
/: USING &QVERSION&
```

```
/: USING &ECC_LEVEL&
```

```
/: USING &X_DIM&
```

```
/: CHANGING &QRNAME&
```

```
/: CHANGING &QRWIDTH&
```

```
/: CHANGING &RESULT&
```

```
/: ENDPERFORM
```

```
/: BITMAP &QRNAME& OBJECT GRAPHICS ID BMAP TYPE BMON XPOS &XPOS& MM
```

```
/: PERFORM DEL_QR_SS IN PROGRAM Z_SET_QRCODE
```

```
/: USING &QRNAME&
```

```
/: CHANGING RESULT
```

```
/: ENDPERFORM
```

In the implementation shown above, initially a type is assigned to each variable (eg. **&LIKP-VBELN&** was defined as text). The separator "**GS**" was designated as a function code with **DFLAG = 'F'**.

In the next step bar code properties **RES**, **QVERSION**, **ECC_LEVEL** and **X_DIM** were defined.

All defined variables and parameters were passed in one step to the form routine in **GEN_QRCODE_SS** in the program **Z_SET_QRCODE**

.

The graphic generated by **RBarc / QR-Code** with the name **&QRNAME&** is included in the form with the **BITMAP...** statement.

Finally, the image is deleted from the system by calling the form routine **DEL_QR_SS** in program **Z_SET_QRCODE**.

8 Example of a job definition for a Smart Forms Form.

8.1 Job.

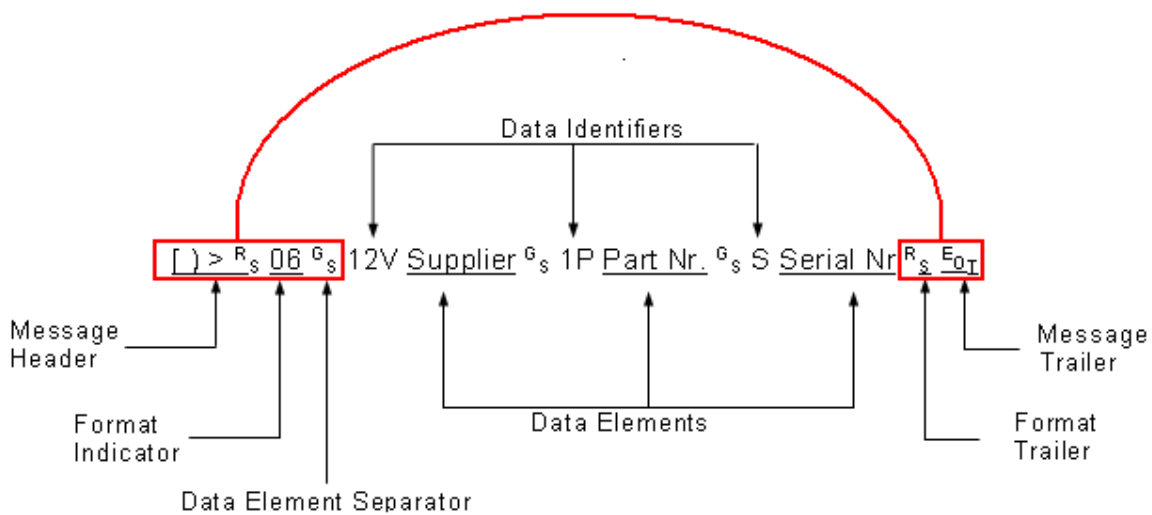
3 variables shall be encoded:

The supplier number "**SupplierNo**",

The part number "**PartNo**"

The serial number of the part "**SerNo**".

All variables shall be encapsulated in the **Message Envelop** (see picture below). The module shall be **1 Mm** wide and **1 Mm** high. The matrix size shall be calculated automatically.



The "**Message Envelop**" is a standardized way of data transfer to the ANSI standard MH10.8.7. The encoding data will be preceded by a message header – up to the standard "[] > R_s 06 G_s" or "[] > R_s 05 G_s" end finished by the Trailer R_s E_{0T}. Each variable will be preceded by an indicator (eg. 12V for Supplier) and all variables will be separated by the fields separator G_s.

Solution:

1. First create a bar code window in the Smart Forms form with 3 nodes as described in chapter [QR-Code printing with a Smart Forms form](#)
2. Only the first node have to be adapted to a new job. The node 2 and 3 remain unchanged.

go to the next page.

3. Enter the following program code:

Data declaration

```
data: inputdata type table of input_data with header line.  
clear inputdata.  
refresh inputdata.
```

Encoding of '</' as Text

```
inputdata-qrvar = '(>'.  
inputddata-dflag = 'T'.  
append inputdata.
```

Encoding of 'RS' as function code

```
inputdata-qrvar = 'RS'.  
inputddata-dflag = 'F'.  
append inputdata.
```

Encoding of '06' as Text

```
inputdata-qrvar = '06'.  
inputdata-dflag = 'T'.  
append inputdata.
```

Encoding of 'GS' as function code

```
inputdata-qrvar = 'GS'.  
inputdata-dflag = 'F'.  
append inputdata.
```

Encoding of '12V' + suplierno as Text

```
concatenate '12V' suplierno into inputdata-qrvar.  
inputdata-dflag = 'T'.  
append inputdata.
```

Encoding of 'GS' as function code

```
inputdata-qrvar = 'GS'.  
inputddata-dflag = 'F'.  
append inputdata.
```

Encoding of '1P' +partno as Text

```
concatenate '1P' partno into qrvar.  
inputdata-dflag = 'T'.  
append inputdata.
```

Encoding of 'GS' as function code

```
inputdata-qrvar = 'GS'.  
inputddata-dflag = 'F'.  
append inputdata.
```

Encoding of 'S' + serialno as Text

```
concatenate 'S' serialno into qrvar.  
inputdata-dflag = 'T'.  
append inputdata.
```

Encoding of 'RS' as function code

```
inputdata-qrvar = 'RS'.  
inputdata-dflag = 'F'.  
append inputdata.
```

Encoding of 'EOT' as function code

```
inputdata-qrvar = 'EOT'.  
inputdata-dflag = 'F'.  
append inputdata.
```

Definition of bar code properties

```
read table inputdata index 1.  
inputdata-res = 150.  
inputdata-qversion = 0.  
inputdata-ecc_level = 'L'  
inputdata-x_dim = 4.  
modify inputdata index 1.
```

Passing the interface table inputdata to z_set_qrcode

```
perform gen_qrcode_sf in program z_set_qrcode  
tables inputdata  
changing qrcode qwidth result.
```

In the implementation shown above, initially a type is assigned to each variable (eg **12 + SUPPLIERNO** defined as text). The separator "**GS**" was defined as a function code with the type designation **DFLAG = 'F'**..

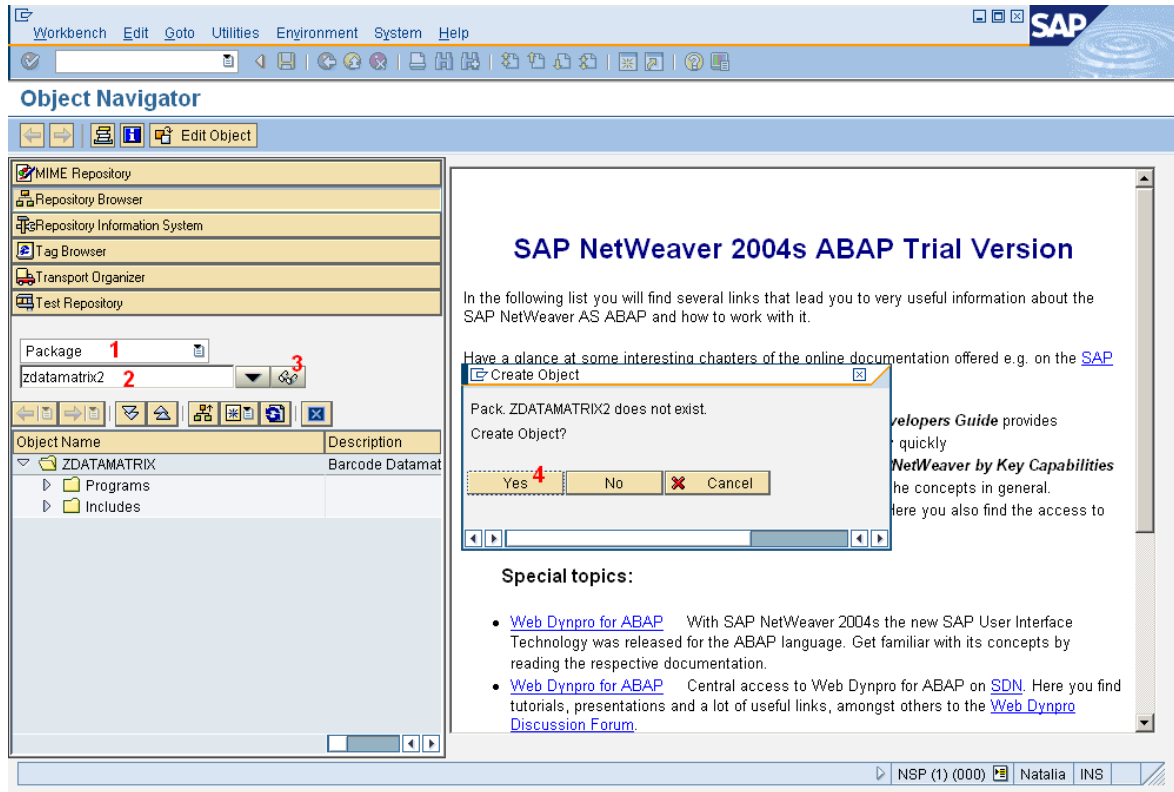
In the next step bar code properties with **RES**, **QVERSION**, **ECC_LEVEL** and **X_DIM** defined and written into the first record of interface table **INPUTDATA**.

The interface table **INPUTDATA** is passed to the form routine **gen_qrcode_sf** in the program **z_set_qrcode**.

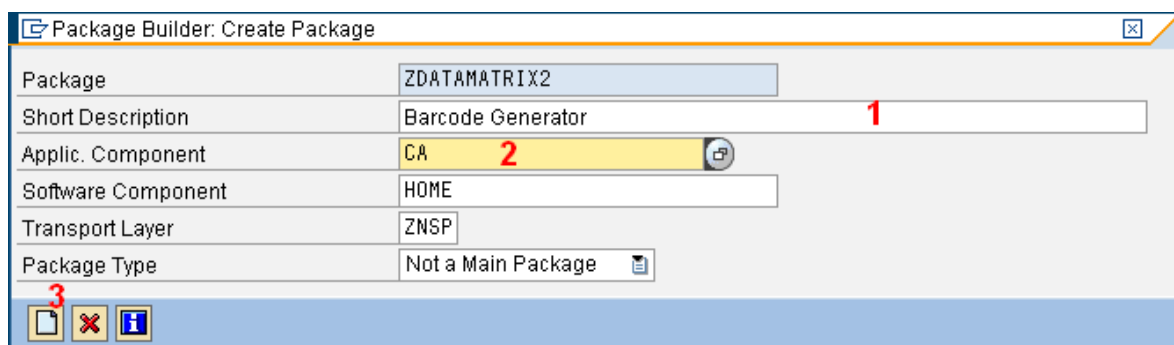
The return parameter is the name of the bar code **QRNAME**. The bar code is dynamically included into the form during further form processing.


9 Annex 1: Creating a new package (development class).

1. Start the transaction **SE80 (Object Navigator)**.
2. The following dialog appears:




3. Select the object „Package“ (1) (on older Systems "development class").
4. Enter the object name **ZQRCODE** (2).
5. Click on the symbol "glasses" (3).
6. It appears a message with the information, that the package doesn't exist and asks you, if you want to create it. Click on **Yes** (4).
7. The following dialog appears:



8. Enter a short description (1).
9. Select „CA“ (2) as application component.
10. Click on the symbol  (3) to create the new package.

11. The prompt for transportable workbench request appears.

The dialog box has a title bar with a close button. It contains three input fields: 'Package' with the value 'ZDATAMATRIX2', 'Request' with the value 'NSPK900050' (highlighted in yellow), and 'Short Description' with the value 'RBarc2006'. Below the fields is a toolbar with a green checkmark, a floppy disk icon, a document icon, a button labeled 'Own Requests', and a red X icon. A red number '1' is placed above the document icon.

12. If no request for this installation has been created yet, create the request, otherwise go to pt. 16.
13. Click on the symbol  to create the new request.

The dialog box has a title bar with a close button. It contains several input fields: 'Request' with the value 'Workbench request', 'Short description' with the value 'Datamatrix' (highlighted in yellow), 'Project' (empty), 'Owner' with the value 'BCUSER', 'Status' with the value 'New', 'Last changed' with the value '04.05.2007 21:42:36', 'Source client' with the value '000', and 'Target' with the value 'CL5'. Below these fields is a 'Tasks' section with a list box containing 'User' and 'BCUSER'. At the bottom is a toolbar with a floppy disk icon, a document icon, and a red X icon. A red number '2' is placed above the floppy disk icon.

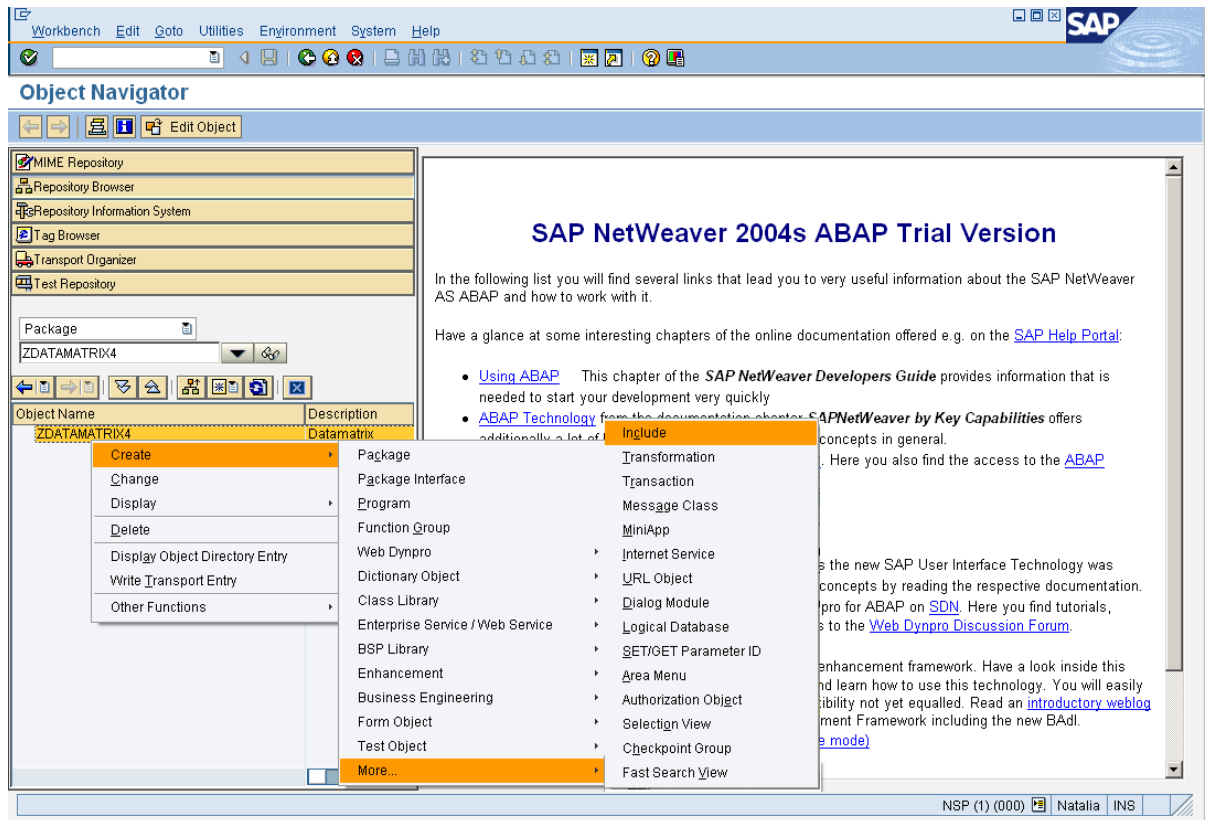
14. Enter a short description for the request (1).
15. Click on the floppy symbol (2) to save the request.
16. The request number appears in the field „Request“

The dialog box is the same as in step 11, but the 'Request' field now contains the value 'NSPK900053' (highlighted in yellow) and the 'Short Description' field contains the value 'Datamatrix'. A red number '1' is placed above the green checkmark icon in the toolbar.

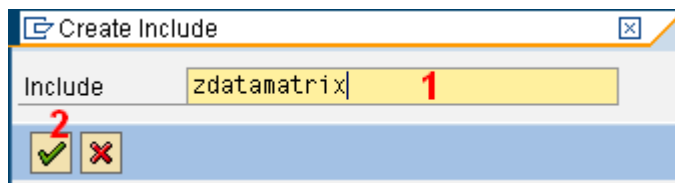
17. Click on the green check (1) to save the request.

10 Annex 2: Creating an include in the ABAP Workbench .

1. Start the **Object Navigator** (transaction **SE80**) and select the new Package (development class) **ZQRCODE**.
2. Select the object name **ZQRCODE** and click on the right mouse button.
3. Select from the context menu **Create/More/Include** (on older systems **Include** appears already on the second place).



4. Enter in the following dialog the name of the include, which has to be created (1) and click on the green check (2).



5. Click on the button "Save" (1) to create the include.

ABAP: Program Attributes ZDATAMATRIX2 Change

Title

Original language English

Created

Last changed by

Status

Attributes

Type

Status

Application

Authorization Group

☐ Editor lock

1 Save

6. Take care to save the include in the new package (1) and click on the floppy symbol (2) to save the include.

Create Object Directory Entry

Object

Attributes

Package 1

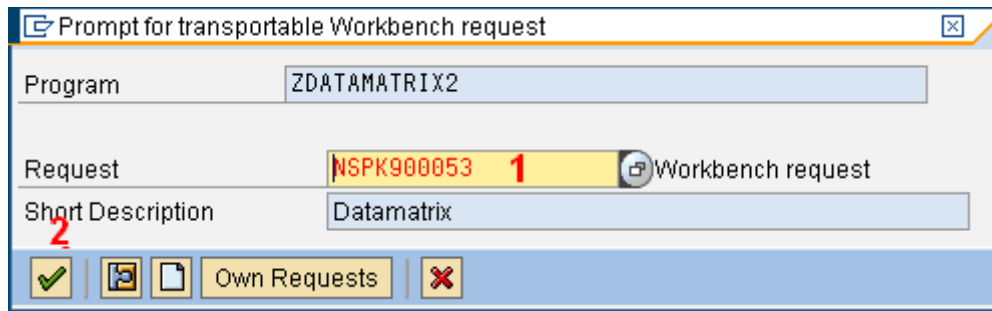
Person Responsible

Original System

Original language English

2 Local Object Lock Overview

7. The prompt for transportable workbench request appears. The request should be at this point already present, because it was created during saving the new Package (development class) **ZQRCODE**. The request number should be the same as before (1).



Prompt for transportable Workbench request

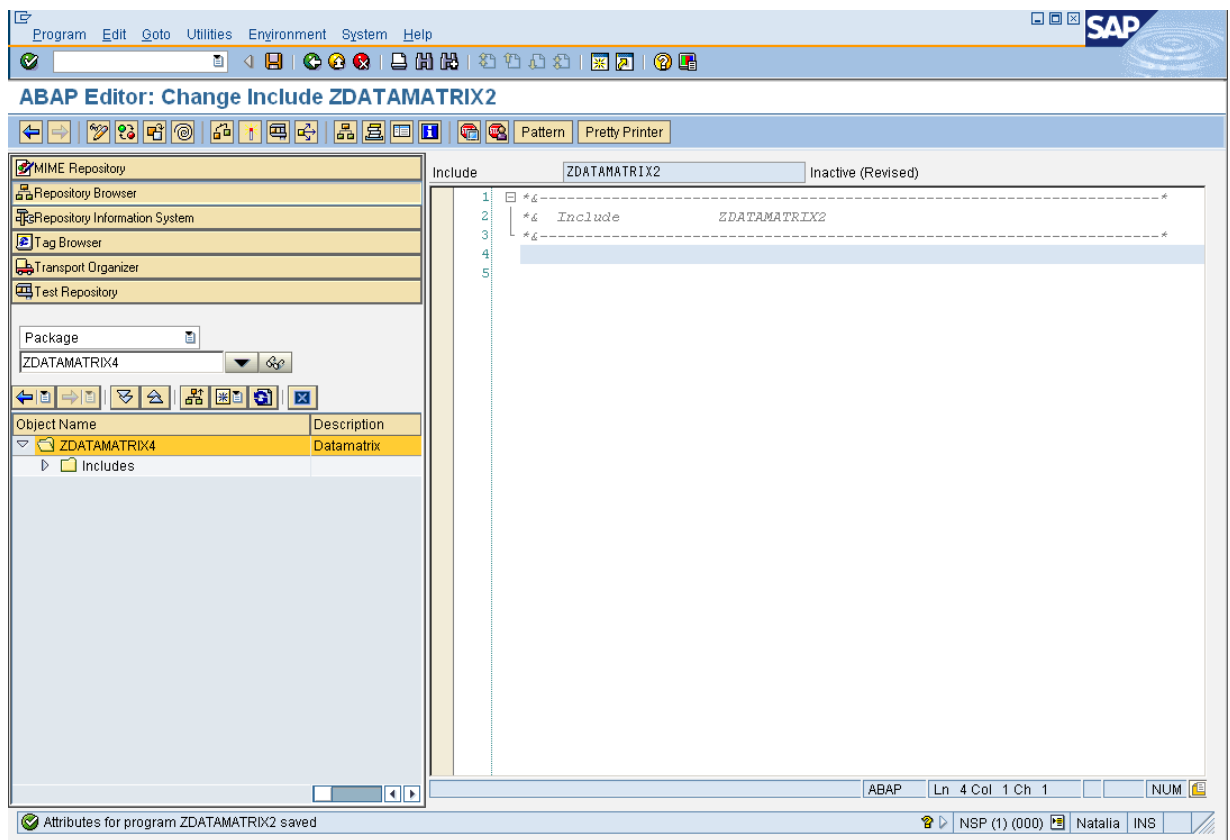
Program: ZDATAMATRIX2

Request: NSPK900053 1 Workbench request

Short Description: Datamatrix

Buttons: [Green Checkmark], [Icon], [Icon], Own Requests, [Red X]

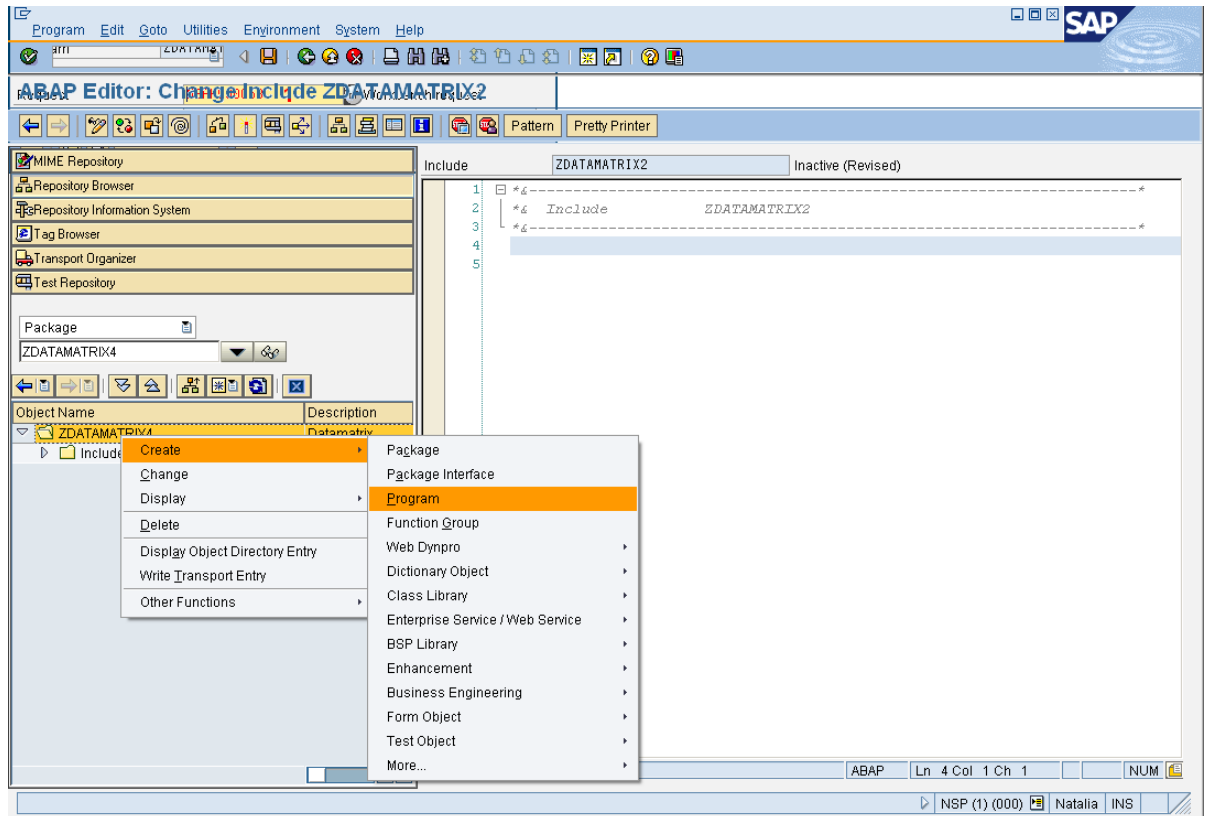
8. Click on the green check (2), to save the request.
9. Now the include will be finally saved and the window with the source code will appear.



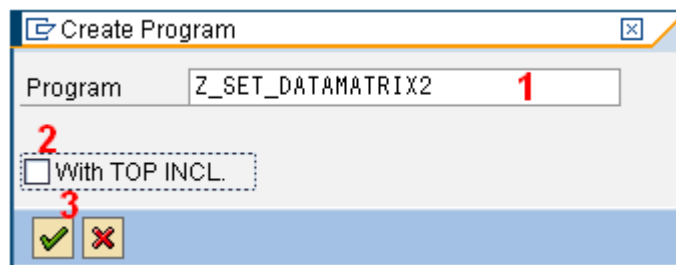
10. Delete the content of the source code, which was generated automatically.
11. Copy and paste the source code from the appropriate installation file and save the include. (go to page 60: [Annex 4: inserting content of installation files](#) into an ABAP Program.)

11 Annex 3: Creating a program in the ABAP workbench.

1. Start the **Object Navigator** (transaction **SE80**) and select the new Package (development class) **ZQRCODE**.



2. Select the object **ZQRCODE** and click on the right mouse button.
3. Select from the context menu **Create/Program**.
4. Enter in the next dialog the name of the new program (1), deselect the check box „With TOP-Include“ (2) and click on the green check (3).



5. Click on the button "Save" (1) to create the new program.

ABAP: Program Attributes Z_SET_DATAMATRIX2 Change

Title **Report Z_SET_DATAMATRIX2**

Original language **EN** English

Created **04.05.2007** **BCUSER**

Last changed by

Status **New(Revised)**

Attributes

Type **Executable program**

Status

Application

Authorization Group

Logical database

Selection screen

☐ Editor lock ☒ Fixed point arithmetic

☒ Unicode checks active ☐ Start using variant

1

Save

6. Take care to save the program in the new package (development class) **ZQRCODE** (1) and click on the floppy symbol (2) to save the program.

Create Object Directory Entry

Object **R3TR PROG Z_SET_DATAMATRIX2**

Attributes

Package **ZDATAMATRIX4** **1**

Person Responsible **BCUSER**

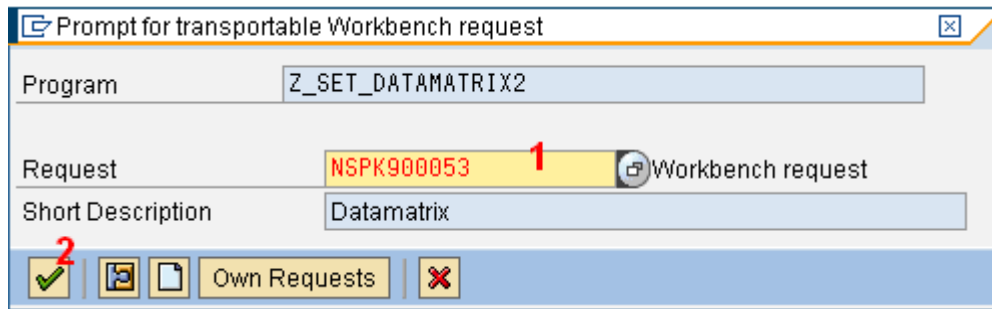
Original System **NSP**

Original language **EN** English

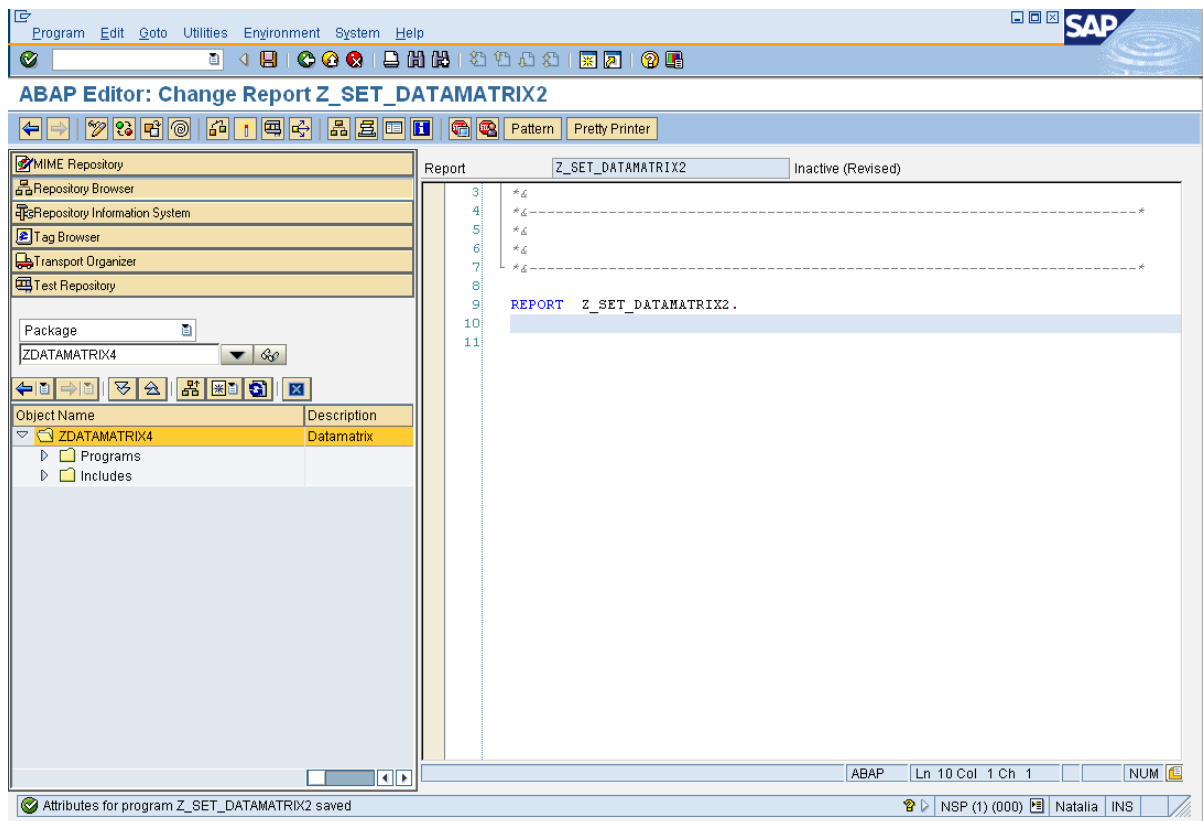
2

Local Object Lock Overview

7. The prompt for transportable workbench request appears. The request should be at this point already present, because it was created during saving the new Package (development class) **ZQRCODE**. The request number should be the same as before (1)..



8. Click on the green check (2), to save the request.
9. Now the include will be finally saved and the window with the source code will appear.

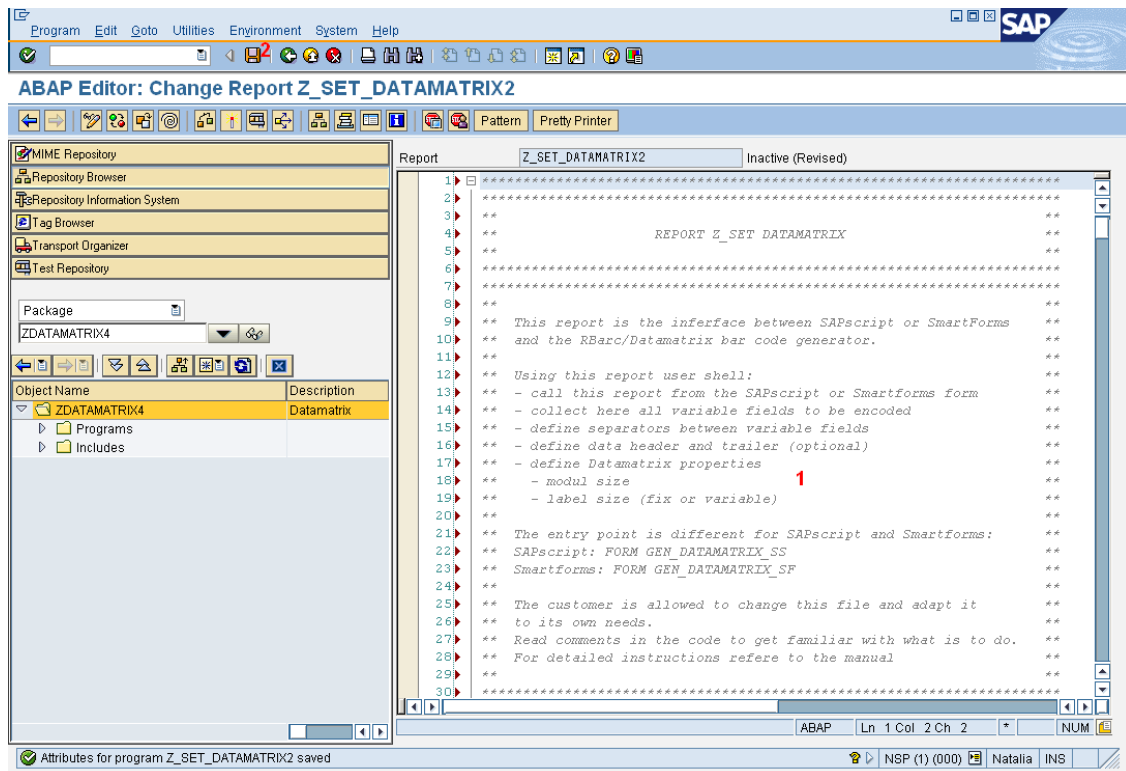


12. Delete the content of the source code, which was generated automatically.
10. Copy and paste the source code from the appropriate installation file and save the include.
(go to page 60: [Annex 4: inserting content of installation files](#) into an ABAP Program.)

12 Annex 4: inserting content of installation files into an ABAP Program.

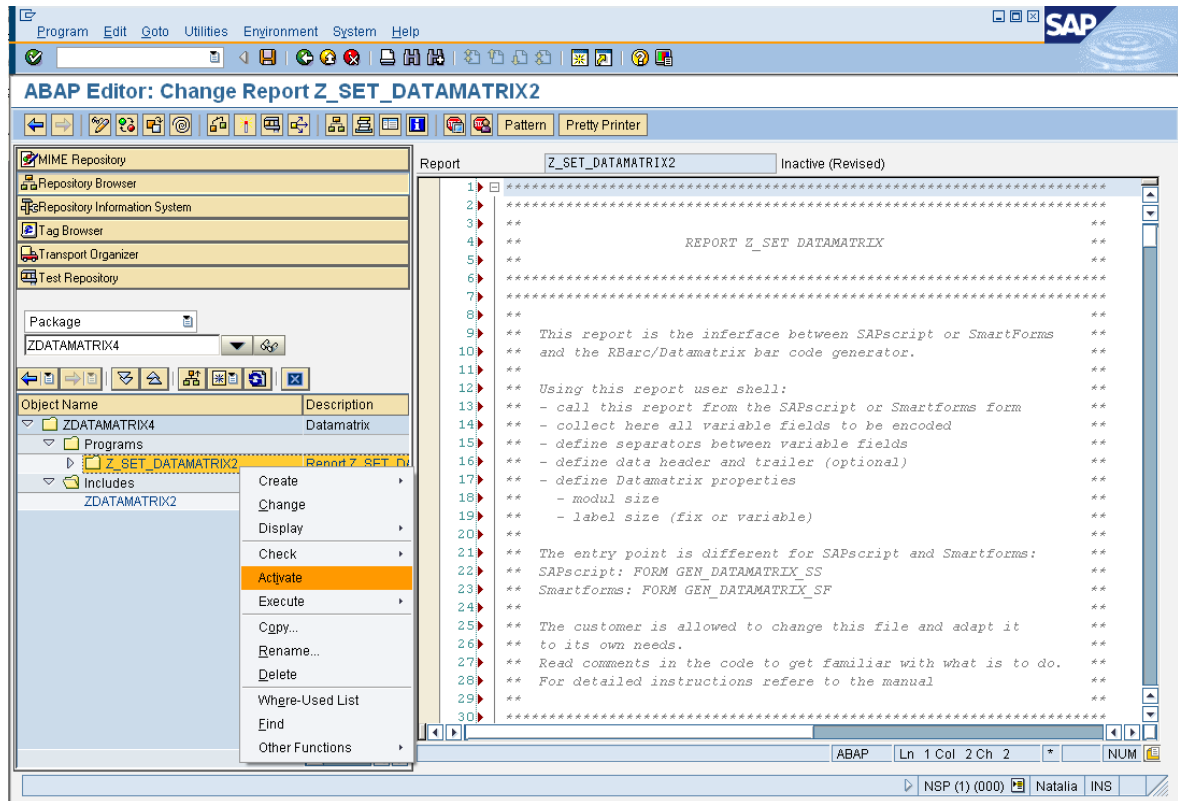
This procedure concerns as well programs as includes. As example we use **Z_SET_QRCODE**.

1. Open the file **Z_SET_QRCODE.PRG** from the subdirectory **Install** with Windows Notepad (or other editor if another system as windows is used).
2. Select the content of the file with **Ctrl+A**.
3. Copy the selection into the clipboard by pressing **Ctrl+C**.
4. Change to SAP GUI to the opened program **Z_SET_QRCODE**.
5. Click with the left mouse button into the window with the ABAP source code (1) and paste the content of the clipboard by pressing **Ctrl+V**.
6. Click on the floppy symbol (2), to save the source code.

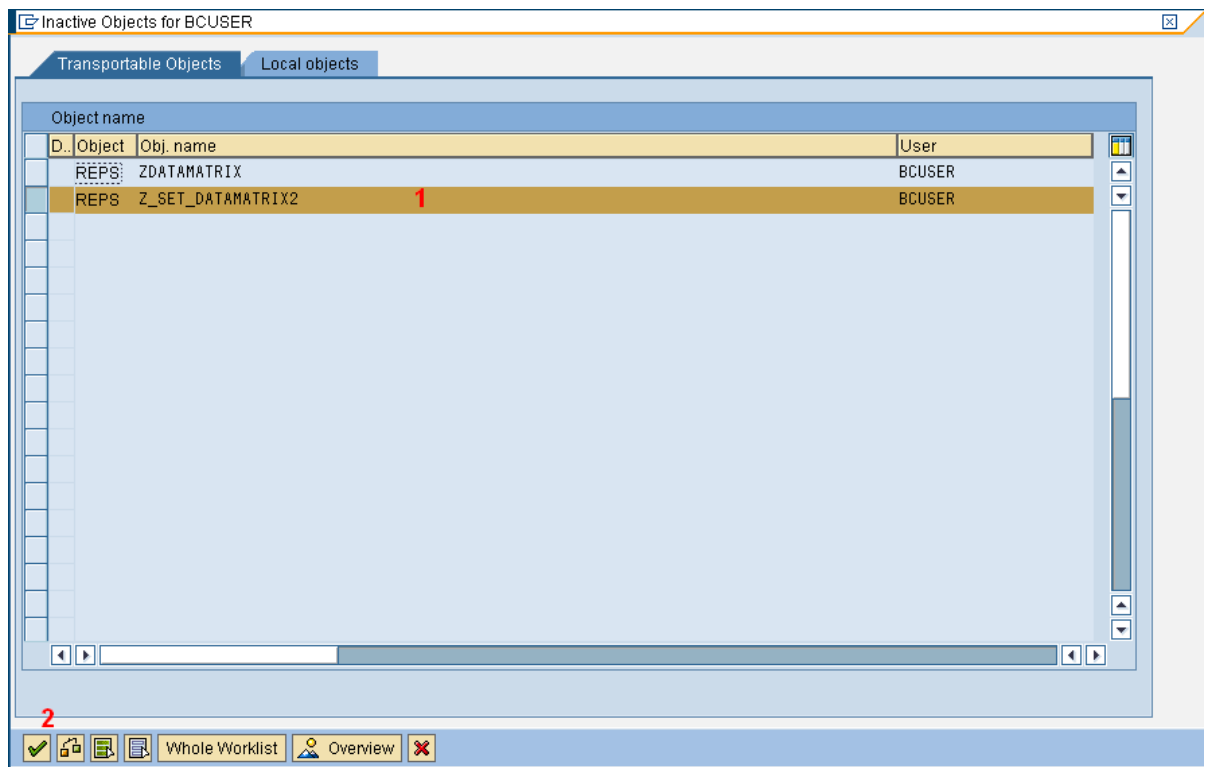


13 Annex 5: Activating of programs and includes

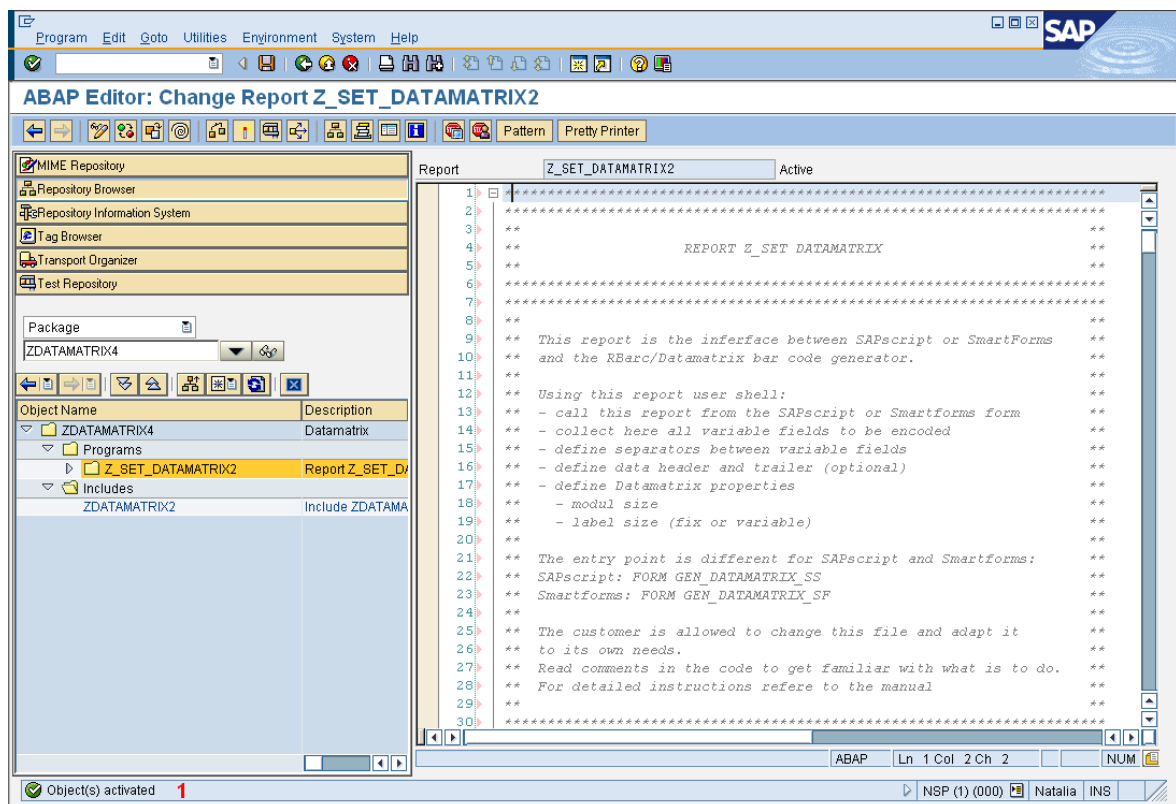
1. Open the **Object Navigator** (transaction **SE80**).
2. Select the object type „**Package**“
3. Enter the package name **ZQRCODE** .
4. The tree nodes „**Programs**“ and „**Includes**“ appears.
5. Expand the tree node **Programs** (1) und **Includes** (2).
6. Select the object, which shall be activated, click on the right mouse button and select „**Activate**“



7. The list of all inactive object appears. The object to be activated is already selected (1).click on the green check (2) to proceed the activating.

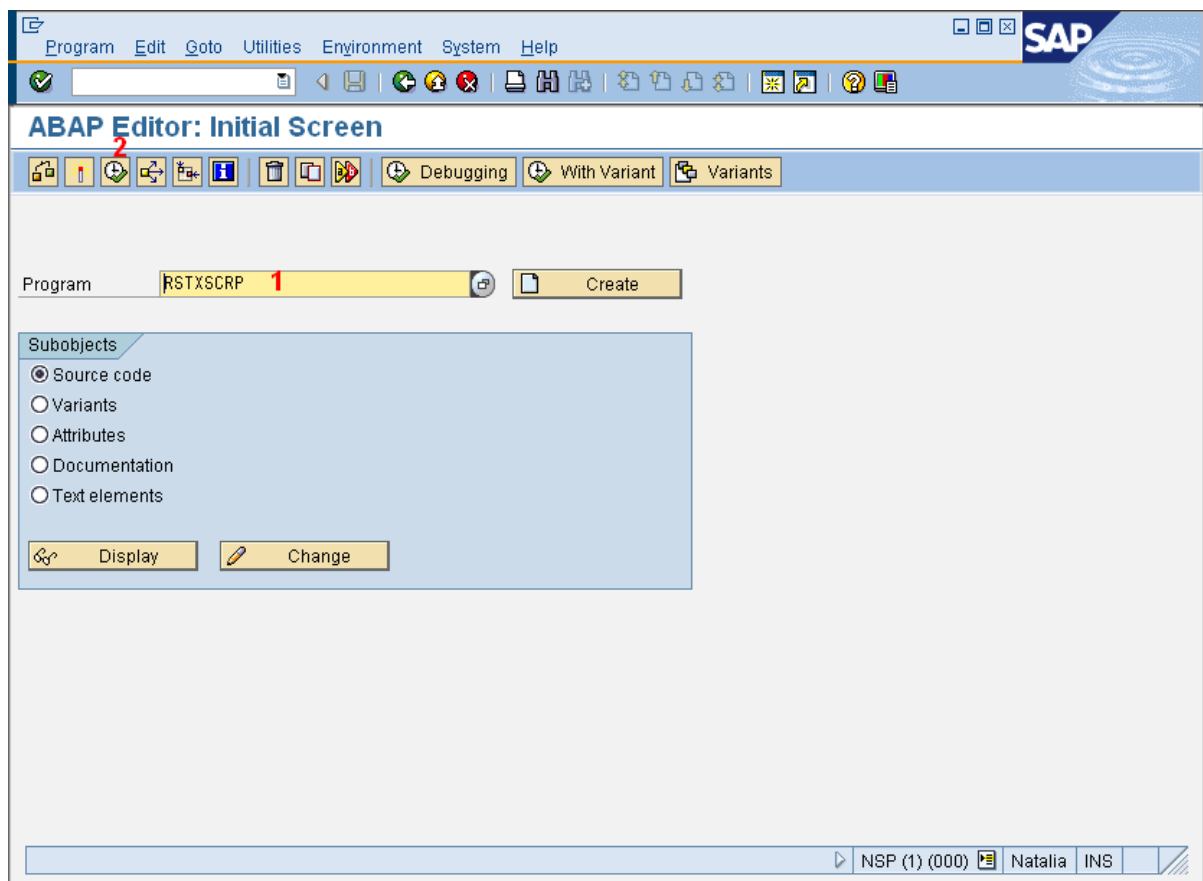


8. After the activating, a message appears in the status line (1). Programs can work only after a successful activating.



14 Anhang 6: Executing of an ABAP program.

1. Start the transaction **SE38**.
2. Enter the name of the program in the field „**Program**“ (1).
3. Click on the execute symbol in the task bar (2).



15 Annex 7: Importing of a SAPscript form .

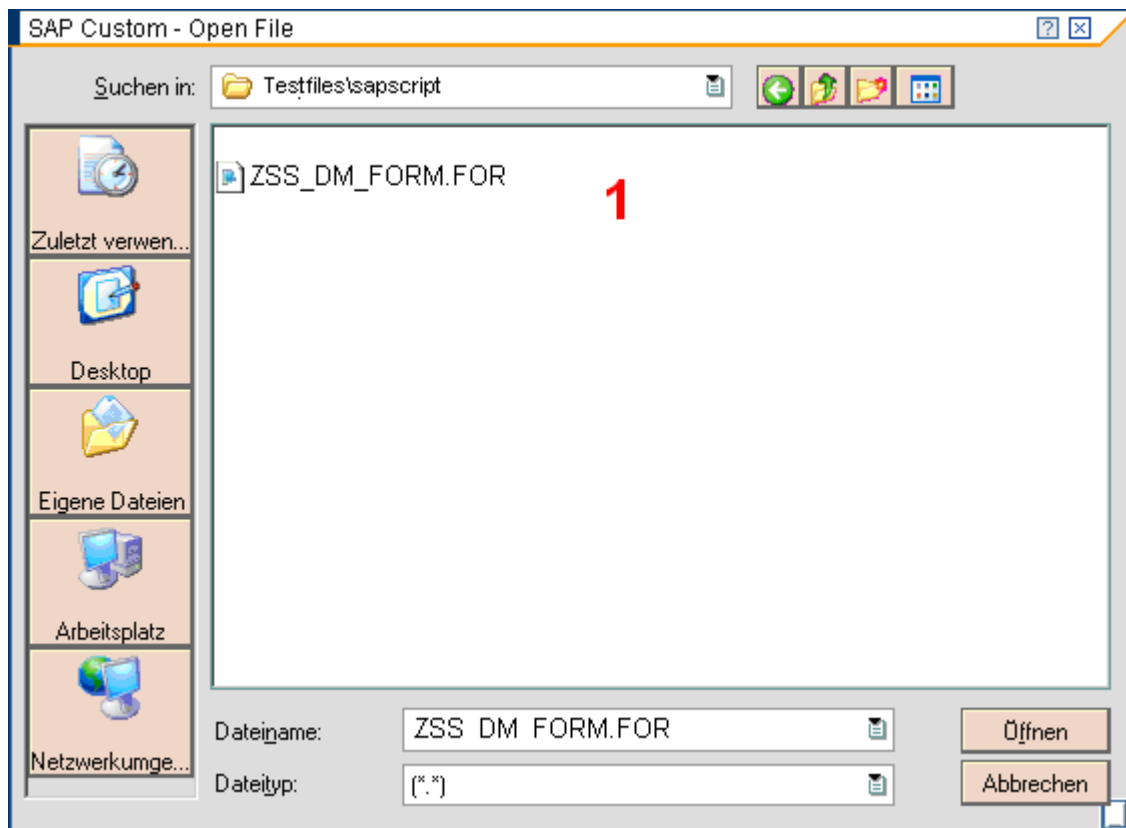
1. Start the SAP program **RSTXSCR**.
(Go to: [Anhang 6: Executing of](#) an ABAP program)
2. Select the object type „**Form**“ (1).
3. Enter **ZSS_QR_FORM** in the field "Object name" (2).
4. Enter the mode „**IMPORT**“ (3).
5. Click on the symbol Execute (4)

The screenshot shows the SAPscript Export to Dataset / SAPscript Import from Dataset dialog box. The title bar includes the SAP logo and window controls. The main area is divided into three sections: "Object selection and session ctrl", "Ctrl parameters for file operation", and "Control of language versions". In the "Object selection and session ctrl" section, the "Form" radio button is selected (1). The "Object name" field contains "ZSS_DM_FORM" (2). The "Mode (EXPORT/IMPORT)" dropdown is set to "IMPORT" (3). The "Ctrl parameters for file operation" section shows "From/on frontend" selected. The "File Name" field contains "C:\temp*****&&&". The "Control of language versions" section has an empty "Language vector" field. The status bar at the bottom shows "NSP (1) (000)", "Natalia", and "INS".

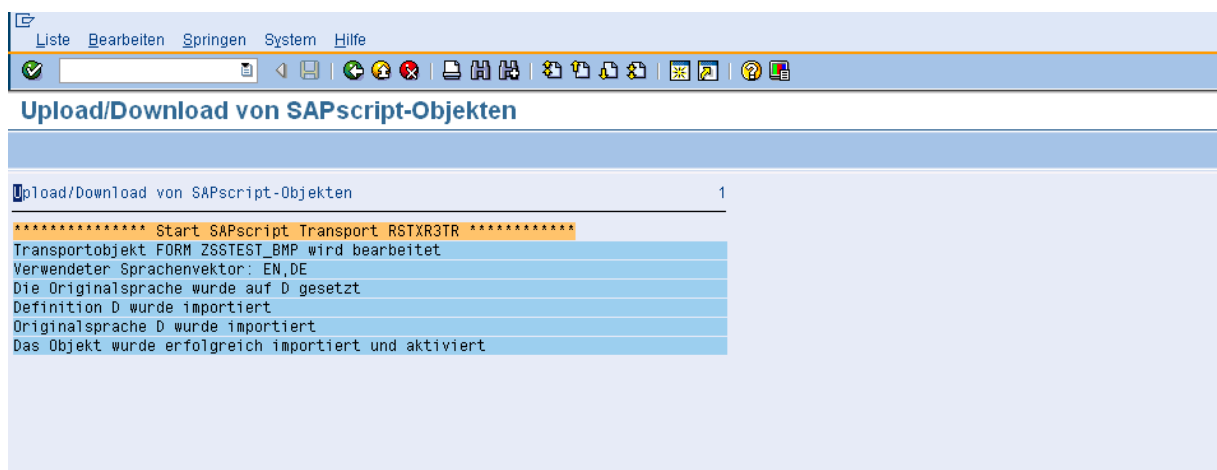
6. The dialog "Create Object Directory Entry" appears . Add the form to the package **ZQRCODE** (1). Click on the floppy symbol (2) to save the entry.

The screenshot shows the "Create Object Directory Entry" dialog box. The title bar includes the SAP logo and window controls. The "Object" field contains "R3TR FORM ZSS_DM_FORM2". The "Attributes" section has the "Package" field set to "ZDATAMATRIX4" (1). The "Person Responsible" field contains "BCUSER". The "Original System" field contains "NSP". The "Original language" field is empty. At the bottom, there is a floppy disk icon (2) and buttons for "Local Object", "Lock Overview", and a close button.

7. The prompt for transportable workbench request papers. Use the existing request number (1) and click on the green check (2).
8. The file selection dialog appears. Select the file **ZSS_QR_FORM.FOR** from the subdirectory **TestForms** (1) and click on the button **Open**.

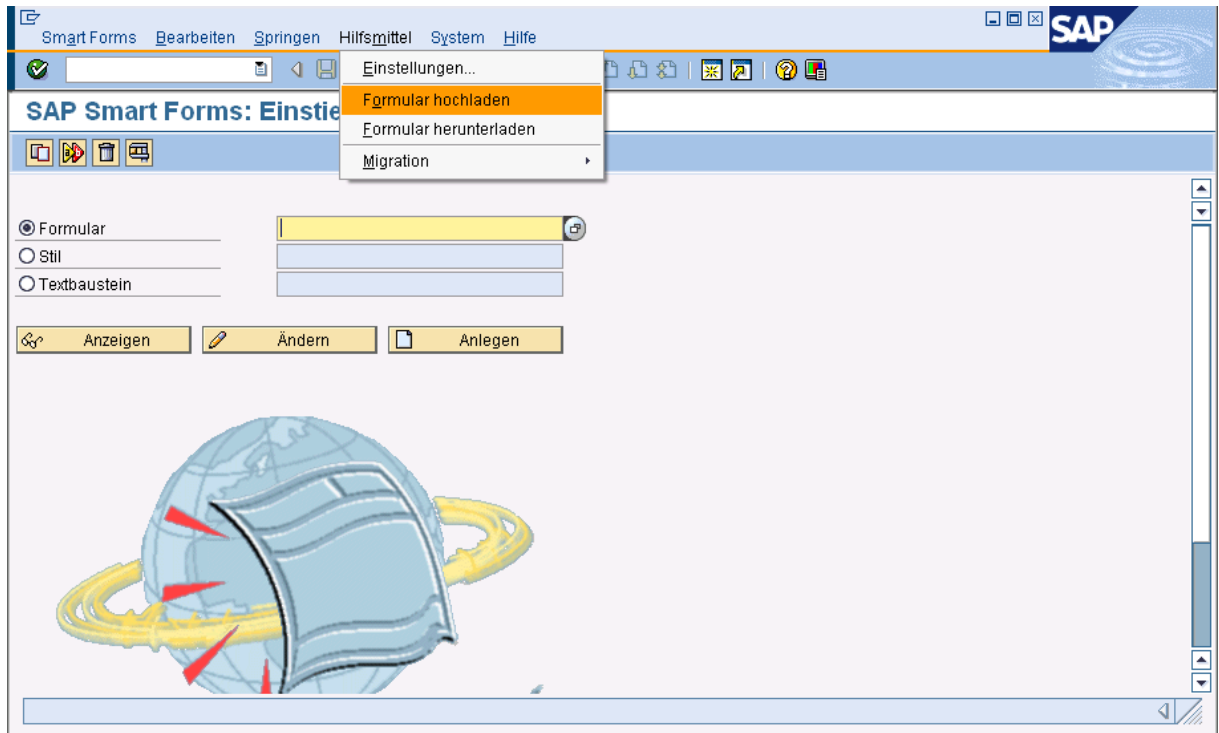


9. The form will be imported and at the end a message, similar to the following, appears:

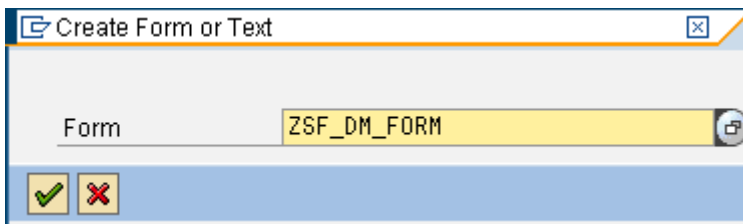


16 Annex 8: Uploading of a Smart Forms Form.

1. Start the transaction **Smartforms**.
2. Select **Utilities / Upload Form** from the menu.

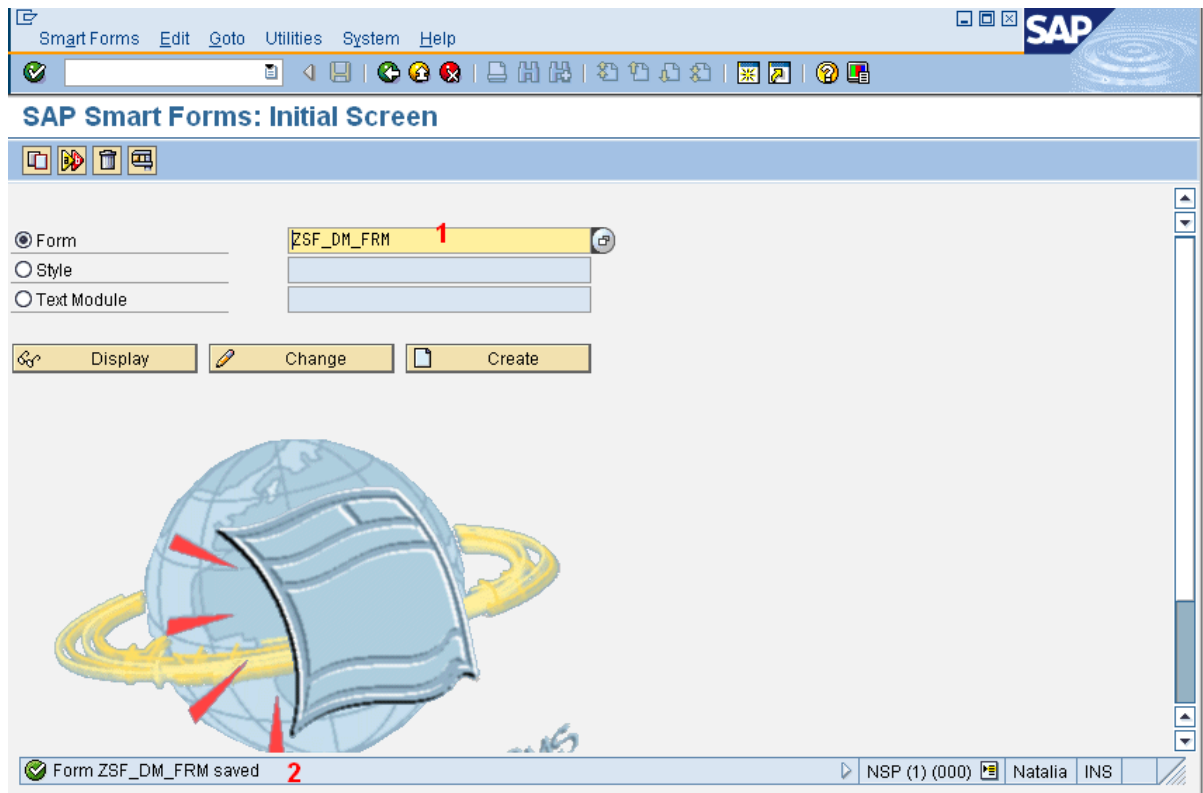


3. Enter the name of the form **ZSF_QR_FORM** in the next dialog and click on the green check.



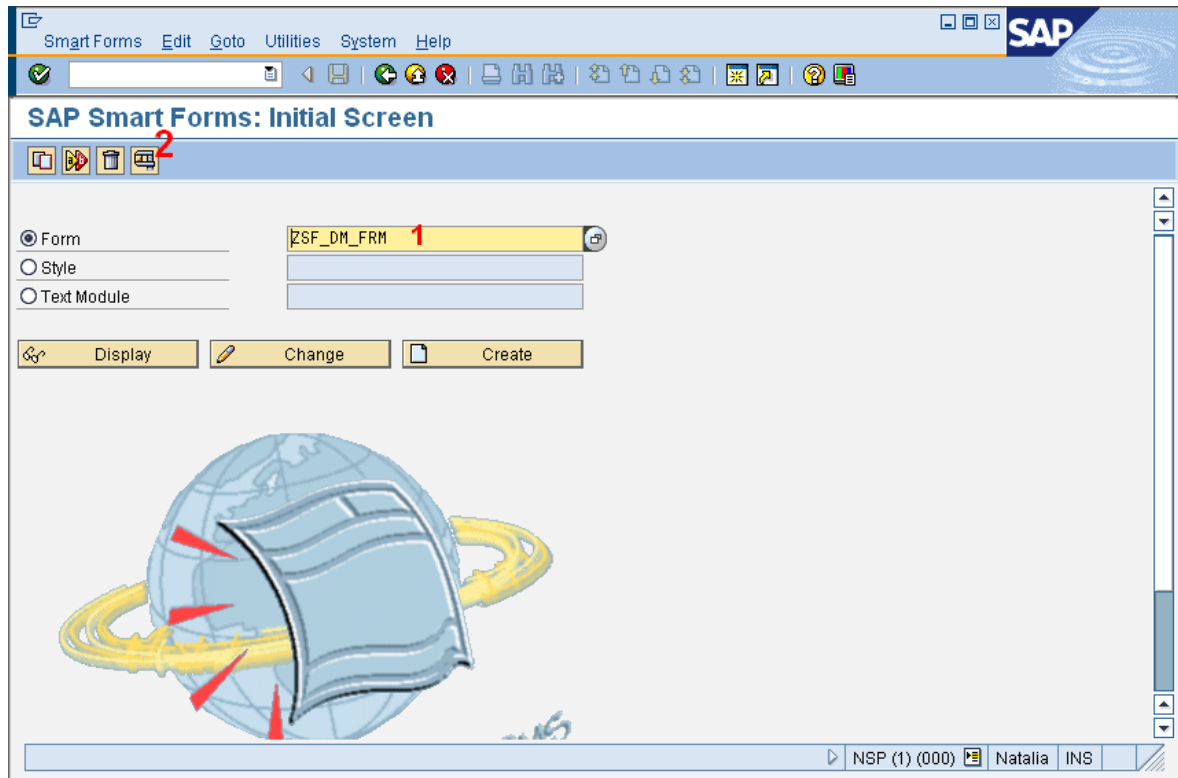
4. The file selection dialog appears. Select the file **ZSF_QR_FORM.XML** from the subdirectory **TestForms** and click on the button **Open**.

5. The form will be uploaded. The name of the form appears automatically in the field "Form" (1) and a message in the status bar (2).

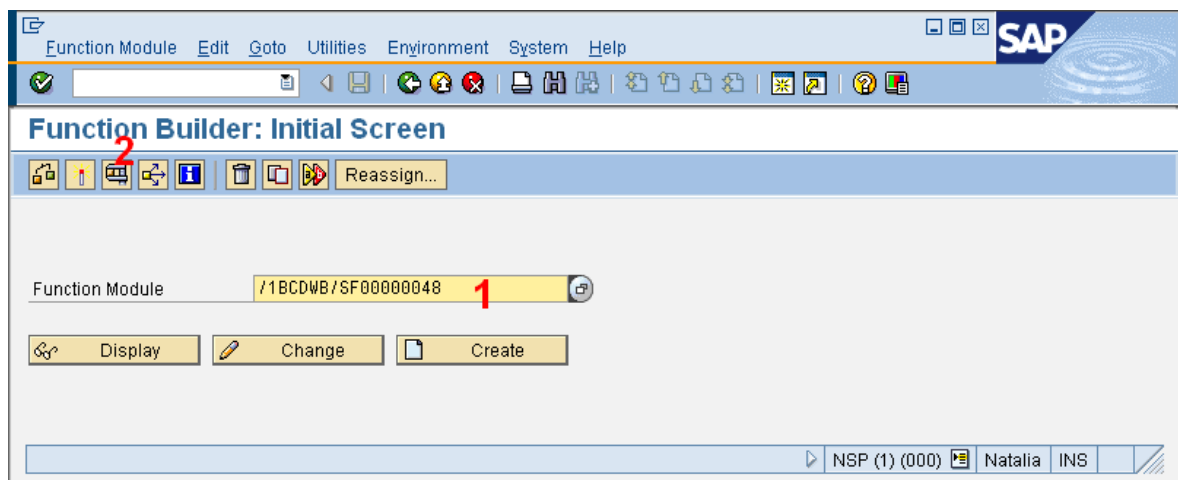


17 Annex 9: Testing a Smart Forms form.

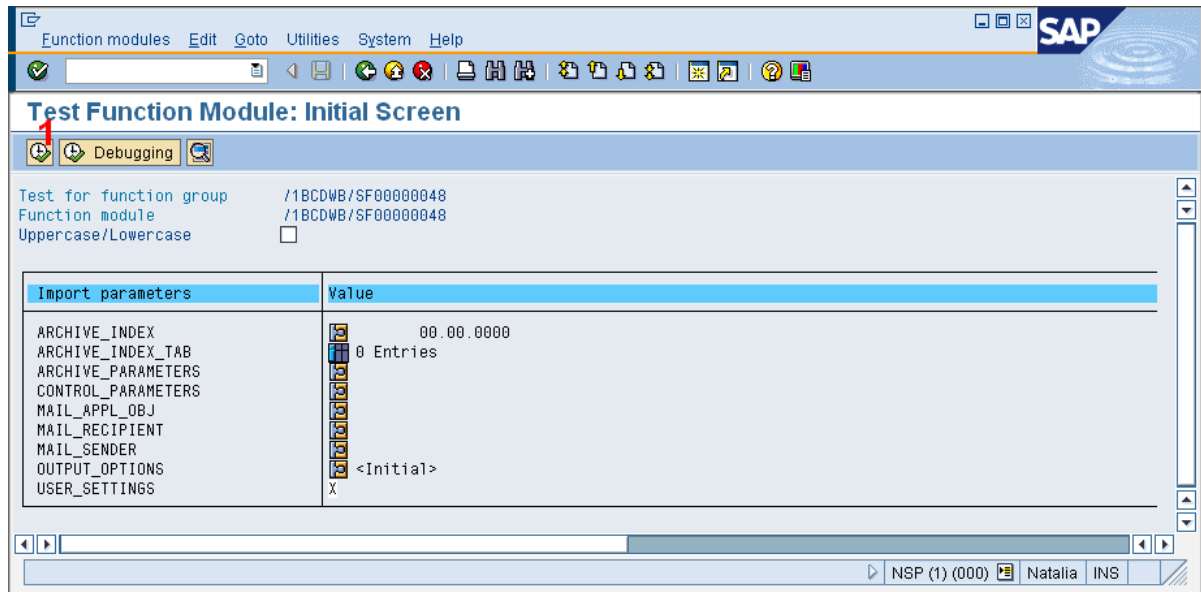
1. Start the transaction **Smartforms**.
2. Enter the name of the form **ZSF_QR_FORM** (1) and click on the test button (2).



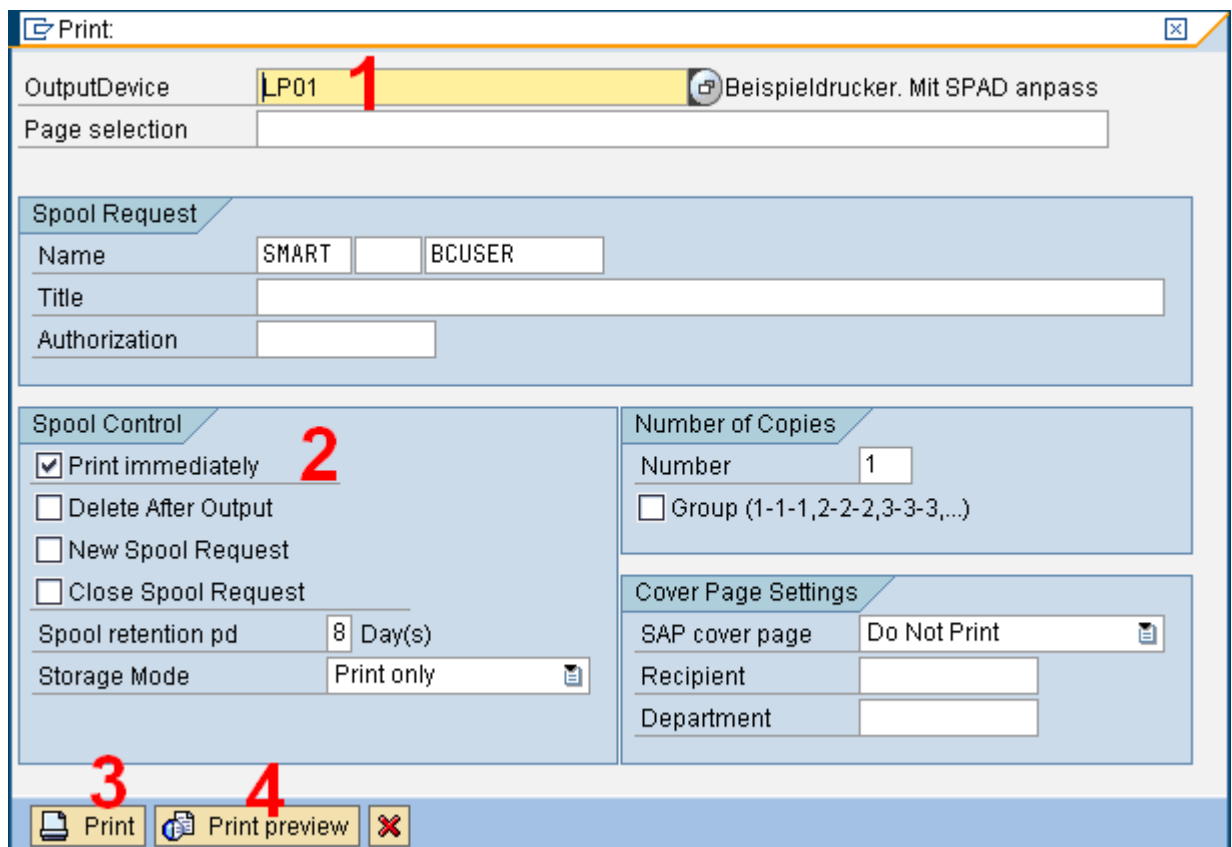
3. The system generates automatically a function, which name appears in the field "Function Module" (1) of the next dialog. Click against on the symbol **Test** (2).



4. Click on the button execute (1).

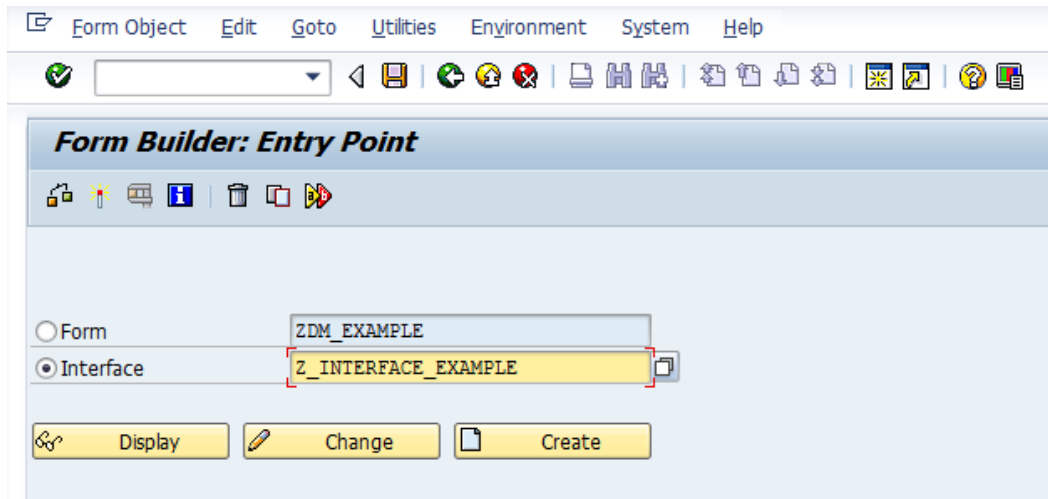


5. The print program starts now. Enter the name of the printer, where the form shall be printed out (1).
6. Check the box **Print immediately** (2)
7. Click on the button Print (3), to print the form on the printer, or on Print preview (4), to watch the form on the screen.

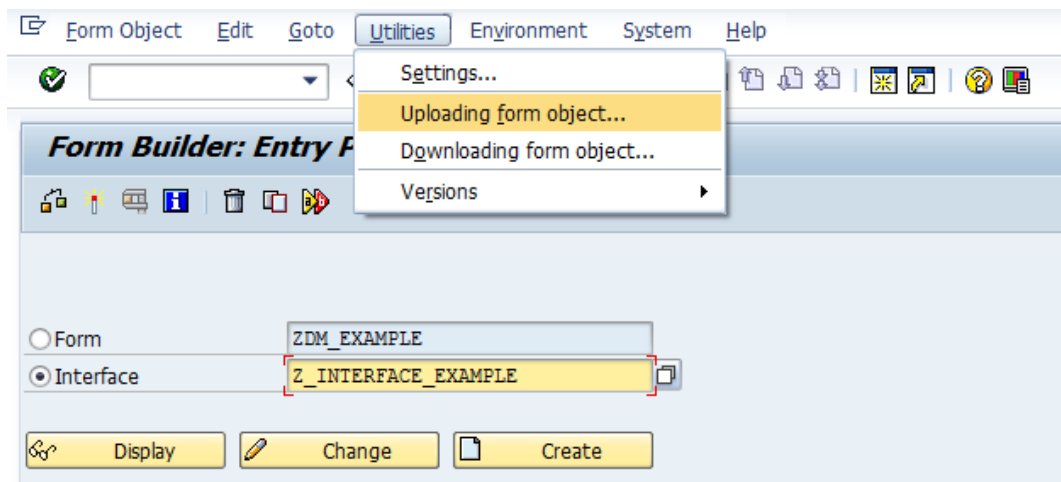


18 Annex 10: Upload an Interactive Forms Interface.

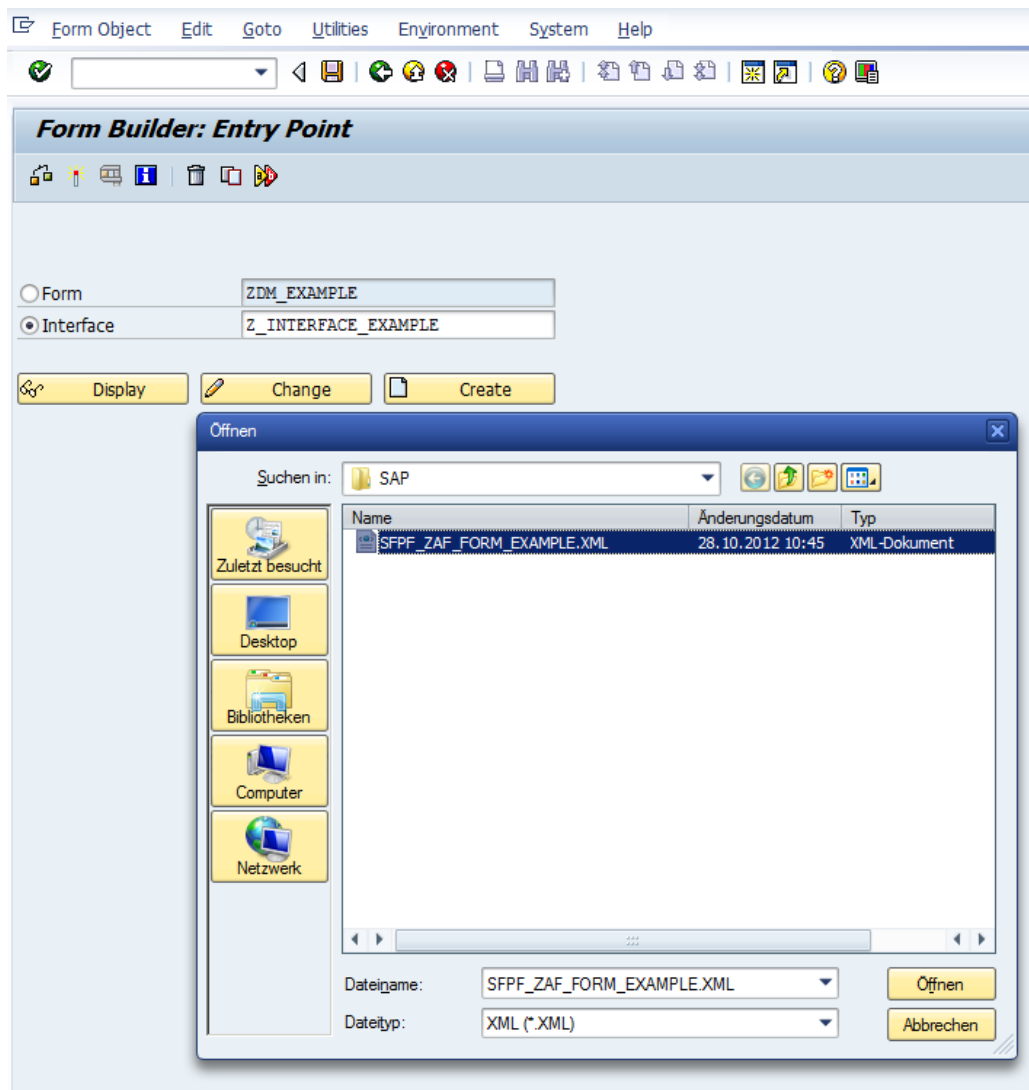
1. Start the FormBuilder by calling the transaction **SFP**.
2. Mark the radio button **Interface** and give the interface a name.



3. Select from the menu **Tool** → **Upload Form Object**.

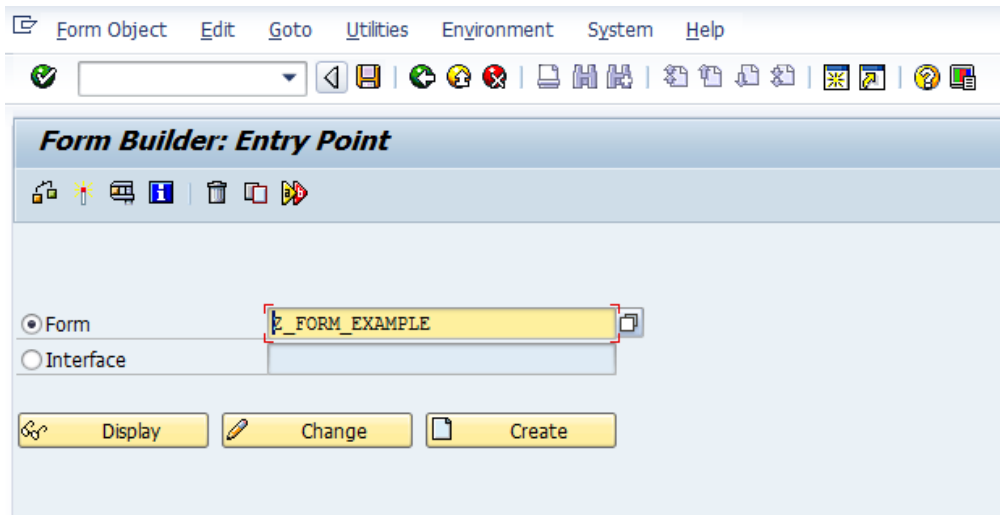


4. From the file selector, select the XML file and press the button **Open**.

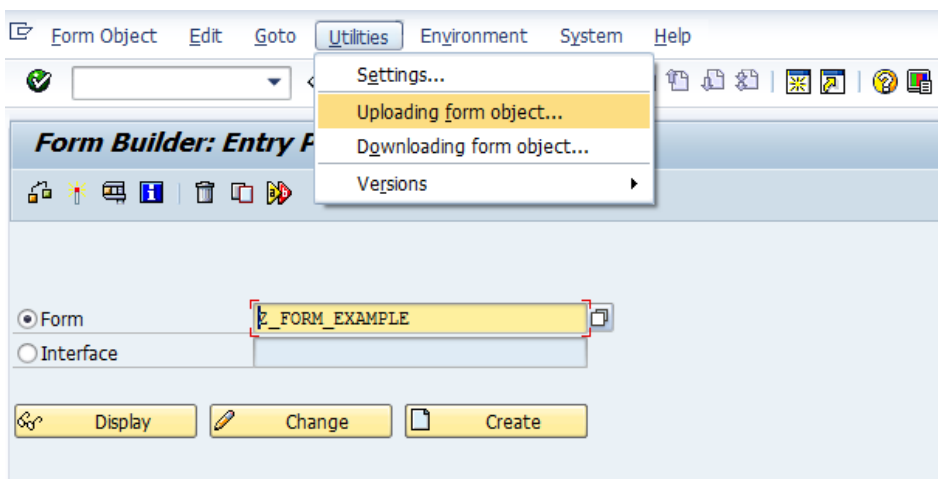


19 Annex 11: Upload an Interactive Forms Form.

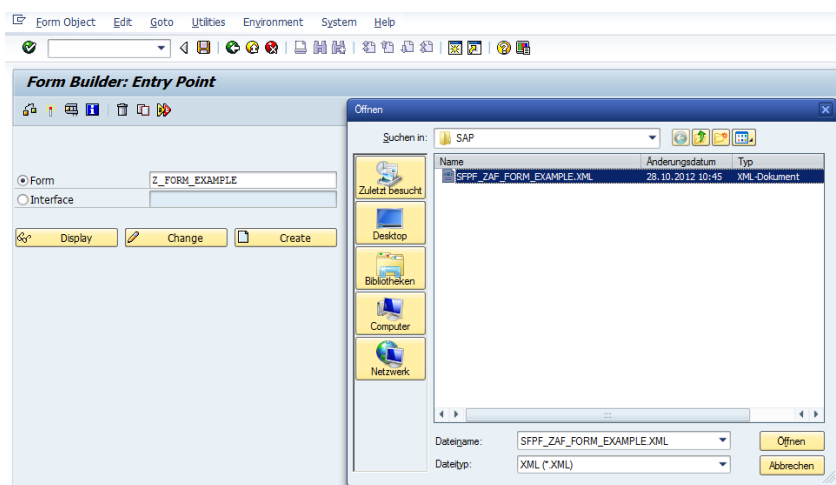
1. Start the FormBuilder by calling the transaction **SFP**.
2. Mark the radio button **Form** and give the form a name.



3. Select from the menu **Tool** → **Upload Form Object**.

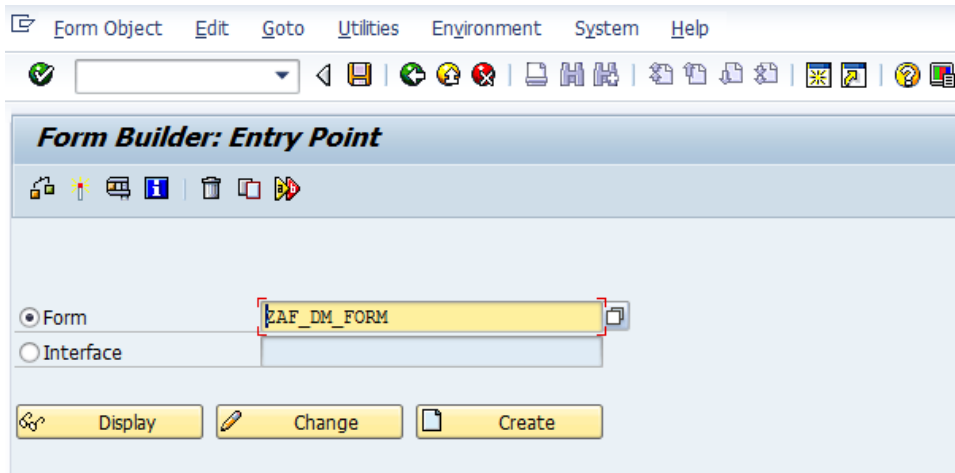


4. From the file selector, select the XML file and press the button **Open**.

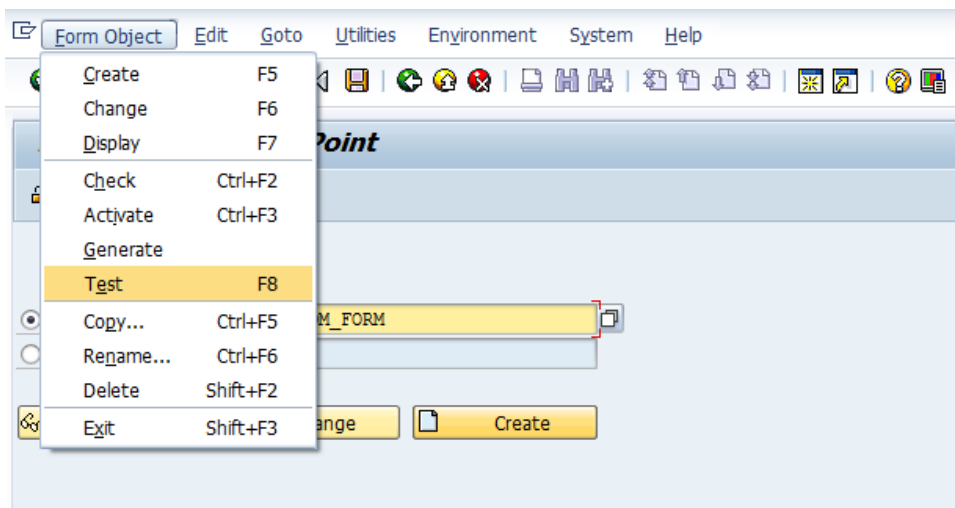


20 Annex 12: Test an Interactive Forms Form

1. Start the FormBuilder by calling the transaction **SFP**.
2. Mark the radio button **Form** and enter the name of the form (in the example: **ZIF_QR_FORM**).

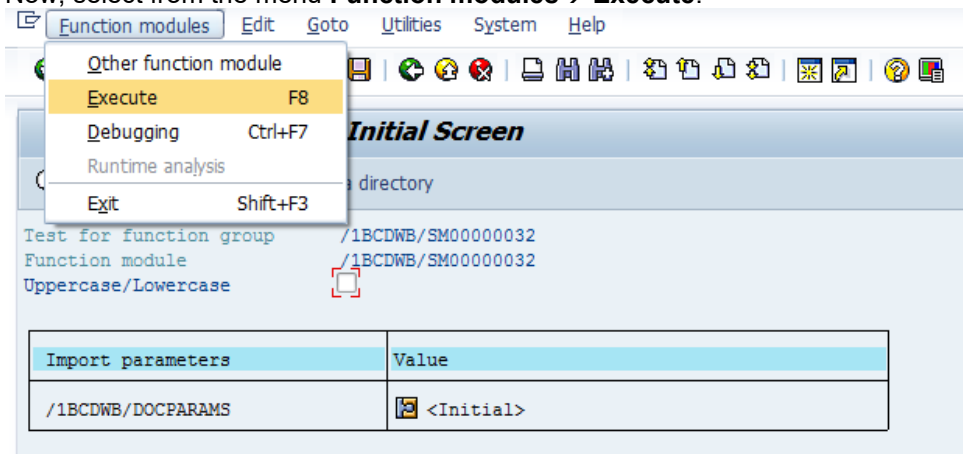


3. Select from the menu **Form Object** → **Test**.

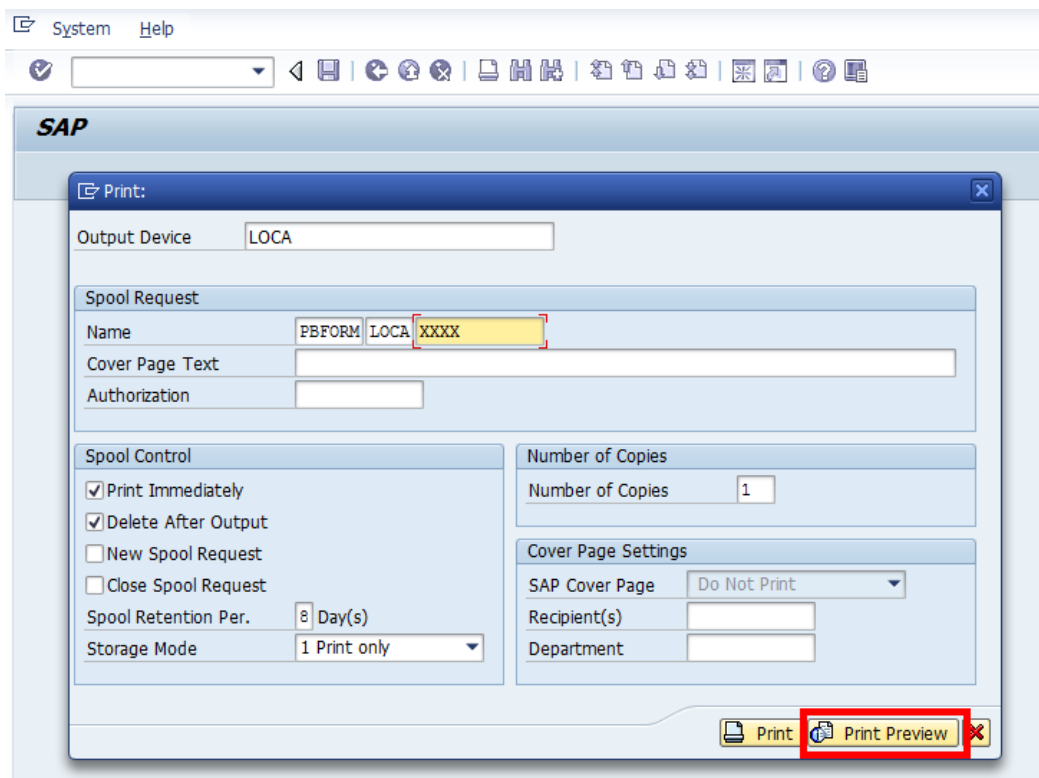


The generated function block is now shown.

4. Now, select from the menu **Function modules** → **Execute**.

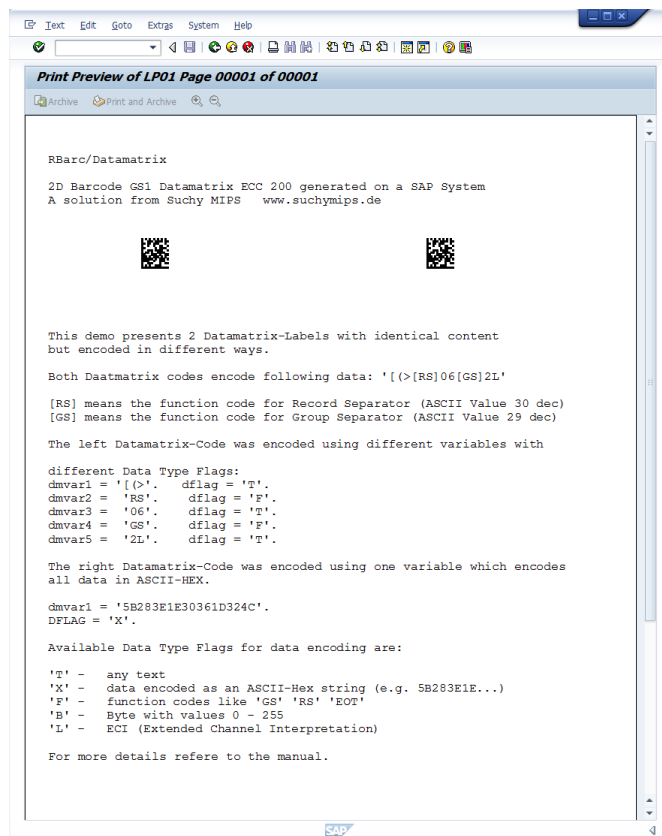


5. Enter a printer that was not allocated with an SAPWIN or SWIN device type and press the button **Print View Display**.



The image shows the SAP 'Print' dialog box. The 'Output Device' field is set to 'LOCA'. In the 'Spool Request' section, the 'Name' field contains 'PBFORM LOCA XXXX', with 'XXXX' highlighted in yellow. The 'Spool Control' section has 'Print Immediately' and 'Delete After Output' checked, 'New Spool Request' and 'Close Spool Request' unchecked, 'Spool Retention Per.' set to '8 Day(s)', and 'Storage Mode' set to '1 Print only'. The 'Number of Copies' is '1'. The 'Cover Page Settings' section has 'SAP Cover Page' set to 'Do Not Print', and empty fields for 'Recipient(s)' and 'Department'. At the bottom, the 'Print' button is disabled, and the 'Print Preview' button is highlighted with a red rectangle.

6. It should now be possible to view the Print View Display.



21 Annex 13: Swiss QRCode

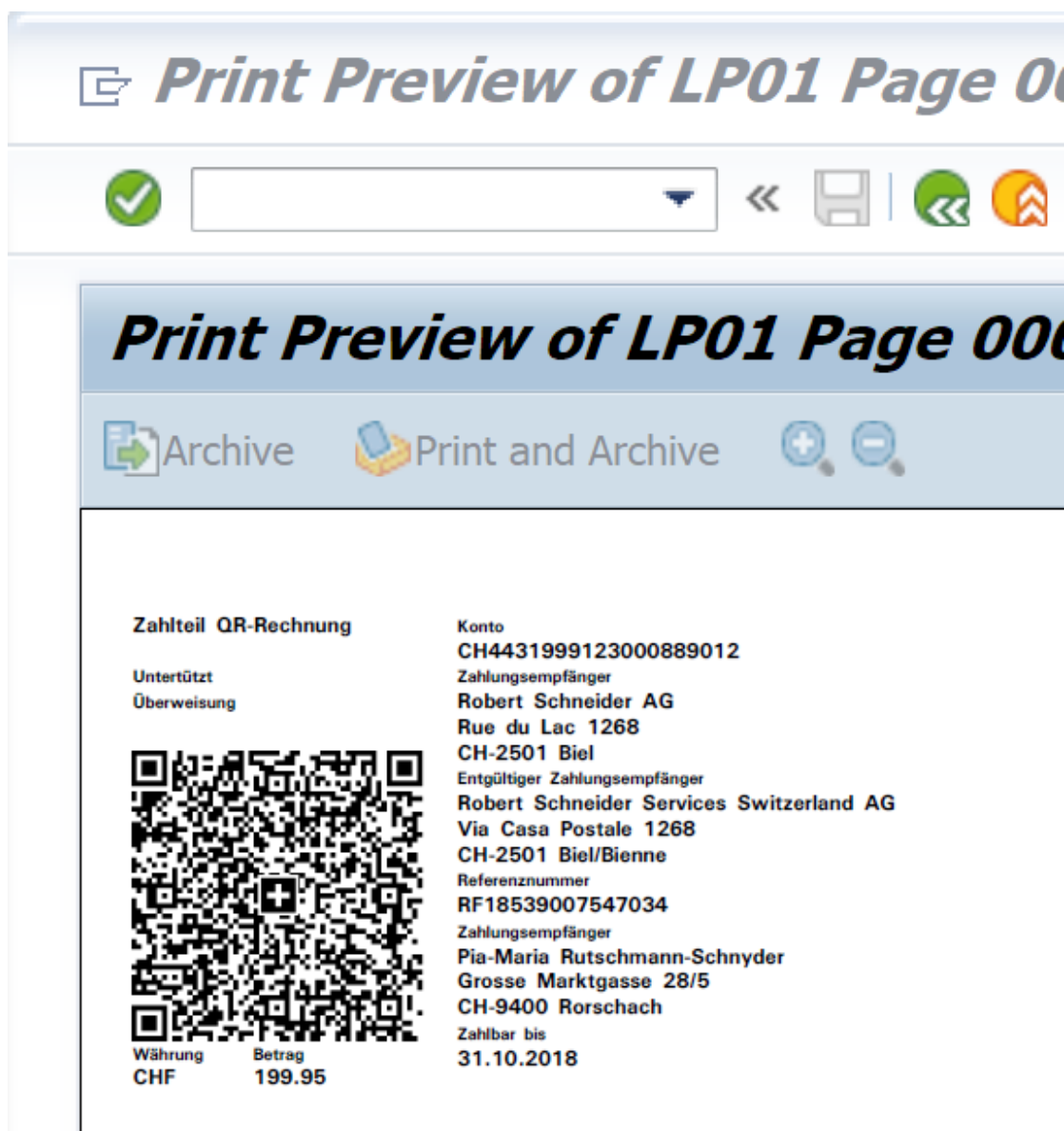
RBarc/QRCode includes in option for printing the Swiss QRCode.

The Swiss QR/Code is not fully compliant with ISO 18004 which specifies the QRCode. It prints a Swiss Cruise in the middle of the barcode and requires - regardless of the amount of encoded data – a fix size of 46 x 46 mm. This causes, that the module size cannot be predefined as at ISO-QRCode, but must be calculated dinamically.

To print the Swiss QRCode with RBarc/QRCode, only the parameter X_DIMM must be set to value 1046.

In this case the X_DIMM value will be calculated automatically, so that the whole Barcode will be of size 46 x 46 mm.

The parameter ECC_Level must be set to the value of "M".



Example of Swiss QRCode generated by RBarc/QRCode

Please note:

- The "Swiss Payment Standard" allows either <LF> (linefeed) or <CR> <LF> (carriage return + linefeed) as field separation.
- Some QR codes contain a so-called QR reference. The structure of the QR reference corresponds to the ESR reference (26 numeric characters followed by a check digit according to modulo 10 recursively, and can be used by the biller as a structured reference. For the calculation of the check digit for the QR reference, we provide the ABAP program Z_QR_REFERENCE. The QR reference can be determined from the form, for example:

```
REPORT Z_QR_REFERENCE_GET.
```

```
data: refnumber(26).
```

```
data: checksum.
```

```
refnumber = '21000000000313947143000901'.
```

```
perform qr_reference in program z_qr_reference using refnumber  
                                                    changing checksum.
```

```
write:/'This is my refnumber: ', checksum.
```

Summary

- ✓ The Swiss QRCode was implemented as an option in RBarc / QRCode (from version 1.7)
- ✓ The following parameters must be used for the Swiss QR Code
 - x_dim = 1046.
 - ecc_level = 'M'.
- ✓ **LF** or **CR+LF** must be used as fields separator.
- ✓ The carriage return is eliminated after the final element