ORACLE

Oracle® Documaker

Implementing PDF417 Bar Codes

12.7.0

Part number: F51808-01

December 2021

Contents

Chapter 1

Producing PDF417 Bar Codes with Documaker

This document describes how to create a PDF417 bar code that can contain any type of information. For instance, this capability makes your Documaker system compatible with the New York State Insurance Department's (NYSID) regulation that requires bar codes on driver ID cards.

NOTE: The ability to produce PDF417 bar codes is an add-on feature for Documaker versions 9.7 through 11.1, but the functionality was included in Documaker version 11.2. In version 11.5 PDF417 became a built-in bar code type, and the rules described in this document were enhanced to support a specified bar code object.

Included is information on these topics:

OVERVIEW

In July, 2000, the state of New York announced the Insurance Information & Enforcement System (IIES) for the purpose of enforcing compulsory insurance laws.

A key element of the provision is that all insurance companies that provide auto insurance in the state of New York must add a bar code to driver ID cards and other documents.

The bar code must contain information that is specific to the insured and to the insurance company. The bar code is a public domain form of a 2-dimensional bar code known as *Portable Data File 417* or *PDF417*.

Oracle Insurance's PDF417 solution produces an industry standard PDF417 bar code which can be scanned by any industry standard scanner that supports PDF417 symbology. The Oracle applications along with Oracle-supplied fonts generate a valid and scannable bar codes you can print on AFP, Xerox, PostScript, and PCL printers.

The implementation described in this document was based on the document Encrypted 2D Bar Coded ID Card Update II, which was published on September 29, 2000 by the New York State Department of Motor Vehicles. For additional information contact the New York State Department of Motor Vehicles.

Reference documents

These documents also provide information on this regulation:

Title	Date	Published by
Guide to the Use of 2-D Bar Codes and Insurance ID Cards	December 2000	New York State Department of Motor Vehicles, Information Technology
Programmer's Guide to use of IDCardGen Libraries	January, 2001	New York State Department of Motor Vehicles, Information Technology
IIC Security Specifications	June, 2000	New York State Department of Motor Vehicles, Information Technology
What's New — IDCardGen Module Changes	January 16, 2000	New York State Department of Motor Vehicles, Information Technology
Encrypted 2D Bar Coded ID Card Update II	September 29, 2000	New York State Department of Motor Vehicles, Information Technology
Best Practices Recommendation for the Use of Bar-Codes	April 1996	AAMVA - American Association of Motor Vehicle Administrators

Implementing a PDF417 Solution

Oracle Insurance's PDF417 software, for use with Documaker Server, lets you generate the PDF417 bar code required by the New York State Insurance Department (NYSID) for automobile insurance ID cards.

There are numerous steps involved in implementing a PDF417 bar code solution for the NYSID application.

Things You are Responsible For

- Serving as the authorized contact (a *go-between*) for the implementation team to get answers to questions that can only be answered by the NYSID.
- Providing the implementation team with a list of forms requiring the addition of the bar code, or initiating a requirements study to make that determination.
- Providing the implementation team with access to the current MRL, including forms, rules, and data definitions for the production system so necessary changes can be made.
- Authorizing your IT group (or other internal control group) to install the new PDF417 fonts provided by Oracle for the high-volume printers, network printers, workstations, or servers.
- Getting the company's unique security key from the NYSID and providing it to the implementation team.
- Providing the implementation team with the authorizations and access to local facilities to perform the necessary implementation and testing, including login rights, and access to test data regions.
- Receiving the print out test results from the implementation team and submitting them to the NYSID for approval as a part of the certification process.
- Receiving and relaying the results of the certification tests back to the implementation team, and re-submitting subsequent tests as necessary until certification is achieved.
- Managing the migration of the updated programs and resources into production.

Implementation Team Responsibilities

The implementation team consists of you and system integrators such as Oracle Consulting.

- Determining which forms will need to have the bar code added and the size, location, and purpose of the bar code. (That is, choosing the font size based on business requirement.)
- Analyzing the existing forms to determine if you can add the bar code to them or if redesign work is necessary. Performing the redesign, if necessary.
- Analyzing the form data to determine if changes are necessary to make the data meet the bar code requirements.
- Analyzing the extract data to determine if the necessary data is available or if some re-implementation work is necessary.
- Modifying the scripts, INI files, and so on, as necessary to invoke the new rules, in the batch system or the online system.
- Installing the printer or display fonts, or providing them to IT for installation, as necessary for either online or batch implementations.
- Testing the implementation, getting output that can be successfully scanned.
- Delivering clean test output for submission to the NYSID for certification.
- Making subsequent changes as needed until certification is achieved.
- Assisting as necessary in moving the implementation into production.

Oracle Product Development Responsibilities

- Providing the code required for the two processes: creating the New York bar code information and creating a PDF417 bar code. The exact code provided will vary based on the product and platform.
- Providing the necessary fonts for either batch printing, printing from document output history, or embedding in PDF, as needed.
- Providing documentation and other support to make sure the implementation is successful.
- Working with Oracle to make sure the necessary knowledge is available and transmitted to implementation team members.

Ways to Implement the PDF417 Bar Code Solution

There are several ways you can implement a PDF417 bar code solution in a Documaker environment. For instance, you can do it via...

- Documaker batch processing
- Document output history retrieve and view
- Documaker Workstation

- IDS (iDocumaker Workstation, iPPS, or other Documaker Shared Objects usage)

Here is a brief overview of the necessary steps

Batch processing

To implement a batch processing solution, you must...

1. Update your FAP and INI files.
2. Use printer fonts (PCL, AFP, PostScript, or Metacode).

Output history retrieve and view

To implement an output history retrieve and view solution, you must...

1. Use PCL fonts for printing.
2. Use TTF fonts for viewing purposes (optional) or faxing (required).

Documaker Workstation

To implement a Documaker Workstation solution, you must...

1. Use PCL fonts for printing.
2. Use TTF fonts for faxing (required) or viewing (optional).

IDS (iDocumaker Workstation/iPPS)

1. Use TTF fonts for embedding in PDF.
2. Update rule configurations to invoke the new IDS rule at all locations necessary to create the bar codes in the necessary data import implementation.
3. Test printing using Adobe Acrobat.

Since each scenario uses different software, platforms, and fonts, each may require certification by New York.

PDF417 Rules

The PDF417 software for Documaker includes these section level rules:

- CreateNYAAMVAData
- CreatePDF417Barcode on page 17
- GetNYAAMVAVar on page 20

The first rule lets you create the data while the second rule lets you create the bar code. These processes are separate to enhance performance. The third rule lets you retrieve members of the AAMVA structure after it is built. See Turning on Tracing on page 50 for debugging information.

In most cases, you will need to call the CreateNYAAMVAData rule only once in order to populate the AAMVA structure and copy it to the GVM variable. You can then call the CreatePDF417Barcode rule once for each bar code that you need to create. If you need to retrieve members from the AAMVA structure after it is populated, you can then call the GetNYAAMVAVar rule.

The screen shot below shows an example of using all three of these rules.

Section rules	
CreateNYAAMVAData	BARCODEVAR
CreatePDF417Barcode	N=P417BX1,F=912,G=BARCODEVAR
CreatePDF417Barcode	F=1216,G=BARCODEVAR,T=9444,L=8649,B=15204,R=19689
CreatePDF417Barcode	N=P417BX3,F=1216,G=BARCODEVAR
GetNYAAMVAVar	G=BARCODEVAR,ICOVERAGESTARTDATE,F=CSDATE,RNAMEASONE1,F=NAMEA1,RNAMEASONE1,V=NMA1

- The CreateNYAAMVA rule is called once to populate the AAMVA structure and store it in the GVM named BARCODEVAR.
- The CreatePDF417Barcode rule is called three times (to create three different bar codes), each time using the data stored in the GVM variable named BARCODEVAR.
- The GetNYAAMVAVar rule is called once. It retrieves the ICOVERAGESTARTDATE member of the AAMVA structure and copies its value to the field named CSDATE, copies the RNAMEASONE1 member to the field named NAMEA1, and also copies the RNAMEASONE1 member to the GVM named NMA1.

NOTE: The NY AAMVA driver ID card application must store encrypted binary data into a PDF417 bar code. The standard Documaker method of mapping data from a data source to a variable field will not work in this case because the data is binary and not plain text. The rules and functions described here are designed to provide a method of creating the binary buffer needed and then inserting the data into the PDF417 bar code.

This alternate approach works by using the Documaker technique known as GVM (Global Variable in Memory). A GVM symbol is a symbolically named dynamic memory buffer. The CreateNYAAMVAData rule creates the data buffer and attaches it to a GVM symbol. The other rules and functions reference the GVM data buffer by its symbolic name. You can use the PDF417 bar code object with rules such as the Move_it rule in other applications, but the NY driver ID card application must use the GVM approach described here.

CREATENYAAMVADATA

To receive this rule, you must:

- Be authorized to produce auto insurance policies in the state of New York.
- Have executed the non-disclosure agreement with New York.
- Have licensed Documaker technology from Oracle.

The CreateNYAAMVAData rule uses application and encryption routines provided by the state of New York for purposes of porting or including into custom applications.

Syntax

```
;CreateNYAAMVAData; GVMOutputVariable , CheckForRequired , INS_Key;
```

Parameters

Parameter	Description
GVMOutputVariable	Specifies the name for the GVM variable that will contain the 570 bytes of formatted data be stored into a PDF417 bar code by a later call. You can make several calls using this data to create bar codes of different sizes and locations, but which contain the same data. Keep in mind that this rule is typically run once, while you can call the CreatePDF417Barcode rule several times using the same source GVM data.
CheckForRequired	The New York application defines a number of variable data elements to be included in the bar code. Some of these elements are required. This parameter lets you decide when and to what extent to check for these required data elements. If you enter one (1), if any required fields are missing from the data, an error is issued for each missing field. If you enter two (2), if any required fields are missing from the data, a warning is issued for each missing field. If you enter three (3), processing continues and no errors or warnings are issued if required fields are missing. The default is 3. Other INI options control whether the Rules Processor continues processing or stops on errors and warnings. These options are documented in the Documaker Administration Guide.

INS_Key	Each insurance company that writes auto insurance for drivers in New York will receive a unique key signature. This value is an *armored* hex string. This parameter specifies that key.
	If you omit this parameter, you must include this information in the INI file. For security reasons, you may want to have the source code to this rule customized to hardcode the constant key value into the rule. The base version does not provide this option.
	If you use the INI file to provide the information, specify the value as shown here:
	`< NYAAMVA>`
	`Ins_Key = value`

Parameter	Description

PassPhrase (implicit parameter)	The INS_KEY parameter is an armored 36-character hex string value. Internally, it must be de-armored using a *pass phrase* before it can be used. The rule does not support pass phrase as an explicit rule parameter, but the pass phrase is an implicit parameter necessary for compliance, and it must be specified in the INI file. The PassPhrase is a plain text message string that can only be obtained by an authorized insurance company from the NYSID. For security reasons, you may want to have the source code to the rule customized to hardcode the constant pass phrase string value into the rule. The base version does not provide this option.
	Specify the pass phrase value in the INI file as follows:
	`< NYAAMVA >`
	`PassPhrase = value`

The rule begins by finding and destroying any existing GVM variable named by the GVMOutputVariable parameter.

1 The rule provides an interface between the Documaker batch process and application logic developed and provided by the NYSID. To interface with the New York application code, an interface structure must be filled out. This definition of this internal structure is specified by the New York application, and is called *InsuranceInfo* in C, and Oracle provides a COBOL version known as *NYID-INSURANCE-INFO*.

   The rule automates the process of moving data within the Rules Processor into the proper position in the interface structure. The rule uses an internal table of variable names and has the necessary offsets and lengths to assemble the structure. The internal table also has flags to indicate which fields are required.

2 The rule checks each field in the table. The rule first looks in the INI file to see if you have specified alternate mapping. The rule uses the name as the option and looks in the INI file. If found, it examines the value to see if you specified cross-mapping. You can specify a variable field, a constant value, a GVM variable, a DAL variable, a DAL script, or any other existing built-in INI function. If there is no INI entry for the variable, the rule assumes the name will be used as is and it then looks for a field with that name.

3 The rule first looks for a variable field with a given name on the current section (FAP file). If the field is found on the section, that field will be used. If the field does not have any data or is blank, then the matching structure element will also be blank.

   If the field is not found on the section, the rule next searches the entire document set for an occurrence of that named field which contains data. If no occurrence of the field can be found with data, the rule looks for a GVM variable with that name. If still no data is found, the rule looks for a DAL variable with that name. If it does not find a DAL variable, the search ends and no data will be used.

   If an appropriate value is found, the rule uses the offset and length stored in the table to move the data into the interface structure. If no data was found, the corresponding structure member is left blank.

   If the structure member is empty (blank), the rule examines the CheckForRequired parameter to see what action should be taken.

If the CheckForRequired parameter is	Then
1 and the data is empty	An error is issued
2	A warning is issued
3	No warning or error is issued

If an error is issued, other INI options for the Rules Processor determine whether processing continues or stops.

4 Once the InsuranceInfo structure has been assembled, the structure is passed in to the New York application API, along with the INS_Key parameter, the PassPhrase option, and the output structure, which is called *InsuranceAAMVA*.

5 The InsuranceAAMVA output structure contains 570 bytes of formatted data to be encoded into the bar code. The rule stores this formatted data in a GVM variable, named by the GVMOutputVariable parameter, for later use.

INI Options

The easiest implementation is one that provides the required data elements using the reserved names assigned by Oracle. These names are also used as C structure names in the New York application. Below, for convenience, a table provides a cross-reference of these names to COBOL structure member names used by other Oracle Insurance applications.

Many implementations will have some data elements available under different names. Some data elements require manipulation before being included into the bar code. To avoid requiring an existing form to be re-implemented or changes to the extract file, you may want to map the reserved variable names to other names or other sources of data.

This rule supports a number of INI options and ways to configure each of the 30 or so variables that are necessary to feed the New York structure to alternate sources.

```
< NYAAMVA >
    RLastName1      = field name
    RFirstName1     = ;constant value
    RMiddleName1    = ;
    RNameSuffix1    = ;
    RNameAsOne1     = ;
    RClientId1      = ;~(any existing INI built-in)
    RLastName2      =
    RFirstName2     =
    RMiddleName2    =
    RNameSuffix2    =
    RNameAsOne2     =
    RClientId2      =
    RStreet         =
    RCity           =
    RState          =
    RZipCode        =
    ROrganizationLine1=
    ROrganizationLine2=
```

```
ROrganziationFein=
VVinNumber     =
VYear          =
VMake          =
VRepInd        =
VHistInd       =
VTowInd        =
VSeats         =
IIssuerId      =
IInsCompanyCode=
IInsIssuanceDate=
ICoverageStartDate=
ICoverageEndDate=
IPolicyNumber  =
```

Using the INI mapping options

The INI file offers many ways to get the necessary interface variables. Here are some examples:

```
< NYAAMVA >
    RLastName1= LAST NAME
```

The above example looks for data named LAST NAME. The rule first looks for a field on this section. Then it looks for a global field. Next, it looks for a GVM variable. Finally, it looks for a DAL variable.

```
VTowInd     = ;N
```

The above example shows how a company that never insures tow trucks could short cut this part and hardcode a constant N as the tow truck indicator value.

This table shows all of the variables accessed via the table, the corresponding C-structure member name, and whether the variable is required.

NYID-INSURANCE-INFO name Used by the COBOL copy book record structure.	**Corresponding NY InsuranceInfo C-structure member name and INI option.**	**Required?**
NYID-R-LAST-NAME-1	RLastName1	Yes
NYID-R-FIRST-NAME-1	RFirstName1	Yes
NYID-R-MIDDLE-NAME-1	RMiddleName1	No
NYID-R-NAME-SUFFIX-1	RNameSuffix1	No
NYID-R-NAME-AS-ONE-1	RNameAsOne1	Yes
NYID-R-CLIENT-ID-1	RClientId1	Yes
NYID-R-LAST-NAME-2	RLastName2	No
NYID-R-FIRST-NAME-2	RFirstName2	No
NYID-R-MIDDLE-NAME-2	RMiddleName2	No
NYID-R-NAME-SUFFIX-2	RNameSuffix2	No
NYID-R-NAME-AS-ONE-2	RNameAsOne2	No

NYID-INSURANCE-INFO name Used by the COBOL copy book record structure.	Corresponding NY InsuranceInfo C-structure member name and INI option.	Required?
NYID-R-CLIENT-ID-2	RClientId2	No
NYID-R-STREET	RStreet	Yes
NYID-R-CITY	RCity	Yes
NYID-R-STATE	RState	Yes
NYID-R-ZIP-CODE	RZipCode	Yes
NYID-R-ORGANIZATION-LINE-1	ROrganizationLine1	Yes
NYID-R-ORGANIZATION-LINE-2	ROrganizationLine2	No
NYID-R-ORGANIZATION-FEIN	ROrganizationFEIN	Yes
NYID-V-VIN-NUMBER	VVinNumber	Yes
NYID-V-YEAR	VYear	Yes
NYID-V-MAKE	VMake	Yes
NYID-V-REP-IND	VRepInd	Yes
NYID-V-HIST-IND	VHistInd	No
NYID-V-TOW-IND	VTowInd	No
NYID-V-SEATS	VSeats	No
NYID-I-ISSUER-ID	IIssuerID	Yes
NYID-I-INS-COMPANY-CODE	IInsCompanyCode	Yes
NYID-I-INS-ISSUANCE-DATE	IInsIssuanceDate	Yes
NYID-I-COVERAGE-START-DATE	ICoverageStartDate	Yes
NYID-I-COVERAGE-END-DATE	ICoverageEndDate	Yes
NYID-I-POLICY-NUMBER	IPolicyNumber	Yes

CREATEPDF417BARCODE

Use this rule to create a PDF417 bar code.

Syntax

```
;CreatePDF417Barcode;N=BarCodeName,F=FontID,E=ECCLevel,G=GVMSourceV
ariable,T=TopFAPCoord,L=LeftFAPCoord,B=BottomFAPCoord,
R=RightFAPCoord,W=Warning;
```

Parameter	Description
BarCodeName	Specifies a name for the bar code. This name typically is the name of the associated PDF417 bar code variable field object located on the current section. Older implementations that do not have a bar code object may specify the name of an associated box object located on the current section, and if there is no box object the name is still required to provide the root name used by a series of dynamic text labels that contain the bar code text data.
FontID	Specifies the PDF417 font you want to use for this instance of the bar code. This parameter is optional if you specified the name of an associated PDF417 variable data field, in which case the object already specifies the font to be used. You can choose between two sizes of PDF417 bar codes fonts: 9 dots high by 12 dots wide and 12 dots high by 16 dots wide. While Oracle does not mandate the IDs you can use, it does provide an FXR file which contains the attributes for the two fonts. The IDs for these fonts are: 912 and 1216. For more information on the included FXR files, see PDF417 FXR Files on page 38.
ECCLevel	Specifies the degree of error correction to be used in the bar code. This parameter is optional if you specified the name of an associated PDF417 variable data field, in which case the object already specifies the ECCLevel value to be used. The range is from zero (0) to eight (8). The default is four (4), which is the value recommended by the NYSID. For more information, see Sizing a PDF417 Bar Code on page 42.
GVMSourceVariable	Specifies the name of a GVM variable which contains the data to be embedded into the bar code. The CreateNYAAMVAData rule must have previously created this data and stored it in this GVM variable. The New York application currently specifies 570 bytes of data. It is likely that you will need to use the same source data to create a bar code multiple times on the same page, or on multiple forms in the form set. The bar codes can be different sizes and use different fonts yet contain the same source data. This is why it is important for performance that data creation and bar code creation steps are performed by separate rules.

Parameter	Description
TopFAPCoord LeftFAPCoord BottomFAPCoord RightFAPCoord	Each bar code instance must specify a location and a rectangular size. You can specify this information from... • A named PDF417 bar code variable field object on the FAP file • A named box object on the FAP file • Explicit rectangle FAP coordinates The rectangle coordinate parameters are optional if you specified the name of an associated PDF417 bar code variable field, in which case the object already specifies the coordinates. When included, the coordinates explicitly provide the dimensions of the rectangular area to contain the bar code. If you omit the rectangle parameters, the system gets the coordinates from either the associated variable bar code field or box object.
Warning	If the system cannot locate the specified bar code or box for the rule and the section is a multi-page section, you will get a warning. To suppress this warning, set this parameter to No (W=No). The default is Yes.

NOTE: Do not specify a location and a rectangular size using explicit rectangle coordinates in a multi-page section. This causes the system to create the bar code on every page of the multi-page section.

This rule performs these actions:

1 The rule first checks for a bar code variable field object with the specified name. If found, this field provides the coordinates, font, and error correction level.

NOTE: This rule was originally introduced before the system supported PDF417 as a built-in bar code object type and therefore alternate methods of specifying the bar code location and parameters had to be provided. If this is an implementation from a prior version of the product, there may not be a bar code variable field object configured for use on the section. In this case the rest of the steps are applicable.

2 The rule first checks to see if the rectangle coordinates were provided. The most important ones specify the bottom and right. If the bottom or right coordinates are zero or blank. the system assumes the coordinates are provided by a box object. The box object must be on the same section and must have the name specified in the BarCodeName parameter.

3 The rule uses the rectangle coordinates that were specified explicitly or specified by the named box. Using these coordinates, the rule determines the height and width of the bar code.

4 The rule uses the font specified in the FontID parameter to locate the requested bar code font. The FXR font information provides the cell height, width, and baseline values.

5 The rule uses the font cell width to calculate the number of text characters (COLS) that can fit within the width of the bar code rectangle.

6 The rule uses the cell height to calculate the number of text character text labels (ROWS) that can fit within the height of the bar code rectangle.

7 The rule checks for and destroys any existing bar code text. These text labels will be internally named with a root name matching the name specified by BarCodeName.

8 The rule retrieves the bar code data from a GVM variable specified by the GVMSourceVariable parameter.

9 The rules passes the bar code data, along with the calculated number of rows and columns, and the requested error correction level, to internal APIs that create the text characters necessary to render the bar code.

   An error at this point could mean that the bar code area was too small for the source data and ECC level.

10 The rule creates dynamic text labels, row by row, for all of the returned bar code text. Each dynamic text label will have a root name specified by the BarCodeName parameter, the font specified by the FontID parameter, and its cell parameters, data length specified by COLS, and will contain the text for that row.

The dynamic text labels are unloaded automatically into the NAFILE.DAT file.

GETNYAAMVAVAR

Use this rule to retrieve members of the AAMVA structure after it is built. Certain fields in the AAMVA structure that are not supplied by the user are resolved during the processing of the CreateNYAAMVAData rule. This rule lets you retrieve the data from those fields and place it in a field or a GVM variable.

NOTE: The field must exist before this rule is executed. You can create a field using the MK_Hard rule on the field. You cannot create an empty field.

Syntax

```
;GetNYAAMVAVar;GVMOutputVariable,AAMVAMember, Field/
GVMVariable;
```

You can retrieve more than one AAMVA data member in a single call to GetNYAAMVAVar. To do this, simply use another pair of AAMVAMEMBER and FIELD/GVMVARIABLE combinations.

Parameter	Description
GVMOutputVariable	This is the name of the GVM variable supplied to CreateNYAAMVAData where the AAMVA structure is stored.
AAMVAMember	This is the name of the AAMVA field you want to retrieve. It should match the names used in the INI file, such as RNameAsOne1, RLastName2, and so on.
Field/GVMVariable	This is where the data will be placed. If you are sending the data to a field, must use the F switch, and for GVM variables use the V switch.

Example

```
;GetNYAAMVAVar;G=CustomerInfo,RNameAsOne1,F=FULLNAMEFIELD,
RLastName2,V=SECONDNAME;
```

In this example, the rule gets the *RNameAsOne1* data from the CustomerInfo AAMVA data and places it in the field *FULLNAMEFIELD*. It will also get the *RLastName2* data member and place it in the GVM variable *SECONDNAME*.

PDF417 IDS Rules

To implement a PDF417 bar code solution with IDS, you use the P417NYPDF417 rule. This rule handles both gathering the data and building the bar code. See Turning on Tracing on page 50 for debugging information.

P417NyPDF417

Use this rule to create PDF417 bar codes in IDS implementations.

Syntax

```
P417NyPDF417 FontID, ECCLevel, BarCodeName
```

Parameter	Description
FontID	Specifies PDF417 font you want to use for this instance of the bar code. You can choose between two sizes of PDF417 bar codes fonts: 9 dots high by 12 dots wide and 12 dots high by 16 dots wide. While Oracle does not mandate the IDs you can use, it does provide an FXR file which contains the attributes for the two fonts. The IDs for these fonts are: 912 and 1216. **Note**: Prior to release 11.2, the PDF417_2.FXR and PDF417.FXR files were provided to licensed PDF417 customers. In release 11.2, the new REL112.FXR file contains the PDF417 font references found in the two PDF417 FXR files previously included in this product. See for more information on the included FXR files, see PDF417 FXR Files on page 38.
ECCLevel	Specifies the degree of error correction to be used in the bar code. The range is from zero (0) to eight (8). The default is four (4), which is the value recommended by the NYSID. For more information, see Sizing a PDF417 Bar Code on page 42.
BarCodeName (or BoxName)	Specifies a name for the bar code. The name typically is the name of a PDF417 barc ode variable field object. Older implementations that pre-date the built-in PDF417 bar code may specify the name of a box object located in the current section or simply a name for the bar code that is used internally when creating the bar code text.

The rule works by first searching for all of the bar code variable field objects with the name given in the BarCodeName parameter and applying the data to the bar codes.

NOTE: Older implementations that predate the built-in PDF417 bar code type may specify the name of a box object instead. In this case, the rules works by first searching for all the boxes with the name given in the BarCodeName parameter. It then builds the bar code by looking at the INI file, then the current section, and then the form set.

Once it makes the bar code, the rule populates it for all boxes in the section. When the rule finds a box that is in a different section, it remakes the bar code. So you can make each bar code different by putting them in different sections.

Add this rule under the ReqType control group in the DOCSERV.INI file as shown here:

```
< ReqType:SAMPCO >
```

```
function = dpros2->DPRLoadImportFile
function = p417w32->P417NyPDF417,1216,4,barcode
function = dpros2->DPRPrint
```

where SAMPCO is the group name. Be sure to add this rule after the DPRLoadImportFile rule.

PDF417 DAL Functions

You can also use the following DAL functions to implement a PDF417 solution:

- P417DalCreateNYAAMVAData
- P417DalCreatePDF417Barcode

Registering the functions

Since these DAL functions are not registered by default, you must register them in the INI file. This will look something like this:

```
< DALFunctions >
    Keyword = DLL->FunctionName
```

Where *Keyword* left of the equals sign represents the DAL script function name you want to register.

To the right of the equals sign, *DLL->FunctionName* represents a DLL to load and the exported function name associated with the script function.

Registered in this manner, you can then reference these functions in a DAL script just like any other built-in function. For instance, you could register these functions as shown here:

```
< DALFunctions >
    CreateNYAAMVAData  = P417W32->P417DalCreateNYAAMVAData
    CreatePDF417Barcode= P417W32->P417DalCreatePDF417Barcode
```

And then refer to them as CreateNYAAMVAData() and CreatePDF417Barcode() in your DAL scripts.

You could also register the functions as shown here:

```
< DALFunctions >
    NYAAMVA = P417W32->P417DalCreateNYAAMVAData
    PDF417  = P417W32->P417DalCreatePDF417Barcode
```

And then refer to them as NYAAMVA() and PDF417() in your scripts. How you register these functions is up to you, just make sure the name you use is descriptive.

The DAL functions provide PDF417 functionality to Entry (Documaker Desktop). They work similar to the PDF417 rules; except when mapping data it searches fields on the section, fields in the formset, then for a DAL variable with the name of the field. The DAL functions are not intended to be used in batch processing; rules should be used instead.

P417DalCreateNYAAMVAData

Use this function to create the data you want to place in the bar code.

Syntax

```
P417DalCreateNYAAMVAData (DALVariableName, RequiredLevel, INS_Key)
```

Parameters	Description
DALVariableName	Specifies the name for the DAL variable that will contain the 570 bytes of formatted data be stored into a PDF417 bar code by a later call. You can make several calls using this data to create bar codes of different sizes and locations, but which contain the same data. Keep in mind that this function is typically run once, while the CreatePDF417Barcode function can be called several times using the same source DAL data.
RequiredLevel	The New York application defines a number of variable data elements to be included in the bar code. Some of these elements are required. This parameter lets you decide when and to what extent to check for these required data elements. If you enter one (1), if any required fields are missing from the data, an error is issued for each missing field. If you enter two (2), if any required fields are missing from the data, a warning is issued for each missing field. If you enter three (3), processing continues and no errors or warnings are issued if required fields are missing. The default is 3. Other INI options control whether the Rules Processor continues processing or stops on errors and warnings. These options are documented in the Rules Processor System Guide.
INS_Key	Each insurance company that writes auto insurance for drivers in New York will receive a unique key signature. This value is an *armored* hex string. This parameter specifies that key. If you omit this parameter, you must include this information in the INI file. For security reasons, you may want to have the source code to this rule customized to hardcode the constant key value into the rule. The base version does not provide this option. If you use the INI file to provide the information, specify the value as shown here: `< NYAAMVA>` Ins_Key = value

This function returns DALERR_SUCCESS if successful. Otherwise, it returns DALERR_USER and shows a dialog.

P417DalCreatePDF417Barcode

Syntax

```
P417DalCreatePDF417Barcode (DALVariableName, FontID, BarCodeName or
BoxName, ECCLevel, TopCoord, BottomCoord, LeftCoord, RightCoord)
```

Parameter	Description
DALVariableName	Specifies the name for the DAL variable that will contain the 570 bytes of formatted data be stored into a PDF417 bar code by a later call. You can make several calls using this data to create bar codes of different sizes and locations, but which contain the same data. Keep in mind that this function is typically run once, while the CreatePDF417Barcode function can be called several times using the same source DAL data.
FontID	Specifies PDF417 font you want to use for this instance of the bar code. You can choose between two sizes of PDF417 bar code fonts: 9 dots high by 12 dots wide and 12 dots high by 16 dots wide. While Oracle does not mandate the IDs you can use, it does provide an FXR file which contains the attributes for the two fonts. The IDs for these fonts are: 912 and 1216. **Note**: Prior to release 11.2, the PDF417_2.FXR and PDF417.FXR files were provided to licensed PDF417 customers. In release 11.2, the new REL112.FXR file contains the PDF417 font references found in the two PDF417 FXR files previously included in this product. See for more information on the included FXR files, see PDF417 FXR Files on page 38.
BarCodeName (or BoxName)	Specifies a name for the bar code or box. This name can be the name of a box object located on the current section. It is required to provide the root name used by a series of dynamic text labels that contain the bar code text data.
ECCLevel	Specifies the degree of error correction to be used in the bar code. The range is from zero (0) to eight (8). The default is four (4), which is the value recommended by the NYSID. For more information, see Sizing a PDF417 Bar Code on page 42.
TopFAPCoord LeftFAPCoord BottomFAPCoord RightFAPCoord	Each bar code instance must specify a location and a rectangular size. There are two ways to specify this information: - from a named box object on the FAP file - from explicit rectangle coordinates The rectangle coordinate parameters are optional. When included, the coordinates explicitly provide the dimensions of the rectangular area to contain the bar code. If you omit the rectangle parameters, the system loads the FAP file for this section and searches for a box named BarCodeName. For optimal performance, specify the rectangle's coordinates.

This function returns DALERR_SUCCESS if successful. Otherwise, it returns DALERR_USER and shows a dialog.

Input Data Structure

The New York state application code expects to receive the NYID-INSURANCE-INFO input data structure. This data structure should be in a native format (such as EBCDIC on MVS, ASCII on Windows NT) and will be converted to ASCII, if necessary by Oracle routines.

NOTE: The New York state required data formatting is adopted from public industry standards. These formatting requirements are accomplished in the software you get from the NYSID. These formats include the American Association of Motor Vehicle Administrators (AAMVA) compliant bar code, MD5 message digest algorithms, and the public domain encryption algorithm called *Triple DES* which generates the required internal digital signature.

The New York application code is written in C/C++ and uses many conventions specific to those languages. Many Oracle implementations will use COBOL programs to interface to this application code. Therefore, part of the interface project involves inter-language communication. Oracle provides COBOL layouts that match the format expected by the C/C++ programs, and will work across the inter-language procedure calls.

The COBOL copybook record structure is defined in NYINSINF.CBL as shown below. COBOL application programs should first set the structure to all NULLs (low values such as binary zero).

This makes sure the FILLER members remain set to NULL as each data member is assigned. As each data member is assigned a value, COBOL will blank fill any unused character positions. The receiving program (written in C/C++) will use a structure view and treat each field as a null-terminated C string.

```
01  NYID-INSURANCE-INFO.
          02  NYID-REGISTRANT-SUB-FILE.
              04  NYID-R-LAST-NAME-1          PIC X(18).
              04   FILLER                     PIC X.
              04  NYID-R-FIRST-NAME-1         PIC X(16).
              04   FILLER                     PIC X.
              04  NYID-R-MIDDLE-NAME-1        PIC X(16).
              04   FILLER                     PIC X.
              04  NYID-R-NAME-SUFFIX-1        PIC X(3).
              04   FILLER                     PIC X.
              04  NYID-R-NAME-AS-ONE-1        PIC X(20).
              04   FILLER                     PIC X.
              04  NYID-R-CLIENT-ID-1          PIC X(9).
              04   FILLER                     PIC X.
              04  NYID-R-LAST-NAME-2          PIC X(18).
              04   FILLER                     PIC X.
              04  NYID-R-FIRST-NAME-2         PIC X(16).
              04   FILLER                     PIC X.
              04  NYID-R-MIDDLE-NAME-2        PIC X(16).
              04   FILLER                     PIC X.
              04  NYID-R-NAME-SUFFIX-2        PIC X(3).
              04   FILLER                     PIC X.
              04  NYID-R-NAME-AS-ONE-2        PIC X(20).
              04   FILLER                     PIC X.
              04  NYID-R-CLIENT-ID-2          PIC X(9).
              04   FILLER                     PIC X.
```

```
    04  NYID-R-STREET                PIC X(20).
    04   FILLER                      PIC X.
    04  NYID-R-CITY                  PIC X(15).
    04   FILLER                      PIC X.
    04  NYID-R-STATE                 PIC X(2).
    04   FILLER                      PIC X.
    04  NYID-R-ZIP-CODE              PIC X(9).
    04   FILLER                      PIC X.
    04  NYID-R-ORGANIZATION-LINE-1   PIC X(20).
    04   FILLER                      PIC X.
    04  NYID-R-ORGANIZATION-LINE-2   PIC X(20).
    04   FILLER                      PIC X.
    04  NYID-R-ORGANIZATION-FEIN     PIC X(9).
    04   FILLER                      PIC X.
02  NYID-VEHICLE-SUB-FILE.
    04  NYID-V-VIN-NUMBER            PIC X(25).
    04   FILLER                      PIC X.
    04  NYID-V-YEAR                  PIC X(4).
    04   FILLER                      PIC X.
    04  NYID-V-MAKE                  PIC X(5).
    04   FILLER                      PIC X.
    04  NYID-V-REP-IND               PIC X(1).
    04   FILLER                      PIC X.
    04  NYID-V-HIST-IND              PIC X(1).
    04   FILLER                      PIC X.
    04  NYID-V-TOW-IND               PIC X(1).
    04   FILLER                      PIC X.
    04  NYID-V-SEATS                 PIC X(2).
    04   FILLER                      PIC X.
    04  NYID-V-VIN-OVERRIDE          PIC X(1).
    04   FILLER                      PIC X.
02  NYID-INSURANCE-SUB-FILE.
    04  NYID-I-DOCUMENT-TYPE         PIC X(15).
    04   FILLER                      PIC X.
    04  NYID-I-PAPER-SELECTION       PIC X(15).
    04   FILLER                      PIC X.
    04  NYID-I-ISSUER-ID             PIC X(10).
    04   FILLER                      PIC X.
    04  NYID-I-INS-COMPANY-CODE      PIC X(3).
    04   FILLER                      PIC X.
    04  NYID-I-INS-COMPANY-NAME      PIC X(40).
    04   FILLER                      PIC X.
    04  NYID-I-INS-ISSUANCE-DATE     PIC X(8).
    04   FILLER                      PIC X.
    04  NYID-I-COVERAGE-START-DATE   PIC X(8).
    04   FILLER                      PIC X.
    04  NYID-I-COVERAGE-END-DATE     PIC X(8).
    04   FILLER                      PIC X.
    04  NYID-I-POLICY-NUMBER         PIC X(15).
    04   FILLER                      PIC X.
    04  NYID-I-AGENCY-NAME           PIC X(40).
    04   FILLER                      PIC X.
    04  NYID-I-AGENCY-ADDRESS-LINE-1 PIC X(40).
    04   FILLER                      PIC X.
    04  NYID-I-AGENCY-ADDRESS-LINE-2 PIC X(40).
```

```
    04   FILLER                      PIC X.
    04  NYID-I-IS-ORGANIZATION       PIC X(1).
    04   FILLER                      PIC X.
02  NYID-DIGITAL-CERT-SUB-FILE.
    04  NYID-S-SIGNATURE-TYPE        PIC X(4).
    04   FILLER                      PIC X.
    04  NYID-S-DIGITAL-SIGNATURE     PIC X(32).
    04   FILLER                      PIC X.
```

Output Data Structure

The New York state ID application code creates the NY-AAMVA-INSURANCE-INFO output data structure. This data structure, in ASCII format, represents the actual content of the PDF417 bar code that will be printed on the ID card.

The COBOL copybook record structure is defined in NYAAMVA.CBL as follows:

```
01  NY-AAMVA-INSURANCE-INFO.
          02  NY-AAMVA-HEADER-DEF.
              04  NY-AAMVA-CMPL-IND            PIC X(1).
              04  NY-AAMVA-DATA-SEP            PIC X(1).
              04  NY-AAMVA-RECORD-SEP          PIC X(1).
              04  NY-AAMVA-SEGMENT-TERM        PIC X(1).
              04  NY-AAMVA-FILE-TYPE           PIC X(5).
              04  NY-AAMVA-IIN                 PIC X(6).
              04  NY-AAMVA-VERSION-NUMBER      PIC X(2).
              04  NY-AAMVA-NO-OF-SUBFILES      PIC X(2).
          02  NY-AAMVA-SUBFILE-DESIGNATOR.
              04  NY-AAMVA-SUB-REG-TYPE        PIC X(2).
              04  NY-AAMVA-REG-OFFSET          PIC X(4).
              04  NY-AAMVA-REG-LEN             PIC X(4).
              04  NY-AAMVA-SUB-VEH-TYPE        PIC X(2).
              04  NY-AAMVA-VEH-OFFSET          PIC X(4).
              04  NY-AAMVA-VEH-LEN             PIC X(4).
              04  NY-AAMVA-SUB-INS-TYPE        PIC X(2).
              04  NY-AAMVA-INS-OFFSET          PIC X(4).
              04  NY-AAMVA-INS-LEN             PIC X(4).
              04  NY-AAMVA-SUB-SIG-TYPE        PIC X(2).
              04  NY-AAMVA-SIG-OFFSET          PIC X(4).
              04  NY-AAMVA-SIG-LEN             PIC X(4).
          02  NY-AAMVA-REGISTRANT-SUBFILE.
              04  NY-AAMVA-SUB-REG-START       PIC X(2).
              04  NY-AAMVA-DES-R-LAST-NAME-1   PIC X(4).
              04  NY-AAMVA-SUB-R-LAST-NAME-1   PIC X(18).
              04  NY-AAMVA-DES-R-GIVEN-NAME-1  PIC X(4).
              04  NY-AAMVA-SUB-R-GIVEN-NAME-1  PIC X(16).
              04  NY-AAMVA-DES-R-MIDDLE-INIT-1 PIC X(4).
              04  NY-AAMVA-SUB-R-MIDDLE-INIT-1 PIC X(1).
              04  NY-AAMVA-DES-R-NAME-SUFX-1   PIC X(4).
              04  NY-AAMVA-SUB-R-NAME-SUFX-1   PIC X(3).
              04  NY-AAMVA-DES-R-NAME-AS-ONE-1 PIC X(4).
              04  NY-AAMVA-SUB-R-NAME-AS-ONE-1 PIC X(20).
              04  NY-AAMVA-DES-R-CLIENT-ID-1   PIC X(4).
              04  NY-AAMVA-SUB-R-CLIENT-ID-1   PIC X(9).
              04  NY-AAMVA-DES-R-LAST-NAME-2   PIC X(4).
              04  NY-AAMVA-SUB-R-LAST-NAME-2   PIC X(18).
              04  NY-AAMVA-DES-R-GIVEN-NAME-2  PIC X(4).
              04  NY-AAMVA-SUB-R-GIVEN-NAME-2  PIC X(16).
              04  NY-AAMVA-DES-R-MIDDLE-INIT-2 PIC X(4).
              04  NY-AAMVA-SUB-R-MIDDLE-INIT-2 PIC X(1).
              04  NY-AAMVA-DES-R-NAME-SUFX-2   PIC X(4).
              04  NY-AAMVA-SUB-R-NAME-SUFX-2   PIC X(3).
              04  NY-AAMVA-DES-R-NAME-AS-ONE-2 PIC X(4).
              04  NY-AAMVA-SUB-R-NAME-AS-ONE-2 PIC X(20).
              04  NY-AAMVA-DES-R-CLIENT-ID-2   PIC X(4).
              04  NY-AAMVA-SUB-R-CLIENT-ID-2   PIC X(9).
              04  NY-AAMVA-DES-R-ORG-LINE-1    PIC X(4).
```

```
    04  NY-AAMVA-SUB-R-ORG-LINE-1    PIC X(20).
    04  NY-AAMVA-DES-R-ORG-LINE-2    PIC X(4).
    04  NY-AAMVA-SUB-R-ORG-LINE-2    PIC X(20).
    04  NY-AAMVA-DES-R-ORG-FEIN      PIC X(4).
    04  NY-AAMVA-SUB-R-ORG-FEIN      PIC X(9).
    04  NY-AAMVA-DES-R-SEATS         PIC X(4).
    04  NY-AAMVA-SUB-R-SEATS         PIC X(2).
    04  NY-AAMVA-DES-R-STREET        PIC X(4).
    04  NY-AAMVA-SUB-R-STREET        PIC X(20).
    04  NY-AAMVA-DES-R-CITY          PIC X(4).
    04  NY-AAMVA-SUB-R-CITY          PIC X(15).
    04  NY-AAMVA-DES-R-STATE         PIC X(4).
    04  NY-AAMVA-SUB-R-STATE         PIC X(2).
    04  NY-AAMVA-DES-R-ZIP-CODE      PIC X(4).
    04  NY-AAMVA-SUB-R-ZIP-CODE      PIC X(5).
    04  NY-AAMVA-DES-REG-END         PIC X(1).
02  NY-AAMVA-VEHICLE-SUBFILE.
    04  NY-AAMVA-SUB-VEH-START       PIC X(2).
    04  NY-AAMVA-DES-V-VIN-NUMBER    PIC X(4).
    04  NY-AAMVA-SUB-V-VIN-NUMBER    PIC X(25).
    04  NY-AAMVA-DES-V-YEAR          PIC X(4).
    04  NY-AAMVA-SUB-V-YEAR          PIC X(4).
    04  NY-AAMVA-DES-V-MAKE          PIC X(4).
    04  NY-AAMVA-SUB-V-MAKE          PIC X(5).
    04  NY-AAMVA-DES-V-REPL-IND      PIC X(4).
    04  NY-AAMVA-SUB-V-REPL-IND      PIC X(1).
    04  NY-AAMVA-DES-V-HIST-IND      PIC X(4).
    04  NY-AAMVA-SUB-V-HIST-IND      PIC X(1).
    04  NY-AAMVA-DES-V-TOW-IND       PIC X(4).
    04  NY-AAMVA-SUB-V-TOW-IND       PIC X(1).
    04  NY-AAMVA-DES-VEH-END         PIC X(1).
02  NY-AAMVA-INSURANCE-SUBFILE.
    04  NY-AAMVA-SUB-INS-START       PIC X(2).
    04  NY-AAMVA-DES-I-ISSUER-ID     PIC X(4).
    04  NY-AAMVA-SUB-I-ISSUER-ID     PIC X(10).
    04  NY-AAMVA-DES-I-INS-CO-CODE   PIC X(4).
    04  NY-AAMVA-SUB-I-INS-CO-CODE   PIC X(3).
    04  NY-AAMVA-DES-I-INS-ISSUE-DT  PIC X(4).
    04  NY-AAMVA-SUB-I-INS-ISSUE-DT  PIC X(8).
    04  NY-AAMVA-DES-I-COV-START-DT  PIC X(4).
    04  NY-AAMVA-SUB-I-COV-START-DT  PIC X(8).
    04  NY-AAMVA-DES-I-COV-END-DT    PIC X(4).
    04  NY-AAMVA-SUB-I-COV-END-DT    PIC X(8).
    04  NY-AAMVA-DES-I-POLICY-NUMBER PIC X(4).
    04  NY-AAMVA-SUB-I-POLICY-NUMBER PIC X(15).
    04  NY-AAMVA-DES-INS-END         PIC X(1).
02  NY-AAMVA-DIG-CERT-SUBFILE.
    04  NY-AAMVA-SUB-SIG-START       PIC X(2).
    04  NY-AAMVA-DES-S-NY-BARCODE    PIC X(4).
    04  NY-AAMVA-SUB-S-NY-BARCODE    PIC X(8).
    04  NY-AAMVA-DES-S-SIG-TYPE      PIC X(4).
    04  NY-AAMVA-SUB-S-SIG-TYPE      PIC X(3).
    04  NY-AAMVA-DES-S-DIG-SIG       PIC X(4).
    04  NY-AAMVA-SUB-S-DIG-SIG       PIC X(32).
    04  NY-AAMVA-DES-SIG-END         PIC X(1).
```

Examining the Output Data

The NY-AAMVA-INSURANCE-INFO structure represents 570 characters of data. This table describes in more detail each of the data elements and its content or source.

NY-AAMVA-INSURANCE-INFO

Output element	Offset	Length	Description
Header information			
NY-AAMVA-CMPL-IND	1	1	"\x40"
NY-AAMVA-DATA-SEP	2	1	"\x0A"
NY-AAMVA-RECORD-SEP	3	1	"\x1C"
NY-AAMVA-SEGMENT-TERM	4	1	"\x0D"
NY-AAMVA-FILE-TYPE	5	5	"AAMVA"
NY-AAMVA-IIN	10	6	"636001"
NY-AAMVA-VERSION-NUMBER	16	2	"03"
NY-AAMVA-NO-OF-SUBFILES	18	2	"04"
Sub-file information			
NY-AAMVA-SUB-REG-TYPE	20	2	"RG"
NY-AAMVA-REG-OFFSET	22	4	"0060"
NY-AAMVA-REG-LEN	26	4	"0310"
NY-AAMVA-SUB-VEH-TYPE	30	2	"VH"
NY-AAMVA-VEH-OFFSET	32	4	"0370"
NY-AAMVA-VEH-LEN	36	4	"0064"
NY-AAMVA-SUB-INS-TYPE	40	2	"FR"
NY-AAMVA-INS-OFFSET	42	4	"0434"
NY-AAMVA-INS-LEN	46	4	"0079"
NY-AAMVA-SUB-SIG-TYPE	50	2	"SI"
NY-AAMVA-SIG-OFFSET	52	4	"0513"
NY-AAMVA-SIG-LEN	56	4	"0058"
Registrant sub-file			
NY-AAMVA-SUB-REG-START	60	2	"RG"
NY-AAMVA-DES-R-LAST-NAME-1	62	4	"\x0ARBD"

Output element	Offset	Length	Description
NY-AAMVA-SUB-R-LAST-NAME-1	66	18	From NYID-R-LAST-NAME-1
NY-AAMVA-DES-R-GIVEN-NAME-1	84	4	“\x0ARBE”
NY-AAMVA-SUB-R-GIVEN-NAME-1	88	16	From NYID-R-FIRST-NAME-1
NY-AAMVA-DES-R-MIDDLE-INIT-1	104	4	“\x0ARBF”
NY-AAMVA-SUB-R-MIDDLE-INIT-1	108	1	From NYID-R-MIDDLE-NAME-1
NY-AAMVA-DES-R-NAME-SUFX-1	109	4	“\x0ARBG”
NY-AAMVA-SUB-R-NAME-SUFX-1	113	3	From NYID-R-NAME-SUFFIX-1
NY-AAMVA-DES-R-NAME-AS-ONE-1	116	4	“\x0ARBC”
NY-AAMVA-SUB-R-NAME-AS-ONE-1	120	20	From NYID-R-NAME-AS-ONE-1
NY-AAMVA-DES-R-CLIENT-ID-1	140	4	“\x0ADAQ”
NY-AAMVA-SUB-R-CLIENT-ID-1	144	9	From NYID-R-CLIENT-ID-1
NY-AAMVA-DES-R-LAST-NAME-2	153	4	“\x0ARBD”
NY-AAMVA-SUB-R-LAST-NAME-2	157	18	From NYID-R-LAST-NAME-2
NY-AAMVA-DES-R-GIVEN-NAME-2	175	4	“\x0ARBE”
NY-AAMVA-SUB-R-GIVEN-NAME-2	179	16	From NYID-R-FIRST-NAME-2
NY-AAMVA-DES-R-MIDDLE-INIT-2	195	4	“\x0ARBF”
NY-AAMVA-SUB-R-MIDDLE-INIT-2	199	1	From NYID-R-MIDDLE-NAME-2
NY-AAMVA-DES-R-NAME-SUFX-2	200	4	“\x0ARBG”
NY-AAMVA-SUB-R-NAME-SUFX-2	204	3	From NYID-R-NAME-SUFFIX-2
NY-AAMVA-DES-R-NAME-AS-ONE-2	207	4	“\x0ARBC”
NY-AAMVA-SUB-R-NAME-AS-ONE-2	211	20	From NYID-R-NAME-AS-ONE-2
NY-AAMVA-DES-R-CLIENT-ID-2	231	4	“\x0ADAQ”
NY-AAMVA-SUB-R-CLIENT-ID-2	235	9	From NYID-R-CLIENT-ID-2
NY-AAMVA-DES-R-ORG-LINE-1	244	4	“\x0AZBC”
NY-AAMVA-SUB-R-ORG-LINE-1	248	20	From NYID-R-ORGANIZATION-LINE-1
NY-AAMVA-DES-R-ORG-LINE-2	268	4	“\x0AZBD”
NY-AAMVA-SUB-R-ORG-LINE-2	272	20	From NYID-R-ORGANIZATION-LINE-2
NY-AAMVA-DES-R-ORG-FEIN	292	4	“\x0AZBE”

Output element	Offset	Length	Description
NY-AAMVA-SUB-R-ORG-FEIN	296	9	From NYID-R-ORGANIZATION-FEIN
NY-AAMVA-DES-R-SEATS	305	4	"\x0ARAP"
NY-AAMVA-SUB-R-SEATS	309	2	From NYID-R-V-SEATS
NY-AAMVA-DES-R-STREET	311	4	"\x0ARBN"
NY-AAMVA-SUB-R-STREET	315	20	From NYID-R-STREET
NY-AAMVA-DES-R-CITY	335	4	"\x0ARBP"
NY-AAMVA-SUB-R-CITY	339	15	From NYID-R-CITY
NY-AAMVA-DES-R-STATE	354	4	"\x0ARBQ"
NY-AAMVA-SUB-R-STATE	358	2	From NYID-R-STATE
NY-AAMVA-DES-R-ZIP-CODE	360	4	"\x0ARBR"
NY-AAMVA-SUB-R-ZIP-CODE	364	5	From NYID-R-ZIP-CODE
NY-AAMVA-DES-REG-END	369	1	"\x0D"
Vehicle sub-file			
NY-AAMVA-SUB-VEH-START	370	2	"VH"
NY-AAMVA-DES-V-VIN-NUMBER	372	4	"\x0AVAD"
NY-AAMVA-SUB-V-VIN-NUMBER	376	25	From NYID-V-VIN-NUMBER
NY-AAMVA-DES-V-YEAR	401	4	"\x0AVAL"
NY-AAMVA-SUB-V-YEAR	405	4	From NYID-V-YEAR
NY-AAMVA-DES-V-MAKE	409	4	"\x0AVAK"
NY-AAMVA-SUB-V-MAKE	413	5	From NYID-V-MAKE
NY-AAMVA-DES-V-REPL-IND	418	4	"\x0AZZD"
NY-AAMVA-SUB-V-REPL-IND	422	1	From NYID-V-REP-IND
NY-AAMVA-DES-V-HIST-IND	423	4	"\x0AZZE"
NY-AAMVA-SUB-V-HIST-IND	427	1	From NYID-V-HIST-IND
NY-AAMVA-DES-V-TOW-IND	428	4	"\x0AZZF"
NY-AAMVA-SUB-V-TOW-IND	432	1	From NYID-V-TOW-IND
NY-AAMVA-DES-VEH-END	433	1	"\x0D"
Insurance sub-file			

Output element	Offset	Length	Description
NY-AAMVA-SUB-INS-START	434	2	"FR"
NY-AAMVA-DES-I-ISSUER-ID	436	4	"\x0AFAA"
NY-AAMVA-SUB-I-ISSUER-ID	440	10	From NYID-I-ISSUER-ID
NY-AAMVA-DES-I-INS-CO-CODE	450	4	"\x0AZZC"
NY-AAMVA-SUB-I-INS-CO-CODE	454	3	From NYID-I-INS-COMPANY-CODE
NY-AAMVA-DES-I-INS-ISSUE-DT	457	4	"\x0AFAB"
NY-AAMVA-SUB-I-INS-ISSUE-DT	461	8	From NYID-I-INS-ISSUANCE-DATE
NY-AAMVA-DES-I-COV-START-DT	469	4	"\x0AFAC"
NY-AAMVA-SUB-I-COV-START-DT	473	8	From NYID-I-COVERAGE-START-DATE
NY-AAMVA-DES-I-COV-END-DT	481	4	"\x0AFAD"
NY-AAMVA-SUB-I-COV-END-DT	485	8	From NYID-I-COVERAGE-END-DATE
NY-AAMVA-DES-I-POLICY-NUMBER	493	4	"\x0AZZB"
NY-AAMVA-SUB-I-POLICY-NUMBER	497	15	From NYID-I-POLICY-NUMBER
NY-AAMVA-DES-INS-END	512	1	"\x0D"
Signature sub-file			
NY-AAMVA-SUB-SIG-START	513	2	"SI"
NY-AAMVA-DES-S-NY-BARCODE	515	4	"\x0AZZZ"
NY-AAMVA-SUB-S-NY-BARCODE	519	8	"IC200010"
NY-AAMVA-DES-S-SIG-TYPE	527	4	"\x0ASAA"
NY-AAMVA-SUB-S-SIG-TYPE	531	3	"001"
NY-AAMVA-DES-S-DIG-SIG	534	4	"\x0ASAB"
NY-AAMVA-SUB-S-DIG-SIG	538	32	From NYID-S-DIGITAL-SIGNATURE
NY-AAMVA-DES-SIG-END	570	1	"\0x0D"

NOTE: The hexadecimal values \x0D and \x0A, shown in the table above, represent the '*Carriage Return*' and '*Line Feed*' characters. When the PDF417 barcode generated from the NY-AAMVA-INSURANCE-INFO structure is scanned, these \x0D and \x0A hexadecimal characters are not displayed in the scanned output, so the output element offsets in the output will differ slightly from the offsets listed in the table above.

Mapping Input Data to Output Data

The NY-INSURANCE-INFO structure is the source of the transaction data that will define the output data in the NY-AAMVA-INSURANCE-INFO structure. A subset of the data elements is moved to the output structure. In some cases, the source data is truncated in the output structure.

This table details which data elements will be transferred to the output structure (and thereby the bar code). This will help define which data elements are actually necessary to assign in the application to create a NYSID bar code.

Source record	Length	Destination record	Length	Required
NYID-R-LAST-NAME-1	18	NY-AAMVA-SUB-R-LAST-NAME-1	18	Yes
NYID-R-FIRST-NAME-1	16	NY-AAMVA-SUB-R-GIVEN-NAME-1	16	Yes
NYID-R-MIDDLE-NAME-1	16	NY-AAMVA-SUB-R-MIDDLE-INIT-1	1	No
NYID-R-NAME-SUFFIX-1	3	NY-AAMVA-SUB-R-NAME-SUFX-1	3	No
NYID-R-NAME-AS-ONE-1	20	NY-AAMVA-SUB-R-NAME-AS-ONE-1	20	Yes *
NYID-R-CLIENT-ID-1	9	NY-AAMVA-SUB-R-CLIENT-ID-1	9	Yes
NYID-R-LAST-NAME-2	18	NY-AAMVA-SUB-R-LAST-NAME-2	18	No
NYID-R-FIRST-NAME-2	16	NY-AAMVA-SUB-R-GIVEN-NAME-2	16	No
NYID-R-MIDDLE-NAME-2	16	NY-AAMVA-SUB-R-MIDDLE-INIT-2	1	No
NYID-R-NAME-AS-ONE-2	20	NY-AAMVA-SUB-R-NAME-AS-ONE-2	20	No
NYID-R-CLIENT-ID-2	9	NY-AAMVA-SUB-R-CLIENT-ID-2	9	No
NYID-R-ORGANIZATION-LINE-1	20	NY-AAMVA-SUB-R-ORG-LINE-1	20	Yes
NYID-R-ORGANIZATION-LINE-2	20	NY-AAMVA-SUB-R-ORG-LINE-2	20	No
NYID-R-ORGANIZATION-FEIN	9	NY-AAMVA-SUB-R-ORG-FEIN	9	Yes
NYID-R-V-SEATS	2	NY-AAMVA-SUB-R-SEATS	2	No**
NYID-R-STREET	20	NY-AAMVA-SUB-R-STREET	20	Yes
NYID-R-CITY	15	NY-AAMVA-SUB-R-CITY	15	Yes
NYID-R-STATE	2	NY-AAMVA-SUB-R-STATE	2	Yes
NYID-R-ZIP-CODE	9	NY-AAMVA-SUB-R-ZIP-CODE	5	Yes
NYID-V-VIN-NUMBER	25	NY-AAMVA-SUB-V-VIN-NUMBER	25	Yes
NYID-V-YEAR	4	NY-AAMVA-SUB-V-YEAR	4	Yes

* If you omit data from this field, it will be added by software routines provided by the New York State DMV.

**The number of seats is a required field for FH-1 and FH-1B ID cards issued to For Hire vehicles (taxis, liveries, rentals, school cars, buses, and so on).

Source record	Length	Destination record	Length	Required
NYID-V-MAKE	5	NY-AAMVA-SUB-V-MAKE	5	Yes
NYID-V-REP-IND	1	NY-AAMVA-SUB-V-REP-IND	1	Yes
NYID-V-HIST-IND	1	NY-AAMVA-SUB-V-HIST-IND	1	No
NYID-V-TOW-IND	1	NY-AAMVA-SUB-V-TOW-IND	1	No
NYID-I-ISSUER-ID	10	NY-AAMVA-SUB-I-ISSUER-ID	10	Yes
NYID-I-INS-COMPANY-CODE	3	NY-AAMVA-SUB-I-INS-CO-CODE	3	Yes
NYID-I-INS-ISSUANCE-DATE	8	NY-AAMVA-SUB-I-INS-ISSUE-DT	8	Yes
NYID-I-COVERAGE-START-DATE	8	NY-AAMVA-SUB-I-COV-START-DT	8	Yes
NYID-I-COVERAGE-END-DATE	8	NY-AAMVA-SUB-I-COV-END-DT	8	Yes
NYID-I-POLICY-NUMBER	15	NY-AAMVA-SUB-I-POLICY-NUMBER	15	Yes
Hard code: “IC200010”		NY-AAMVA-SUB-S-NY-BARCODE	8	No
NYID-S-SIGNATURE-TYPE (“001”)	4	NY-AAMVA-SUB-S-SIG-TYPE	3	Yes
NYID-S-DIGITAL-SIGNATURE	32	NY-AAMVA-SUB-S-DIG-SIG	32	Yes
NYID-V-VIN-OVERRIDE	1	Not used in the bar code	-	-
NYID-I-DOCUMENT-TYPE	15	Not used in the bar code	-	-
NYID-I-PAPER-SELECTION	15	Not used in the bar code	-	-
NYID-I-INS-COMPANY-NAME	40	Not used in the bar code	-	-
NYID-I-AGENCY-NAME	40	Not used in the bar code	-	-
NYID-I-AGENCY-ADDRESS-LINE-1	40	Not used in the bar code	-	-
NYID-I-AGENCY-ADDRESS-LINE-2	40	Not used in the bar code	-	-
NYID-I-IS-ORGANIZATION	1	Not used in the bar code	-	-

* If you omit data from this field, it will be added by software routines provided by the New York State DMV.

**The number of seats is a required field for FH-1 and FH-1B ID cards issued to For Hire vehicles (taxis, liveries, rentals, school cars, buses, and so on).

Input Data Formatting Rules

Some of the data fields in the input structure have special formatting requirements specified by the NYSID.

These requirements are listed below.

Source record	Length	Destination record	Length	Required
NYID-R-NAME-AS-ONE-1	20	NY-AAMVA-SUB-R-NAME-AS-ONE-1	20	Yes
NYID-R-NAME-AS-ONE-2	20	NY-AAMVA-SUB-R-NAME-AS-ONE-2	20	No
NYID-R-ORGANIZATION-LINE-1	20	NY-AAMVA-SUB-R-ORG-LINE-1	20	Yes
NYID-R-ORGANIZATION-LINE-2	20	NY-AAMVA-SUB-R-ORG-LINE-2	20	No

Format the source data with "@" to separate each field as follows:

```
"LASTNAME@FIRSTNAME@MI"
```

Format the organization name using "@" character to separate each word. The name can overflow into the second line. For example:

```
"JOHN@SMITH@DBA@SMITH@TRUCKING"
```

NOTE: Supply the input dates in the InsuranceInfo structure (which is passed to the PNYLoadAAMVA API) in this format: *MMDDYYYY*.

The New York routines, when moving information from the InsuranceInfo structure to the InsuranceAAMVA structure, convert these dates into *YYYYMMDD* format. This format swapping is built into those routines.

PDF417 FXR Files

The PDF417 resources you receive from Oracle Insurance for Documaker include FXR files you can use. Prior to version 11.2, the PDF417_2.FXR and PDF417.FXR files were provided to licensed PDF417 customers. Included in version 11.2, the REL112.FXR file contains the PDF417 font references found in the two PDF417 FXR files previously included in this product.

The 0912 and 1216 font IDs from the PDF417.FXR file (used for 300 dpi printing) are included in the REL112.FXR file.

The 0912 and 1216 font IDs from the PDF417_2.FXR file (only used if your primary printer is an AFP 240 dpi printer) are included in the REL112.FXR file as font IDs 0911 and 1215 to avoid conflicting with the 0912 and 1216 font IDs used for 300 dpi printing.

This table summarizes the differences in the FXR files:

FXR file	Used in version	For these printers	Contains these font IDs
PDF417.FXR	11.1 or lower	300 dpi	0912 and 1216
PDF417_2.FXR	11.1 or lower	240 dpi AFP only	0912 and 1216
REL112.FXR	11.2 and higher	300 or 240 dpi	0912 and 1216 (300 dpi) 0911 and 1215 (240 dpi)

If you are running version 11.1 or lower:

- Only use the PDF417_2.FXR file when your primary printer is an AFP 240 dpi printer. You can also use this FXR file to print to other printers, but only if your primary printer is an AFP 240 dpi printer.
- Use the PDF417.FXR file in all other cases. For instance, if your primary printer is a PCL, Metacode, or 300 dpi AFP printer, you would use the PDF417.FXR file.

If you are running version 11.2 or higher, use the REL112.FXR file.

You can include the contents of these FXR files into the FXR file you are currently using. Studio's Font manager lets you insert fonts from another FXR file. For more information, see the Documaker Studio User Guide.

NOTE: If the font IDs conflict with existing font IDs, renumber them.

PDF417 Fonts

There are 16 characters represented in each of the fonts provided. Each font contains bitmaps for the character set consisting of the underscore character and the uppercase letters A through O.

The characters are fixed pitch. Each cell is evenly divisible by four, to represent four bar-space pattern strokes. The cell height establishes a three-to-one ratio over the stroke width.

Oracle offers two font character cell sizes:

- 12 dots wide by 9 dots tall (approximately a 2 point font size)
- 16 dots wide by 12 dots tall (approximately a 3 point font size)

Oracle offers bitmap fonts for the following printer families:

- AFP (both 240dpi and 300dpi)
- Metacode 300dpi (both portrait and landscape)
- PCL 300dpi (PCL Level 5 (LJ3) and up)

Oracle also offers scalable vector TrueType and PostScript fonts.

Printer family	Cell dimensions	Printer resolution	Page orientation	Font files provided
PostScript Type One	Specify one of these point sizes: 2.16pt or 2.88pt	Not applicable	Not applicable	PDF417__.PFB
TrueType (RP runtime)	Specify one of these point sizes: 2.16pt or 2.88pt	Not applicable	Not applicable	PDF417__.TTF
AFP	9 H x 12 W	240 dpi	Standard	Char set: C0P09X12.240 Coded font: X0P09X12.FNT Code page: T1DOC037.
AFP	12 H x 16 W	240 dpi	Standard	Char set: C0P12X16.240 Coded font: X0P12X16.FNT Code page: T1DOC037.
AFP	9 H x 12 W	300 dpi	Standard	Char set: C0P09X12.300 Coded font: X0P09X12.FNT Code page: T1DOC037.
AFP	12 H x 16 W	300 dpi	Standard	Char set: C0P12X16.300 Coded font: X0P12X16.FNT Code page: T1DOC037.
AFP	15 H x 20 W	300 dpi	Standard	Char set: C0P15X20.300 Coded font: X0P15X20.FNT Code page: T1DOC037

Printer family	Cell dimensions	Printer resolution	Page orientation	Font files provided
Metacode	9 H x 12 W	300 dpi	Portrait	P09X12.FNT
Metacode	12 H x 16 W	300 dpi	Portrait	P12X16.FNT
Metacode	15 H x 20 W	300 dpi	Portrait	P15X20.FNT
Metacode	9 H x 12 W	300 dpi	Landscape	L09X12.FNT
Metacode	12 H x 16 W	300 dpi	Landscape	L12X16.FNT
Metacode	15 H x 20 W	300 dpi	Landscape	L15X20.FNT
PCL	9 H x 12 W	300 dpi	Standard	P09X12.PCL
PCL	12 H x 16 W	300 dpi	Standard	P12X16.PCL
PCL	15 H x 20 W	300 dpi	Standard	P15X20.PCL

The 15x20 fonts, with the exception of the AFP 300 dpi font, are only referenced in the PDF417_2.FXR file in version 11.1 or lower.

NOTE: Prior to version 11.2, the PDF417_2.FXR and PDF417.FXR files were provided to licensed PDF417 customers. Included in version 11.2, the REL112.FXR file contains the PDF417 font references found in two PDF417 FXR files previously included in the product.

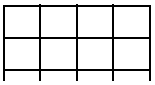
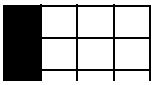
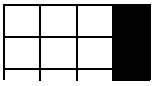
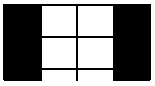
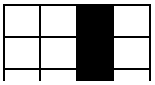
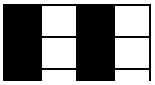
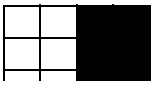
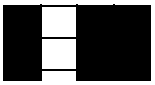
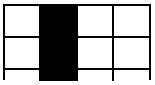
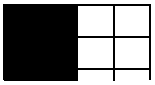
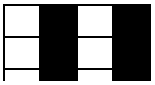
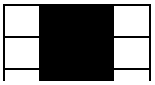
The 0912 and 1216 font IDs from the PDF417.FXR file (used for 300 dpi printing) are included in the REL112.FXR file. The 0912 and 1216 font IDs from the PDF417_2.FXR file (only used if your primary printer is an AFP 240 dpi printer) are included in the REL112.FXR file as font IDs 0911 and 1215 to avoid conflicting with the 0912 and 1216 font IDs used for 300 dpi printing.

Only use the PDF417_2.FXR file when your primary printer is an AFP 240 dpi printer. You can use this FXR to print to other printers (300 dpi) as well but only use it when your primary printer is an AFP 240 dpi printer.

NOTE: This is accomplished by using a different font when printing to a 300 dpi printer. For example, font ID 912 uses a 9x12 font when printing to an AFP 240 dpi printer but uses the 12x16 font when printing to 300 dpi printers (PCL, Metacode). Font ID 1216 uses a 12x16 font when printing to an AFP 240 dpi printer but uses the 15x20 font when printing to 300 dpi printers (PCL, Metacode).

PDF417 Character Set

Here is a representation of the Oracle font character set for PDF417.

Character		Character	
Underscore (_)		H	
A		I	
B		J	
C		K	
D		L	
E		M	
F		N	
G		O	

Sizing a PDF417 Bar Code

A PDF417 bar code can represent a limited amount of data. This data is converted internally and stored in what are called *code words*. There is a maximum of 928 internal code words per bar code. Some of the code words are dedicated to the data content itself, and some of the code words are usually dedicated to error correction (ECC) encoding.

The ECC level can range between zero (0) and eight (8). Each increase in level requires an increase in the amount of the bar code's code words dedicated to error correction. Since there is a maximum number of code words allowed, increasing the amount of error correction decreases the maximum space available for data.

The number of code words dedicated to error correction is calculated in this fashion:

```
CW = 2 ** (ECC + 1)
```

For example:

```
ECC = 0, CW = 2 ** (0+1) = 2 ** 1 = 2
ECC = 1, CW = 2 ** (1+1) = 2 ** 2 = 4
ECC = 2, CW = 2 ** (2+1) = 2 ** 3 = 8
ECC = 3, CW = 2 ** (3+1) = 2 ** 4 = 16
ECC = 4, CW = 2 ** (4+1) = 2 ** 5 = 32
ECC = 5, CW = 2 ** (5+1) = 2 ** 6 = 64
ECC = 6, CW = 2 ** (6+1) = 2 ** 7 = 128
ECC = 7, CW = 2 ** (7+1) = 2 ** 8 = 256
ECC = 8, CW = 2 ** (8+1) = 2 ** 9 = 512
```

The NYSID requires an ECC level of at least 4, equating to 32 internal code words.

The total number of code words (up to the maximum of 928) that can be contained in a PDF417 bar code is computed by determining how many code words are represented by the number of columns in each row, and then multiplying by the number of rows.

The number of rows can range from 3 to 90.

The number of code words per row can be calculated from the specified columns using the following formula:

```
CW=INT(((COLS*100)+425)/425)-5
```

The resultant code words per row can range from 1 to 30. This puts a practical range on the number of columns per row of from 22 to 145.

There is an absolute limit to the size of a bar code of 928 internal code words. All input data and error correction information must be able to be represented by no more than 928 internal code words. Generally, two code words are always consumed by overhead.

The bar code application for the state of New York currently stores 570 characters of text into the bar code. A PDF417 bar code will compact the data somewhat. The fastest and safest method of encoding the data is called *binary mode*. Binary mode compacts the data by approximately a factor of 1.2. For example:

570 / 1.2 = 475 data code words
ECC level 4 = 32 code words
Overhead = 2 code words.
Total required = 509 code words

There are many size and shape combinations of PDF417 bar codes that are just large enough to represent this information. By multiplying a ROWS value times a code words per row value, it is possible to find many combinations greater than 509.

This table shows how to equate text columns to code words per row.

Text columns	Internal code words
149 and more	Too big
145-148	30
141-144	29
136-139	28
132-135	27
128-131	26
124-127	25
119-123	24
115-118	23
111-114	22
107-110	21
102-106	20
98-101	19
94-97	18
90-93	17
85-89	16
81-84	15
77-80	14
73-76	13
68-72	12
64-67	11
60-63	10
56-59	9
51-55	8
47-50	7
43-46	6
39-42	5

Text columns	Internal code words
34-38	4
30-33	3
26-29	2
22-25	1
21 and less	Too small

For example 30 rows times 18 code words per row equals 540, which should be large enough. And, 18 code words per row equate to 94 text columns. So, a bar code of 30 rows and 94 columns should be safely large enough for the New York application.

From the previous tables, assuming the smaller font and 300 dpi, the bar code would be 9/10ths of an inch tall and about 3 and 7/10ths inches wide.

Determining the Columns per Row

Use this table to determine the number of font character columns that will be required to create a bar code of the desired width. Notice that because of the way the PDF417 bar code works, only certain numbers of columns can be returned.

Programs should be written to use one of the choices listed in the table. If the program requests a different size, the remaining columns at the end of the row will be padded with underscores.

For example, if the program establishes a row of one hundred character columns in width, the Oracle API returns 98 columns of bar code data and fills the remaining characters at the end of each row with underscores.

If each row in the receiving buffer is this many bytes (PIC X(39))	Here is the resulting bar code width in inches, using a...			
	12-dot character width font at 300 dpi	12-dot character width font at 240 dpi	16-dot character width font at 300 dpi	16-dot character width font at 240 dpi
39-42	1.52	1.90	2.03	2.53
43-46	1.68	2.10	2.24	2.80
47-50	1.88	2.35	2.51	3.13
51-55	2.04	2.55	2.72	3.40
56-59	2.20	2.75	2.93	3.67
60-63	2.36	2.95	3.15	3.93
64-67	2.56	3.20	3.41	4.27
68-72	2.72	3.40	3.63	4.53
73-76	2.88	3.60	3.84	4.80
77-80	3.04	3.80	4.05	5.07
81-84	3.24	4.05	4.32	5.40
85-89	3.40	4.25	4.53	5.67
90-93	3.56	4.45	4.75	5.93
94-97	3.72	4.65	4.96	6.20
98-101	3.92	4.90	5.23	6.53
102-106	4.08	5.10	5.44	6.80
107-110	4.24	5.30	5.65	7.07
111-114	4.40	5.50	5.87	7.33
115-118	4.60	5.75	6.13	7.67
119-123	4.76	5.95	6.35	7.93

If each row in the receiving buffer is this many bytes (PIC X(39))	Here is the resulting bar code width in inches, using a...			
	12-dot character width font at 300 dpi	12-dot character width font at 240 dpi	16-dot character width font at 300 dpi	16-dot character width font at 240 dpi
124-127	4.92	6.15	6.56	8.20
128-131	5.08	6.35	6.77	8.47
132-135	5.28	6.60	7.04	8.80

Determining the Number of Rows

Use this table as a guideline to determine the number of font character rows that will be required to create a bar code of the desired height.

Other values are allowed, and the table can be a guide on extrapolating other row sizes. The PDF417 bar code specification has a maximum limit of 90 rows.

For each row in the receiving buffer (OCCURS 20 TIMES)	Here is the resulting bar code height in inches, using a...			
	9-dot character height font at 300 dpi	9-dot character height font at 240 dpi	12-dot character height font at 300 dpi	12-dot character height font at 240 dpi
10	0.3000	0.3750	0.4000	0.5000
11	0.3300	0.4125	0.4400	0.5500
12	0.3600	0.4500	0.4800	0.6000
13	0.3900	0.4875	0.5200	0.6500
14	0.4200	0.5250	0.5600	0.7000
15	0.4500	0.5625	0.6000	0.7500
16	0.4800	0.6000	0.6400	0.8000
17	0.5100	0.6375	0.6800	0.8500
18	0.5400	0.6750	0.7200	0.9000
19	0.5700	0.7125	0.7600	0.9500
20	0.6000	0.7500	0.8000	1.0000
21	0.6300	0.7875	0.8400	1.0500
22	0.6600	0.8250	0.8800	1.1000
23	0.6900	0.8625	0.9200	1.1500
24	0.7200	0.9000	0.9600	1.2000
25	0.7500	0.9375	1.0000	1.2500
26	0.7800	0.9750	1.0400	1.3000
27	0.8100	1.0125	1.0800	1.3500
28	0.8400	1.0500	1.1200	1.4000
29	0.8700	1.0875	1.1600	1.4500
30	0.9000	1.1250	1.2000	1.5000
31	0.9300	1.1625	1.2400	1.5500
32	0.9600	1.2000	1.2800	1.6000

For each row in the receiving buffer (OCCURS 20 TIMES)	Here is the resulting bar code height in inches, using a...			
	9-dot character height font at 300 dpi	9-dot character height font at 240 dpi	12-dot character height font at 300 dpi	12-dot character height font at 240 dpi
33	0.9900	1.2375	1.3200	1.6500
34	1.0200	1.2750	1.3600	1.7000
35	1.0500	1.3125	1.4000	1.7500
36	1.0800	1.3500	1.4400	1.8000
37	1.1100	1.3875	1.4800	1.8500
38	1.1400	1.4250	1.5200	1.9000
39	1.1700	1.4625	1.5600	1.9500
40	1.2000	1.5000	1.6000	2.0000
41	1.2300	1.5375	1.6400	2.0500
42	1.2600	1.5750	1.6800	2.1000
43	1.2900	1.6125	1.7200	2.1500
44	1.3200	1.6500	1.7600	2.2000
45	1.3500	1.6875	1.8000	2.2500

TIPS

This section provides information you made need to get the best results and to resolve any problems which may occur.

PRODUCING RELIABLE BAR CODES

The most reliable bar codes are produced when you use one of Documaker's standard print drivers (AFP, Xerox Metacode, PCL, or PostScript) to produce a print stream that is sent directly to the appropriate printer. Bar codes scanned directly from these printouts are more reliable when scanned because...

- Documaker completely controls the print stream produced
- The printouts are of high resolution (240 or 300 dots per inch)
- Most AFP, Xerox Metacode, PCL, or PostScript printers are very reliable and consistent

You lessen the reliability of scanned bar codes when you...

- Produce bar codes in an alternate document format such as PDF, RTF, or HTML
- Fax the bar code

Producing bar codes in PDF, RTF, or HTML format

When you produce bar codes in an alternate document format such as PDF, RTF, or HTML, you reduce the reliability of the bar codes being produced. Applications such as Internet Explorer, Microsoft Word, and Adobe Acrobat must first interpret and display the document format. The interpretation and display of the document will often vary depending upon the version of the software used. These applications rely on other applications, such as Windows print drivers, to print the document which includes the PDF417 bar codes. High reliability and consistency can be more difficult to achieve in this environment as there are many factors that affect the final printed results.

Faxing

Most fax machines scan and transmit documents at a far lower resolution than the original printed documents. High resolution is defined as 196 x 204 dpi and low resolution is defined as 96 x 104 dpi (dots per inch).

Improving readability

Here are some things you can do to improve the readability of PDF417 bar codes:

- If possible, increase the Error Correction (ECC) Encoding when producing the PDF417 bar code.
- If you must fax the document, fax at the highest possible resolution.
- Make sure you do not have any fit to width options enabled on your Window's print or FAX driver. These options shrink output and that can distort the bar code.
- Turn off any graphic smoothing features. Smoothing is used by some print or FAX drivers to make bitmaps look better. This can distort the bar code.
- If your FAX driver will accept PCL or PostScript output, try sending PCL or PostScript output to eliminate the PDF step.
- Use the option to print as a graphic image if the application or Windows print driver allows it. This approach is slower but may give you a more precise printout.

- Experiment with various advanced print settings within the 3rd party application and test using different printers and Windows print drivers.

TURNING ON TRACING

To help you spot errors, you can turn on tracing. To turn on tracing for the PNYLIB and PSYLIB modules, set the INI options shown below. Setting one or both of these options to Yes tells the system to create a trace file named PDF417.LOG.

```
< Debug_Switches >
   Enable_Debug_Options = Yes
   PNYLIB = Yes
   PSYLIB = Yes
```

NOTE: Omit these options or set them to No during normal operation. Tracing slows performance.

ERROR MESSAGES

Here is a summary of the error messages you may encounter.

PNY01A

This table shows the error messages that might possibly be returned in the error return-code variable parameter passed to the PNY01A API that prepares the bar code data into the AAMVA-compliant format.

Error	Description
0	Successful completion.
8	Error in creating the document signature. Check the KEY and PASS PHRASE parameters.

P4172FNT

This table shows the error messages that might possibly be returned in the error return-code variable parameter passed to the P4172FNT API that creates the font text for a bar code.

The errors that have descriptions denoted by ***INTERNAL ERROR* should ordinarily never be observed by an application invoking the Oracle API. These errors should be reported to Oracle Support. The other errors may result from incorrect usage, and the description notes where to begin to determine the cause of the error.

Error	Symbolic name	Description
0	ERR_NOERROR	Successful completion.
-4	ERR_NULLDATASOURCE	**INTERNAL ERROR. No input routine was specified.
-5	ERR_INVALIDINPUTOBJECT	**INTERNAL ERROR. The PDF object is corrupt.
-7	ERR_TOOMANYCW	**INTERNAL ERROR. Re-size of bar code cannot contain data.

Error	Symbolic name	Description
-8	ERR_NOTINPUTDEVICE	**INTERNAL ERROR. The PDF object was not input type.
-9	ERR_OUTPUTDEVICE	**INTERNAL ERROR. Error initializing output object -- invalid type or memory allocation failure.
-10	ERR_NOFILTERSELECTED	**INTERNAL ERROR. No output driver setup/installed.
-11	ERR_TOOMUCHDATA	The input data length exceeds the bar code. The bar code size requested (via the rows and columns parameters) is not large enough to contain the input data. Check the form design and see if you can make the bar code area larger.
-12	ERR_NOACTIVEBARCODE	**INTERNAL ERROR. Attempt to print/query without bar code.
-14	ERR_MPCBTOOMUCH	** INTERNAL ERROR. The bar code is too small for the MPDF information.
-16	ERR_TOOMANYMPDF	**INTERNAL ERROR. There are too many bar codes for the MPDF information.
-17	ERR_ECCEXCEEDSCAPACITY	The bar code is too small for the ECC specified. The bar code size requested (via the rows and columns parameters) might be large enough to contain the data, but is not large enough to represent the data given the error correction value you specified. Increase the size of the bar code or make sure the ECCLevel parameter is correct.
-20	ERR_MEMORYALLOC	**INTERNAL ERROR. Failed to allocate a temporary buffer.
-21	ERR_ARGUMENT	**INTERNAL ERROR. Invalid argument: not supplied, out of range, and so on. If this occurs, make sure the input parameters are valid.
-22	ERR_INVALIDECIESCAPE	**INTERNAL ERROR. Ill-formed ECI escape sequence.
-23	ERR_C128CONTENT	**INTERNAL ERROR. Invalid data in the code128 emulation input.
-24	ERR_INVALIDMACROCHAR	**INTERNAL ERROR. Ill-formed macro character substitution sequence in input.
-99	ERR_FATALINTERNAL	**INTERNAL ERROR. Internal API error.
-100	ERR_FONTFATALINTERNAL	**INTERNAL ERROR. Internal conversion error.
-101	ERR_FONTOBJALLOC	**INTERNAL ERROR. A failure occurred during internal buffer memory allocation.
-102	ERR_FONTROWCOL	Incorrect row/column combination. The ROW parameter can range from 3 to 90. The COLS parameter is converted into an internal code word using this formula: `CW=INT(((COLS*100)+425)/425)-5` The resultant internal code words can range from 1 to 30. This puts a practical range on the COLS parameter of from 22 to 145. The absolute limit to the size of a bar code is 928 internal code words. All input data and error correction information must be able to be represented by no more than 928 internal code words.

Error	Symbolic name	Description
-103	ERR_FONTECC	Incorrect ECC level. Check the ECCLevel parameter value. The valid range is zero (0) through eight (8).
-104	ERR_FONTOUTBUFFTOOSMALL	The output buffer is too small. Check the parameter that defines the size of the output buffer. Make sure it is large enough to represent the size you need.

If you have multi-page sections, you may get one of these messages, depending on how you set the W parameter in the CreatePDF417Barcode rule.

If you have a multi-page section, you will not get a warning message when the system cannot locate the specified bar code or box if you set the Warning parameter of the CreatePDF417Barcode rule to No. Otherwise, you get the following warning message:

```
DM20402:  Warning In CREATEPDF417BARCODE():  The system is unable to
find the specified name for the barcode or box <P417BX1> for image
<AAA> on page <2>.
```

Index

F

G

I

K

L

M

N

O

P

R

S

T

U

V