THE DESIGNER

User Reference Manual

The vpxPrint Designer

Version 4.2

4GL
vpxPrint Studio

The vpxPrint Designer is a software designed, developed and distributed by 4GL.

This software is a graphical extension, it complements the vpxPrint software, essential for its operation for the creation of reports and projects.

The reference site for downloading software is https://www.4GL.fr

Do not hesitate to consult the videos on the Designer at https://www.4GL.fr/designer.html

Support and information are available for all our users registered on our site or by email at support@4GL.fr

Copyright © 4GL

vpxPrint version 10.47 *is the minimum version compatible with version 4.2 of the Designer, it is necessary to make sure that your license allows you to use it.*

TABLE OF CONTENTS

1 Introduction 5
1.1 History 5
1.2 Evolution 5
1.3 The basic principles 5
2 History of changes 7
3 Programming concepts 10
3.1 The working environment 10
3.1.1 Data schema 10
3.1.2 XML Data 13
3.1.3 SQL-Server data file format: 18
3.1.4 Database interfaces (mySQL, SQLServer,Postgres…) 19
3.2 Run a project 26
3.2.1 vpRun.dll parameters: 26
3.2.2 Calling from C# 29
3.2.3 Calling from VB 30
3.2.4 Calling from Delphi 31
3.3 USE OF THE PROJECT 32
3.3.1 Project processing 32
4 The Designer 33
4.1 Graphical Interface 33
4.1.1 Main screen 33
4.1.2 Mail Interface 34
4.1.3 PDF Interface 34
4.1.4 Language and theme of the interface 35
4.1.5 Objects 36
4.1.6 Data schema 36
4.1.7 The workspace 38
4.1.8 The pages 38
4.1.9 Basic objects 42
4.1.10 Complex objects 44
4.2 PROGRAMMING AN OBJECT 58
4.2.1 Properties window 58
4.2.2 Definition of object content 60

4.2.3 Programming an array object 62
4.2.4 Array Object format 65
4.2.5 Labels Object 69
4.3 Processing 72
4.3.1 Logic 72
4.3.2 Each master record is merged with the "Main_xxx" pages 73
5 Functions 74
5.1 Formats 74
5.2 Programming, expressions 75
6 Table of illustrations 77
7 Index 78

1 INTRODUCTION

1.1 HISTORY

Historically, vpxPrint was created in 2000 to allow PROGRESS users to generate graphical prints while this language only supported character-based printing at the dot-matrix printer standards of the time. And we programmed our editions with ESC escape sequences ...

When the inkjet and laser printers appeared, Progress's printing system did not follow this evolution, leaving us in the hands of Report Builder and other software that is unsuited to the needs of developers.

When the first versions of vpxPrint (former name xPrint) were proposed, this simple and effective system was very quickly a resounding success in 46 countries and thousands of sites and users.

Over the years, the system has improved considerably. Without going into details, today is a system without equivalent. Capable of making edits with or without preview, creating PDF files, emails with attached files and HTML text, TIF or graphic files, Excel or PowerPoint files, rearranging the page order of PDF files, it integrates with finesse to your applications, it can do all this with user intervention or silently.

When you look at the power and the complicity that vpxPrint has with your application, you quickly realize that it is irreplaceable.

That's why many prestigious computer companies have chosen it.

32 or 64 bits, characters ANSI or Unicode, Active-X or dll, vpxPrint is a complete and powerful system.

1.2 EVOLUTION

Today, the Designer allows to generate a vpxPrint file (* .xpr) with a graphical interface, thus combining the best of both worlds.

The Designer is a professional tool, just like you. We all know the insurmountable problems of footer management, orphan totals alone on a page, partial tabulations, printed papers with fixed formats, backgrounds visible on the screen but not printable, etc ... , professionals problems!

So, we have definitely fixed these issues and integrated all these features for you. Choose your rules, quickly and efficiently change the documents of your users or customers.

1.3 THE BASIC PRINCIPLES

It is usually difficult to use data from database files directly and without checking. Added to this are the performance issues of data access.

Thus the designer does not connect directly to the databases but uses the data from a structured file in XML format.

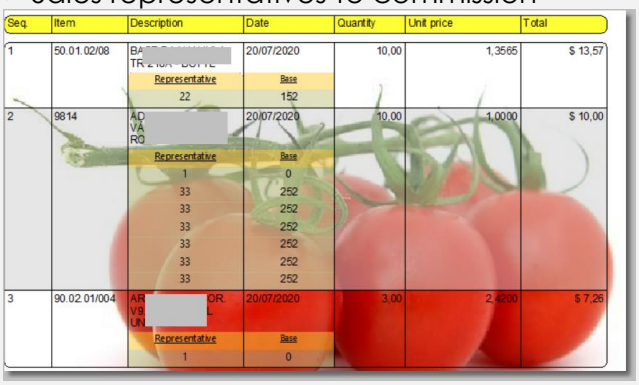
Accomplices of Progress for a long time, we provide you the programs to create these XML files from a simple request « FOR EACH customer WHERE …, EACH order OF customer …»

The Designer is also compatible with the XML AUTO format of SQL-Server.

The following versions will propose this option for all the other languages and management systems of databases, even if the structure of such an XML file is simple to generate.

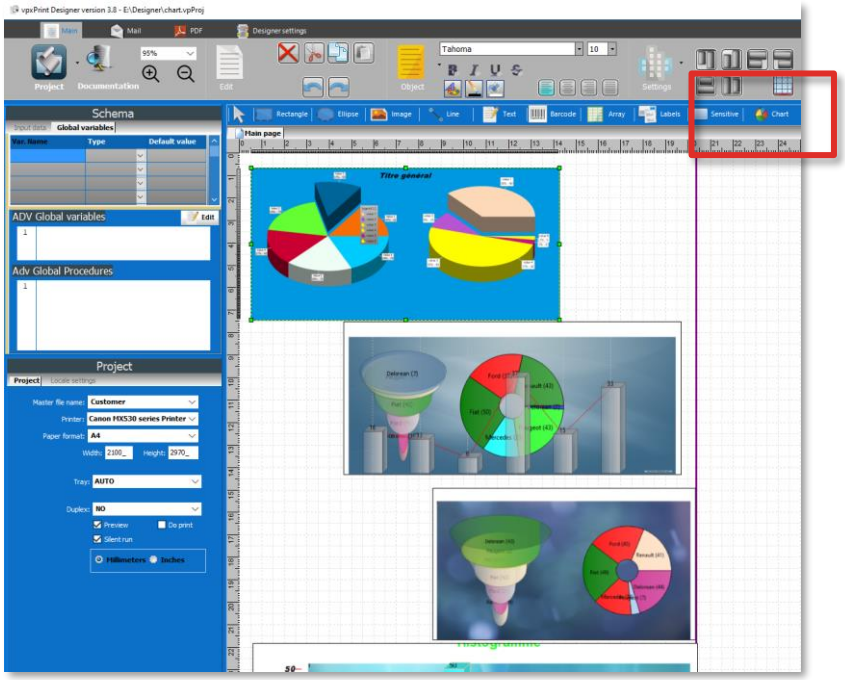
With version 3.1, we provide interfaces with mySQL, SQL-Server, Advantage Database Server (ADS), Firebird, Interbase, IB Lite, MS-Access, PostgreSQL, SQLite and Microsoft Jet & Microsoft OLE DB drivers.

2 HISTORY OF CHANGES

VERSION	
4.2 (APRIL 2024)	• Compatibility with queries including PROGRESS "temp-tables". Data export treats these tables as database tables (XML_Data.w) • Dynamic creation of base schemas, "*.vpschema*" type files: The dynamic creation of schema files makes it possible to select temporary tables and database files based on search criteria (matches) to restrict schemas at the design stage. • Possibility of adding a schema to the active schema during design. • When a data file is supplied in design mode, the hierarchy of its file treeviews is automatically reduced. These adaptations make the design environment extremely easy and pleasant to use when the number of tables is large. • New query level in arrays. Example: Customer -> Customer orders -> Order lines -> Sales representatives to commission  • PDF export corrections: vpxPrint 10.47 recommended • Various improvements and optimizations...
4.1 (SEPTEMBER 2023)	• The Designer is now **free of charge** for all vpxPrint users, • It is available for Windows, Linux and MacOS environments: Linux & MacOS reference Manual

3.8
(MARCH 2023)

- **- DESIGN OF GRAPHICS IN WYSIWYG MODE:**



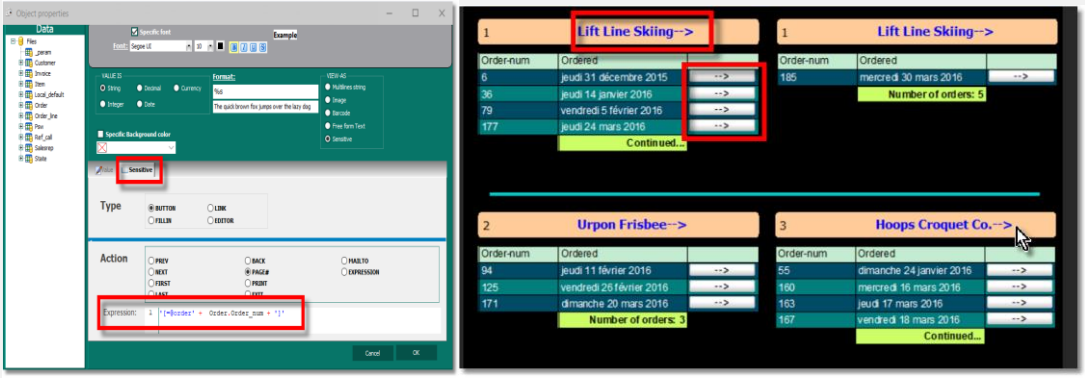
- **- NEW ENTRY POINTS TO ADD CUSTOM CODE IN THE "RUN" OF A PROJECT (VPRUN).**

3.8A – 38E:

- **DIFFERENT FIXES AND OPTIMIZATION,**
- **NEW STYLES : COPPER, COPPERBLACK,**
- **DUPLICATE FILES ARE NOW DETECTED WHEN IMPORTING A SCHEMA,**
- **…**

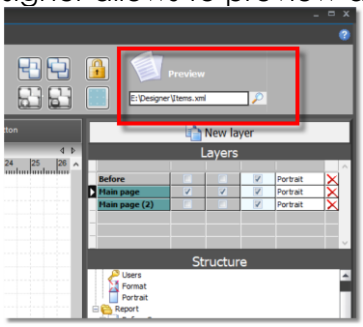
3.6
(JANUARY 2022)

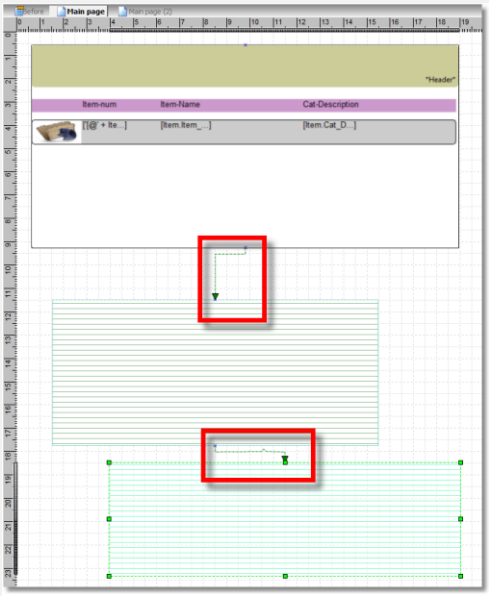
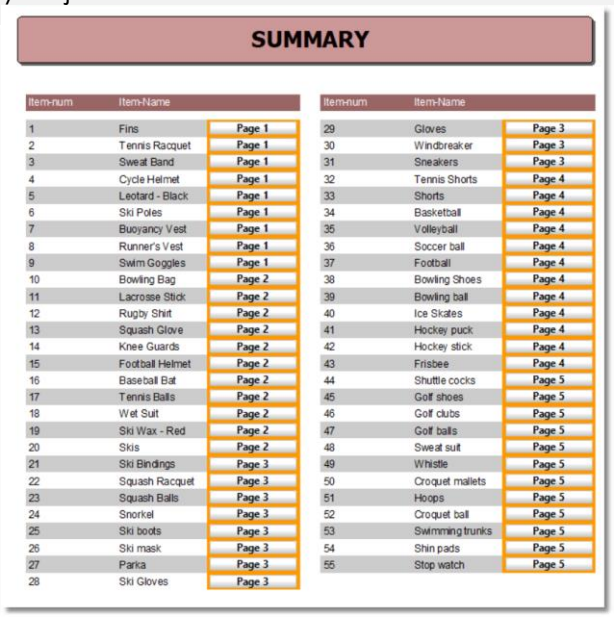
- Sensitive fields with calculated expressions:



3.5
(DECEMBER 2021)

- The Designer allows to preview a project from the conception window:



	• Some additional internal enhancements…
3.4 (DECEMBER 2021)	• Added chained arrays in a page. Several "array" objects can be placed in the same page, the records are automatically dumped from the initial array to the chained objects. These inherit the width of the source array object:   SUMMARY The order of the chained objects can be changed: ▪ "bring to front": places the object in the last position of the chain, ▪ "send to back": places the object in the first position of the chain,
3.2 [AUGUST 2021]	• Considerable improvement in loading time of large db schemas
3.1 [MAY 2021]	• Interface SQL **select** with **mySQL, SQL-Server, Advantage Database Server (ADS), Firebird, Interbase, IB Lite, MS-Access, PostgreSQL, SQLite** and **Microsoft Jet & Microsoft OLE DB drivers.** • Improved vertical cell centering, • Rounded corners of rounded and colored tables,…
2.20	Dynamic modification of database field values when creating XML data with XML_Data.w Stretched images (STRETCH) in image frame (10.31 vpxPrint) Shadow management of rectangles and ellipses New button to adjust objects to the grid ("Snap to grid")
2.19	Internal optimizations
2.18	Export of PROGRESS array fields (extent) in XML_Data.w Syntax check of procedures Adjusting property windows
2.17	SQL-Server data files compatibility

3 PROGRAMMING CONCEPTS

3.1 THE WORKING ENVIRONMENT

3.1.1 Data schema

The data pattern is provided to the designer by a text file with a suffix ".vpSchema". Each file is defined by a line starting with the character **>**

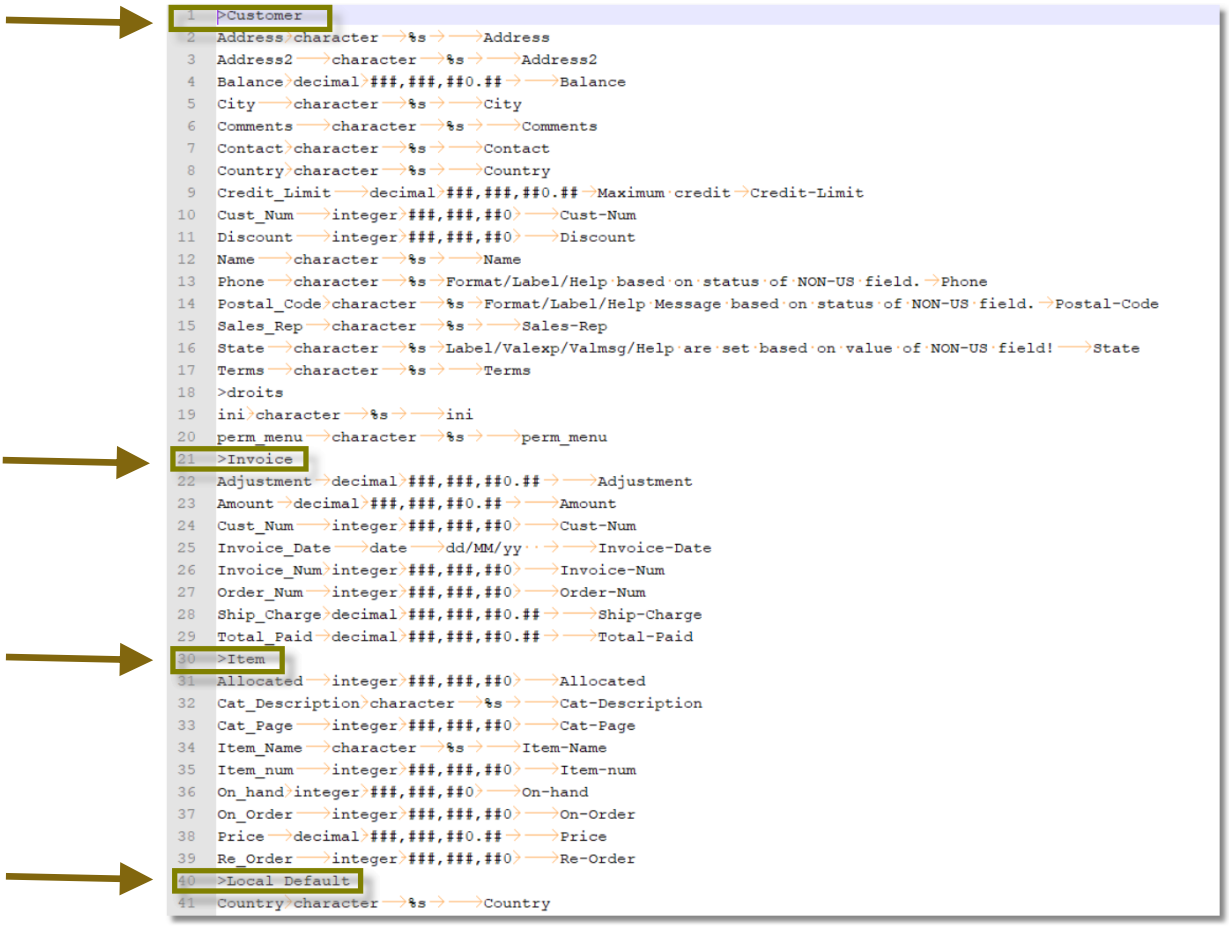


Figure 1 - ".vpSchema" file

Each line defines a file field, it is a list of 5 elements with a delimiter "tab" or chr (10):

1. Name of the data: dashes are replaced by _
2. Type: integer, character, decimal, date or logical
3. Format: presentation format, see Data formats
4. Label: field label
5. Hint: text of the help

A DOUBLE-CLICK ON A ".VPSCHEMA" FILE IN THE WINDOWS EXPLORER OPENS THE DESIGNER WITH THIS DATA DICTIONARY.

3.1.1.1. Static schema of a database

For Progress users, the **exportSchema.w** utility makes it easy to create this file.

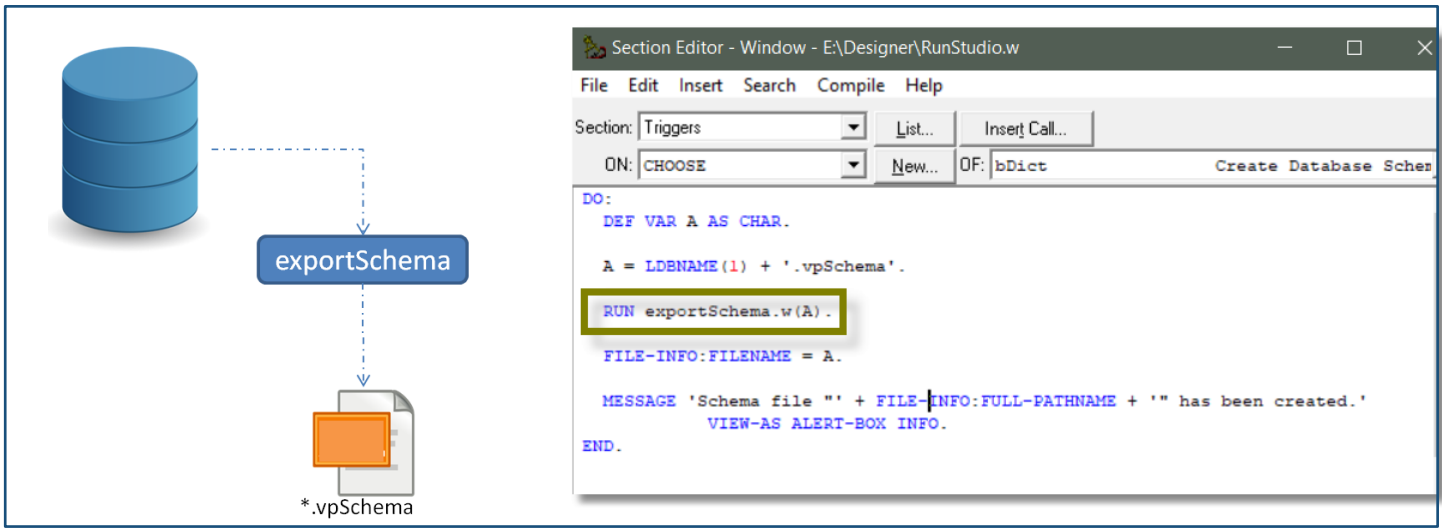


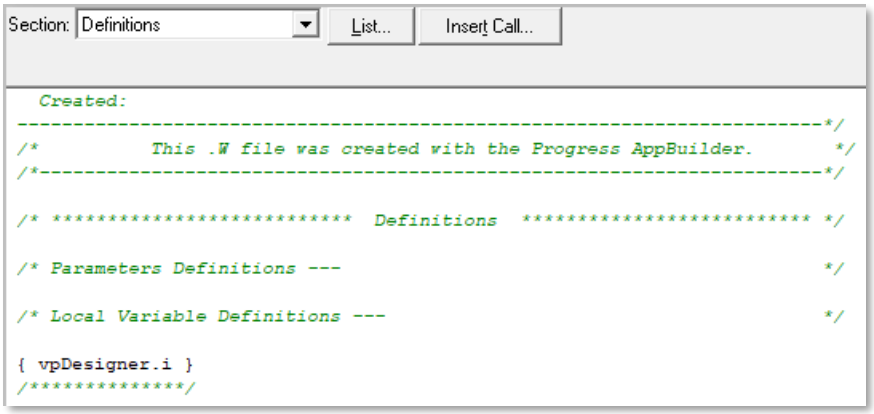
***Figure** 2 - exportSchema*

*(from **RunStudio.w**)*

3.1.1.2. Dynamic schemas

Version 4.2 allows to create dynamic schema files.

- Include **vpDesigner.i** :



vpDesigner.i defines:

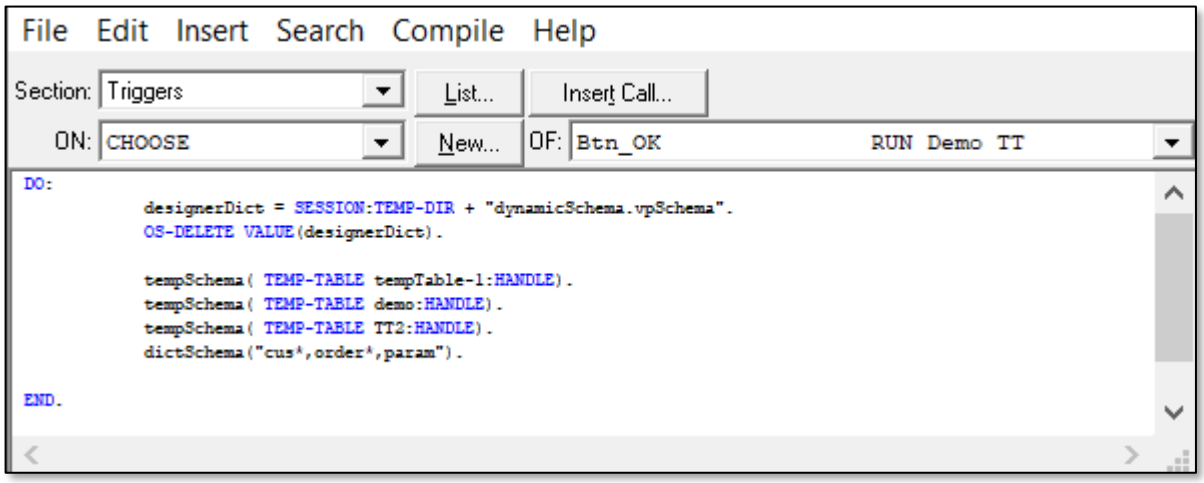
a. A **designerDict** string variable indicating the name of the .vpSchema file to be generated,

b. A **tempSchema**(table_handle) function
   Exports the temp-table schema to the .vpschema file,

- A **dictSchema**(*tables_list*) function.
  Where *tables_list* defines the list of database tables to be included in the .vpschema.
  The rule for constructing this list is identical to **CAN-DO** PROGRESS function.

- The .vpSchema definition file is built in "APPEND" mode, so all combinations of tempSchema and dictSchema are valid.

  Simply delete the .vpschema to create a new one:



3.1.2 XML Data

This file contains the fields of the database corresponding to the query to be processed.

The Designer accepts two XML data file formats, the standard one and the SQL-Server format.

3.1.2.1 Standard format of an XML data file:

We find the file hierarchy of the query in the data structure.

```
FOR EACH customer,
EACH order OF customer,
EACH order-line OF order,
FIRST item OF order-line
```

```
<?xml version="1.0"?>
<data>
<file name="customer">
    <Cust_Num>1</Cust_Num>
    <Country>USA</Country>
    <Name>Lift Line Skiing</Name>
    <Address>276 North Street</Address>
    <Address2></Address2>
    <City>Boston</City>
    <State>MA</State>
    <Postal_Code>02114</Postal_Code>
    <Contact>Gloria Shepley</Contact>
    <Phone>(617) 450-0087</Phone>
    <Sales_Rep>HXM</Sales_Rep>
    <Credit_Limit>66700</Credit_Limit>
    <Balance>42568</Balance>
    <Terms>Net30</Terms>
    <Discount>35</Discount>
    <Comments>This customer is on credit hold.</Comments>
    <file name="order">
        <Order_num>6</Order_num>
        <Cust_Num>1</Cust_Num>
        <Order_Date>31/12/15</Order_Date>
        <Ship_Date>03/02/16</Ship_Date>
        <Promise_Date>29/01/16</Promise_Date>
        <Carrier>Standard Mail</Carrier>
        <Instructions></Instructions>
        <PO></PO>
        <Terms>Net30</Terms>
        <Sales_Rep>HXM</Sales_Rep>
        <file name="order_line">
            <Order_num>6</Order_num>
            <Line_num>1</Line_num>
            <Item_num>9</Item_num>
            <Price>18</Price>
            <Qty>67</Qty>
            <Discount>35</Discount>
            <Extended_Price>783.9</Extended_Price>
            <Backorder>no</Backorder>
            <file name="item">
        </file>
        <file name="order_line">
```

Figure 3 – XML Data file

3.1.2.2 XML_Data.w

FOR PROGRESS USERS, XML_DATA.W ALLOWS CREATING AN XML DATA FILE FROM A CLASSIC PROGRESS REQUEST.

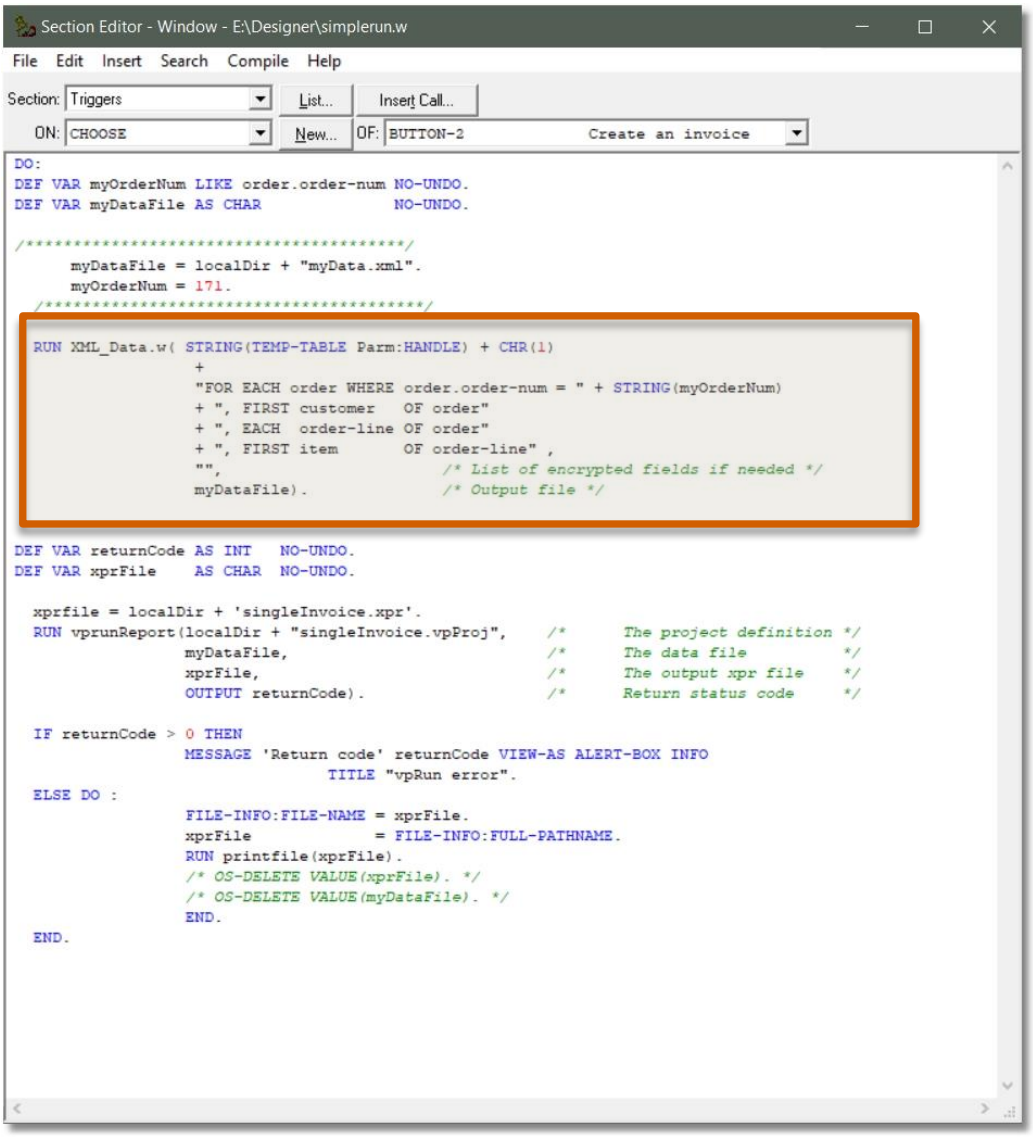


Figure 4 – Creating an XML file

*(from **simpleRun.w**)*

Parameters :

1. Query:
   a. [**String**(**Handle** TEMP-TABLE) + **chr(1)**] -optional-
   *(if this first part is present, XML_Data adds a parameter file _param out of the standard query, it is the possibility to **transmit parameters during the execution**)*
   b. "**FOR EACH**" query
2. List of fields to encrypt,
3. XML file to create

Example :

```
DEF TEMP-TABLE parm NO-UNDO
            FIELD editedBy      AS CHAR
            FIELD editedDate    AS DATE
            FIELD INFO          AS CHAR.
CREATE parm.
ASSIGN      editedBy        = 'Myself'
            editedDate      = TODAY
            INFO            = F.
```

```
<?xml version="1.0"?>
<data>
<file name="_param">
    <editedBy>Myself</editedBy>
    <editedDate>10/12/18</editedDate>
    <INFO>E:\Designer/example.txt</INFO>
</file>
<file name="order">
    <Order_num>171</Order_num>
    <Cust_Num>2</Cust_Num>
    <Order_Date>20/03/16</Order_Date>
    <Ship_Date>?</Ship_Date>
    <Promise_Date>13/04/16</Promise_Date>
    <Carrier>Standard Mail</Carrier>
    <Instructions></Instructions>
    <PO></PO>
    <Terms>Net30</Terms>
    <Sales_Rep>DKP</Sales_Rep>
    <file name="customer">
        <Cust_Num>2</Cust_Num>
```

The _param file is created outside the standard hierarchy of the query.

When exporting a field from the database, XML_Data.w checks if a function named "filename_fieldname" exists in the program that calls XML_Data.

If such a procedure exists, it is called with the following parameters:

- INPUT iRowid AS ROWID
- iNPUT FieldValue AS CHAR

and should return the result in character format.

Example:

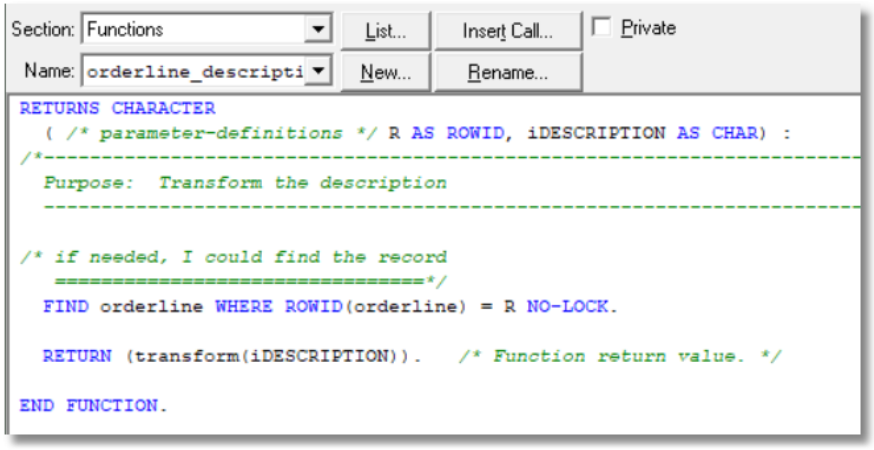


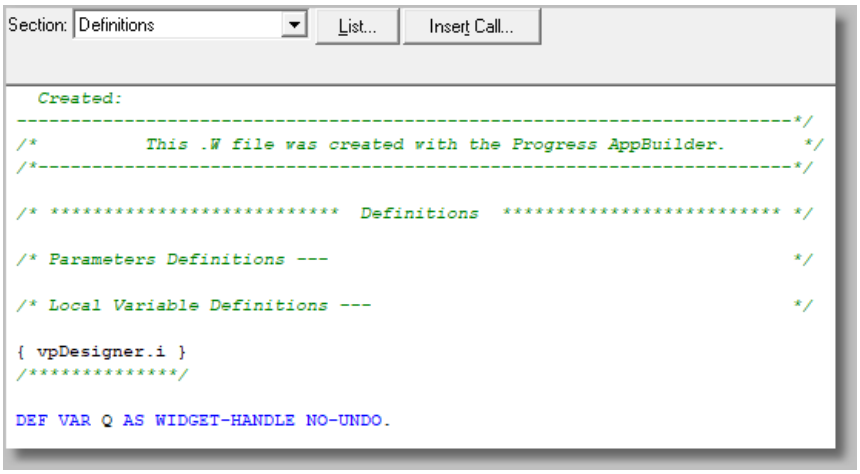
Table OrderLine,champ Description -> FUNCTION **OrderLine_Description**

3.1.2.2.1 Dynamic queries:

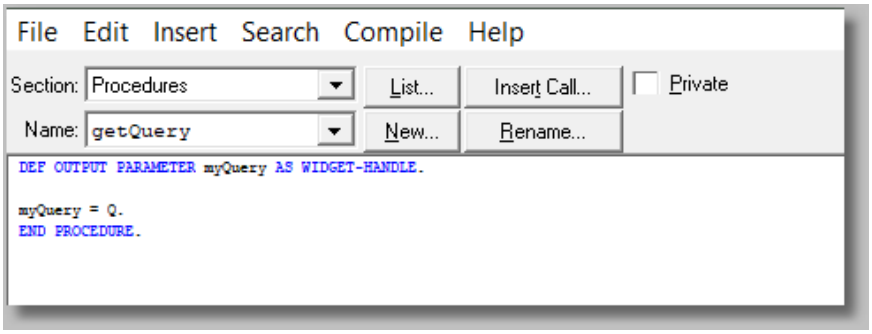
Version 4.2 also enables the generation of XML data files from dynamic queries that include external data.

These queries are created by the calling program and can integrate temporary tables and/or database tables.

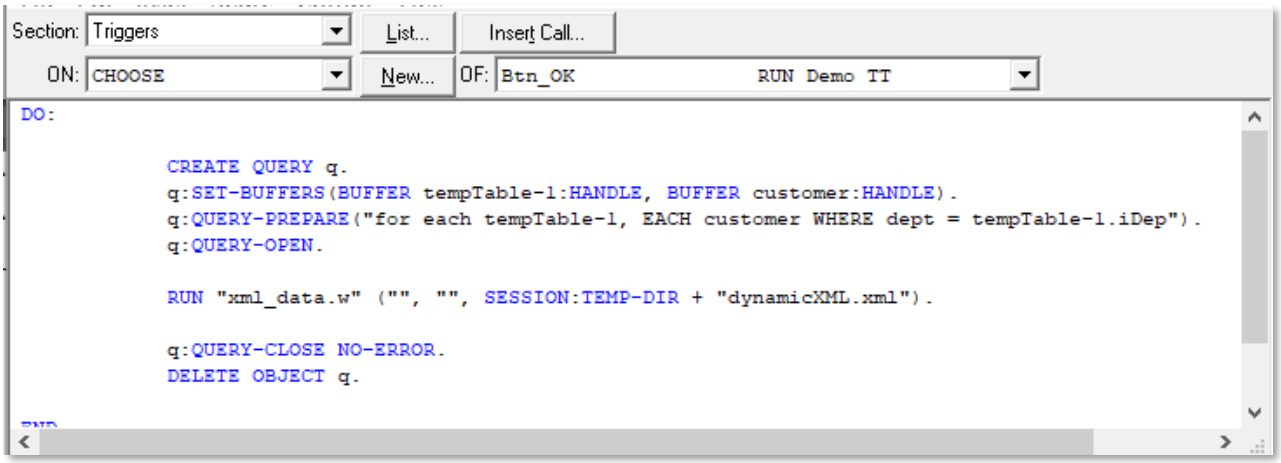
- Include **vpDesigner.i** and define a query **handle**, "Q" in this example :



- Create a **getQuery** procedure that XML_Data will call to connect to the query:



- **The query:**



This dynamic query involves a join between a temporary table and the "customer" table in the database.

3.1.3 SQL-Server data file format:

The Designer also accepts SQL-Server XML format:

```
<?xml version="1.0"?>
<data>
<customer>
    <Cust_Num>1</Cust_Num>
    <Country>USA</Country>
    <Name>Lift Line Skiing</Name>
    <Address>276 North Street</Address>
    <Address2></Address2>
    <City>Boston</City>
    <State>MA</State>
    <Postal_Code>02114</Postal_Code>
    <Contact>Gloria Shepley</Contact>
    <Phone>(617) 450-0087</Phone>
    <Sales_Rep>HXM</Sales_Rep>
    <Credit_Limit>66700</Credit_Limit>
    <Balance>42568</Balance>
    <Terms>Net30</Terms>
    <Discount>35</Discount>
    <Comments>This customer is on credit hold.</Comments
    <order>
        <Order_num>6</Order_num>
        <Cust_Num>1</Cust_Num>
        <Order_Date>31/12/15</Order_Date>
        <Ship_Date>03/02/16</Ship_Date>
        <Promise_Date>29/01/16</Promise_Date>
        <Carrier>Standard Mail</Carrier>
        <Instructions></Instructions>
        <PO></PO>
        <Terms>Net30</Terms>
        <Sales_Rep>HXM</Sales_Rep>
        <order_line>
            <Order_num>6</Order_num>
            <Line_num>1</Line_num>
            <Item_num>9</Item_num>
            <Price>18</Price>
            <Qty>67</Qty>
            <Discount>35</Discount>
            <Extended_Price>783.9</Extended_Price>
            <Backorder>no</Backorder>
            <item>
        </order_line>
        <order_line>
            <Order_num>6</Order_num>
```

Example of query:

```
SELECT customer.cust_num,
       … …     (customer fields)
       order.order_num,
       order.order_date,
       … …     (order fields)
FROM clients AS customer
   INNER JOIN order
   ON order.Cust_Num =
customer.Cust_num
WHERE customer.cust_num = 1
FOR XML AUTO, ROOT ('data'),
ELEMENTS;

<data>
  <customer>
    <cust_num>1</cust_num>
    <order>
      <order_num>10</order_num>
      <order_date>31/12/2015</order_
date>
    </order>
    <order>
      <order_num>12</order_num>
      <order_date>31/12/2018</order_
date>
    </order>
  </customer >
</data>
```

Figure 5 **- SQL-Server XML data file**

3.1.4 Database interfaces (mySQL, SQLServer,Postgres…)

Version 3.1 of the Designer allows you to create .vpschema data schema files and .xml data files presenting the hierarchy of an SQL query for mySQL, SQL-Server, Advantage Database Server (ADS), Firebird databases , Interbase, IB Lite, MS-Access, PostgreSQL, SQLite and Microsoft ADO drivers like dBase, Jet & Microsoft OLE DB.

A connection string must be supplied, native or ADO.

The different connection strings are well documented on the internet, see for example: **https://www.connectionstrings.com/**

The interface is provided by **vpSQLXML.dll** with the following entry points:

1. **connectDB** : (Connects to **mySQL, ADS, FB, IB, MSAccess, Postgres, SQLite, IBLite**)

   returnCode **= connectDB(**connection_string, password**);**

   Params:

   - Connection_string: string,
   - Password: string,
   - returnCode: integer 1 = ok, 0 = error

2. **ADOConnectDB** : (Connects to **SQLServer, dBase, connections Microsoft ADO Jet, Ole DB…**)

   returnCode **= ADOconnectDB(**connection_string, DB_name**);**

   Params:

   - Connection string: string,
   - DB_name: string,
   - returnCode: integer 1 = ok, 0 = error

3. **exportSQL** : (Creates the hierarchical XML data file)

   returnCode **= exportSQL(**SQL_Select_phrase, XML_File**);**

   Params:

   - SQL_Select_phrase, string
   - XML_File: string, output XML file,
   - returnCode: integer 1 = ok, 0 = error

4. **exportSchema** : (Exports the DB schema in Designer format .vpSchema)

   returnCode **= exportSchema(**vpSchema_File**);**

   Params:

   - vpSchema_File, string, .vpSchema file to create
   - returnCode: integer 1 = ok, 0 = error

5. **disconnectDB** : (DB disconnect)

disconnectDB;

Params:

- None

3.1.4.1 Some examples of connection strings:

- **mySQL**
  `DriverID=MySQL;Server=myServer;Database=base;User_name=UserName`

- **SQL-SERVER** *(DataTypeCompatibility=80 is **mandatory**)*
  `Provider=SQLNCLI11.1;DataTypeCompatibility=80;Persist Security Info=False;User ID=UserName;Initial Catalog=base;Data Source= myServer;Initial File Name="";Server SPN=""`

- **dBase**
  `Provider=MSDASQL.1;Persist Security Info=False;Data Source=dBASE Files;Initial Catalog=myDBase_Folder`

- **MSAccess**
  `DriverID=MSAcc;Provider=MSAcc;Persist Security Info=False;Data Source=Access Files;Initial Catalog=myFolder;Database=Base_Access.accdb`

- **Microsoft Jet OLE DB**
  `Provider=Microsoft.Jet.OLEDB.4.0;Data Source=databaseName;User ID=UserName;Password=myPassWord;`
  Microsoft OLE DB Provider for Microsoft Jet | Microsoft Docs
- …

SQL "select" phrases:

The SQL "select" phrase is parsed by vpSQLXML to extract the names of the tables and columns referenced in the query.

As the goal is to constitute an XML file with the hierarchy of the request (e.g. Customer -> Order-> Lines -> Item) and table-column correspondence, some constraints apply:

1. All tables must be specified between "FROM" and "WHERE".
2. The order of the tables must be **identical** to the hierarchy of the query,
3. "AS" column aliases are not supported,
4. If calculated expressions or aliases are specified, these values will be assigned as elements of the last table of the query,
5. A level of "JOIN" is allowed however for simple queries.

3.1.4.2 Usage:

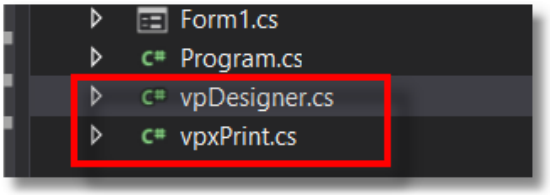
3.1.4.2.1 PROGRESS :

```
/* Include to insert in the definitions
   ===================================*/

{ vprun.i }
```

```
DEF VAR RT AS INT NO-UNDO.

RUN connectDB("DriverID=MySQL;Server=127.0.0.1;Database=sakila;User_name=XX",
              "XX",
              OUTPUT RT).

IF RT = 0 THEN DO:
        MESSAGE "Database is NOT connected."
                    VIEW-AS ALERT-BOX TITLE "mySQL".
        RETURN.
        END

RUN exportSQL("select * from film, film_actor, actor"
                        + " where film_actor.film_id    = film.film_id"
                        + " and   actor.actor_id        = film_actor.actor_id"
                        + " and   film.film_id <= 4"
                        + " order by film.film_id",
                "C:/temp/MySQL.xml",
                OUTPUT RT).

IF RT = 0 THEN
      MESSAGE "Return code" RT
                    VIEW-AS ALERT-BOX TITLE "MySQL error".

RUN exportSchema("C:/temp/sakila_MySQL.vpSchema", OUTPUT RT).

RUN disconnectDB.
```

3.1.4.2.2 VISUAL Studio C# :

Entry points definition:



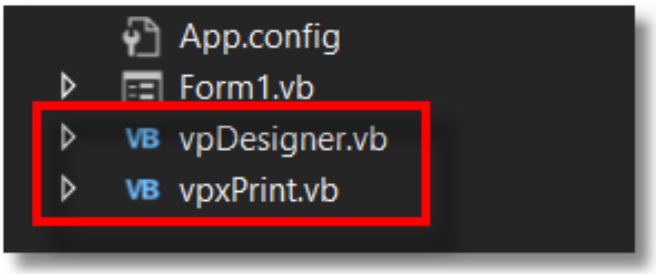

```
[DllImport("vpSqlXML.dll")]
public static extern void connectDB(string connect_string, string password,
                                    ref int returnCode);

[DllImport("vpSqlXML.dll")]
public static extern void ADOConnectDB(string connect_string, string database_name,
                                    ref int returnCode);

[DllImport("vpSqlXML.dll")]
public static extern void exportSQL(string SQL_select, string XML_OutputFile,
                                    ref int returnCode);

[DllImport("vpSqlXML.dll")]
public static extern void exportSchema(string Schema_File, ref int returnCode);

[DllImport("vpSqlXML.dll")]
public static extern void disconnectDB();
```

Functions call:

```
private void bconnect_Click(object sender, EventArgs e)

       {
       int retCode = 0;

       vpDesigner.connectDB("DriverID=MySQL;Server=127.0.0.1;Database=sakila;User_name=XXX"
       ,
                 "XXXX",                          // password
                 ref retCode);
       }

private void bSQL_Click(object sender, EventArgs e)
        {
            int retCode = 0;
            vpDesigner.exportSQL("select * from film, film_actor, actor"
                        + " where film_actor.film_id   = film.film_id"
                        + " and   actor.actor_id       = film_actor.actor_id"
                        + " and   film.film_id <= 4"
                        + " order by film.film_id",
                       "c:/temp/MySQL_CSharp.xml",
                      ref retCode);
        }

private void bSchema_Click(object sender, EventArgs e)
        {
            int retCode = 0;
            vpDesigner.exportSchema("c:/temp/MySQL_CSharp.vpSchema", ref retCode);
         }

private void bdisconnect_Click(object sender, EventArgs e)
        {
            int retCode = 0;
            vpDesigner.disconnectDB();
         }
```

3.1.4.2.3 VISUAL STUDIO VB :

Entry points definition:




```
Declare Sub connectDB Lib "vpSQLXml.dll" (ByVal connectString As String,
                                          Password As String, ByRef retCode As Integer)

Declare Sub ADOconnectDB Lib "vpSQLXml.dll" (ByVal connectString As String,
                                          DBName As String, ByRef retCode As Integer)

Declare Sub exportSQL Lib "vpSQLXml.dll" (ByVal selectphrase As String,
                                          ByVal filename As String, ByRef retCode As Integer)

Declare Sub exportSchema Lib "vpSQLXml.dll" (ByVal filename As String,
                                          ByRef retCode As Integer)

Declare Sub disconnectDB Lib "vpSQLXml.dll" ()
```

Functions call:

```
Dim returnCode As Integer

vpDesigner.connectDB("DriverID=MySQL;Server=127.0.0.1;Database=sakila;User_name=XXX
",
          "XXXX", returnCode)

If returnCode = 0 Then
     MsgBox("Database sakila is NOT connected", vbInformation, Me.Text)
     End If

vpDesigner.exportSQL("select * from film, film_actor, actor” _
                        & " where film_actor.film_id = film.film_id" _
                        & " and actor.actor_id       = film_actor.actor_id" _
                        & " and film.film_id <= 4" _
                        & " order by film.film_id",
                "C:/temp/MySQL_VB.xml", returnCode)

If returnCode = 0 Then
        MsgBox("Error in SQL select", vbInformation, Me.Text)
        End If


vpDesigner.exportSchema("C:/temp/MySQL_VB.vpSchema", returnCode)
If returnCode = 0 Then
     MsgBox("Error creating the schema", vbInformation, Me.Text)
     End If

vpDesigner.disconnectDB
```

3.1.4.2.4 EMBARCADERO Delphi

Entry points definition:

```
{$I vpRun.inc }


DLLs folder of vpSQLXML.dll & vpRun.dll can be specified:

const designerSQL = 'C:\dev\xPrint\' + 'vpSQLXml.dll';

      designerRUN = 'C:\dev\xPrint\' + 'vpRun.dll';
```

```
var returnCode: integer;

returnCode := connectDB('DriverID=MySQL;Server=127.0.0.1;Database=sakila;'
                        + 'User_name=XXX', 'xxx');

if returnCode = 1 then
               showMessage('mySQL database "sakila" is connected')
else begin
               showMessage('Error connecting to mySQL.');
               exit;
               end;

returnCode := exportSQL('select * from film, film_actor, actor'
                            + ' where film_actor.film_id = film.film_id'
                            + ' and actor.actor_id       = film_actor.actor_id'
                            + ' and film.film_id <= 4'
                            + ' order by film.film_id',
                      XML_File);

if returnCode = 1 then
                     showMessage('XML data file has been created')
else
                     showMessage('Error in SELECT');

disconnectDB;
```

Schema export:

```
      returnCode := exportSchema('c:/temp/MySQL_Delphi.vpSchema');

      if returnCode = 1 then
                           showMessage('Schema file has been created')
      else
                           showMessage('Error in Schema');
```

3.2 RUN A PROJECT

A project is an XML file with the suffix ".vpProj". It is built by the graphic designer vpDesigner.exe.

It contains the definition of the data, the pages, the objects and all the management rules to use.

The execution of a project is done by vpRun.dll which generates a file ".xpr"..

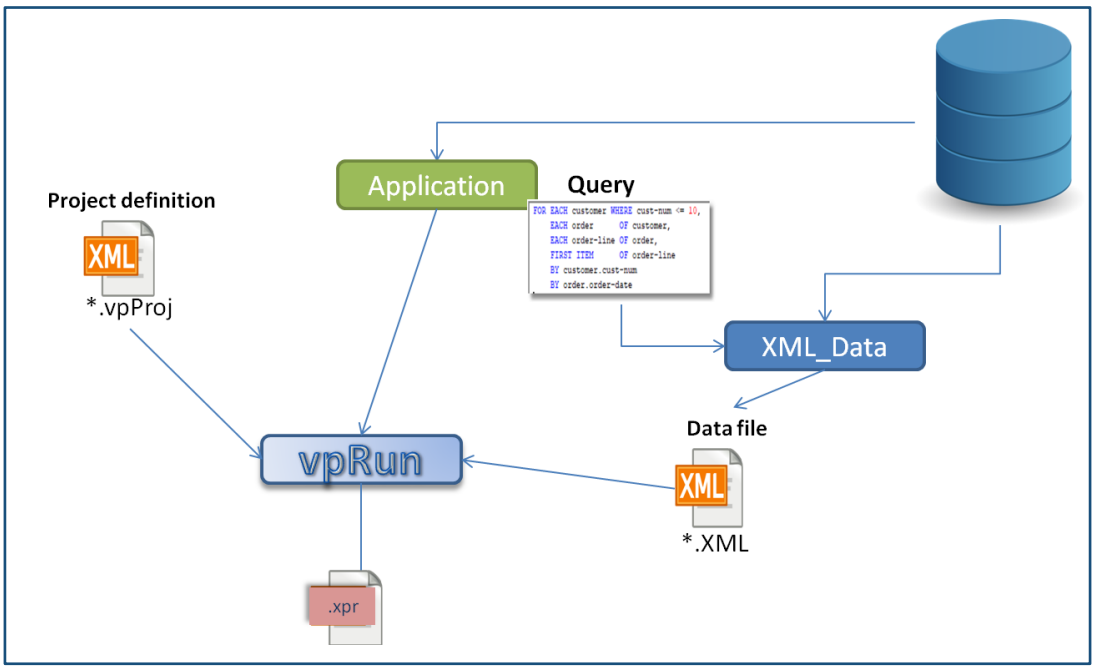


Figure 6 – General schema

3.2.1 vpRun.dll parameters:

Entry points:

- **vpRunReport** :

  projectFile :
  XmlDataFile :
  xprFileName :
  → pointers to AnsiChar strings

  OUTPUT integer → 32-bit integer, execution return code

  (**safecall** calling convention)

- **vpaddTag** :

  xprLine : pointer to command line (AnsiChar pointer)

  Adds one or more lines to the beginning of the .xpr file generated by vpRun.
  Multiple tags can be inserted at once and multiple vpaddTag can be run to create as many lines as needed. Can be used to create a tag with particular attributes, "preprocessed" tags like [@FILE=c:/temp/Data.txt] or data for graphs (see vpxPrint doc)

<**DATA**=SALES>

…

…

</DATA>

After vpRunReport, these lines are automatically removed.

If vpRunReport is not executed, you have to use vpfreeTag to free the resources.

- **vpfreeTag** :

  No parameters

  Releases the lines created by vpaddTag if vpRunReport is not called.

  Useless if a vpRunReport has been executed.

- **vpRunDebug**

  onOff : integer 0 : Debug Off 1 : Debug on

  :

3.2.1.1 Calling from PROGRESS

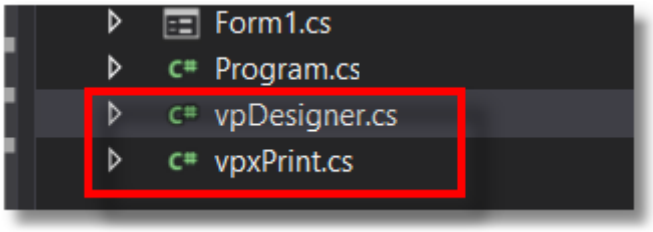
- Include {vpRun.i} that describes vpRun.dll entry points in Progress format,
- Generate the ".xpr" file with a call to vpRunReport,,

```
DEF VAR returnCode AS INT    NO-UNDO.
DEF VAR xprFile    AS CHAR   NO-UNDO.

 xprfile = localDir + 'singleInvoice.xpr'.
 RUN vprunReport(localDir + "singleInvoice.vpProj",     /*      The project definition */
                 myDataFile,                            /*      The data file          */
                 xprFile,                               /*      The output xpr file    */
                 OUTPUT returnCode).                    /*      Return status code     */

 IF returnCode > 0 THEN
                 MESSAGE 'Return code' returnCode VIEW-AS ALERT-BOX INFO
                                 TITLE "vpRun error".
 ELSE DO :
                 FILE-INFO:FILE-NAME = xprFile.
                 xprFile             = FILE-INFO:FULL-PATHNAME.
                 RUN printfile(xprFile).
                 /* OS-DELETE VALUE(xprFile). */
                 /* OS-DELETE VALUE(myDataFile). */
                 END.
 END.
```

Figure 7 - RUN vpReport

3.2.2 Calling from C#

Include **vpDesigner.cs** and **vpxPrint.cs** classes in the project:

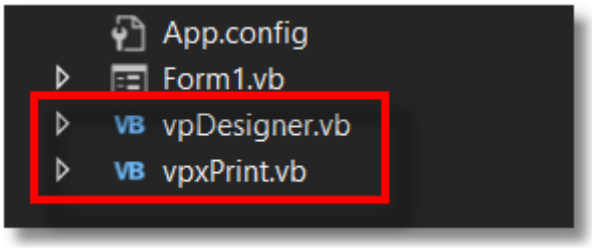



```
vpxPrint.cs | vpDesigner.cs | Designer:1 | Designer:2 | Program.cs | Form1.cs | Form1.vb
Designer | Designer.Form1 | button3_Click(object sender, EventArgs e

36
        1 référence
37      private void button3_Click(object sender, EventArgs e)
38      {
39          int retCode = 0;
40          vpDesigner.exportSQL("select * from film, film_actor, actor"
41                      + " where film_actor.film_id = film.film_id"
42                      + " and actor.actor_id        = film_actor.actor_id"
43                      + " and film.film_id <= 4"
44                      + " order by film.film_id",
45              "e:/temp/MySQL_CSharp.xml",
46              ref retCode);
47      }
48
        1 référence
49      private void button4_Click(object sender, EventArgs e)
50      {
51          int retCode = 0;
52          vpDesigner.exportSchema("e:/temp/MySQL_CSharp.vpSchema", ref retCode);
53      }
54
        1 référence
55      private void bxPrint_Click(object sender, EventArgs e)
56      {
57          int retCode = 0;
58          vpDesigner.vpRunReport("e:/temp/sakila.vpProj",
59                      "e:/temp/MySQL_CSharp.xml",
60                      "e:/temp/MySQL_CSharp.xpr",
61                      ref retCode);
62
63          vpxPrint.printFileStat("e:/temp/MySQL_CSharp.xpr", ref retCode);
64      }
65  }
66  }
67
```

Figure 8 - C# RUN vpReport

3.2.3 Calling from VB

Include **vpDesigner.vb** and **vpxPrint.vb** classes in the project:



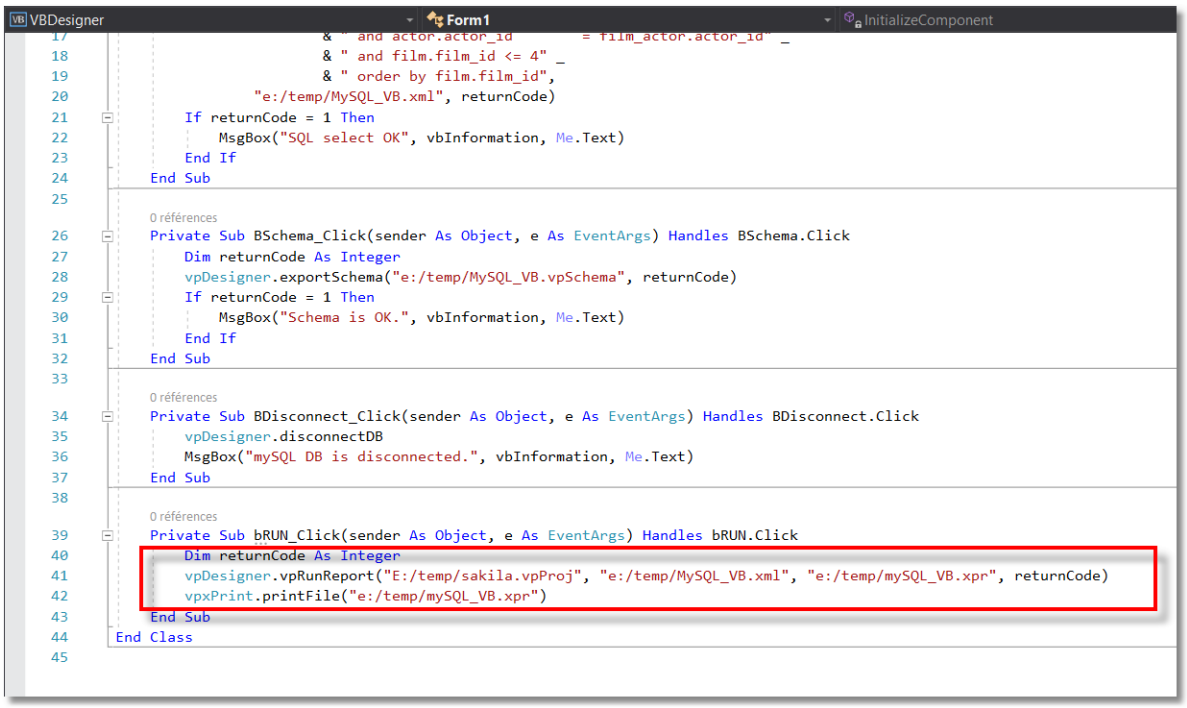


Figure 9 - VB RUN vpReport

3.2.4 Calling from Delphi

Include **vpRun.inc** and **vpxPrint.inc** :

```
var
  Form1: TForm1;
  returnCode:   integer;

implementation

////////////////////////////////////////////////
                {$I vpRun.inc }
////////////////////////////////////////////////
                {$I vpxPrint.inc }
////////////////////////////////////////////////

{$R *.dfm}
```

```
end;

procedure TForm1.bSchemaClick(Sender: TObject);
begin
      returnCode := exportSchema('c:/temp/MySQL_Delphi.vpSchema');

      if returnCode = 1 then
                showMessage('Schema file has been created')
      else
                showMessage('Error in Schema');
end;

procedure TForm1.bSelectClick(Sender: TObject);
begin
      returnCode := exportSQL('select * from film, film_actor, actor'
                        + ' where film_actor.film_id = film.film_id'
                        + ' and actor.actor_id       = film_actor.actor_id'
                        + ' and film.film_id <= 4'
                        + ' order by film.film_id',
                XMLData.Text);

      if returnCode = 1 then
                showMessage('XML data file has been created')
      else
                showMessage('Error in SELECT');
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
        vpRunReport(vpProj.text, XMLData.Text, 'c:/temp/sakila.xpr');
        printFile('c:/temp/sakila.xpr');
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
        vpProj.text  := getCurrentDir + '\sakila.vpProj';
        XMLData.Text := getCurrentDir + '\sakila.xml';
end;
```

Figure 10 – DELPHI RUN vpReport

3.3 USE OF THE PROJECT

3.3.1 Project processing

When the ".xpr" file is created by vpRun, vPxPrint's printFile () or printFileStat () call processes the resulting file: edit, preview, mail, pdf creation, and different combinations.

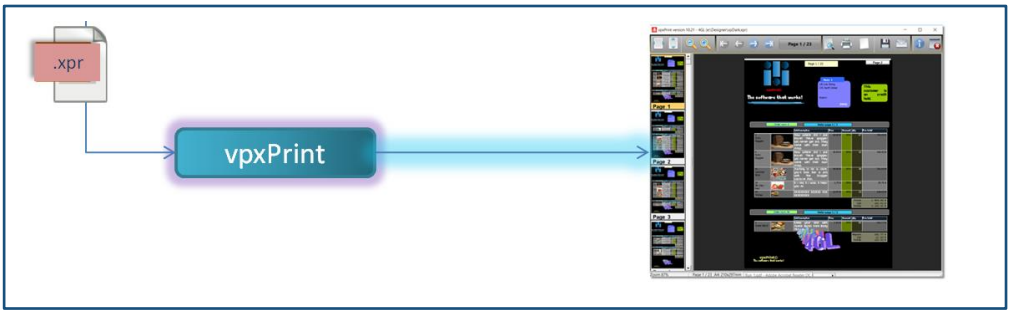


Figure 11 - Calling vpxPrint

This same ".xpr" file can also be used by vpxPrint Active-X, it's as simple as that!

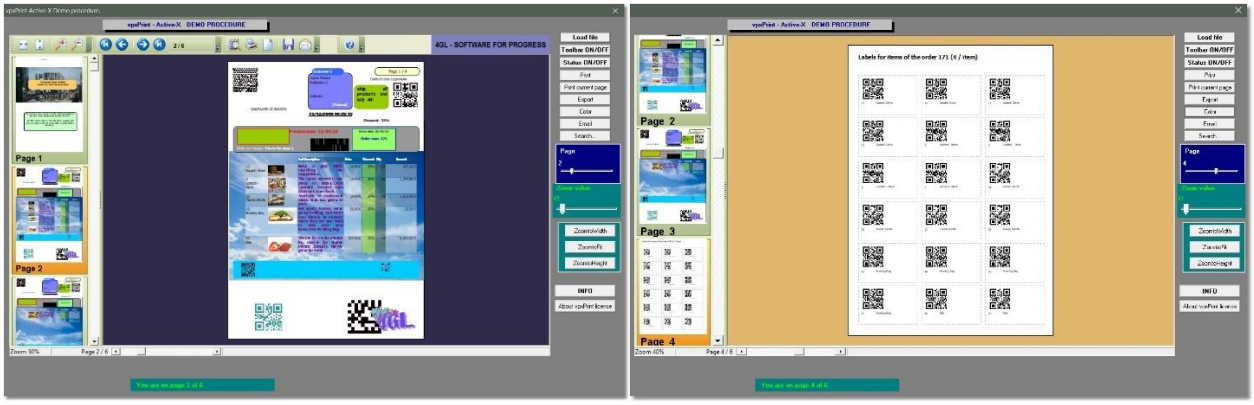


Figure 12 - Active-X

ACCESS TO THE ENTIRE VPXPRINT FUNCTIONS:
- **PREVIEW, PRINT, CREATE MULTIPLE PDF, FAX FILES, IMAGES, MAIL, ETC ...**
- **ACTIVE-X**

4 THE DESIGNER

4.1 GRAPHICAL INTERFACE

4.1.1 Main screen

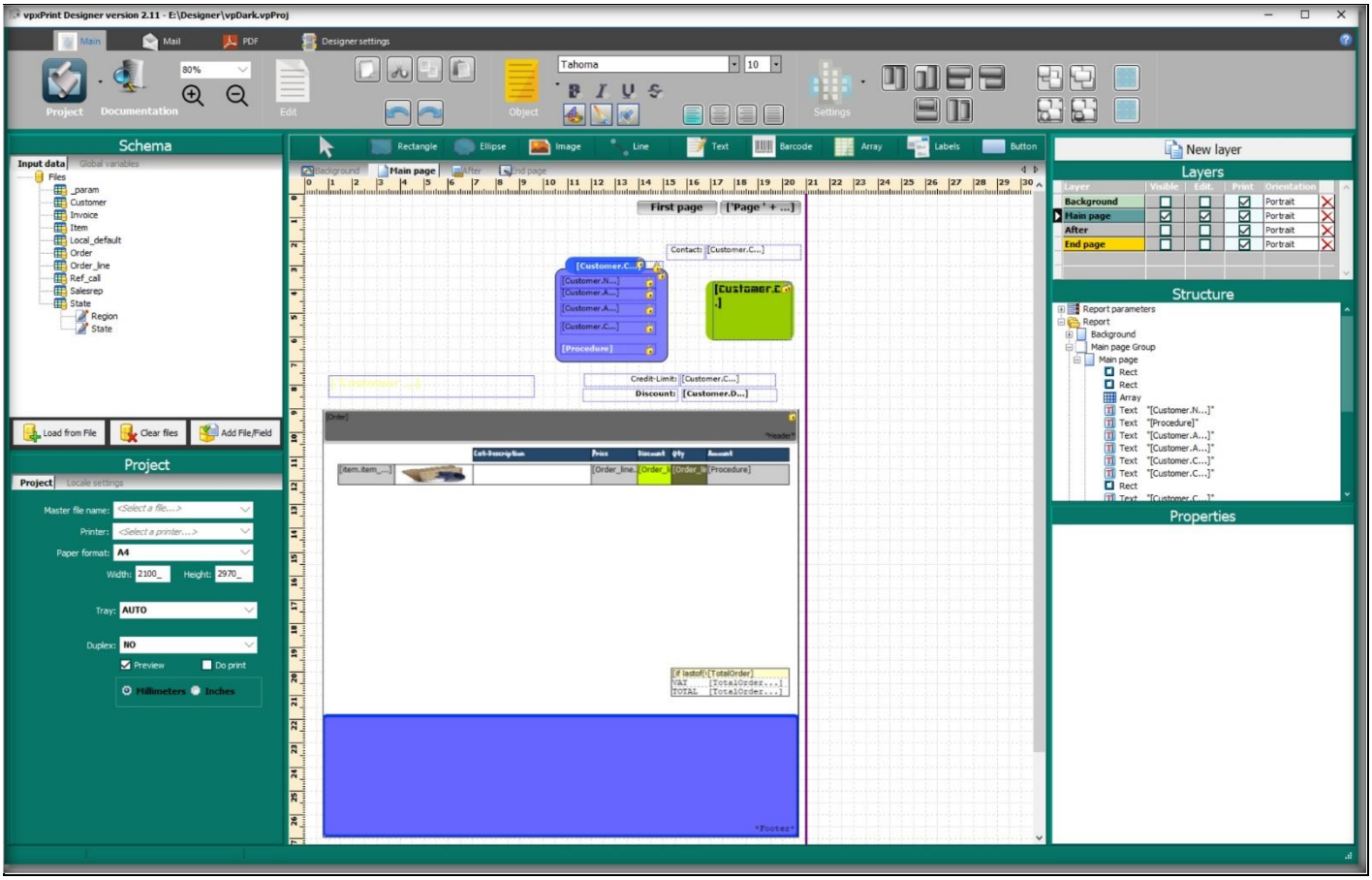


Figure 13 - vpDesigner

4.1.2 Mail Interface

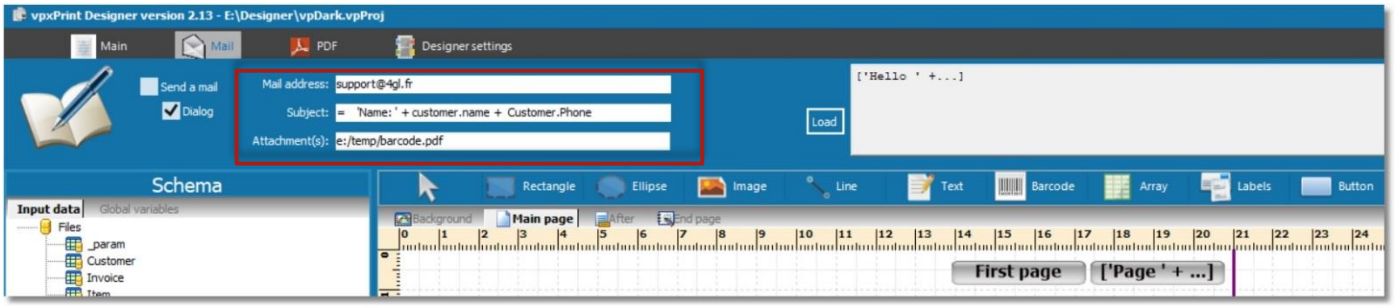


Figure 14 – Mail Interface

In this page, we specify the parameters of sending mail. If an area starts with =, it is evaluated as an expression. The drag-and-drop mode is active on these values.

The "Load" button loads a file from your computer, a click on the "body" part makes it possible to modify the body of the mail message **like any other object of the Designer**.

If a **<html>** tag is present in the text, the mail will be sent in html mode, otherwise it will be sent in plain text mode.

4.1.3 PDF Interface

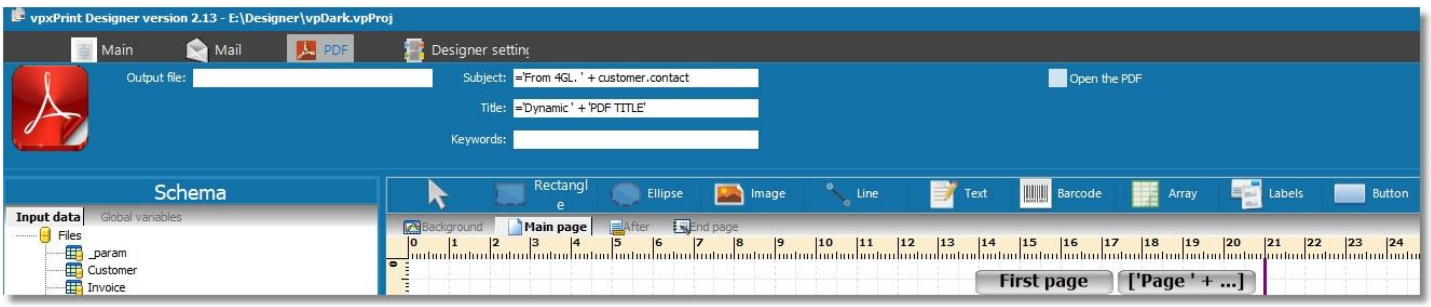


Figure 15 – PDF Attributes

Same mode of use as for the Mail tab.

The drag-and-drop mode is active on these values.

4.1.4 Language and theme of the interface

The "Designer settings" tab allows you to choose the language of the interface and its theme:

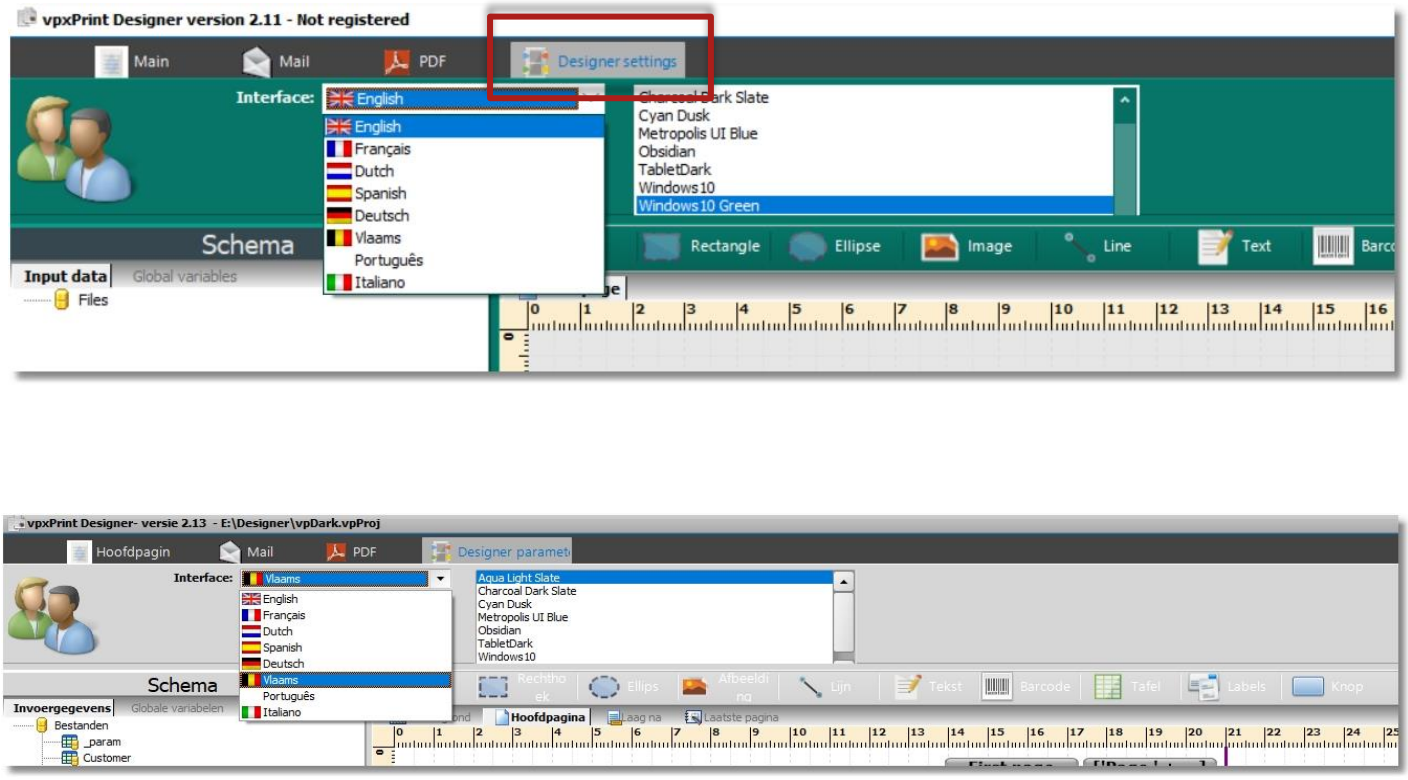


Figure 16 - Language and theme of the interface

4.1.5 Objects

The "objects" toolbar presents the different types of objects:

Figure 17 – Objects toolbar

Instructions for use: click on an object then draw the outline on the screen area. The arrow is used to cancel the current operation.

- **rectangle, ellipse and ligne**
- **Images, texts, barcodes and buttons are objects of the same nature that can be transformed from one to another.**
- **array inserts a "band-line" array on a page,**
- **labels creates a label board,**
- **button is a hyper-text object (vpxPrint sensitive area)**

4.1.6 Data schema

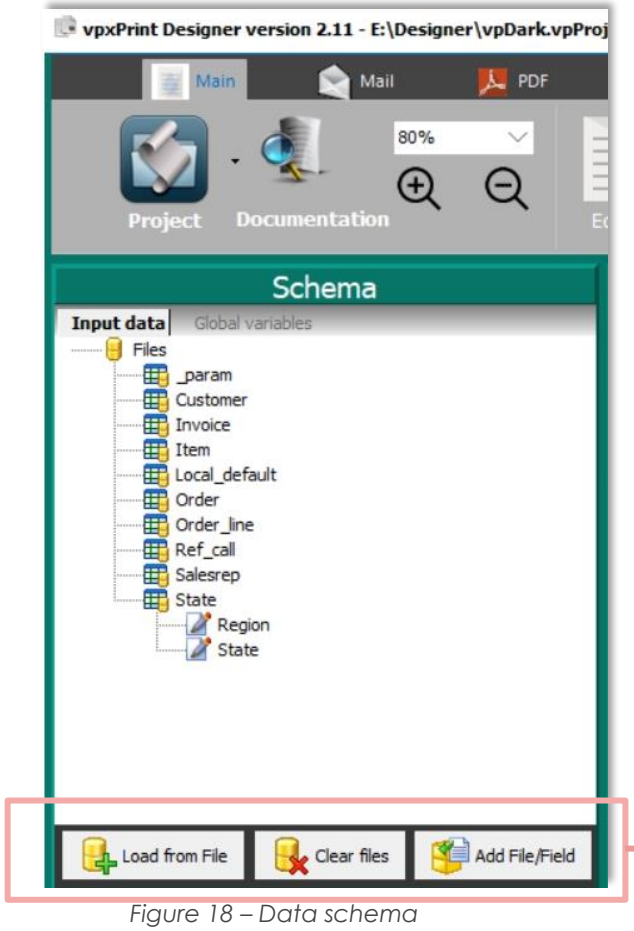


Figure 18 – Data schema

Presents the data structure of the database, from a ".vpSchema" file.

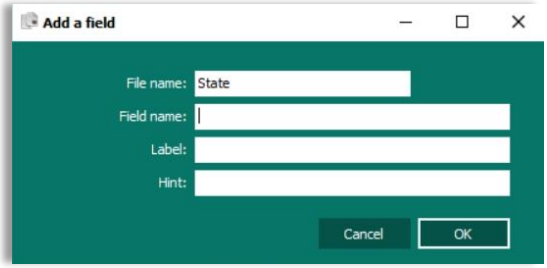
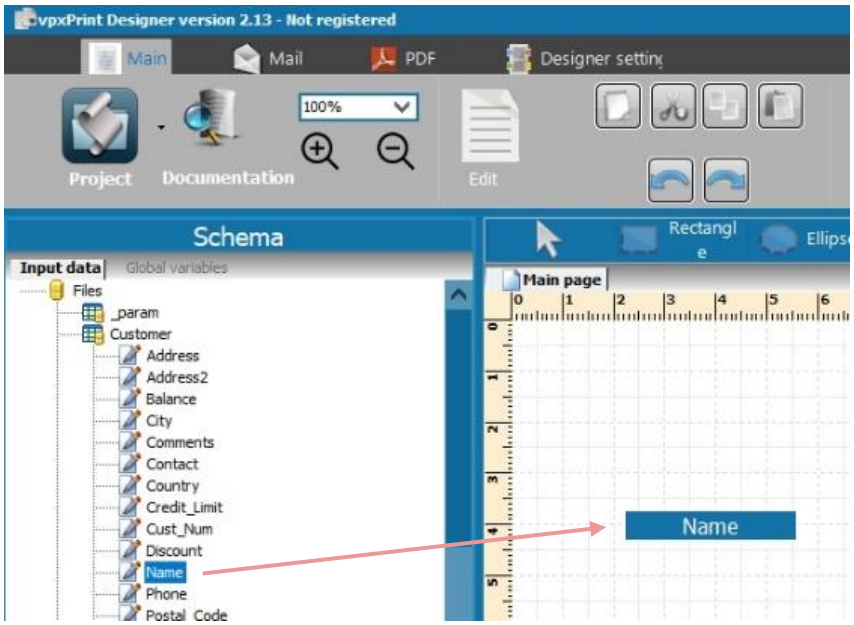


Figure 19 – Adding a file or a field

Drag and drop from the diagram allows:

1. to insert a field in the main screen.

Associated with the "SHIFT" key, the field is copied with its label



2. to insert a field in the mail, pdf or printer zones.

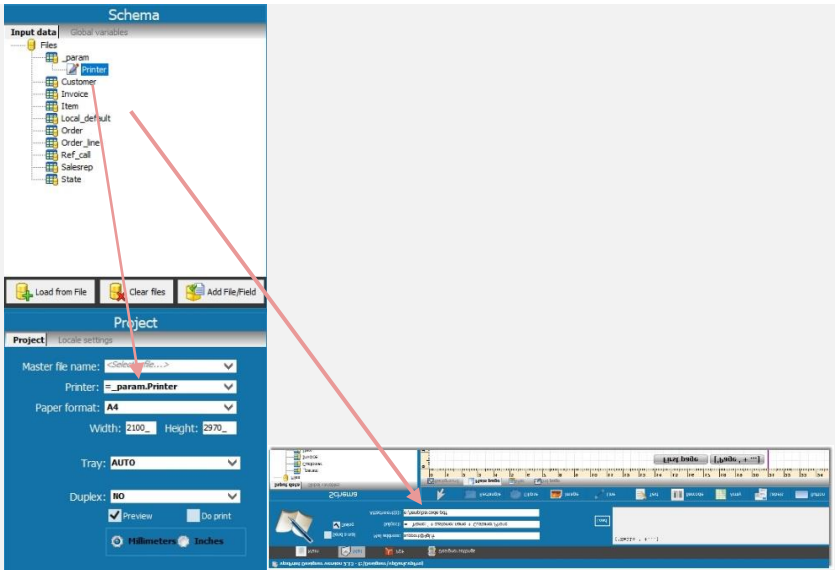
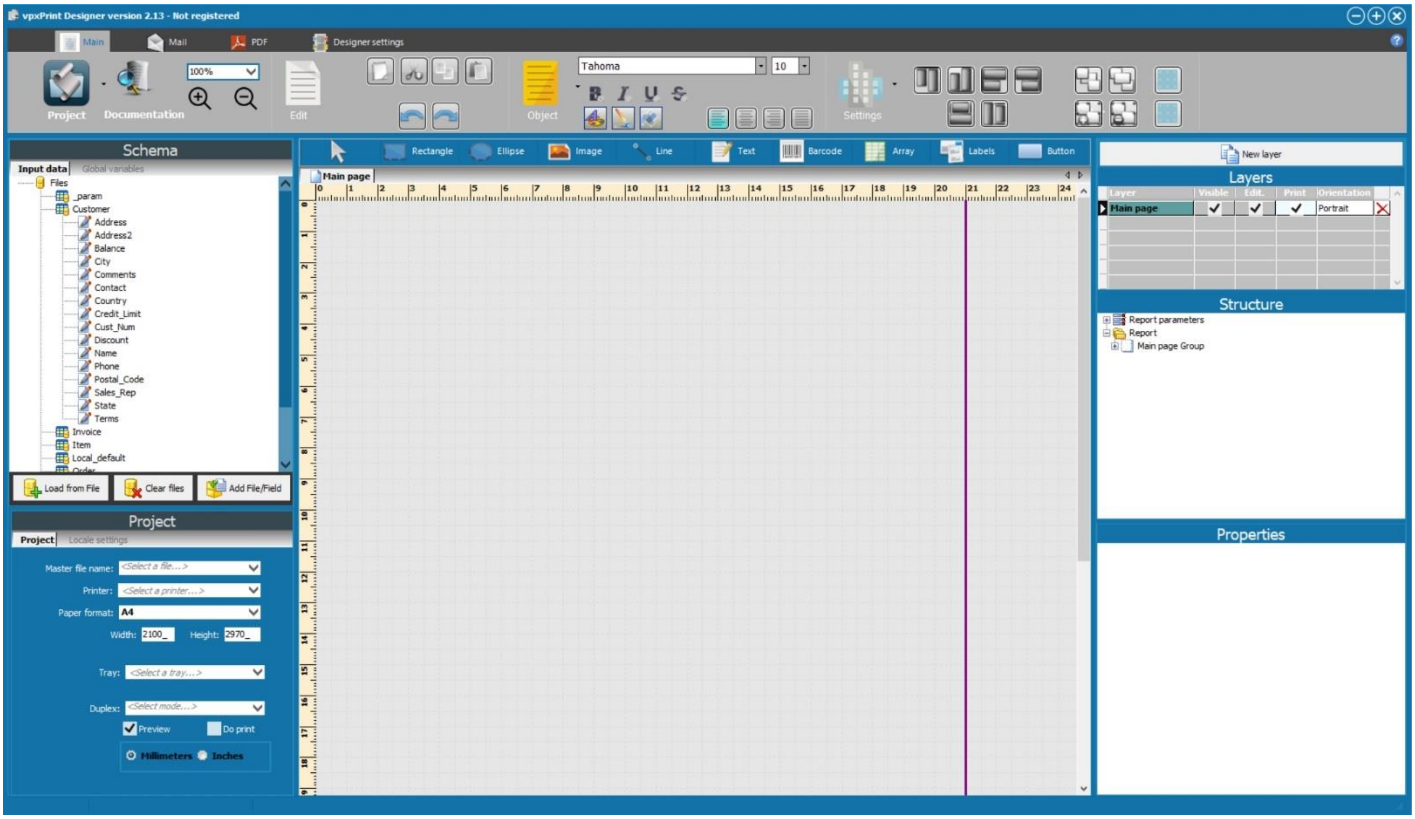


Figure 20 – Drag-n-Drop fields

4.1.7 The workspace

The workspace is organized into pages ("layers").

An empty project consists of only one main page, there can be up to 5 main pages (from "main_1" to "main_5")



4.1.8 The pages

4.1.8.1 Page types, the « layers »

The Designer is organized in multi-layers and has 6 types of different layers. Their use is optional, only a main page is required.

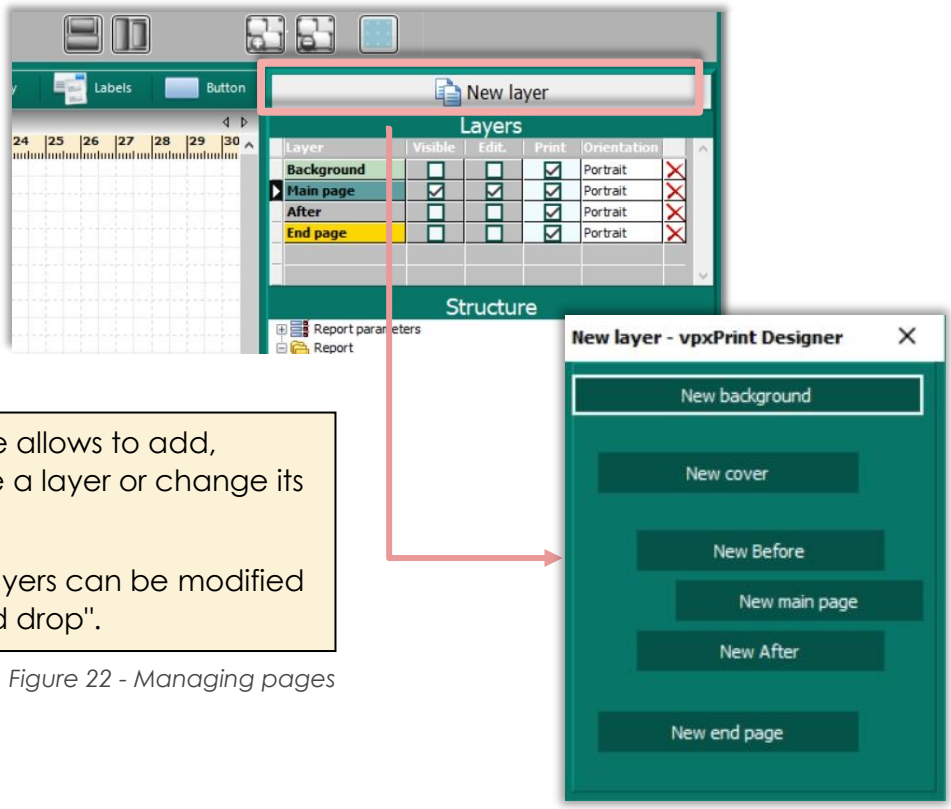
Type de page	Nom interne	Particularités
Background	Back_1 to Back_5	1. A background is calculated when it is defined, it's not updated if its contents change, 2. It can be visualized and not edited (use of pre-printed), to allow the user to better see the rendering of his impression, 3. It can be associated with the pages "cover", "main" and "summary" 4. Is printed only in association with a cover, a main page or a summary page. *(= <BACKGROUND>)*
Cover	Cover_1 to Cover_5	1. Conditionnai printing, 2. Processed at the beginning of the project

Before Page	~~Header_1 to Header_5~~ Before_1 to Before_5	1. Not printable directly, associated with a main page, 2. It contains the objects to print on the page background, after the background and before the main page, 3. Conditionally printable, 4. Can contain variable areas like a title, a variable background
Main Page	Main_1 to Main_5	1. Conditionally printable, 2. Can be associated with a group of pages, 3. Advance the query to the next master record ("master file") when a group ends
After Page	~~Footer_1 à Footer_5~~ After_1 to After_5	1. Not printable directly, associated with a main page, 2. It contains the objects to print in the foreground of the main page, 3. Conditionally printable 4. Can contain variable areas like a page number
Summary Page	Summary_1 to Summary_5	1. Conditionally printable, 2. Edited at the end of the project

Figure 21 - Types of pages

4.1.8.2 Managing pages

Adding a page is done by pressing the "new layer" button on the right panel.



The layer table allows to add, modify, delete a layer or change its orientation.

The order of layers can be modified with "drag and drop".

Figure 22 - Managing pages

The different layers are shown in the tab list on the main screen:

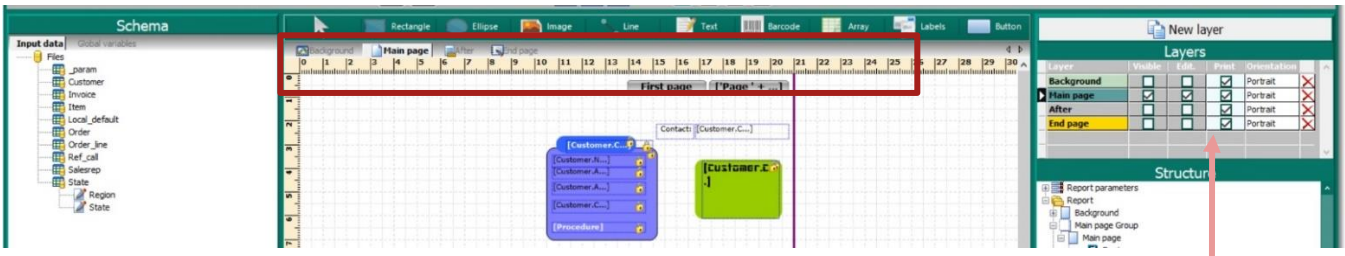
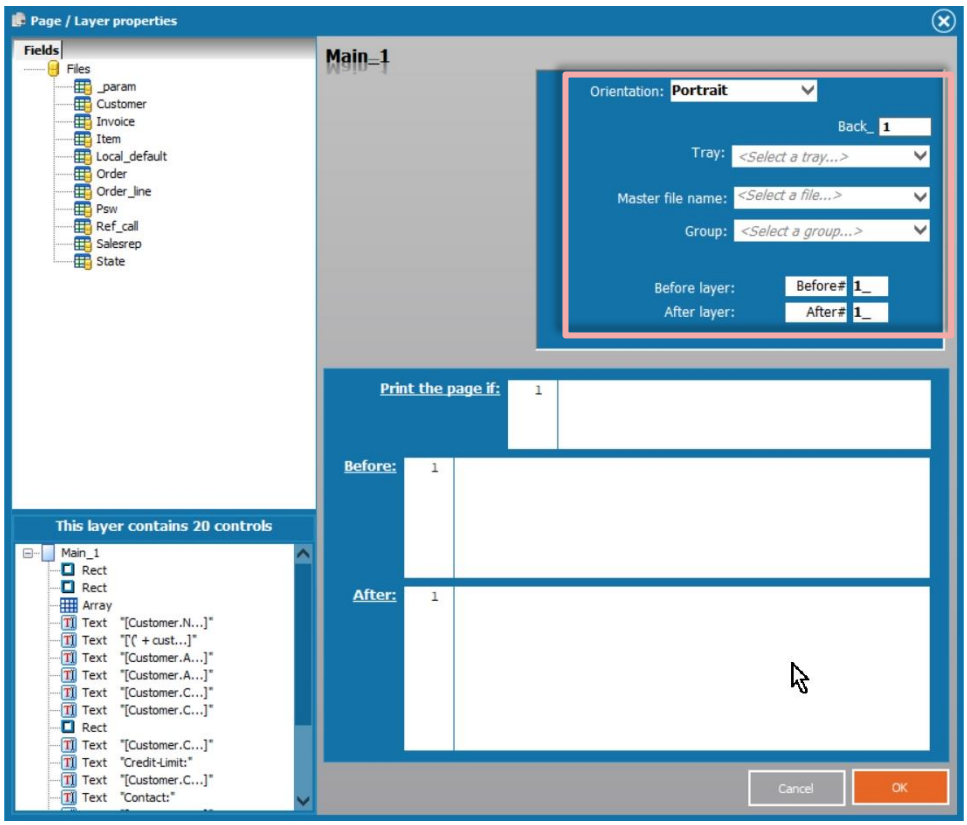


Figure 23 - Page tabs

By selecting the options in the right pane, you can display several pages simultaneously, which aligns objects between pages.

4.1.8.3 Page properties

Double-clicking on a page opens the properties window of the page:



Orientation,
Background,
Paper tray,

Group,
Page before,
Page after

Printing condition

"Before" instructions

"After" instructions»

Figure 24 – Page properties

A page of type "Background" can be declared "unprintable". It will then be seen in a preview but not printed. Useful in the case of using pre-printed paper:

Figure 25 – Background properties

Example:

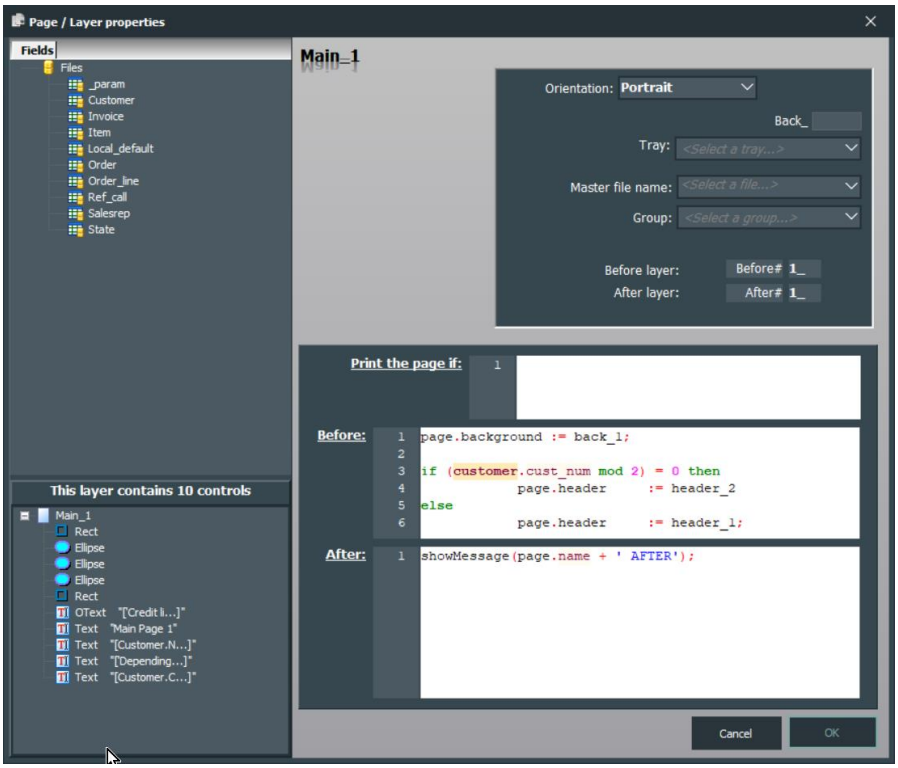


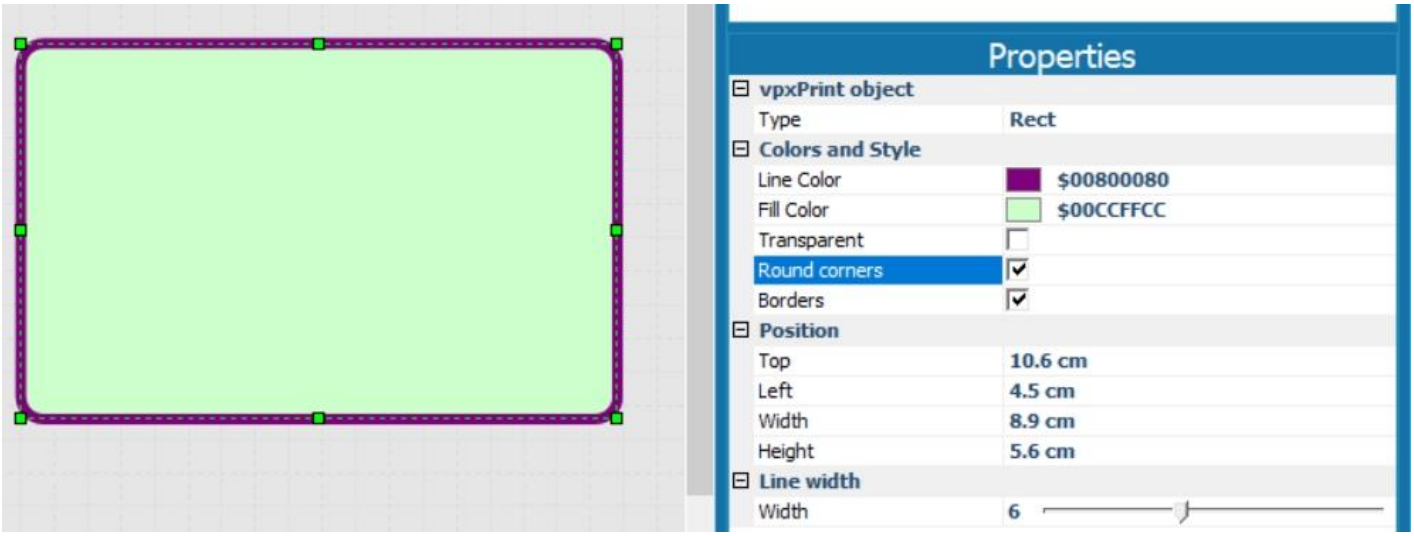
Figure 26 - Page properties - Example

In this example, the page selects its background and its before-after objects, depending on whether the customer number is even or odd. A message is sent to the screen after each page. Note that "header" and "footer" can be replaced by "before" and after ".

4.1.9 Basic objects

4.1.9.1 Graphical Objects

Rectangles, ellipses, and lines are basic graphical objects. You can only specify their shape, colors, thickness and filling. The rectangles may have rounded corners and / or be transparent.



The colors can be specified in the properties window:

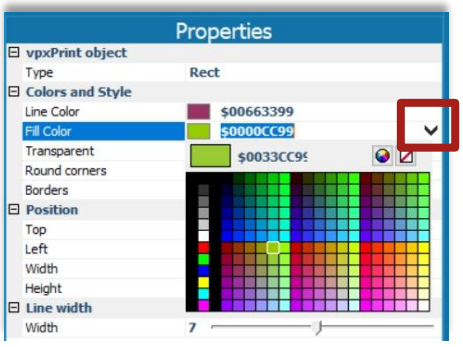
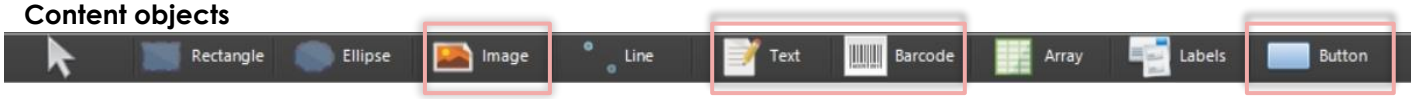


Figure 27 - Object Colors

or in the top panel of the tools:

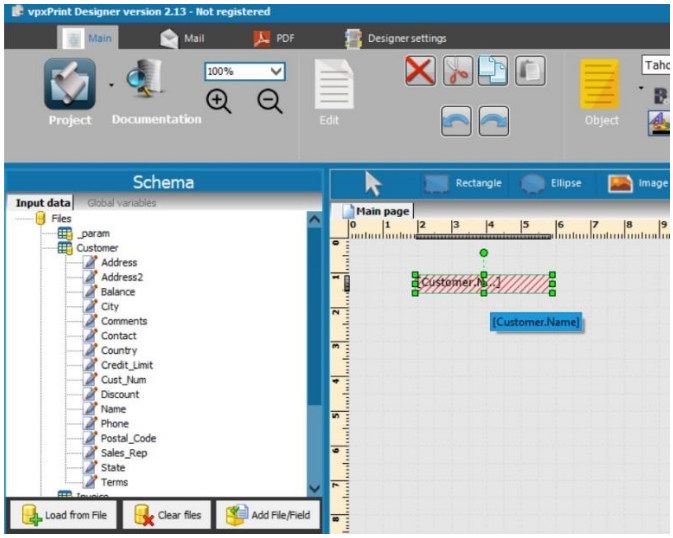
4.1.9.2 Content objects



These are objects whose content is dynamic, linked to field values or management rules: texts, barcodes, images or buttons.

An object of this type is created by drag and drop from the diagram or created by selecting a type from the toolbar to draw it on the surface of the page.

Any object of this type can evolve to any object of this category: a text can be transformed into a barcode or an image, all combinations are allowed. This is the "view-as" property of the object that defines its physical representation mode.



When the object comes from the schema, a value is associated with it automatically, that of the field of the file, as an expression.

Double-clicking on an object opens the window of its properties:

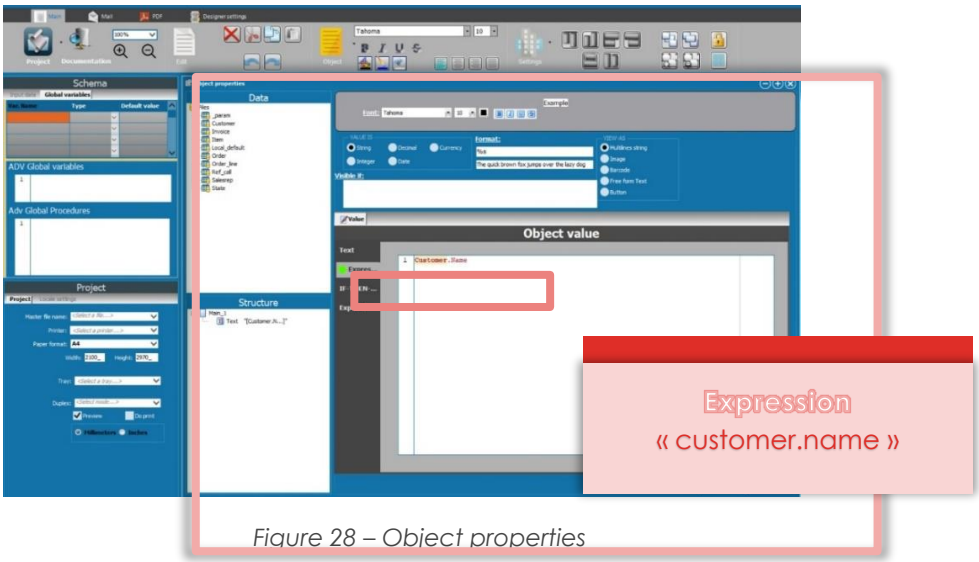
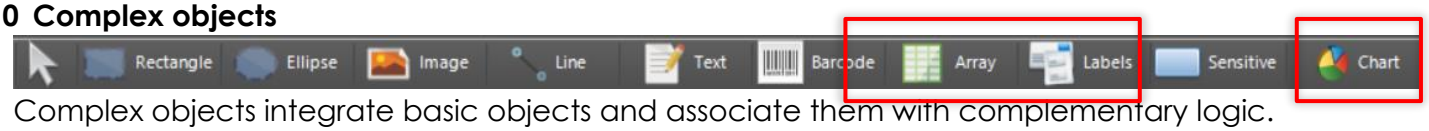


Figure 28 – Object properties

It is in this window that the content of the object, its format and presentation are defined. It can also be associated with actions.

4.1.10 Complex objects



Complex objects integrate basic objects and associate them with complementary logic. These are tables ("arrays"), labels and charts.

4.1.10.1 The arrays

1. These are band-type objects. They are organized in relation master file - detail file, the master file is optional.
2. The table objects are multi-page, they propagate from page to page until the master-detail query is fully processed.
3. 1. If no master file is defined, the last detail record will indicate the end,
4. 2. If a master file is defined, the end will be the last detail record of the last master record of the table object.

In our example, the main page uses the customer file as the master file. The table object uses the file "order" and "order_line" in detail.

Each table object will process all "order" and "order_line OF order" records, before moving on to the next main page for a new "customer". It will generate for this as many pages as necessary.

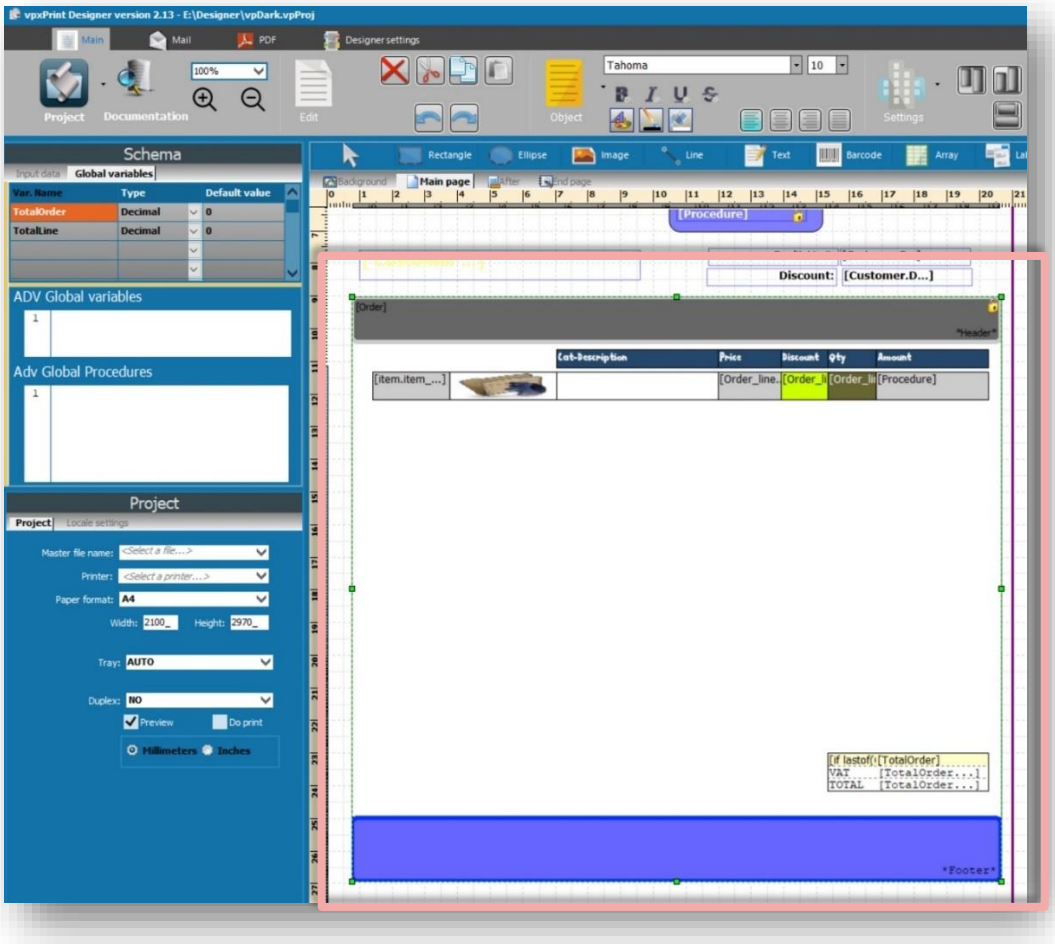


Figure 29 – Array Object

4.1.10.2 Array poperties

Double-clicking on a "table" object opens the properties window.

A table consists of 3 areas:

1. **A header area,** optional. It can contain basic objects like the main page. This field is associated with the master file (**firstof** or first line)
2. **A "detail" zone,** mandatory. With:
   a. A title, optional, consisting of one or more lines.
   b. Iterative "detail" lines that run through the corresponding detail file,
   c. Summary or total lines, optional, consisting of one or more lines..
3. **A "footer" area**, optional. It can contain basic objects like the main page. This field is associated with the master file (**lastof** or bottom of table)

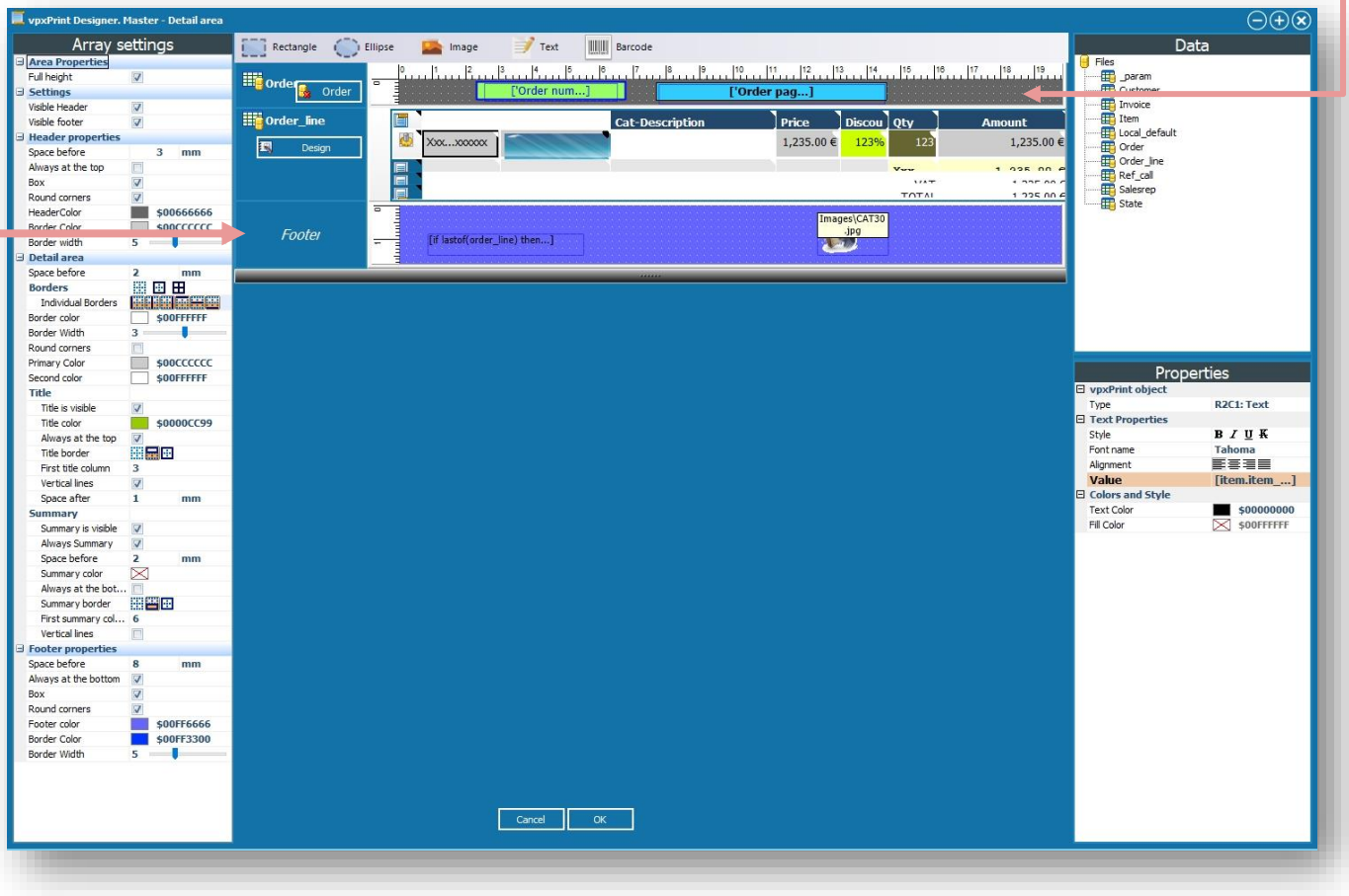


Figure 30 – Array Object – Properties

A cell can contain simple text (characters, numeric, currency, date), multi-line text, an image, a barcode.

It's in the properties window that you have to define how the cell will be viewed ("view-as").

4.1.10.2.1 Defining master and detail files for a table object

The association of files to a table object is done by a simple "drag and drop" from the data schema:

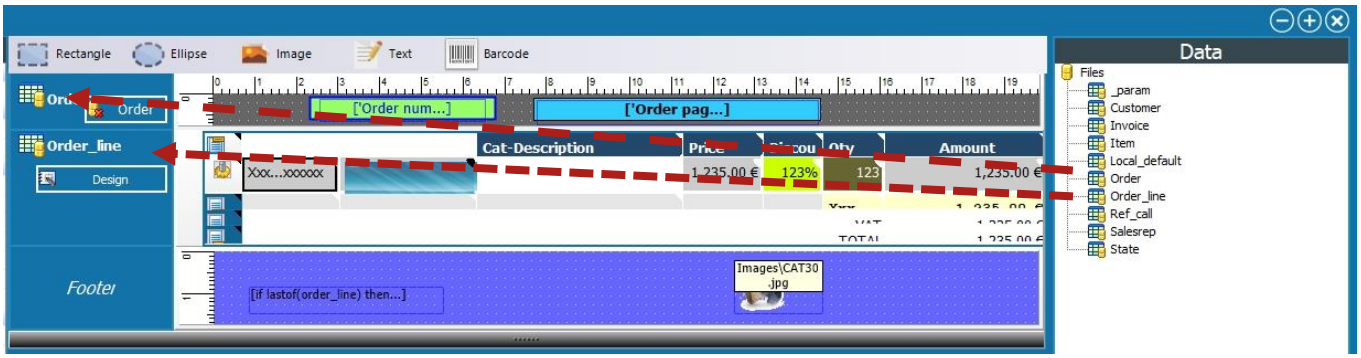


Figure 31 - Association file to a table object

You can delete the association with the master file by clicking on the corresponding button.

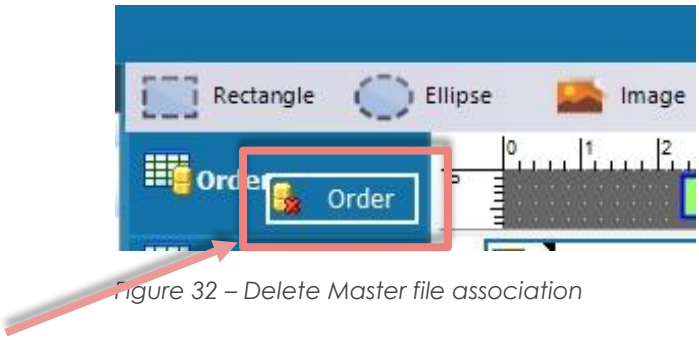


Figure 32 – Delete Master file association

4.1.10.2.2 Objects in a table

We insert the objects in the cells of the table by "drag and drop" from the diagram:

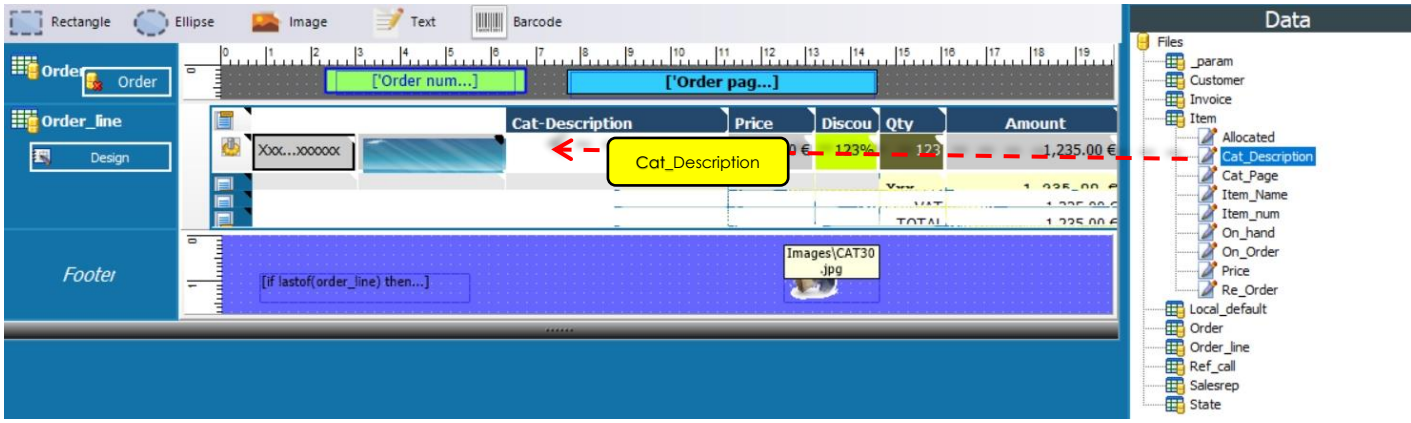


Figure 33 – Adding an object in an array

The label is created automatically in the title line.

Double-clicking on a cell opens an object properties window, identical to the one on the main page:

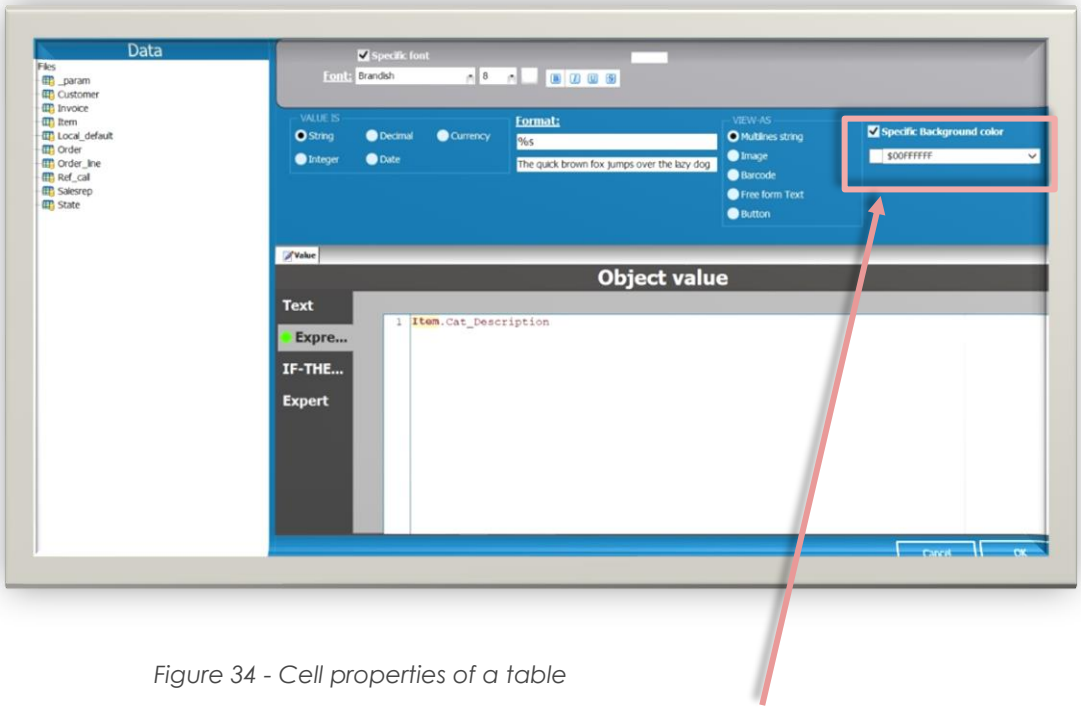
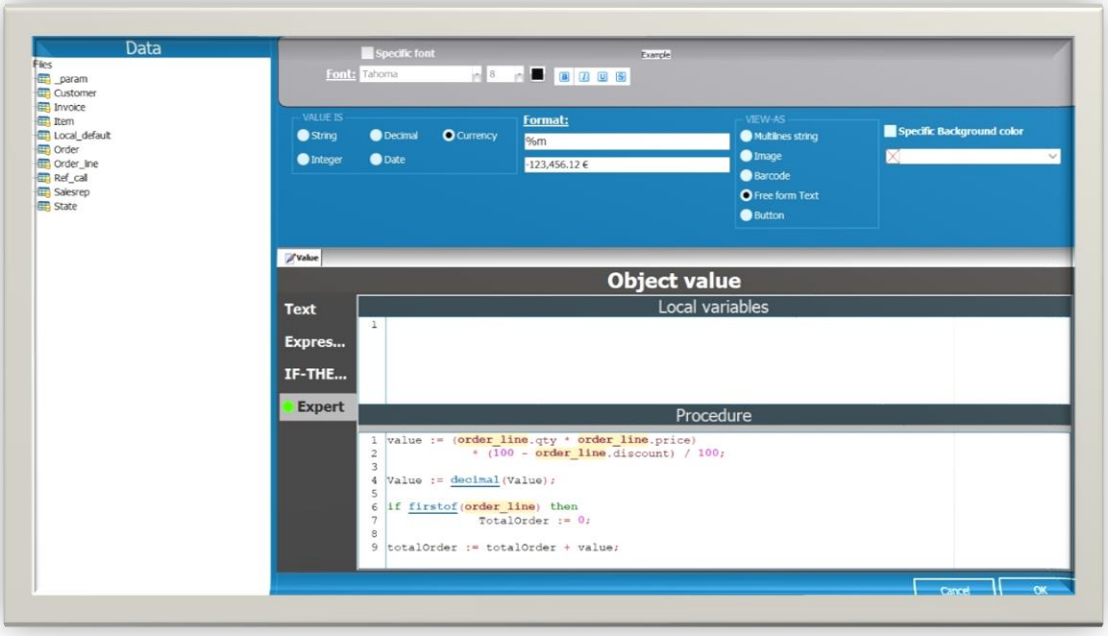


Figure 34 - Cell properties of a table

A difference, the cell can have a background color different from that of the line.

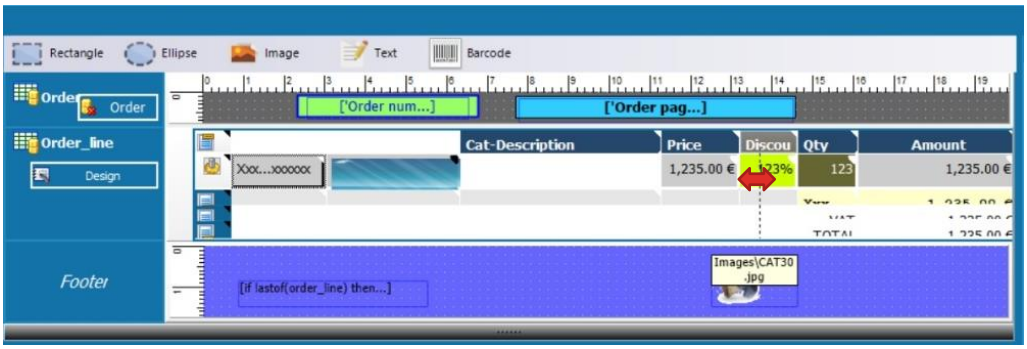
If the cell is a calculated value, a double-click makes it possible to specify its contents:



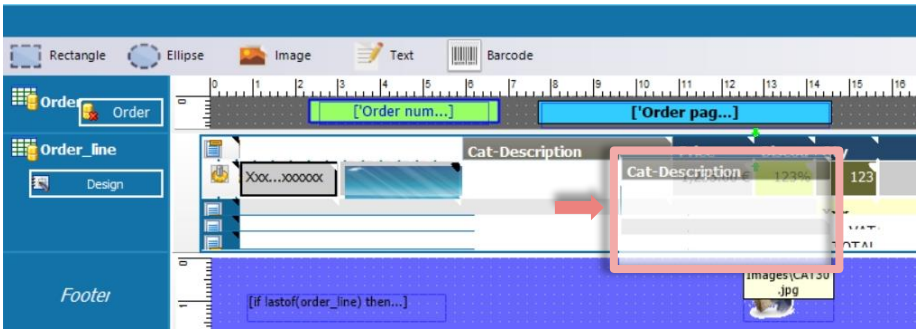
4.1.10.2.3 Formatting a table object

To adapt the table to the desired look, use the mouse:

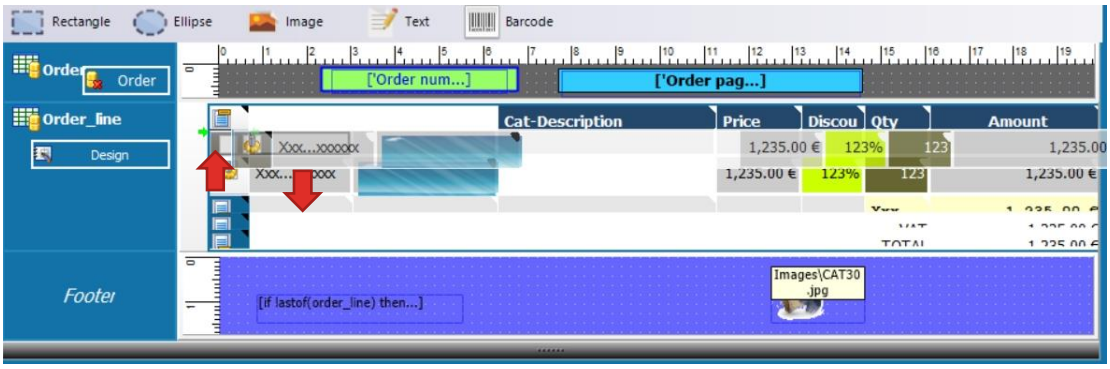
Changing the width of a column



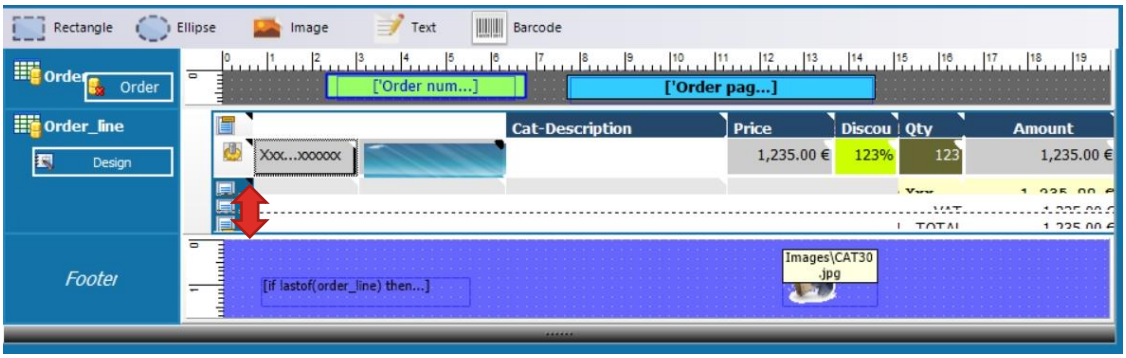
Column move



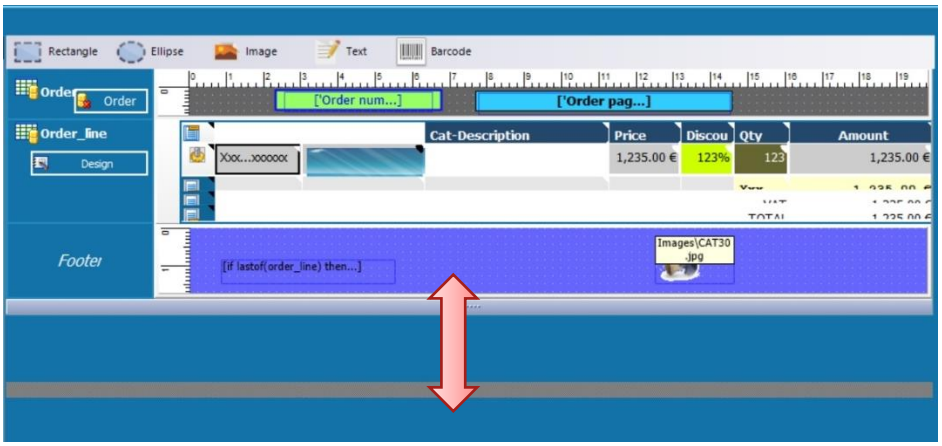
Line move



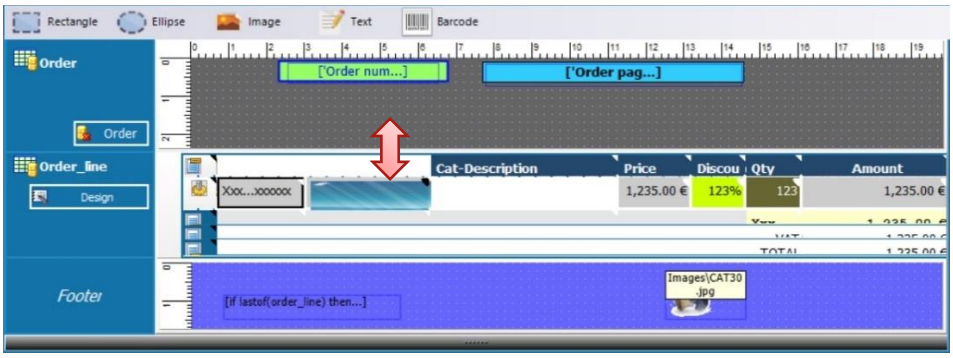
Line height modification
(In some cases, it may be necessary to add a line to adjust the height, delete it after)



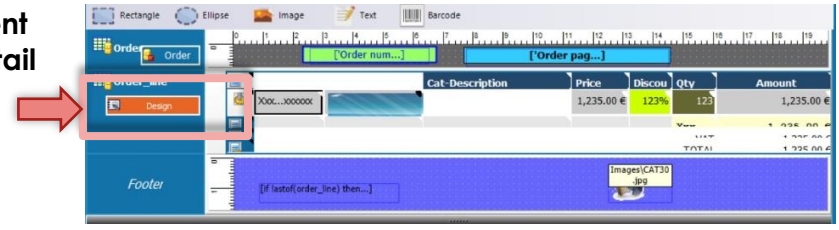
Adaptation of the size of the footer



Adaptation of the size of the header

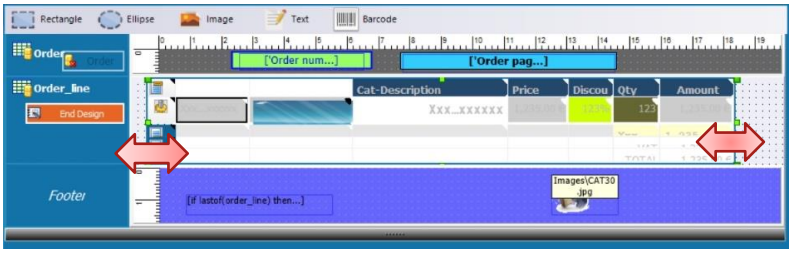


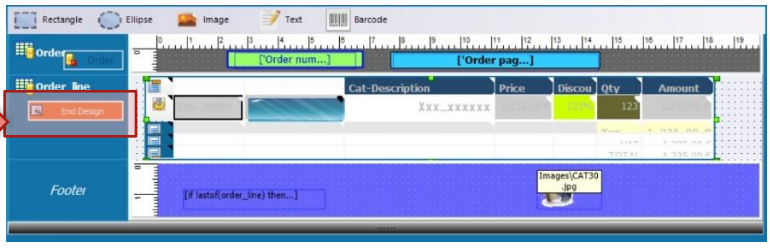
Adjustment of the detail area



Starting the « design » mode

Adjustment of the detail area with the handles





And stop the « design » mode

4.1.10.2.4 Detail lines

The lines of the detail area are accessible by the individual buttons of each line:

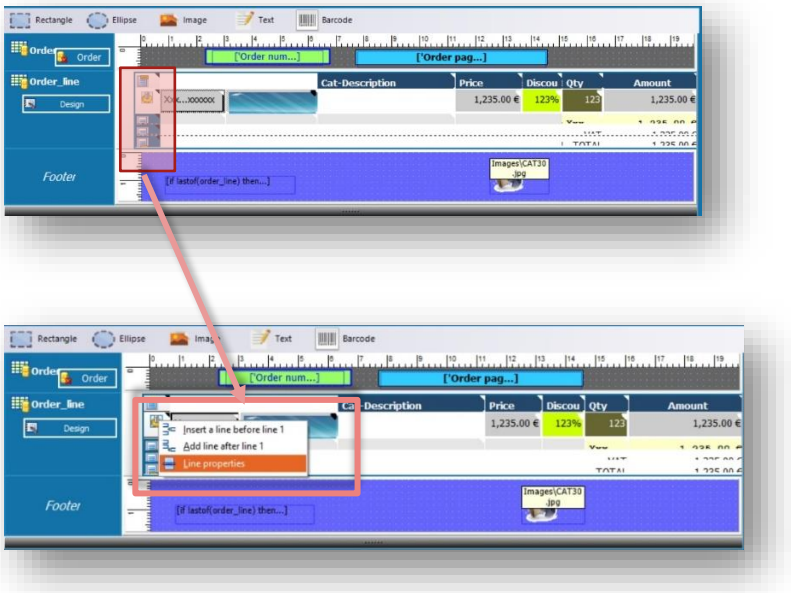


Figure 35 - Table Object - Lines

4.1.10.2.5 Detail columns

Inserting and deleting a column can be accessed by right-clicking on a cell in the title line:

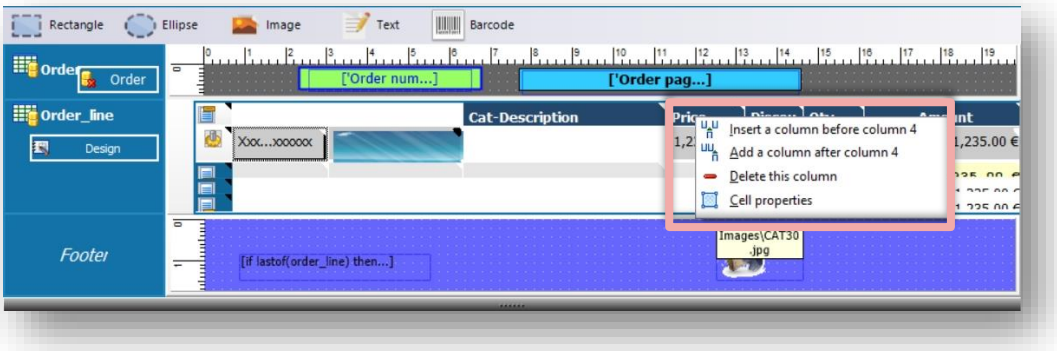
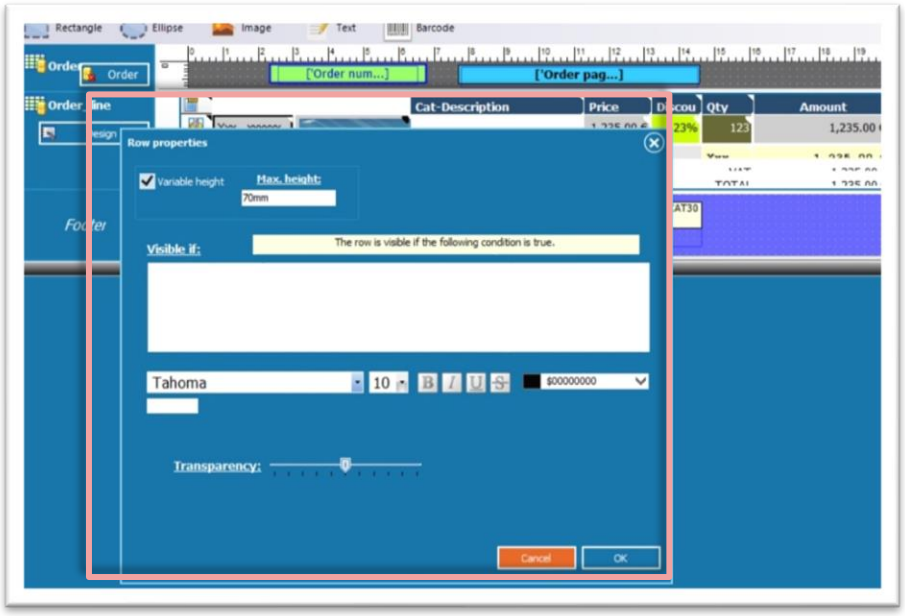


Figure 36 - Table Object - Columns

4.1.10.2.6 Detail line properties

Line properties window

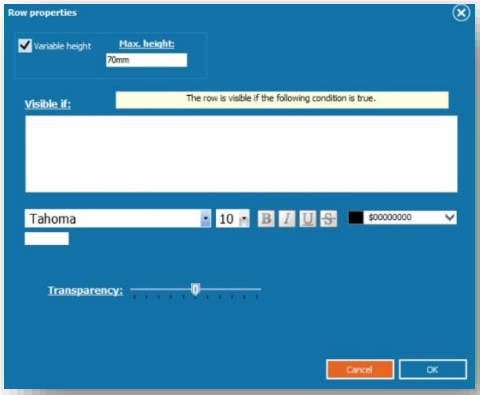


:

Depending on the type of line, some options are available.

First detail line:

1. Variable height. If the line contains a multi-line cell, the height of the line will automatically adjust to the height of the text,
2. Max height, if variable height,
3. Condition of visibility of the line,
4. Font,
5. Level of transparency



Next detail lines:

A detail line can be superimposed on the previous lines, which makes it possible to have several layers on a line.

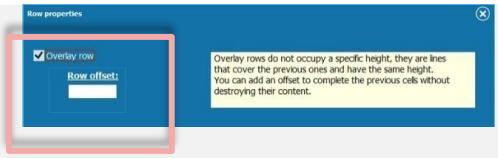
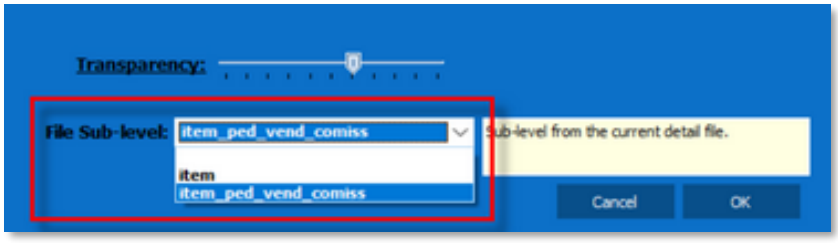
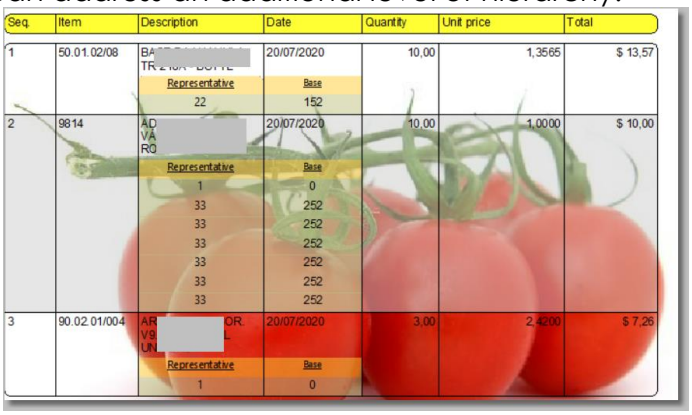


Figure 37 - Properties of the detail lines

Additional level of detail (version 4.2):

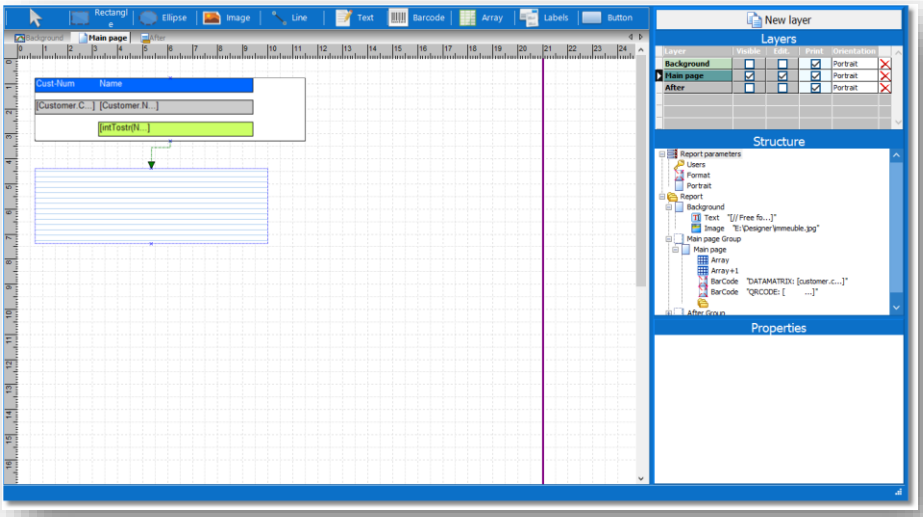


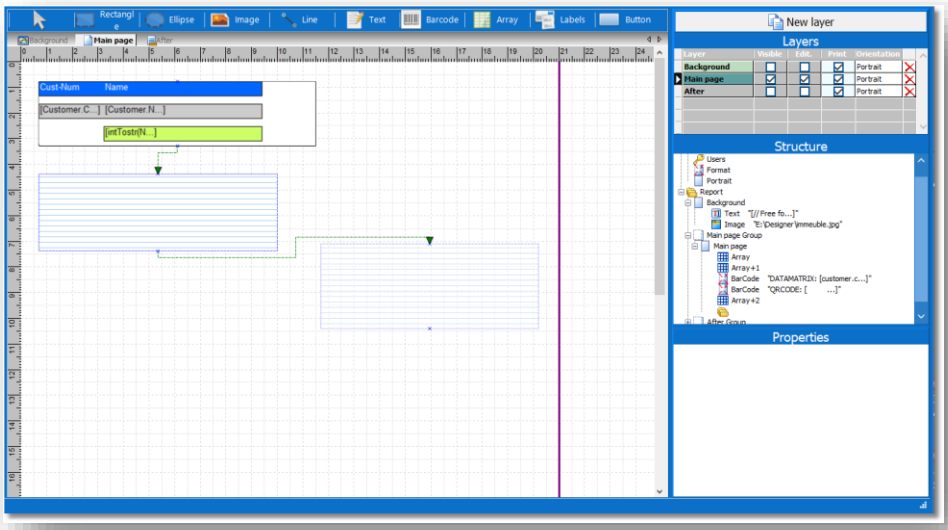
A detail line can address an additional level of hierarchy:



4.1.10.3 **Array chaining** (version 3.4)

When several array objects are defined in the same page, they refer to the query of the initial array. They are considered as overflow areas of the query.
vpDesigner indicates graphically the order of the objects.



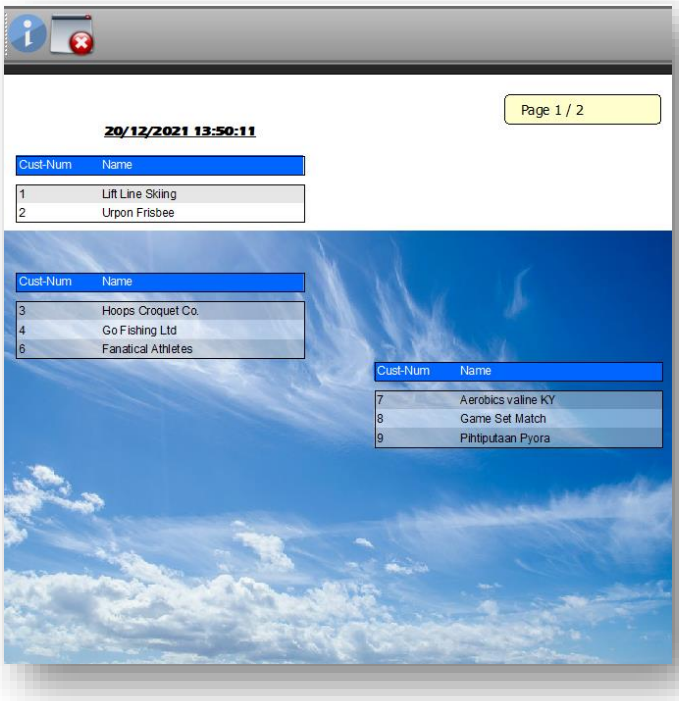


The order can be changed by modifying the Z-order of the objects with the buttons "bring to front" and "send to back":

The first array object is always presented at the beginning of the chain.

Preview:

Page 1



Page 2

4.1.10.4 Graphic object (version 3.8) :

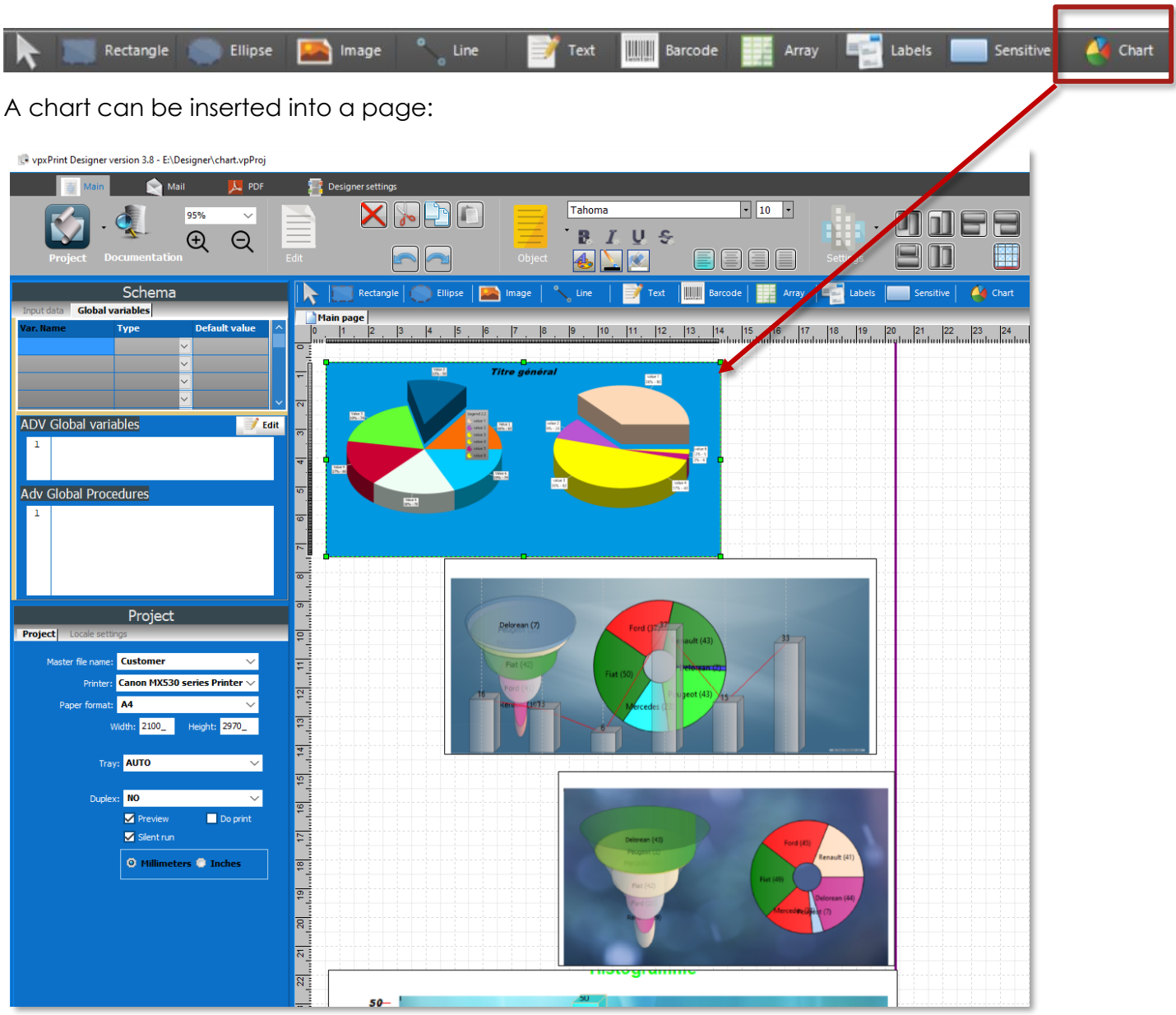
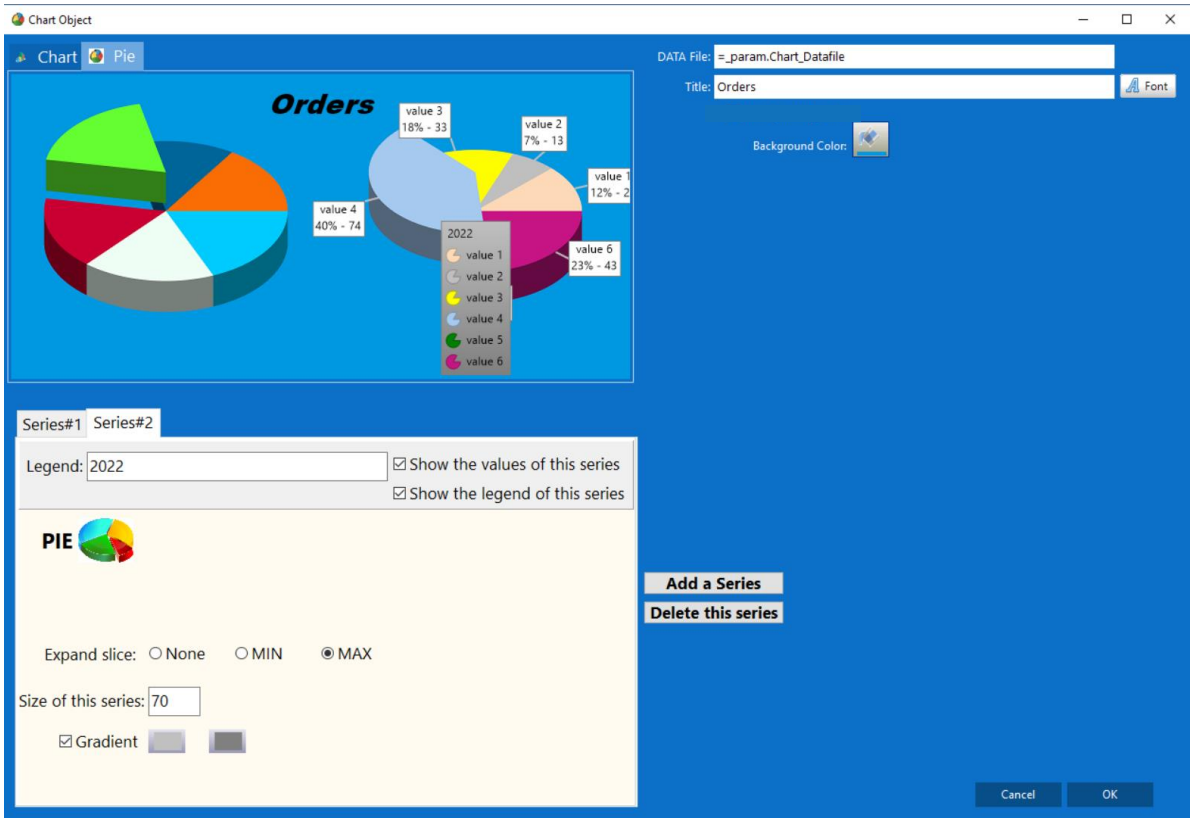


Figure 38 – Charts and Pies

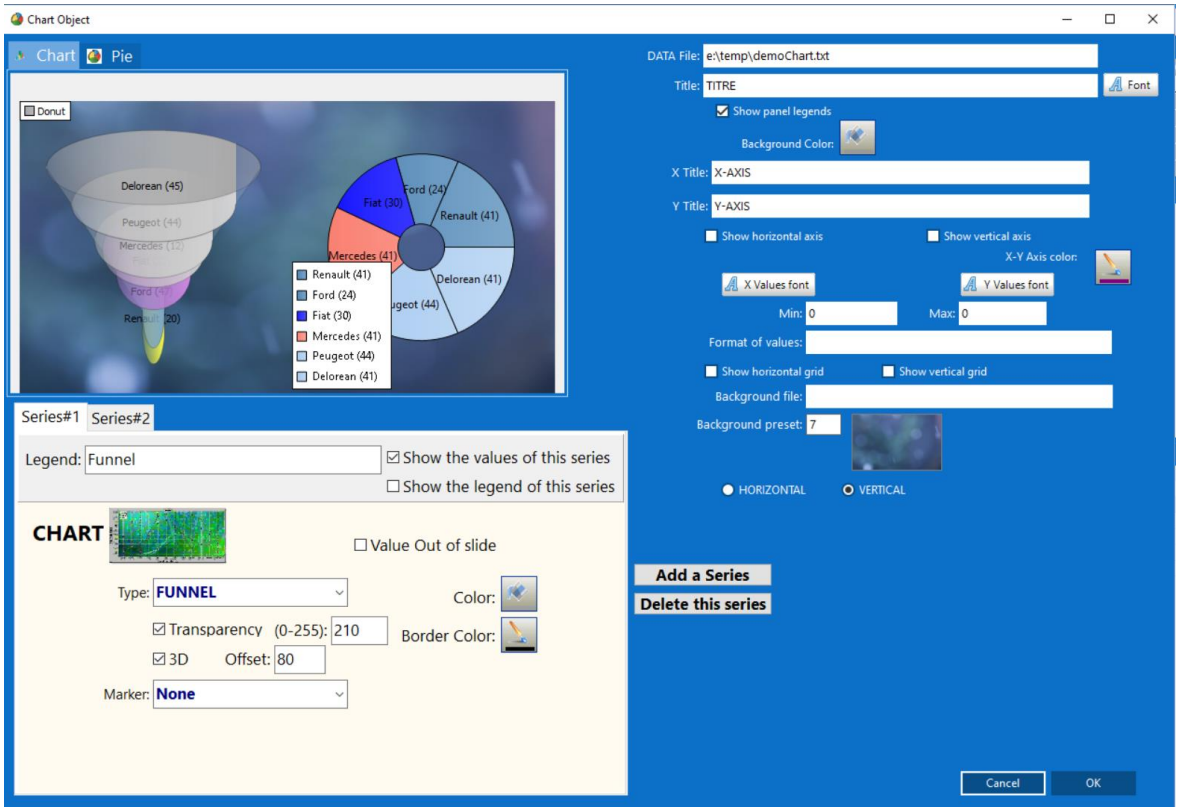
A double-click on an object calls the graphic design window.

6 random series are generated by The Designer to allow visualization of the result in WYSIWYG mode.

4.1.10.4.1 Pies :

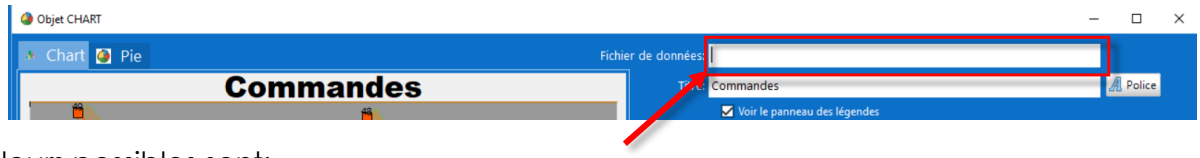


4.1.10.4.2 Charts :



4.1.10.4.3 Data files(data) :

Each chart must have its data file (<DATA> in vpxPrint format), it defines the different points of the charts.



Les valeurs possibles sont:

i. **An external DATA file name:**
Example : **c:/temp/demochart.txt**

Interest: the same data file can be shared between several documents.
Notes : management of unique file names if shared directory,
the external DATA file must be deployed with the .xpr. 0

ii. **A <DATA> name from the .xpr:**
Example : **SALES**

Interest: the data file is embedded in the .xpr file
unless <DATA=SALES,**File=c:/temp/demoChart.txt**>
.
Note :you have to insert the data in the .xpr with **vpaddTag** of the vpRun.

iii. **A "pre-processed" tag:**
Example : **[=@DATACustomer]**

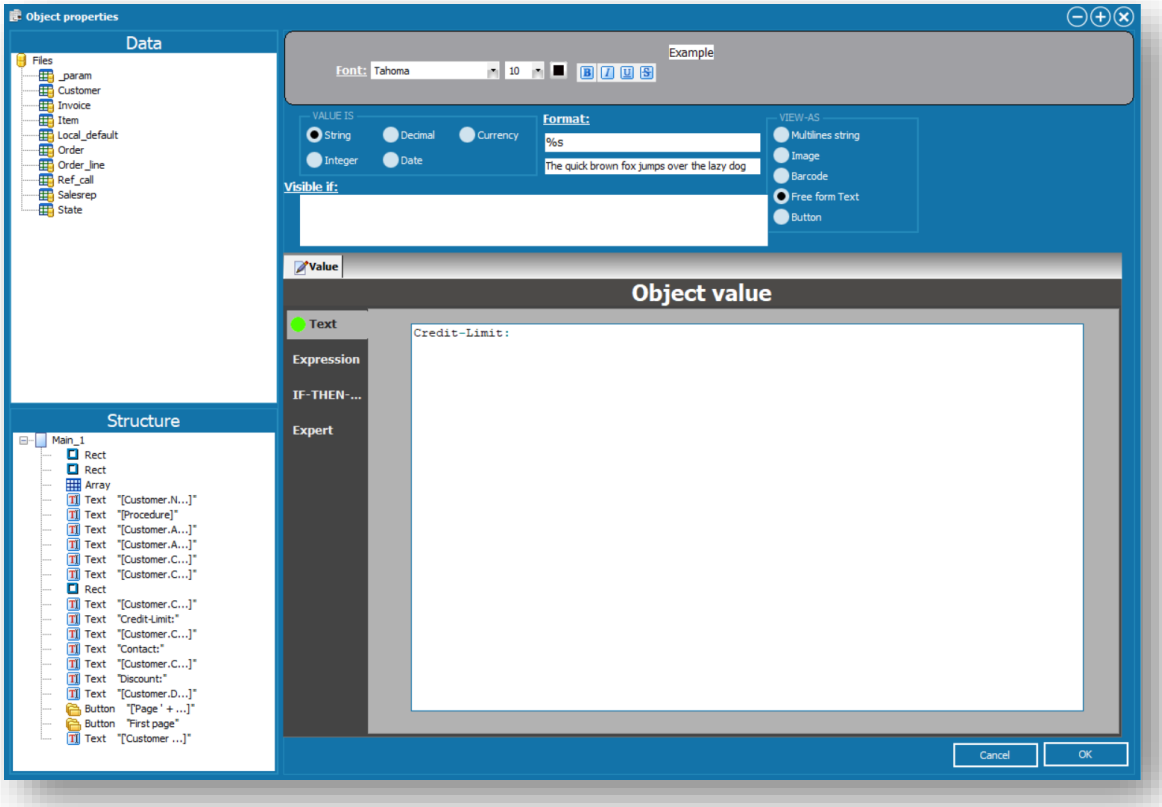
Note : the tag must be defined **[@DATACustomer=*XXXXXX*]** in the .xpr in a text object or inserted with a **vpaddTag** of vpRun.
XXXXXX is the name of an external DATA file (i) or embedded in .xpr (ii).

iv. **A field name containing any of the above:**
Example : **=customer.sales** or **=_param.dataFile**

Note : The content of the field indicates the name of the <DATA> file or the name of an embedded <DATA>.
The "=" sign indicates the reference to a data item in the schema.

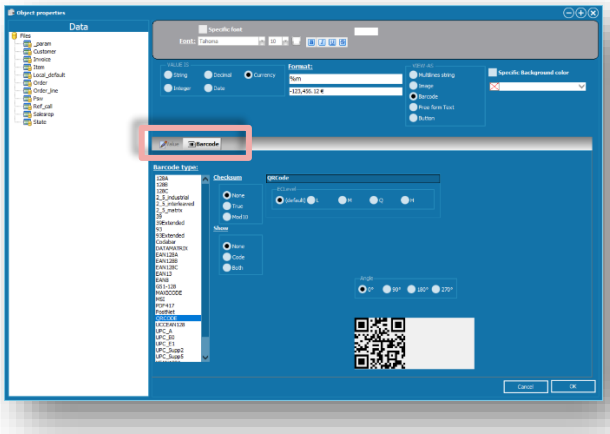
4.2 PROGRAMMING AN OBJECT

4.2.1 Properties window

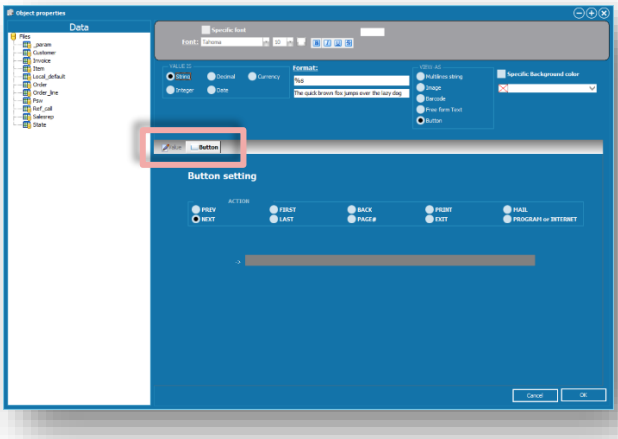


If the object is a barcode or a button, an additional tab is displayed

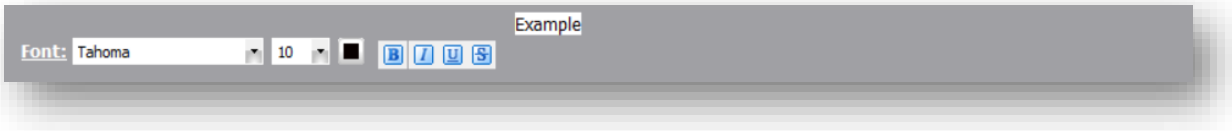
BarCode



Hypertext button



An object has its own font and color.



It is visible or not, according to a condition checked at execution time.

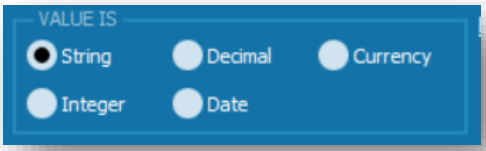
It is defined in a property window and can be expressed as:

1. A literal value (a fixed value such as a label),
2. An expression,
3. An expression of the type "if - then - else"
4. A procedure programmed in Pascal

(the calculated value will serve as a basis for its representation)

using a **format**:

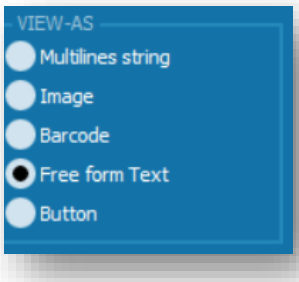
1. Alphanumeric
2. Digital with decimals,
3. Integer,
4. Currency,
5. Date



See formats

View as:

1. A text on several lines (frame vpxPrint),
2. A text on a single line,
3. An image,
4. A barcode
5. A button (vpxPrint sensitive area)



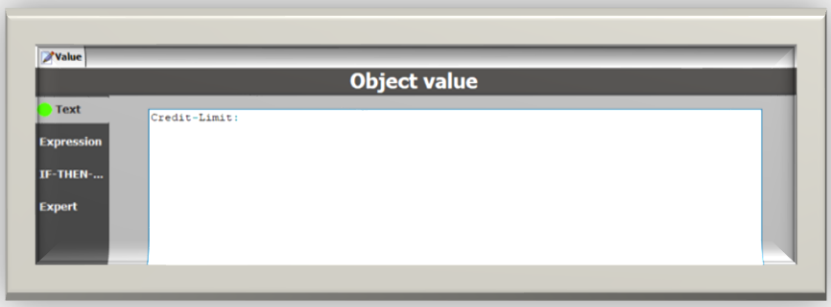
If the object is an expression or procedure, it is possible to encode actions such as inserting vpxPrint sequences, accumulating, totalizing, counting, and so on.

4.2.2 Definition of object content

The content of an object is organized around a system with 4 tabs: text, expression, if-then-otherwise and expert mode. A green LED ● indicates how the object will be evaluated, an orange LED ● indicates that there is an expression for this tab but not activated, and therefore not used.

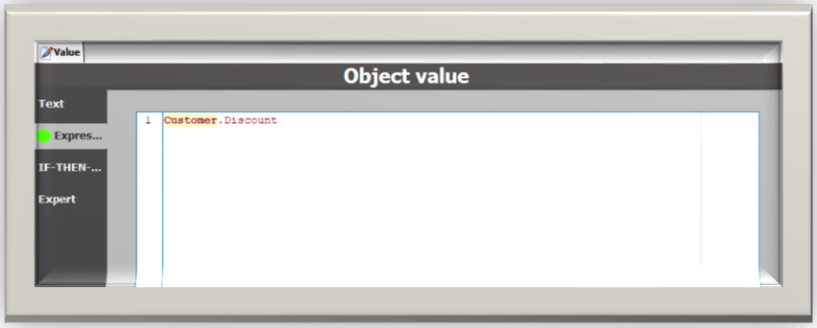
1. **Text**
   Fixed value reproduced "as is" in the page.

« Credit-Limit: »

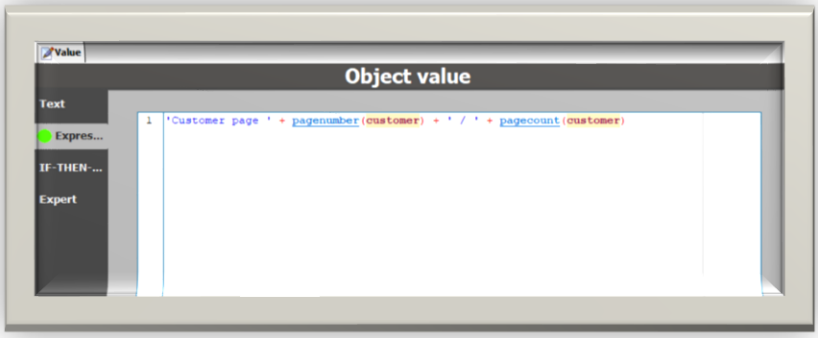


2. **Expression**
   Combination of instructions and / or functions in Pascal format.

Field 'discount' of 'customer' file



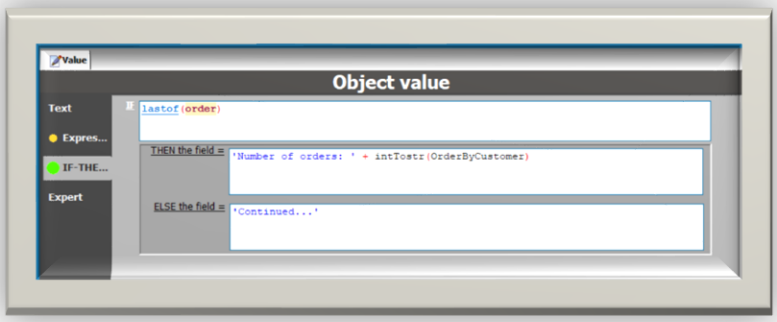
« customer page 2 / 4 »



Path of an image file

« images/CAT/001.jpg »

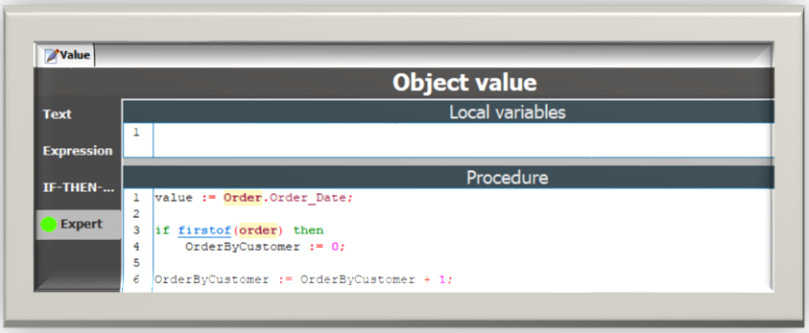
3. If-Then-Else
Simple way to express a value subject to a condition



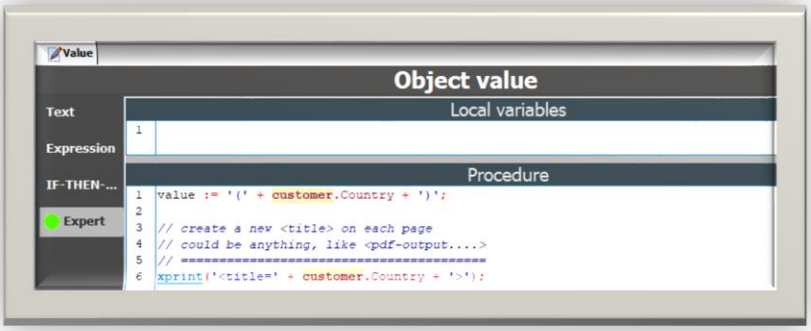
4. Procedure (expert mode)
The value is calculated by a procedure written in Pascal.
This value must be returned in the variable called "**value**".

In these two examples:

- Count of the number of orders per customer using the **FIRSTOF**(order) function



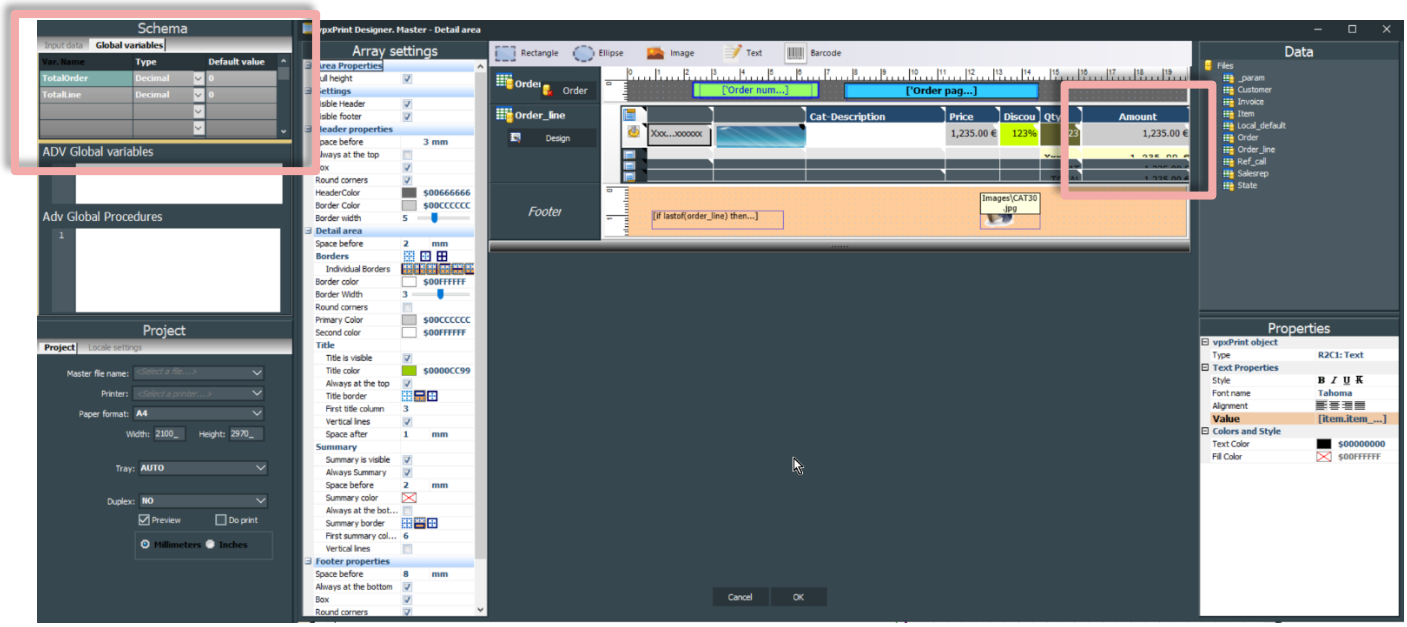
- We return a value of type "(USA)" and we insert a vpxPrint tag to put the customer's country in the thumbnail of the preview.



4.2.3 Programming an array object

The Table object is particularly important for building lists or tables.

	Type of query	Main Page	Master,of the Array object	Detail file of the array
Simple list (lcustomers or items list)	FOR EACH customer	--	--	**-- The file to list --** *Ex : customer, item, etc.*
Structured report (3 levels)	FOR EACH customer, EACH order OF customer, EACH order_line OF order, FIRST item OF order_line	**Base file** *Ex : customer*	**Header file** *Ex : order* *Invoice*	***- Detail file --*** *Ex : order_line*
Structured report (2 levels)	FOR EACH order WHERE xxx, FIRST customer OF order, EACH order_line OF order, FIRST item OF order_line	**--** *Ex : customer*	**Header file** *Ex : order* *Invoice…*	***- Detail file --*** *Ex : order_line*



Double-clicking or right-clicking (context menu) on a cell opens the object properties window.

A bookmark is set on each column of the table: # 1, # 2, ..., #n, which allows you to use vpxPrint tags in expressions.

In this example invoice, we calculate the amount of each line to accumulate in the amount of the invoice. To carry out this operation, we define in the project a global variable "TotalOrder":

We accumulate the amount lines in this global variable.

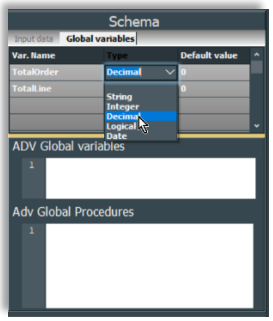
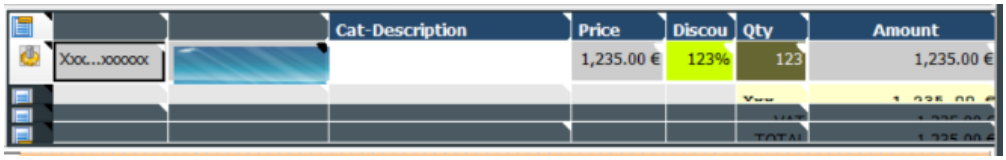


Figure 39 – Global Variables

4.2.3.1 Programming Array objects Cells



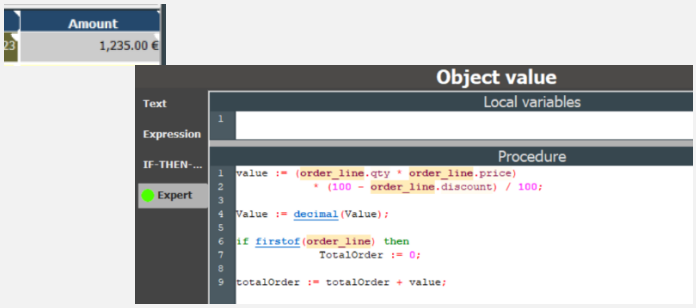
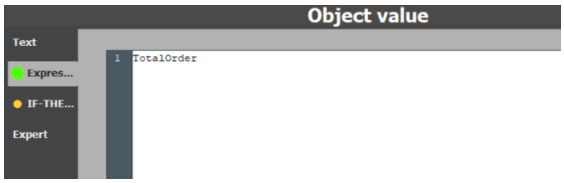
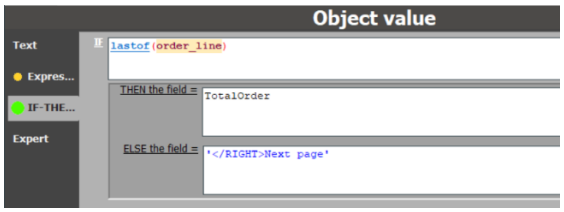
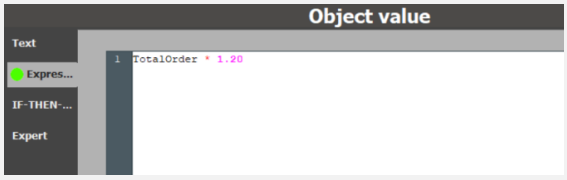
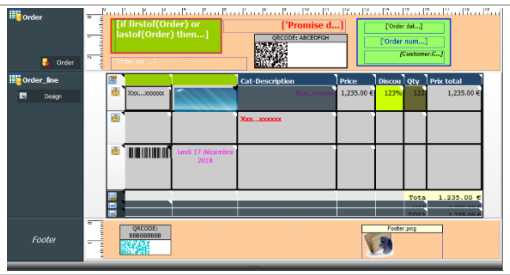
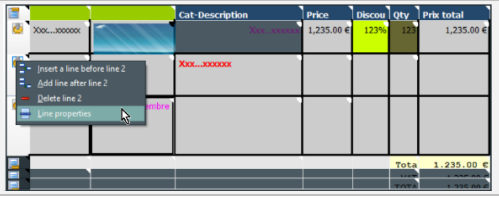
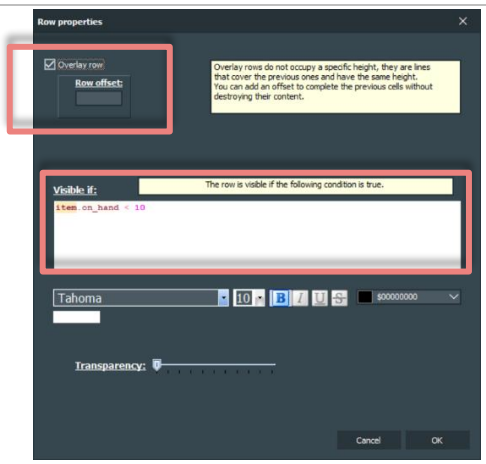
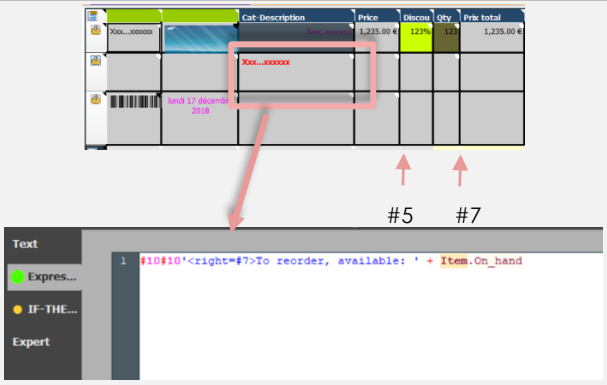
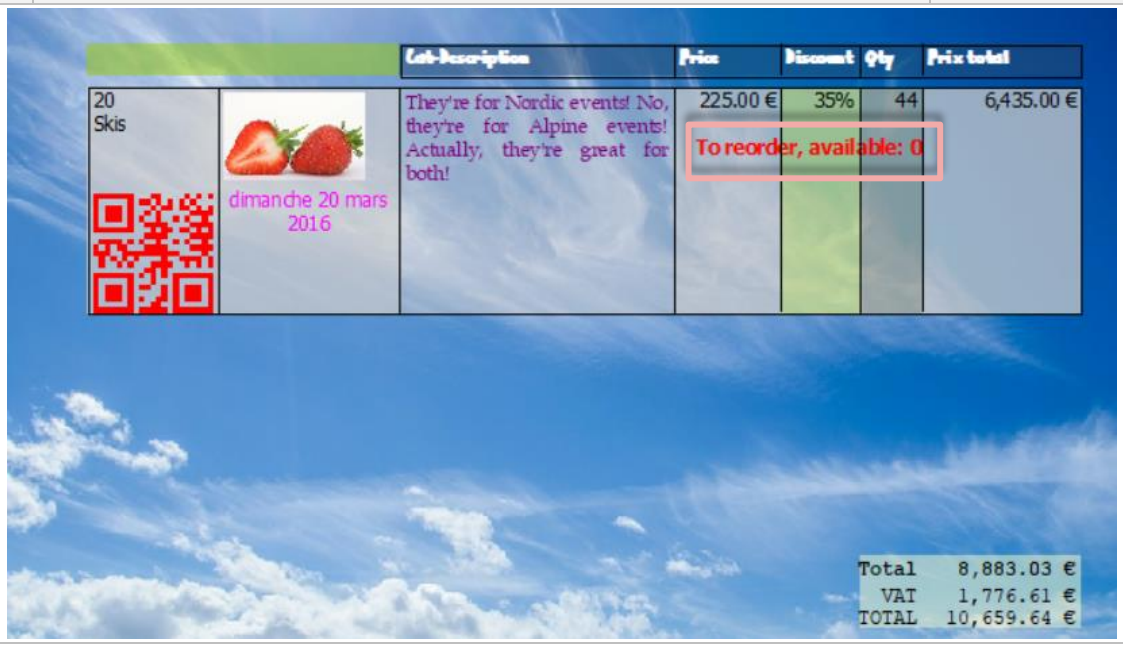
Calculation of the amount of the line		"**Value**" returns the calculated value of the line. Decimal truncates to 2 decimals, The variable "TotalOrder" is reset for the first line of the order The amount of the line is cumulated without "totalOrder"
Total zone excluding taxes	 	Expression « TotalOrder » Here, « totalOrder » is the current value. We could create a line of the type "report next page" by coding thus (for example): If this summary line is not the last of the order, we put a word "Next page" right justified (vpxPrint tag)..
Total zone with taxes		Simple expression representing 120% of TotalOrder *(we could use a "tax" file)*

Figure 40 – Array object programming

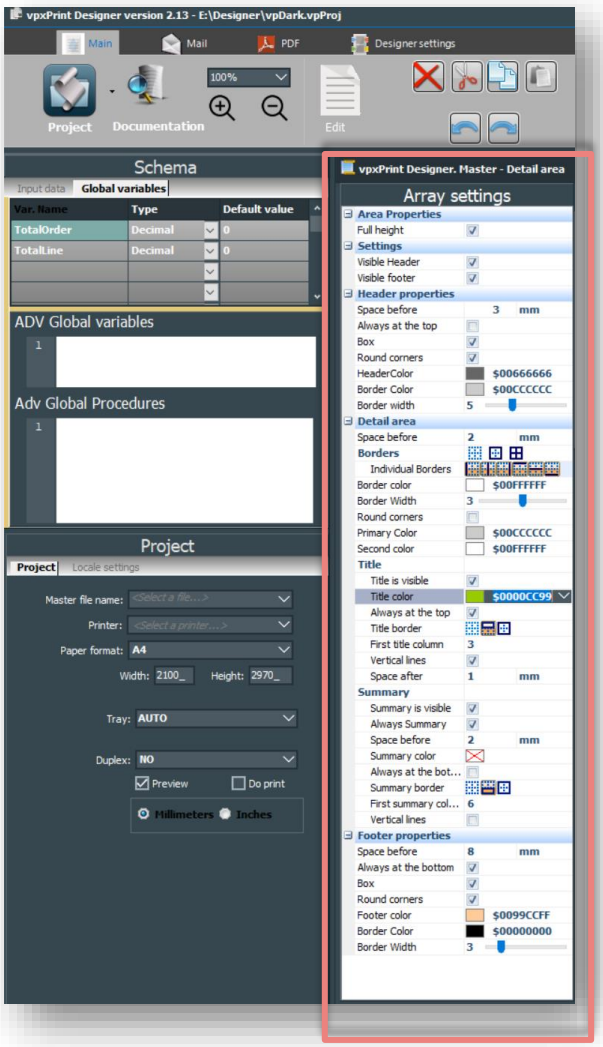
4.2.3.2 Overlay lines

Example of an overlapping line, « vpInvoice.vpProj »

		
Second line defined as overlay		Access to the line menu ...

		Properties of the line: • Line in overlay, • Visibility condition "available <10"
Double-click on the cell		This cell will be superimposed on the main line if the available quantity is less than 10. Here, we do not use the overlay offset, we use two line skips (#10). We use a vpxPrint <RIGHT#7> tag that aligns the value on bookmark # 7, the last column..
		

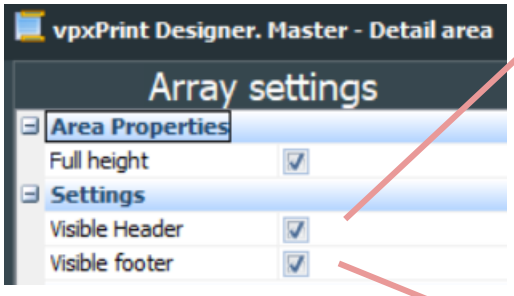
4.2.4 Array Object format

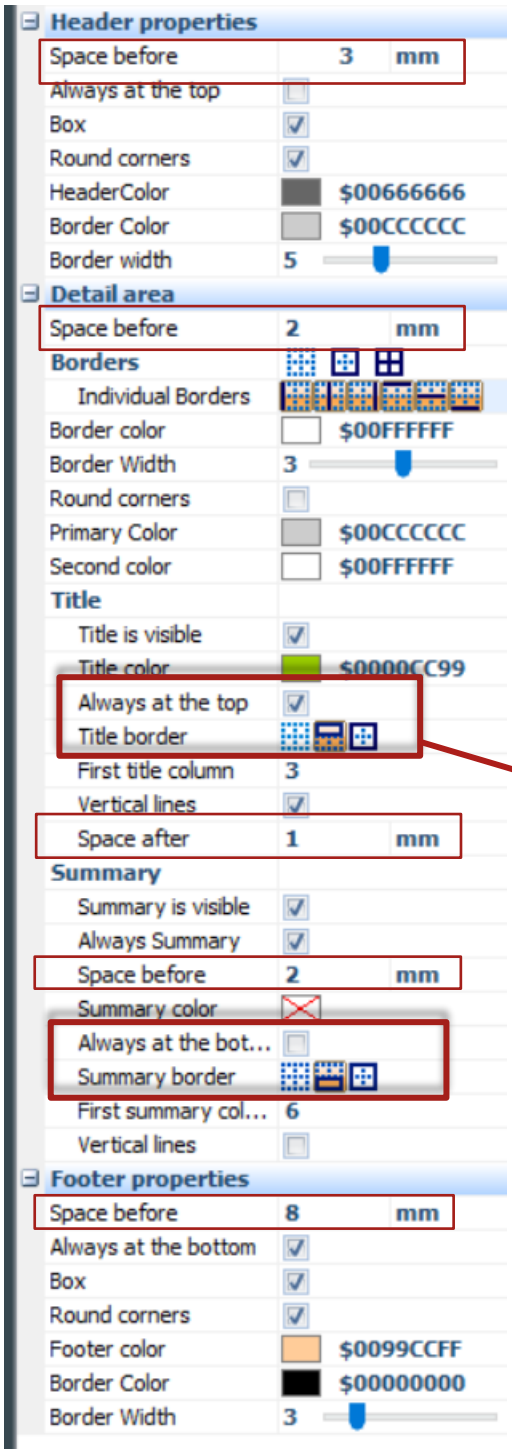


The Table object is extremely configurable and easily adaptable to multiple presentations required in a professional application.

Arrangements according to the settings:

In "Full height" mode, the table is in fixed format





Front space

Borders of the table, color, thickness.

Rounded corners

Primary and secondary color

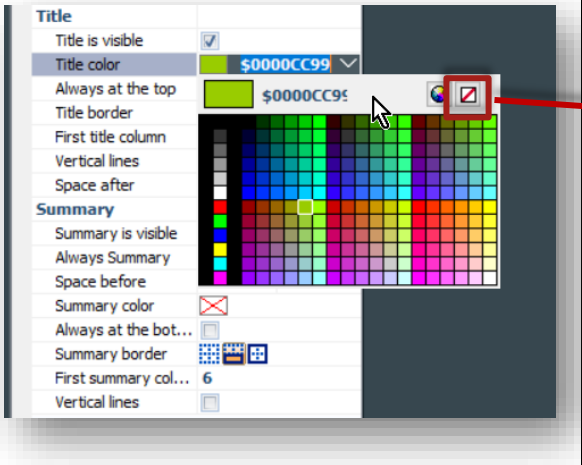
Indicates that the title::

- Must always be at the top,

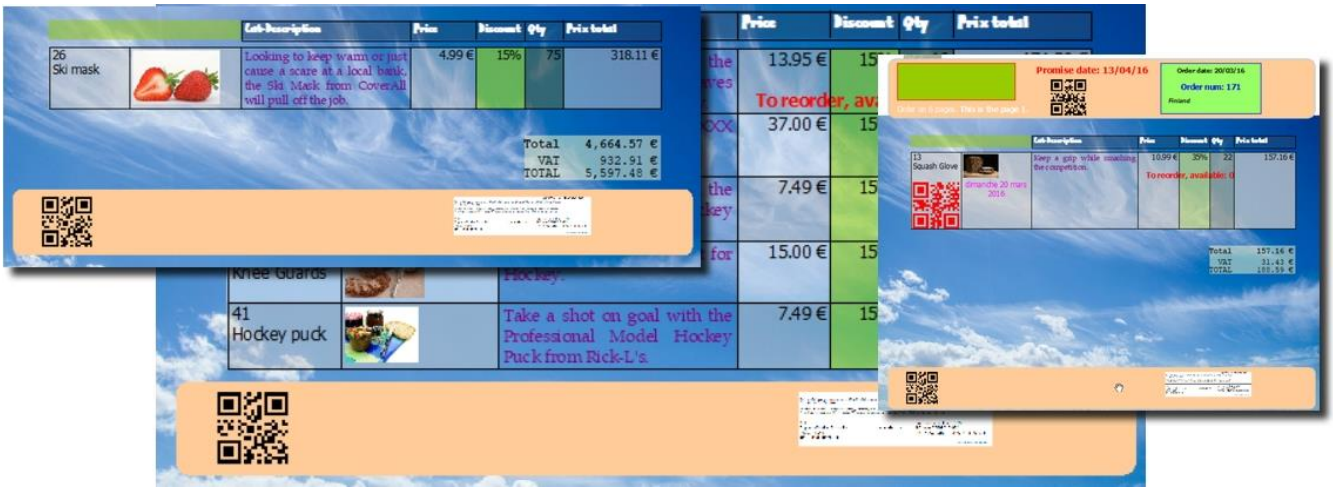
- Has no border,
- Has its own border,
- Has its border included in that of the detail

-id- title

Color adjustment

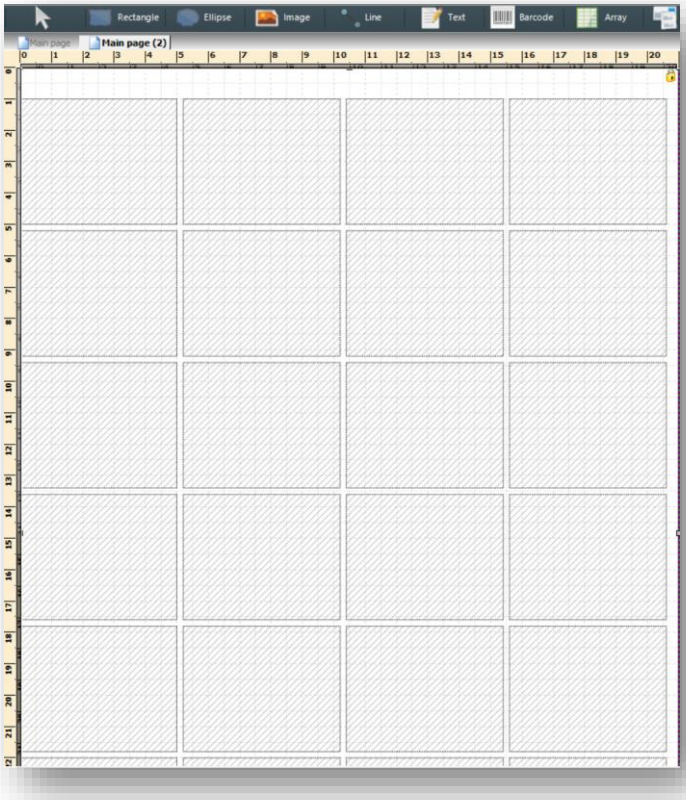


Rendering examples with different parameters:


26
Ski mask
Looking to keep warm or just cause a scare at a local bank, the Ski Mask from CoverAll will pull off the job.
4.99 €
15%
75
318.11 €
Total 4,664.57 €
VAT 932.91 €
TOTAL 5,597.48 €
13.95 €
To reorder, ava
37.00 €
7.49 €
15.00 €
Knee Guards
41
Hockey puck
Take a shot on goal with the Professional Model Hockey Puck from Rick-L's.
7.49 €
Promise date: 13/04/16
Order date: 20/03/16
Order num: 171
Finland
13
Squash Glove
dimanche 20 mars 2016
Keep a grip while smashing the competition.
10.99 €
35%
22
157.16 €
To reorder, available: 0
Total 157.16 €
VAT 31.43 €
TOTAL 188.59 €

Promise date: 27/03/16
Order date: 24/02/16
Order num: 118
Order on 1 page(s). This is the page 1.
16
Baseball Bat
This one's a real slugger. You'll crack them out of the park with this one.
13.95 €
15%
76
901.17 €
36
Soccer ball
15.51 €
15%
73
962.40 €
40
Ice Skates
61.00 €
15%
21
1,088.85 €
Total 2,952.42 €
VAT 590.48 €
TOTAL 3,542.90 €
Promise date: 10/04/16
Order date: 04/03/16
Order num: 141
Order on 2 page(s). This is the page 1.
24
Snorkel
On the reef or in a lake, the Ultra-Snorkel by AirWaves will take your breath away.
13.95 €
15%
40
474.30 €
To reorder, available: 0
Promise date: 16/03/16
Order date: 26/02/16
Order num: 125
Finland
Order on 2 pages. This is the page 1.
41
Hockey puck
vendredi 26 février 2016
Take a shot on goal with the Professional Model Hockey Puck from Rick-L's.
7.49 €
35%
74
360.27 €
28
Ski Gloves
vendredi 26 février 2016
Field tested with the Parka.
23.00 €
35%
98
1,465.10 €
Total 1,825.37 €
VAT 365.07 €
TOTAL 2,190.44 €

4.2.5 Labels Object



The Labels object allows you to edit label pages with multiple parameters. It occupies a complete page but can be positioned precisely in the page by changing its margins in the properties window.

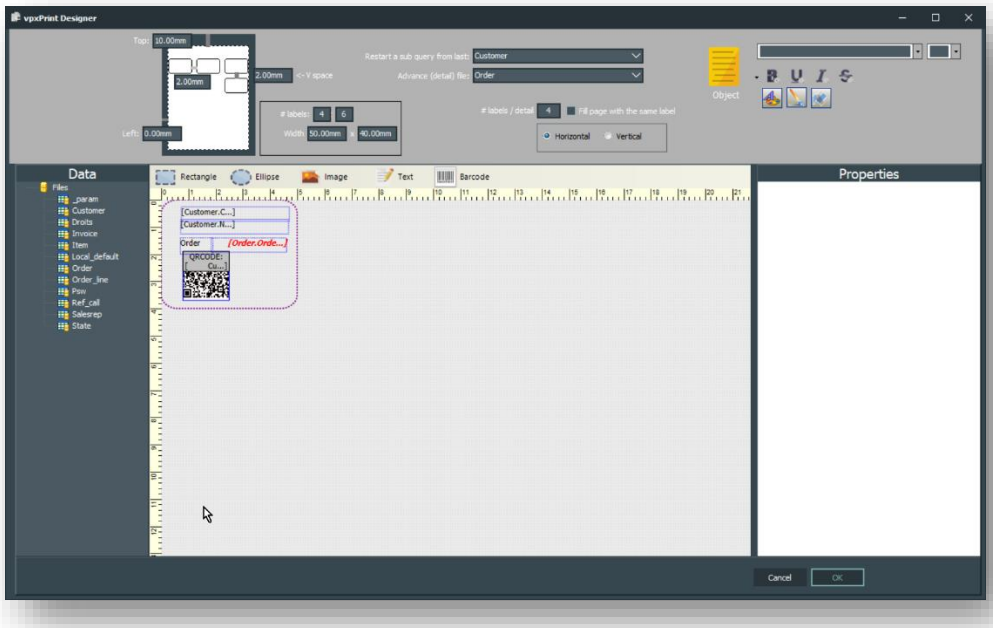
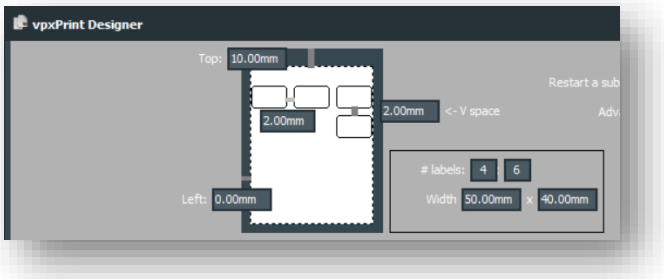
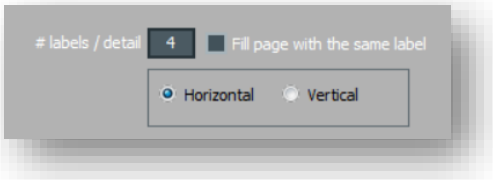


Figure 41 - Labels - Properties



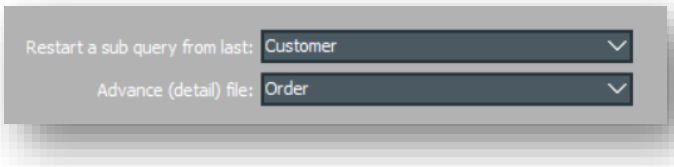
Top and left margins,
Vertical and horizontal spaces

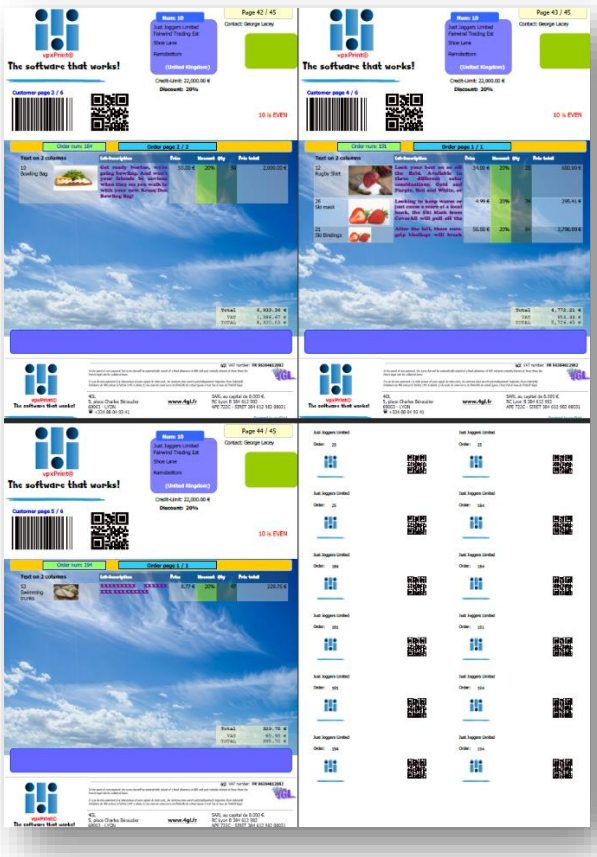
Width - height
Dimensions



Number of labels by detail record,
Full page with the same label,

How to fill the labels in the page



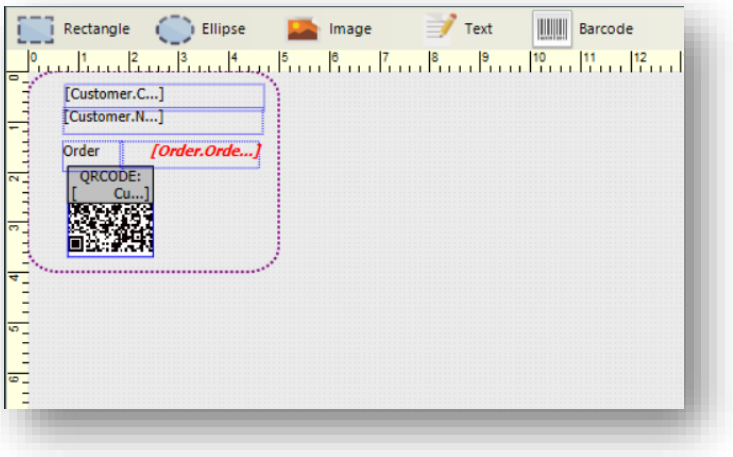


The detail file defines the file to advance for each group of labels, in relation to the number of labels per detail,
You can start querying details on a previous file with a subquery.

<u>Example:</u>

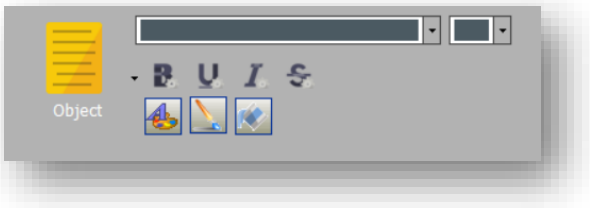
- all the invoices of a customer are printed, followed by pages of labels for each order::
  « Restart » Customer -> Order

- At the end of the edition of an invoice, we edit a page of labels with all the articles of the order.
  « Restart » Order -> Order-Line



Space for defining a label.

Double-clicking an object opens the standard properties window.



Color attributes and font.

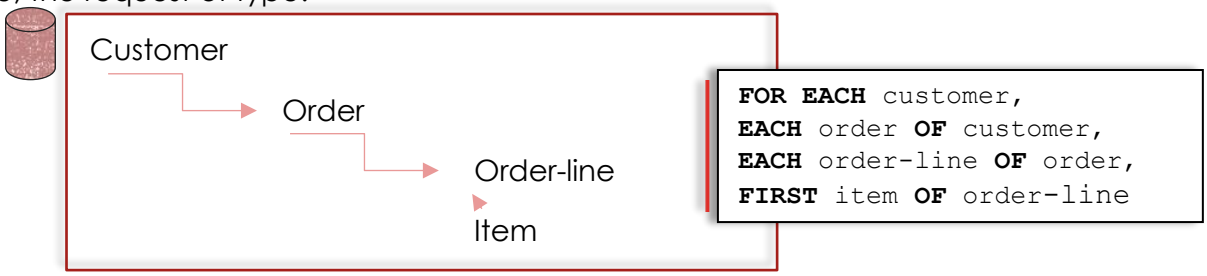
4.3 PROCESSING

4.3.1 Logic

Processing a main page "advances" the master file name ("master file name").

If no file is selected, the first file of the query is used.

In this example, the request of type:



uses « **customer** » as master file.

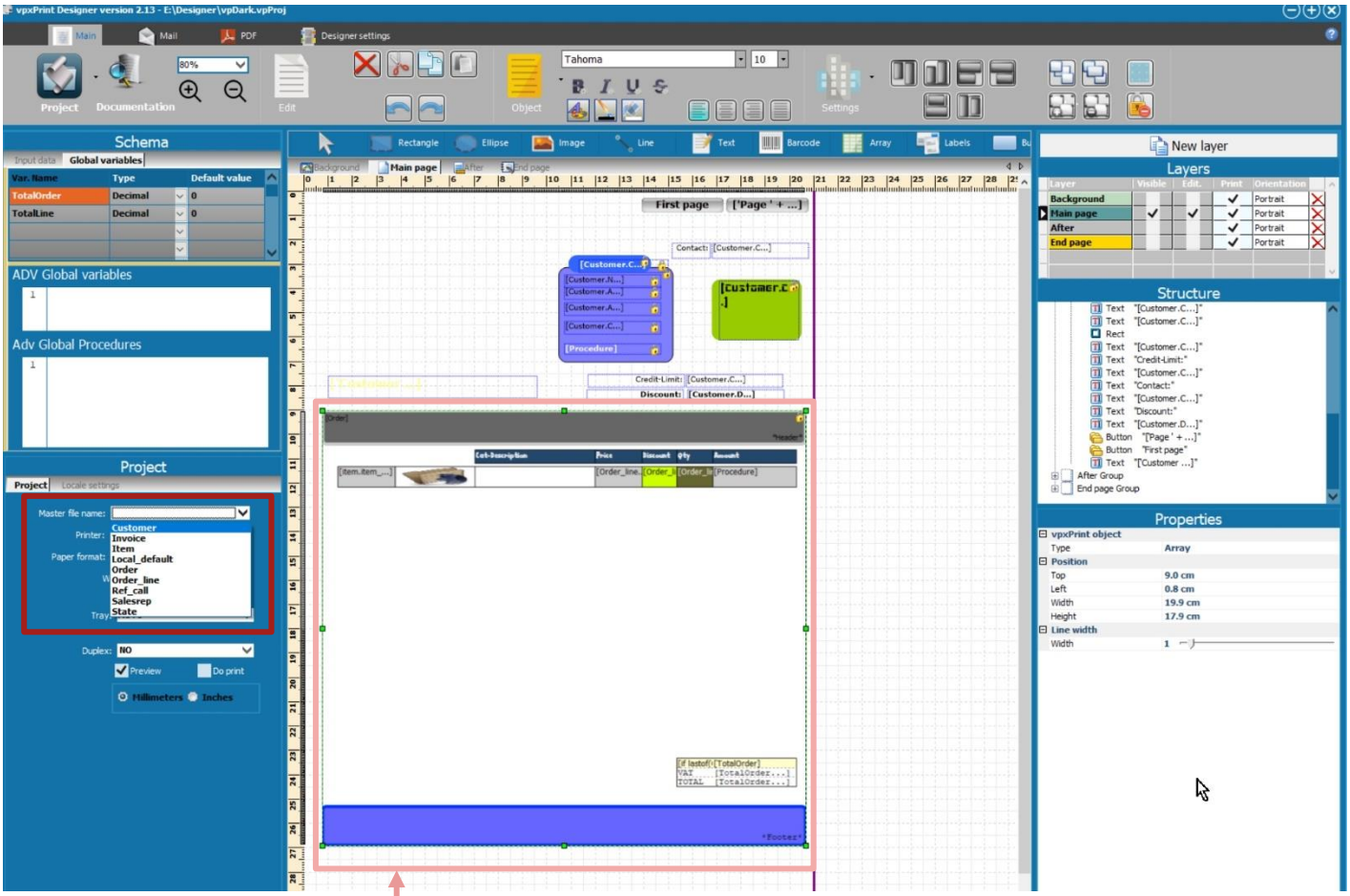


Figure 42 – Master file

An array object uses the commands ("order") and command lines ("order-line") of the current customer.

4.3.2 Each master record is merged with the "Main_xxx" pages

QUERY INIT		
	Process « main_1 »	*Group 1*
	Process « main_2 »	*Group 1*
	…..	…
READ NEXT RECORD		
IF STILL A GROUP, REINIT REQUEST		
	Process « main_3 »	*Group 2*
	Process « main_4 »	*Group 2*
	…..	…
READ NEXT RECORD		
…..		

Each group of "main pages" reopens the request from the beginning, it is a simple way to generate several reports of different natures in the same project. Each group is a kind of subquery.

(An example is provided with the project "main_groups.vpProj", discussed in the video https://ww.4gl.fr/designer.html, button "The Layers & Designer cycle" at the position ~ 1: 06):

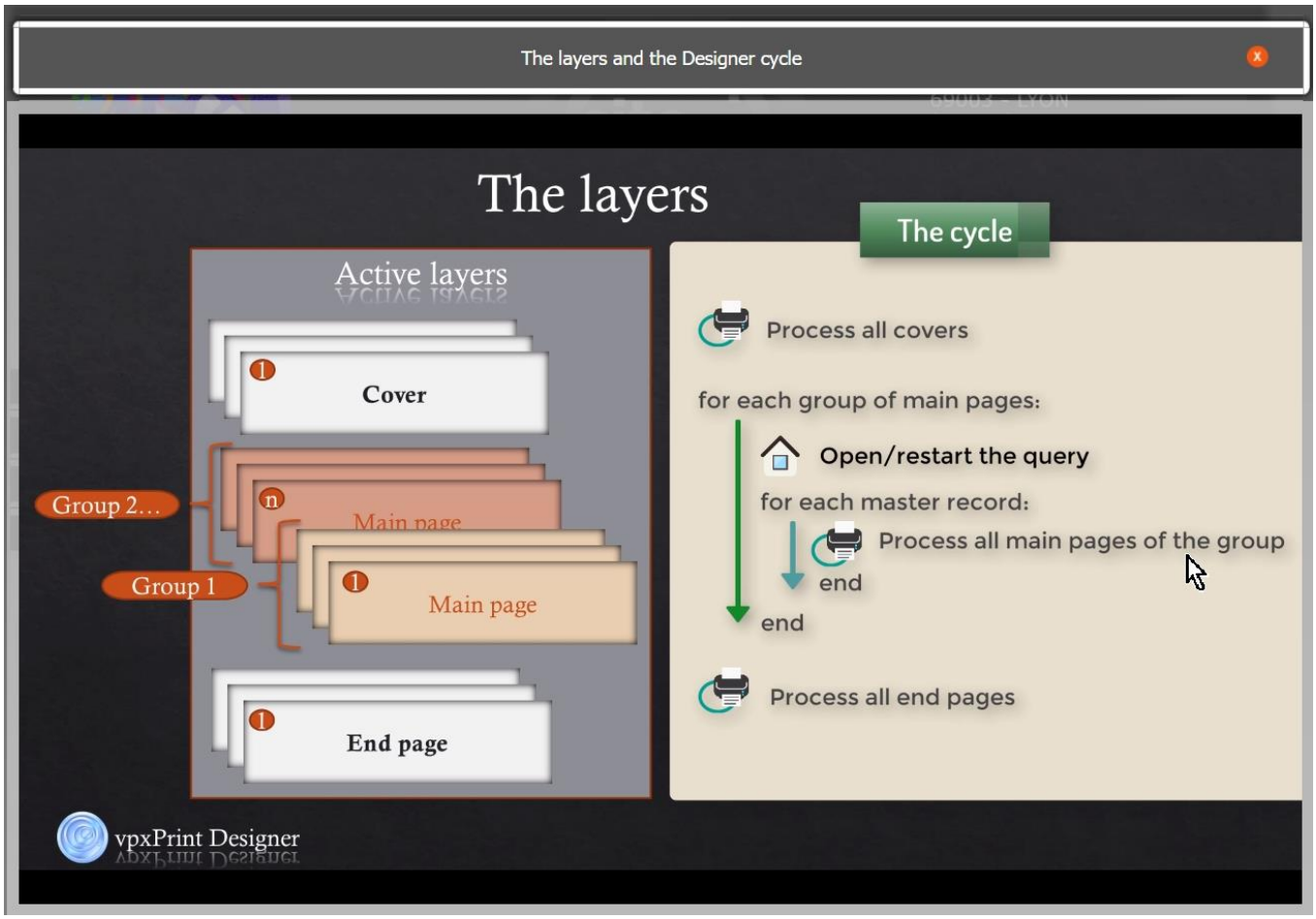


Figure 43 – The cycle

5 FUNCTIONS

5.1 FORMATS

	Format	Result
string	%s	Xxxxxxxxxxxxx xxxxxxxxxxxxxxx xx x x x xxxxxxxxxxxxxx……
	%.8s	XXXXXX = to 'x(8)'
Decimal	###,###,##0.##	123,128.56
Currency	%m	12,345.21€
Integer	###,###,##0	-123
TDateTime	d/m/y	9/2/18
	dd/mm/yy	09/02/18
	ddd d of mmm yyyy	Wed 9 of Feb 2018
	dddd d of mmmm yyyy	Wednesday 9 of February 2018
	ddddd	09/02/2018
	dddddd	09 February 2018
	c	09/02/2018 05:06:07
	h:n:s.z	5:6:7.008
	hh:nn:ss.zzz	05:06:07.008
	t	05 :06
	tt	05 :06 :07
	c	09/02/2018 05:06:07

Figure 44 – Example of data formats

5.2 PROGRAMMING, EXPRESSIONS

Table 1 - Data Conversion Functions

intToStr(integer)	String	Converts an integer to a string
floatToStr(float)	String	Converts a float to a string
currToStr(currency)	String	Converts a currency to a string
datetimeToStr(TDateTime)	String	Converts a TDateTime to a string
strToInt(string)	Integer	Converts a string to an integer
strToFloat (string)	Float	Converts a string to a float
strToDateTime(string)	TDateTime	Converts a string to a TDateTime
strToCurr(string)	Currency	Converts a string to a currency

Table 2 – File functions

extractFilename (filename) **extractFilePath** (filename) **extractFileExt** (filename) **extractFiledir** (filename)	String	File name functions to extract the base name, the file path, the file extension or the file directory of a file.

Table 3 – Specific functions

entry(integer, list [, delimiter])	String	Returns the string of the specified position in the delimited list, '' if not available
numEntries(list [, delimiter])	Integer	Returns the number of elements of a delimited list
begins(expression, startStr)	Boolean	True if expression begins with startStr. **NOT case sensitive**
decimal(Float_value [, n])	Decimal	Rounds a floating point value with n decimals
lookup(expression, list [, delimiter])	Integer	Returns the position of an expression in a list. **NOT case sensitive**
index(source, target [, starting])	Integer	Returns the position of the target string within the source string. **Case sensitive**
replace(source-str, from-str, to-str)	String	Returns a string with specified substring replacements. **NOT case sensitive**
pagecount [*(filename)*]	String	Returns the count of pages. ***Warning: it's a string value.**
pageNumber	Integer	Current page number
pageNumber(filename)	String	Current page number for a file. ***Warning: it's a string value.**
equal(string1, string2)	Boolean	Not case sensitive compare of two strings

substring(source, position [, length]) **substr**	String	Extracts a substring from a source string (relative to 1)
iif(condition, valueIfTrue, valueIfFalse) **iif**(condition,'valueIfTrue/valueIfFalse')	String	If condition is true, returns the true value, else the false value. iif(true, 'yes', 'no') or iif(true, 'yes/no')
xprint(expression)	N/A	Insert **immediately** the expression in the output stream.
loadFromFile(expression)	String	Returns the content of the text file specified by *expression*
available(filename)	Boolean	True if a record is available for this file
firstof (filename)	Boolean	TRUE if the current record is the first iteration at this level
lastof (filename)	Boolean	TRUE if the current record is the last iteration at this level
Today	TDateTime	Date and time at the start of process (**now** is the current value)

6 TABLE OF ILLUSTRATIONS

Figure 1 - ".vpSchema" file 10
Figure 2 - exportSchema 11
Figure 3 – XML Data file 13
Figure 4 – Creating an XML file 14
Figure 5 - SQL-Server XML data file 18
Figure 6 – General schema 26
Figure 7 - RUN vpReport 28
Figure 8 - C# RUN vpReport 29
Figure 9 - VB RUN vpReport 30
Figure 10 – DELPHI RUN vpReport 31
Figure 11 - Calling vpxPrint 32
Figure 12 - Active-X 32
Figure 13 - vpDesigner 33
Figure 14 – Mail Interface 34
Figure 15 – PDF Attributes 34
Figure 16 - Language and theme of the interface 35
Figure 17 – Objects toolbar 36
Figure 18 – Data schema 36
Figure 19 – Adding a file or a field 36
Figure 20 – Drag-n-Drop fields 37
Figure 21 - Types of pages 39
Figure 22 - Managing pages 39
Figure 23 - Page tabs 40
Figure 24 – Page properties 40
Figure 25 – Background properties 40
Figure 26 - Page properties - Example 41
Figure 27 - Object Colors 42
Figure 28 – Object properties 43
Figure 29 – Array Object 44
Figure 30 – Array Object – Properties 45
Figure 31 - Association file to a table object 46
Figure 32 – Delete Master file association 46
Figure 33 – Adding an object in an array 46
Figure 34 - Cell properties of a table 47
Figure 35 - Table Object - Lines 50
Figure 36 - Table Object - Columns 50
Figure 37 - Properties of the detail lines 51
Figure 38 – Charts and Pies 55
Figure 39 – Global Variables 62
Figure 40 – Array object programming 63
Figure 41 - Labels - Properties 69
Figure 42 – Master file 72
Figure 43 – The cycle 73

Figure 44 – Example of data formats 74

7 INDEX

A

Array object
Accum 61
Formatting 45
Array Object
Moving lines 45
Array Object
Column width 45
Moving columns 45
Properties 42
Array Object
Line height 45
Array Object
Footer size 46
Array Object
Header size 46
Array Object
Detail size 46
Array Object
Design mode 46
Array Object
Detail lines 47
Array Object
Columns 47
Array Object
Line properties 48
Array Object
Programming 61

D

Delphi 28
Detail table 68

F

FOR EACH 12
Format
currency 70
decimal 70
integer 70
string 70
TDateTime 70
Formats 70

L

Labels 65
Language 32
Layers 35
Lines
Overlay lines 59

M

Mail 31
Main screen 30

O

Objects 33
Array Object 41
Basic objects 39
Content Objects 40
Labels 65

P

Page Types 35
Pages
1. Background 35
2. Cover 35
3. Before 36
4. Main 36
5. After 36
6. Summary 36
Associating 37
Managing pages 36
Properties 37
Visibility 37
Parameters 14
PDF 31
Processing
Cycle 69
Programming
Array cells 61
Expressions 71
Functions 71

S

Schema 9
SQL-Server 15

T

Tables
1. Data conversion functions 71
2. File functions 71
3. Specific functions 71
Theme 32

V

VB 26
vpRunReport 23, 25

X

XML Data 11
XML_Data.w 12