## VPrint application – Developer’s Guide

1. Presentation  
   VPrint is a Windows Forms application. It requires .NET 4 installed.
2. Installation  
   Install the application by simple copy in folder operation.
   1. Create folder ‘*c:\VPrint’*
   2. Copy the excitable, configuration file and all referenced files in that directory
   3. Right click VPrint executable and choose ‘Send to’->’Desktop (create shortcut)’
   4. Double click the shortcut to run it.
3. Uninstalling  
   Stop application if running. Delete VPrint folder.
4. Servicing
   1. What is a layout template?   
      Layout template is different location of data fields and text in printed paper
   2. How is layout template saved? Where are the templates saved?  
      VPrint uses an xml to save a template. It saves temples in database. During print it creates a call to central server to read a template for retailer. There also is a template in application folder. VPrint uses it in case it can’t find template for retailer.
   3. How template works?  
      Template is a serialized VPrint class caring data and C# executable code. Before printing VPrint deserializes and builds the class. VPrint uses MVC runtime compiling technology to build the class. Currently it supports 9 printing C# types. During binging C# code of temple calls TRS services and calculates values of fields.  
      For each printed voucher it binds a class instance with data and calls corresponding printing architecture to generate printer output. Depending on class type VPrint can call different printer technologies and print on rather different printers. Currently VPrint supports direct printing for IBM ProPrinter driver (good for line printers), MTPL printing commands for Tally printers, and graphics printing.
   4. How many templates does VPrint support?  
      Numbers of templates are not limited. Currently VPrint can link each retailer to its own template. There are 50 different templates already developed.
   5. How many fields can a template have?  
      Numbers of fields are not limited. Neither the number nor types of barcodes. The future version of VPrint will support image objects as well.
   6. How to develop a template?  
      In order to develop a template developer needs to be an administrator of TRS. In VPrint they right-click and choose ‘Crete/Edit template’ context menu option. VPrint shows editor form. User select country. They browse all the templates available for the country. User creates or edits a template. They click save. VPrint builds template to verify its validity, serializes it to xml and call TRS remote service to insert it in database. Now template is linked to a country. There is a configuration screen in TRS that allows a retailer to be linked to a template. VPrint on client side sees new template instantly and starts using it.
   7. How does template look like? What does developer need to know about templates?  
      There are some samples of real temples below. They all use different printing technics and are in production now.
      1. Sample RazorX template. Most complex, powerful and in used.

<?xml version="1.0" encoding="utf-8"?>

<!-- format 1 - FORMAT SPAIN (TESTED)-->

<VoucherPrintLayoutRazX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<DocumentInitialization />

<FormLength>54</FormLength>

<PageSize>

<Width>1100</Width>

<Height>852</Height>

<!-- Was 750 -->

</PageSize>

<Landscape>false</Landscape>

<!-- Name of the template. Should be unique for the system. -->

<TemplateName>D5DF5A25-D3E5-41B8-8555-93C85FD25A3D</TemplateName>

<Context>

<![CDATA[

@{

//Please no code into the body tag!

//Only variables

//Voucher

var voucherNumber = Model.VoucherNo + Model.Printing.CalculateCheckDigit(Model.VoucherNo);

var barcodeNumber = Model.StrVoucherNo.Replace(" ", "").Substring(3);

var barcodeText = Model.StrVoucherNo.Replace(" ", "");

//Retailer

var id = Model.Retailer.Id;

var retailerName = Model.Retailer.TradingName.EscapeXml();

var rLine1 = Model.Retailer.RetailAddress.Line1;

var rLine2 = Model.Retailer.RetailAddress.Line2;

var rLine3 = Model.Retailer.RetailAddress.Line3;

var rLine5 = Model.Retailer.RetailAddress.Line5;

var retailerAddress = string.Concat(rLine1 , '\n', rLine2 , '\n', rLine5, '-', rLine3).EscapeXml();

var retailerPhone = Model.Retailer.Phone.EscapeXml();

var officeData = Model.Manager.RetrieveTableData("ho\_pfs, ho\_Certificate\_1, ho\_Certificate\_2, ho\_Certificate\_3, ho\_category\_title,ho\_add\_id", "HeadOffice",

"where ho\_id={0} and ho\_iso\_id={1}".format(Model.Retailer.HeadOfficeId, Model.Office.CountryId));

var branchData = Model.Manager.RetrieveTableData("br\_category, br\_pfs", "Branch",

"where br\_id={0} and br\_iso\_id={1}".format(Model.Retailer.Id, Model.Retailer.CountryId));

var ho = "HO: " + Model.Retailer.HeadOfficeId;

//Office

var officeName = Model.Retailer.HeadOfficeName.EscapeXml();

var hoData = Model.Manager.RetrieveTableData("hoa\_add\_1,hoa\_add\_2,hoa\_add\_3,hoa\_add\_4,hoa\_add\_5,hoa\_add\_6", "HeadOfficeAddress",

"where hoa\_id = {0} ".format(officeData[5]));

var oLine1 = hoData[0];

var oLine2 = hoData[1];

var oLine3 = hoData[2];

var oLine5 = hoData[4];

var officeAddress = string.Concat(oLine1 , '\n', oLine2 , '\n', oLine5, '-', oLine3 ).EscapeXml();

var vatNumber = string.Concat("", Model.Retailer.VatNumber);

}

<?xml version="1.0" encoding="utf-8"?>

<VoucherPrintRazX xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<!-- -->

<Lines>

<!-- VoucherNumber - VoucherNo -->

<GPrintLine>

<X>360</X>

<Y>60</Y>

<Size>0</Size>

<Text>@voucherNumber</Text>

<Font>

<Value>Arial, 12pt</Value>

</Font>

</GPrintLine>

<!-- RetailerID - StoreNo -->

<GPrintLine>

<X>360</X>

<Y>109</Y>

<Size>0</Size>

<Text>@id</Text>

<Font>

<Value>Arial, 12pt</Value>

</Font>

</GPrintLine>

<!-- RetailerName - RetailerName -->

<GPrintLine>

<X>100</X>

<Y>142</Y>

<Size>0</Size>

<Text>@retailerName</Text>

<Font>

<Value>Arial, 10pt</Value>

</Font>

</GPrintLine>

<!-- RetailerAddress - Address -->

<GPrintLine>

<X>100</X>

<Y>155</Y>

<Size>0</Size>

<Text>@retailerAddress</Text>

<Font>

<Value>Arial, 10pt</Value>

</Font>

</GPrintLine>

<!-- HeadOffice- HeadOffice-->

<GPrintLine>

<X>100</X>

<Y>60</Y>

<Size>0</Size>

<Text>@officeName</Text>

<Font>

<Value>Arial, 10pt</Value>

</Font>

</GPrintLine>

<!-- HeadOfficeAddress - Address -->

<GPrintLine>

<X>100</X>

<Y>73</Y>

<Size>0</Size>

<Text>@officeAddress</Text>

<Font>

<Value>Arial, 10pt</Value>

</Font>

</GPrintLine>

<!-- VatNumber - VatNo -->

<GPrintLine>

<X>100</X>

<Y>208</Y>

<Size>0</Size>

<Text>@vatNumber</Text>

<Font>

<Value>Arial, 10pt</Value>

</Font>

</GPrintLine>

<!-- HO -->

<GPrintLine>

<X>220</X>

<Y>208</Y>

<Size>0</Size>

<Text>@ho</Text>

<Font>

<Value>Arial, 12pt</Value>

</Font>

</GPrintLine>

</Lines>

<Barcodes>

<BarPrintLine>

<X>240</X>

<Y>777</Y>

<Text>@barcodeNumber</Text>

<Height>25</Height>

<Size>2</Size>

<BarText>

<X>0</X>

<Y>5</Y>

<Size>0</Size>

<Text>@barcodeText</Text>

<Font>

<Value>Arial, 11pt</Value>

</Font>

</BarText>

</BarPrintLine>

</Barcodes>

</VoucherPrintRazX>

]]>

</Context>

</VoucherPrintLayoutRazX>

* + 1. Pure Razor template. Simpler but still very efficient.

<?xml version="1.0" encoding="utf-8"?>

<!-- PORTUGAL -->

<VoucherPrintLayoutRaz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<DocumentInitialization />

<FormLength>49</FormLength>

<!-- Name of the template. Should be unique for the system. -->

<TemplateName>D5328596-E6F6-49A4-9ECB-3A4522CB95C4</TemplateName>

<Context>

<![CDATA[

@{

//Please no code into the body tag!

//Only variables

const string COL0 = "<ht><ht><ht><ht>";

const string COL1 = "<ht><ht><ht><ht><ht>";

const string COL2 = "<nbsp><nbsp><nbsp><nbsp>";

string ROW4 = Helper.Repeate(4, "<lf>");

string ROW30 = Helper.Repeate(30, "<lf>");

// var length = MTPL.SetFormLength(49);

var id = Model.Retailer.Id;

var barcodeNumber = Model.StrVoucherNo.Replace(" ", "").Substring(3);

var barcode = MTPL.SetAbsoluteHorizontalPosition(2300) +

MTPL.PrintI2Of5Barcode(barcodeNumber, 1, "000");

var barcodeText = Model.StrVoucherNo.Replace(" ", "");

var shopName = Model.Retailer.TradingName ?? Model.Retailer.Name;

var voucherNumber = string.Concat(Model.VoucherNo, Model.Printing.CalculateCheckDigit(Model.VoucherNo));

var line1 = Model.Retailer.RetailAddress.Line1;

var line2 = Model.Retailer.RetailAddress.Line2;

var line3 = Model.Retailer.RetailAddress.Line3;

var line5 = Model.Retailer.RetailAddress.Line5;

var phone = Model.Retailer.Phone;

var vatNumber = string.Concat(" ", Model.Retailer.VatNumber);

}

<body>

@ROW4

@COL2@string.Format("{0,-25}",shopName)<ht>@voucherNumber<br>

@COL2@line1<br>

@COL2@line2<br>

@COL2@string.Format("{0,-25}",line3)<ht>@id<br>

@COL2@line5<br>

<br>

@COL2@vatNumber<br>

@ROW30

@COL0@barcode

@COL1@barcodeText

</body>

]]>

</Context>

</VoucherPrintLayoutRaz>

* + 1. Out of date France template. No longer in use.

<?xml version="1.0"?>

<!-- FORMAT 1 (tested)-->

<VoucherPrintLayout250 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<DocumentInitialization />

<Context>

<![CDATA[<br><br><br>

<ht>[RetailerID]<ht><nbsp><nbsp>[ShopName]<br><br>

<ht><ht><nbsp><nbsp>[Line1,-26]<nbsp>[VoucherID]<br>

<ht><ht><nbsp><nbsp>[Line2,-26]<br>

<ht><ht><nbsp><nbsp>[Line5]<nbsp>[Line3]<br>

<ht><ht><nbsp><nbsp>Tel:<nbsp>[Phone]<br><br>

<ht><ht><ht>[VATNumber]<ht><ht><ht><ht>PROCEDURE<nbsp>DE<nbsp>SECOURS<br>

<br><br>

<ht><ht><nbsp><nbsp><nbsp>\*<nbsp>[ShopName]<nbsp>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*<br>

<ht><ht><nbsp><nbsp><nbsp>\*<nbsp>[Line1]<nbsp>[Line5]<nbsp>[Line3]<nbsp>\*<br>

<ht><ht><nbsp><nbsp><nbsp>\*<nbsp>Motif<nbsp>d'utilisation<nbsp>de<nbsp>la<nbsp>procedure<nbsp>de<nbsp>secours<nbsp>:<nbsp>...............]]>

</Context>

<FormLength>54</FormLength>

<RetailerID>

<Description>RetailerID</Description>

</RetailerID>

<Line1>

<Description>Line1</Description>

</Line1>

<Line5>

<Description>Line5</Description>

</Line5>

<Line3>

<Description>Line3</Description>

</Line3>

<Phone>

<Description>Phone</Description>

</Phone>

<VoucherID>

<Description>VoucherID</Description>

</VoucherID>

<VATNumber>

<Description>VATNumber</Description>

</VATNumber>

<ShopName>

<Description>ShopName</Description>

</ShopName>

</VoucherPrintLayout250>

* + 1. Out of date UK template. Very simple

<?xml version="1.0"?>

<!-- UK -->

<VoucherPrintLayout826 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<DocumentInitialization />

<FormLength>6120</FormLength>

<BarCodeNo>

<Description>BarCodeNo</Description>

<X>4032</X>

<Y>1</Y>

<Size>1</Size>

<XYZ>000</XYZ>

<Compressed>false</Compressed>

</BarCodeNo>

<BarCodeText>

<Description>BarCodeText</Description>

<X>4304</X>

<Y>221</Y>

<Size>0</Size>

</BarCodeText>

<ShopName>

<Description>ShopName</Description>

<X>4002</X>

<Y>2901</Y>

<Size>0</Size>

</ShopName>

<VoucherID>

<Description>VoucherID</Description>

<X>5845</X>

<Y>2950</Y>

<Size>0</Size>

</VoucherID>

<ShopNo>

<Description>ShopNo</Description>

<X>5845</X>

<Y>3207</Y>

<Size>0</Size>

</ShopNo>

<Line0>

<Description>Line0</Description>

<X>4002</X>

<Y>3001</Y>

<Size>0</Size>

</Line0>

<Line1>

<Description>Line1</Description>

<X>4002</X>

<Y>3103</Y>

<Size>0</Size>

</Line1>

<Line2>

<Description>Line2</Description>

<X>4002</X>

<Y>3204</Y>

<Size>0</Size>

</Line2>

<Line3>

<Description>Line3</Description>

<X>4002</X>

<Y>3306</Y>

<Size>0</Size>

</Line3>

<VATNumber>

<Description>VAT Number</Description>

<X>4002</X>

<Y>3506</Y>

<Size>0</Size>

</VATNumber>

</VoucherPrintLayout826>

* 1. How to debug my template?

There is a test project with VPrint application. To debug a temple do following.

1. Create a test method in project
2. Create a test ‘Test’ event method handler and add it Test event of Printer class.
3. Paste the code of template into the handler.
4. Configure voucher printer class and run debugging.  
   See example below

[TestMethod]

public void italy\_print\_format\_Type\_1()

{

VoucherPrinter printer = new VoucherPrinter();

printer.UseLocalFormat = false;

printer.UseLocalPrinter = false;

printer.PrintOnce = true;

printer.m\_PrinterName = Printers.DELL;

printer.m\_ReportType2 = "VPrinting.Documents.VoucherPrintLayoutRaz";

printer.m\_PrinterXmlFilePath = @"C:\PROJECTS\VPrint\XmlConfigurations\print380\_Type1\_Raz.xml";

printer.Test += new EventHandler(printer\_Test);

printer.SimulatePrint = true;

printer.PrintAllocation(589331, false);//384920//211771

}

private void printer\_Test(object sender, EventArgs e)

{

VoucherPrinter Model = (VoucherPrinter)sender;

var id = Model.Retailer.Id;

var barcodeNumber = Model.StrVoucherNo.Replace(" ", "").Substring(3);

var barcodeText = Model.StrVoucherNo.Replace(" ", "");

//RETAILER

var shopName = Model.Retailer.Name;

var voucher = Model.VoucherNo;

var checkDig = Model.Printing.CalculateCheckDigit(Model.VoucherNo);

var voucherNumber = string.Concat(Model.VoucherNo, checkDig);

var line1 = Model.Retailer.RetailAddress.Line1;

var line2 = Model.Retailer.RetailAddress.Line2;

var line3 = Model.Retailer.RetailAddress.Line3;

var line5 = Model.Retailer.RetailAddress.Line5;

var officeData = Model.Manager.RetrieveTableData("ho\_pfs, ho\_Certificate\_1, ho\_Certificate\_2, ho\_Certificate\_3, ho\_category\_title,ho\_add\_id", "HeadOffice",

"where ho\_id={0} and ho\_iso\_id={1}".format(Model.Retailer.HeadOfficeId, Model.Retailer.CountryId));

var branchData = Model.Manager.RetrieveTableData("br\_category, br\_pfs", "Branch",

"where br\_id={0} and br\_iso\_id={1}".format(Model.Retailer.Id, Model.Retailer.CountryId));

//OFFICE

var officeName = Convert.ToString( Model.Manager.RetrieveTableData("ho\_trading\_name", "HeadOffice",

"where ho\_id={0} and ho\_iso\_id={1}".format(Model.Retailer.HeadOfficeId, Model.Retailer.CountryId)).FirstOrDefault());

var hoData = Model.Manager.RetrieveTableData("hoa\_add\_1,hoa\_add\_2,hoa\_add\_3,hoa\_add\_4,hoa\_add\_5,hoa\_add\_6", "HeadOfficeAddress",

"where hoa\_id = {0} ".format(officeData.Length > 5 ? officeData[5] : "0"));

var oLine1 = Convert.ToString(hoData.Length > 0 ? hoData[0] : "");

var oLine2 = Convert.ToString(hoData.Length > 1 ? hoData[1] : "");

var oLine3 = Convert.ToString(hoData.Length > 2 ? hoData[2] : "");

var oLine5 = Convert.ToString(hoData.Length > 3 ? hoData[4] : "");

//CERTIFICATES

var certificate1 = Convert.ToString(officeData.Length > 1 ? officeData[1] : "");

var certificate2 = Convert.ToString(officeData.Length > 2 ? officeData[2] : "");

var certificate3 = Convert.ToString(officeData.Length > 3 ? officeData[3] : "");

var category = Convert.ToString(officeData.Length > 4 ? officeData[4] : "");

var vatNumber = Model.Retailer.VatNumber;

Assert.AreEqual(id, 123456);

Assert.IsNotNull(Model.Retailer);

Assert.IsTrue(Model.Retailer.HeadOfficeId != 0);

Assert.IsTrue(Model.Retailer.CountryId != 0);

Assert.IsNotNull(oLine1);

Assert.IsNotNull(oLine2);

Assert.IsTrue(!string.IsNullOrWhiteSpace(certificate1));

Assert.IsTrue(!string.IsNullOrWhiteSpace(officeName));

Assert.IsTrue(!string.IsNullOrWhiteSpace(vatNumber));

}

* 1. What developers can test?
     1. They can build a template
     2. They can test one from file
     3. They can connect live services and test live template
     4. They can print live temple on local printer
     5. They can print locally developed template on client printer
     6. They can test live data on local printer
     7. They can reproduce live errors in VS development environment
  2. Which namespaces a developer can use to develop a template?  
     Number of namespaces is intentionally restricted up to the following. Any code that resigns out of those namespaces will not be compiled by the compiler.

System,

System.Data,

System.Collections,

System.Collections.Generic,

System.Linq,

System.Text,

System.Text.RegularExpressions,

System.Xml.Serialization,

VPrinting,

VPrinting.Tools,

VPrinting.Common,

VPrinting.Documents