VisualXSL User Guide

RenderX ®

www.renderx.com

Table of Contents

List of Figures

Copyrights, Trademarks, Registered Trademarks and Acknowledgements

RenderX is a registered trademark of RenderX Inc.

RenderX VisualXSL contains the following third-party software products:

- Barcode components licensed from IDAutomation.com, Inc.
- PDF2Image SDK libraries licensed from VeryPDF, Inc.

These products may be used only as part of and in connection with RenderX XEP and VisualXSL.

RenderX VisualXSL makes use of DockPanel Suite developed by Weifen Luo and licensed under the MIT License.

Certain components are based on part of the reusable wizard component developed by Steven M. Soloff.

iTextSharp is a C# port of the Java-based iText Library. iText is Copyright © 1999-2008 by Bruno Lowagie and Paulo Soares. All Rights Reserved. Read more about the iTextSharp library here: www.itextsharp.sourceforge.net (or www.sourceforge.net/projects/itextsharp/).

Adobe is a trademark and Acrobat Reader is a registered trademark of Adobe Systems, Inc.

Microsoft, .NET Framework, Microsoft Office, Microsoft Word, Microsoft Excel, Microsoft SQL Server 2000, and Windows Vista are registered trademarks of Microsoft Corporation.

Chapter 1. Introduction

This manual describes how to work with the Visual XSL application.

1.1. Overview

VisualXSL is a GUI-based designer for writing eXtensible Stylesheet Language (XSL) stylesheets for use with RenderX *XEP* or *XEPWin*, and *VDPMill*. The product supports creating static overlay forms and flowing documents with variable page layouts based on position such as first, any, last and only. In fact, static and flowing documents can be mixed together into a complete XSL stylesheet project.

VisualXSL allows you to import a PDF or image as a background source or even start with a blank canvas. When using a background you can choose to render the form as part of the output (overlay) or render without the background for imaging on pre-printed stock. Simply associate a sample *XML* file with the project and then drag and drop XML data to design the document directly within the *VisualXSL* layout designer. Common properties like colors, fonts, borders and margins can easily be set with simple-to-use property sheets.

VisualXSL has wizards for many different types of document objects. Text fields, paragraphs with mixed static and dynamic content, tables, barcodes, checkboxes and variable images can all be created in the document layout from the *XML* source. Custom instructions, conditional rendering, date and number formats, and string conversions are all easy to add. *VisualXSL* is capable creating fillable pdf forms, including support of JavaScript.

The *VisualXSL* GUI supports the use of docking windows. You can dock and undock to create an optimal work configuration, with design in one window and preview in another. The design area supports pixel perfect placement to 1/100th of a point.

VisualXSL's architecture and functionality provides for a significant boost in speed of document production when compared to the traditional *XSL-FO* documents created with traditionally made *XSL* stylesheets.

Other features include:

- **Easy-To-Use Layout Designer**

  You design a document visually using *VisualXSL*. The application enables to put data fields onto the document layout. You can drag and drop data directly onto the layout designer.

  The **layout designer** (also called the **layout** or the **designer area**) has a simple view, and is quickly and easily learned.

- **High-Speed Document Generation**

  VisualXSL generates a specific stylesheet for designed document rendering. With this stylesheet, the *XEP* formatter provides high speed document generation for PDF (or PostScript).

- **Single- and Multi-Page Forms Support**

  VisualXSL supports Single- and Multi-Page document creation.

  If you set a multi-page PDF as a background, the application extracts every page of the background to the corresponding page of the resulting PDF. New pages are automatically added if needed.

- **Static and Dynamic flows**

  VisualXSL supports a mixture of static and dynamic templates.

- **Text Frames, Pictures, Barcodes, Paragraphs, Tables, Address blocks, Page numbers, Buttons, Check Boxes, List Boxes, and Combo Boxes.**

  The following types of data fields can be placed into a form:

  - Text Frames are rendered as text from the XML source file.
  - Check boxes are rendered as check marks (text symbol); their appearance (in the final document) depends on the XPath expression result.
  - Pictures are rendered according to the size set on the layout.
  - Barcodes are rendered using RenderX stylesheets and the data from the XML source file. The barcodes can be one of the following types:
    - **`EAN8`**
    - **`EAN13`**
    - **`UPC-A`**
    - **`UPC-E`**
    - **`Interleaved 2 of 5`**
    - **`Code 39`**
    - **`Codabar`**
    - **`Code 128`**
    - **`Australia Post 4-State`**
  - Paragraph blocks can be inserted and edited. They can be formed via XPath expressions, as well as from static text. You can also use different formatting for block text parts.
  - Tables can be inserted and edited. They can be formed via XPath expressions, as well as from static text (for certain cells). You can also use different formatting for different cells.
  - Address Blocks Wizard for adding address labels.The resulting label is aligned vertically to the bottom of the label and consists of four or five address lines. There are three areas in the wizard dialog.
  - Page Numbers are Paragraphs, which contain special texts, to show current page number in the document.
  - Buttons can be two types: Submit and Reset. The button's type is set via Button Type property. By default it is set Submit. All buttons have some common properties.
  - List Boxes and Combo Boxes are tools, which allows the user to select one or more values from the given list.
    - **`Multiselect`** - indicates, if multiple values can be selected
    - **`Values`** - list of selectable values, delimited by the given Separator. By default its value is the inner text of the XML node.
    - **`Separator`**- a symbol, indicating the separator for Values field by default its value is **|**. Also, ComboBox has one more property:
    - **`Editable`** - indicates if a value can be typed, or the value can only be selected from the given list

ℹ Additional formatting can be applied to fields containing data values (in ODBC or ISO8601 format).

- **Conditional Rendering**

  Every data field has properties which can be set to determine its rendering under certain conditions.

- **XPath Validation**

  Almost all XPath expressions are created automatically when you drag and drop XML tree nodes onto the layout designer. You can also manually create or edit XPath expressions. The application validates all XPath expressions and offers to modify incorrect expressions.

- **Docking Windows**

  Docking-window interface is supported. You can dock and undock tool windows to create an optimal work configuration. For example, on a dual-monitor system, some tool windows can be undocked and placed on the second monitor.

- **Print XML Data on a Preprinted Form**

  Printing XML data over paper (preprinted) forms is supported. For more information, see section 6.4 How to Fill in Preprinted Forms with Data.

- **Generation of Fillable PDF Forms**

  The application can generate fillable PDF forms. For more information see section 5.14 PDF Forms.

- **EXSLT Support**

  The application is capable using EXSLT features provided with Mvp.Xml EXSLT.NET module (http://mvp-xml.sourceforge.net/exslt/). This module is not included in VisualXSL distribution and has to be installed separately.

1.2. Intended Audience and User Prerequisites

This manual is intended for personnel who need to generate PDF or PostScript documents from XML files.

To obtain maximum benefit from this manual and from the *VisualXSL* application, the user should have a basic understanding of XML, XSL, and XPath.

1.3. Document Content

This document contains the following:

- Chapter 1: Introduction to the manual
- Chapter 2: Lists what's new in version 2.2 and related issues
- Chapter 3: Describes the system requirements for VisualXSL and how to install it
- Chapter 4: Describes how to get started using VisualXSL
- Chapter 5: Describes data fields and properties
- Chapter 6: Describes how to perform various tasks in VisualXSL
- Chapter 7: Provides examples of how to use VisualXSL
- Glossary

- Index

1.4. Related Publications and References

- W3C XSL Transformation 1.0 Recommendation
- W3C XML Path Language 1.0 Recommendation
- Microsoft .NET Framework 2.0
- Adobe™ Acrobat® Reader®
- RenderX XEP On-line Reference

1.5. Support

For all support-related needs, please go to www.renderx.com/support/index.html.

Chapter 2. What's New

2.1. What's New in Version 2.4.2

2.1.1. Fixed Issues

- Tables have no more border by default;
- Fixed a problem with missing visual styles of checkboxes and radio buttons if they are used as Form fields;
- Fixed wrong reference to base TranslateProject;
- Fixed application crash when loading large documents;
- Performance improved for loading and saving large documents;
- Fixed application crash when loading large documents;

2.2. What's New in Version 2.4.1

2.2.1. New Features

- Added support of XEPWin 3.0;

2.3. What's New in Version 2.4

2.3.1. New Features

- Added support for Mvp.Xml EXSLT.NET module (http://mvp-xml.sourceforge.net/exslt/), providing with EXSLT support;
- Added support of document() XSLT function;

2.3.2. Fixed Issues

- Resolved a problem with image Width attribute setting in dynamic layout;
- Fixed several minor UI-related and logic problems;

2.4. What's New in Version 2.3.1

2.4.1. Fixed Issues

- Fixed severe problems with JavaScript for radiobuttons and action buttons in PDF forms;
- Further improvements and bugfixes over Undo;
- Fixed a display problem after changing frame orientation under certain conditions;
- Fixed a problem of Z-Order property when multiple blocks are selected;
- Fixed an issue with incorrectly relocated Table and Address frames;

- Fixed an exception if browser preview window is no longer accessible under certain conditions;
- Fixed several issues with multiple inlines within paragraphs;

2.5. What's New in Version 2.3

2.5.1. New Features

- Significant improvements over fillable PDF Forms. Now they are built with RenderX XEP and support most features from XEP, including fields, submit buttons, checkboxes, radiobuttons, and more;
- Added feature of editing JavaScript event handlers for PDF Forms, associated with controls, pages, and documents;
- Added support for different output formats, depending on those available in XEP license;
- Major improvements over Paragraph editor (modifying and deleting fields, formatting and aligning the inlines);
- Address Blocks now can be edited like normal paragraphs and have custom layout and additional fields within;
- Improved Page Numbering widgets;
- Improved moving frames with mouse and keyboard (Ctrl and Alt handling);
- New samples and workflows in User Reference;

2.5.2. Fixed Issues

- Fixed several problems with Table builder that prevented toolbar buttons working correctly;
- Improved new and rotated items layout on a dynamic flow;
- Temporary folder leakage (lost files) eliminated;
- Several bugfixes with clipboard operations (Copy and Paste);
- Fixed issues with traces on the layout;
- Fixed critical error during Undo operation with paragraph and address blocks;
- Fixed layout problems for very small frames;
- Fixed an issue that prevented opening older versions of *.vxl files, and source files having CDATA elements;
- Fixed problems with alignment, content deletion, and scaling within the paragraph editor;
- Fixed several issues related to background image rendering;
- Fixed a problem when tooltips remained on screen after the application was deactivated;
- Improved logic of Inherit and Auto length values;
- Fixed problems with incorrect image height in dynamic layout;
- Minor bugfixes and code improvements;

2.6. What's New in Version 2.2

2.6.1. New Features

- Major improvements over Table Builder; it is no longer an extra window but a regular control;
- Improved visual appearance for text blocks in Design layout;
- Added more examples in User's Guide;
- Improved layout of the User's Guide (PDF and CHM versions);
- Improved navigation (modal dialogs, keyboard shortcuts, etc);

2.6.2. Fixed Issues

- Fixed issues with moving multiple objects;
- Fixed clipboard issues;
- Fixed issues with Undo/Redo;
- Fixed issues with Drag-and-Drop operations;
- Fixed several refresh issues with Property Grid;
- Improved navigation (modal dialogs, keyboard shortcuts, etc);
- Fixed update issues with installer;
- Fixed several problems with popup menus;

2.7. What's New in Version 2.1

2.7.1. New Features

- **Preferences** form is now a dockable pane.
- **Preview** is no longer shown when opening a previous project.
- Added transparency control for brightness of hatch that is displayed for background of a data field.
- Concept of 'add page' has been replaced by that of 'add section'.
- Added support for mixed static and dynamic sections.
- Added support for XEP intermediate format as an image.
- Added support for SVG graphics as an image.
- 100% project translation customization, vastly extending the solutions possible with *VisualXSL*. For details, contact Support.
- For the tech-savvy user, added custom attribute feature. There are three new properties, under the new **Custom** Properties group : CustomAttribute, Overridden Attribute, and CustomProcess. Using them, it is possible to add custom attributes (that is, attributes not yet natively supported by *VisualXSL*) to any object type.
- **New Document Wizard** no longer appears at start up.
- Other minor improvements have been added.

2.7.2. Fixed Issues

- Fixed issue of application error when set to load last project on start up and that project is missing.
- Other minor issues have been fixed.

2.8. Limitations

- All project files, including the source XML file, should be located in the same folder.
- To use the print of an address label capability with a PostNet barcode, you have to manually add a font data XML node to the configuration file (`XEP.xml`).
- The PDF file name used as the background for the form layout must differ from the project name.
- There is no validation for barcode values specified by XPath expressions. You should check manually whether the value is acceptable for a barcode. (XPath is validated, but there is no guarantee that the expression result will satisfy the barcode specification. For example, for UPC-A, the result of an expression should be a 12-digit number.)
- A barcode image in the designer area is just a placeholder used to help locate the barcode data field. Only its height and location are taken into consideration. The width of the placeholder is ignored.
- Currently, preview for PostScript files is not supported. Therefore, you cannot see the generated document if you choose the PostScript format. You can find the PostScript file by using the specified path.
- The application considers the first node of the XML-file (source data) to represent its schema and to correspond to the record of the table described by this XML file. Note that this may cause problems with complex XML files whose first record structure is different from other nodes.
- Any operation that directly or indirectly changes the section quantity cannot be undone.
- All file paths to resources outside the project folder are stored as full paths. Be sure that all paths used in project are correct. Wherever possible, place all resources inside the project folder.
- If you are working with several projects and a dialog window is opened for one of them, you cannot switch to another project. For example, when the **Options** window for a project is open, it is not possible to work with any other project.
- An address block formed with the wizard and containing PostNet barcode cannot be edited.
- Fillable forms are only applicable to PDF output format and will not work for other formats.
- The *VisualXSL* application works with non-encrypted PDF only.

2.9. Known Issues

- The list of known issues can be found in the Release Notes. See in ReadMe.txt file that is included in distribution package's Known Issues section.

Chapter 3. System Requirements and Installation

3.1. Hardware/Software Requirements

The *VisualXSL* application requires the following hardware: 512 MB of RAM and 512 MB of free disk space.

Before installing the *VisualXSL* application, the following list of software must be installed:

- Microsoft Windows 2000 SP4, Windows XP SP2, Vista or Windows 7,
- Microsoft .NET Framework 2.0 or later
- *RenderX XEP* formatter 4.17 or later
- *RenderX XEPWin 2.24* (part of the *VisualXSL* installation package; it is automatically installed)
- Adobe Acrobat Reader 7.x. or later(or any web browser compatible PDF viewer, e.g.Foxit).
- Microsoft Internet Explorer 5.5 or later

Recommended.

- MS Office.
- Some third party XML editor (for example Renderx Docbench, oXygen XML editor).
- Any PostScript viewer (fore example Ghostscript v.8.64 and Ghostview v.4.9 , to have the ability to view prepared PostScript documents)
- Windows AFP Viewer Plug-In (to have the ability to view prepared AFP documents)
- Adobe SVG Viewer (to have the ability to view prepared SVG documents)
- Microsoft XPS Viewer(to have the ability to view XPS documents, Microsoft .NET Framework 3.0 is required)
- Mvp.Xml EXSLT.NET module for EXSLT support;

3.2. Product Package Folders

The installation folder (under `C:\Program Files\RenderX\VisualXSL`, if the application's installation default folder is used) contains the following subfolders:

- XSL - Contains the XSL stylesheets used internally by the application; it is not to be used separately from application.
- `Etc` - Contains some data used internally.
- `Doc` - Contains the User Manual.

When the *VisualXSL* application is installed, the following new folder is created:

- `..\All Users\Shared Documents\VisualXSL Projects` – shared documents folder for all users. If the application's installation default folder is used, the `All Users` folder is under `C:\Documents and Settings`.

  For Windows Vista and Windows 7 systems, the shared documents folder for all users is the following : `..\Users\Public\Public Documents\VisualXSL Projects`

3.3. How to Install VisualXSL

The following procedure describes how to install *VisualXSL*.

If the VisualXSL application is already installed it is recommended to uninstall it first. See 3.3.1. How to Uninstall VisualXSL.

- Open the Control Panel
- Add or Remove Programs, or Programs and Features
- Find the *VisualXSL* application in the list of installed applications.
- Click to Remove or Uninstall it.

Install a new version by executing the setup.exe and follow the wizard instructions. Note, that if you don't have . NET Framework installed, wizard will try to download and install it from the Internet. If you fail to set the correct path to license.xml, you can copy this file to the application installation folder (by default: C:\Program Files\RenderX\VisualXSL on 32-bit Windows or C:\Program Files (x86)\RenderX\VisualXSL on 64-bit Windows).

- To enable TrueType Windows fonts the xep.xml (in XEPWin folder, for example C:\Program Files\RenderX\XEPWin \xep\xep.xml on 32-bit Windows or C:\Program Files (x86)\RenderX\XEPWin \xep\xep.xml on 64-bit Windows) file should be modified in the following way:

- Make a backup copy of your xep.xml.
- Open file in an XML Editor or WordPad, not NotePad, and find a string:
- Click to Remove or Uninstall it.

Sample configuration for Windows TrueType fonts.

```
<!--
<font-group xml:base="file:/C:/Windows/Fonts/" label="Windows TrueType"
  embed="true" subset="true">
  <font-family name="Arial">
    <font><font-data ttf="arial.ttf"/></font>
    <font style="oblique"><font-data ttf="ariali.ttf"/></font>
    <font weight="bold"><font-data ttf="arialbd.ttf"/></font>
    <font weight="bold" style="oblique"><font-data ttf="arialbi.ttf"/></font>
  </font-family>
...
</font-group>
-->
```

After modification it should look like:

Sample configuration for Windows TrueType fonts.

```
<font-group xml:base="file:/C:/Windows/Fonts/" label="Windows TrueType"
  embed="true" subset="true">
  <font-family name="Arial">
    <font><font-data ttf="arial.ttf"/></font>
    <font style="oblique"><font-data ttf="ariali.ttf"/></font>
    <font weight="bold"><font-data ttf="arialbd.ttf"/></font>
    <font weight="bold" style="oblique"><font-data ttf="arialbi.ttf"/></font>
  </font-family>
...
</font-group>
```

See *VisualXSL* User Documentation for details, after the installation.

To uninstall *VisualXSL* :

- Open the Control Panel
- Add or Remove Programs (or Programs and Features for Vista and 7)
- Find the *VisualXSL* application in the list of installed applications.
- Click to Remove (or Uninstall) it.
- If you also wish to uninstall RenderX XEPWin, use the Control Panel to uninstall this application

Procedure 3.1. To install VisualXSL:

1. After downloading the `visualxsl.zip` file, unzip it to a temporary folder.

   After unzipping, the folder contains the `setup.exe`, `Readme.txt`, and `WhatsNew.txt` files.

2. It is recommended to review both text files.

3. Go to the temporary folder and double-click `setup.exe`.

   The Welcome screen for **InstallShield Wizard** for *VisualXSL* opens.

4. Click **Next**.

   The **VisualXSL License Agreement** dialog box opens.

5. Select the **I Accept** option, and then click **Next**.

   The **VisualXSL License** dialog box opens.

6. Browsing to the license.xml file, specify its location and then click **Next**.

   > During the installation, a copy of the `license.xml` file is placed in the application installation folder (by default, `C:\Program Files\RenderX\VisualXSL on 32-bit Windows or C:\Program Files(x86)\RenderX\VisualXSL on 64-bit Windows`).

   The **VisualXSL Customer Information** dialog box opens.

7. Fill in the **User Name** and **Organization** fields, select the **Install This Application For** option, and then click **Next**.

   The VisualXSL Setup Type dialog box opens.

8. Select the **Complete** setup type, and click **Next**.

   > The **Custom** setup is recommended only for experienced users.

   The VisualXSL Ready to Install the Program dialog box opens.

9. Click **Install**.

⚠ The installation starts.

The **Installing VisualXSL** screen is displayed while *VisualXSL* is being installed.

10. Click **Finish** to exit the **InstallShield Wizard**

    At the end of the *VisualXSL* installation, the *VisualXSL* icon is placed on the desktop.

ℹ The following steps apply to a **New** installation of XEPWin .

The **welcome** screen of **InstallShield Wizard** of XEPWin opens.

1. Click **Next**.

   The **XEPWin License Agreement** dialog box opens.

2. Select the **I Accept** option, and then click **Next**.

   The **XEPWin License** dialog box opens.

3. Browse to the `license.xml file` specify its location and then click **Next**. During the installation, a copy of the `license.xml` file is placed in the application installation folder (by default, `C:\Program Files\RenderX\XEPWin\xep on 32-bit Windows or C:\Program Files (x86)\RenderX\XEPWin\xep on 64-bit Windows`).

   The **XEPWin Customer Information** dialog box opens.

4. Fill in the **User Name** and **Organization** fields, select the **Install This Application For** option, and then click **Next**.

   The **XEPWin Setup Type** dialog box opens.

5. Select the **Complete** setup, and click **Next**.

   The **Custom** setup is recommended only for experienced users.

   The **XEPWin Ready to Install the Program** dialog box opens.

6. Click **Install**.

   ⚠ The installation starts.

   The **Installing XEPWin** screen is displayed while *XEPWin* is being installed.

7. Click **Finish** to exit the **InstallShield Wizard**

   At the end of the *XEPWin* installation, the **Hot Folder** icon is placed on the desktop and ActiXEP is placed on the taskbar.

ℹ The following steps apply to an **Existing** XEPWin Installation of XEPWin.

The **Welcome** screen of **InstallShield Wizard** of XEPWin opens.

ℹ To exit the **InstallShield Wizard**

1. Click **Cancel**.
2. Click **Yes** to confirm.
3. Click **Finish** to exit the **InstallShield Wizard**.

3.4. How to Uninstall VisualXSL

To uninstall *VisualXSL*, select **Start > Control Panel > Add or Remove Programs, or Programs and Features > VisualXSL > Remove or Uninstall**.

If *XEPWin* is being used by another RenderX product, it should **not** be removed. If *XEPWin* is removed, the system must be restarted for the changes in *XEPWin* to take effect.

Chapter 4. Getting Started

4.1. Opening the VisualXSL Application

Open the *VisualXSL* application by clicking its icon on the desktop.

The application starts, with no project selected.

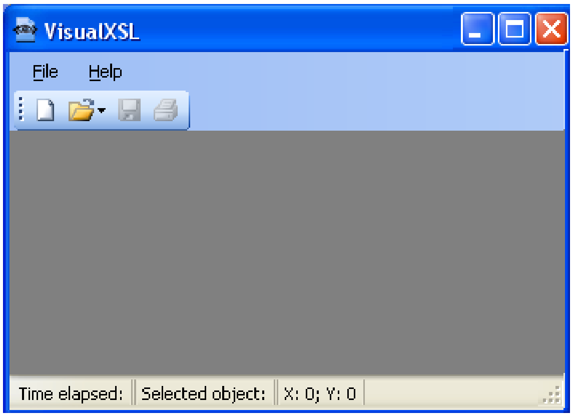


Figure 4.1. *VisualXSL* interface with no project opened

4.2. Opening an Existing VisualXSL Project

To open an existing project, do one of the following:

- Click **File | Open Project**, and select the project that you want to open.
- Click the `.VXL` file of the project that you want to open.
- Before closing *VisualXSL*, ensure that the **Open last project on start** toggle switch (located in the **Options** pane) is enabled (checked). When you launch *VisualXSL* next time, the last closed project will automatically be launched.

Sample projects distributed with *VisualXSL* are located in :

For Windows 2000 and XP in: `All Users\Shared Documents\VisualXSL Projects\Samples` folder.

For Windows Vista and 7 in: `Users\Public\Public Documents\VisualXSL Projects\ Samples` folder .

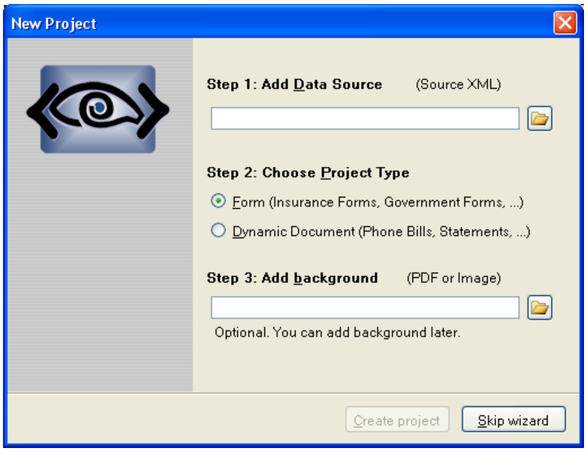
4.3. Creating a New VisualXSL Project

To create a new project:

- If you want to create a new project from an existing project, the following should be done before continuing with the **New Project** dialog box.
  - Create a new folder under `All Users\Document\VisualXSL Projects`.

- Place the data source (XML file) and, optionally, the background file (PDF or image) in this folder.

- Click **File | New** to open the **New Project** dialog box and add a data source (the source XML).



Step 1: Add Data Source (Source XML)

Step 2: Choose Project Type

● Form (Insurance Forms, Government Forms, ...)

○ Dynamic Document (Phone Bills, Statements, ...)

Step 3: Add background (PDF or Image)

Optional. You can add background later.

Figure 4.2. New Project dialog box

(More information about preparing an XML source and PDF background is presented in later chapters.)

- Use the **New Project** dialog box to choose the project type that is applicable to your needs: **static** or **dynamic**. Note that regardless of your choice, you can add both **dynamic** and **static** templates to the project.
- Optionally, use the **Add Background** field to add the background file (PDF or image). The background does not have to be added now; it can be added later.
- In the **New Project** dialog box, click the **Create Project** button. The folder and files to be used for the project are created, and the **Main Window** for *VisualXSL* opens.

Click **Skip wizard** button to create blank project.

Click **Cancel** button to cancel new page addition

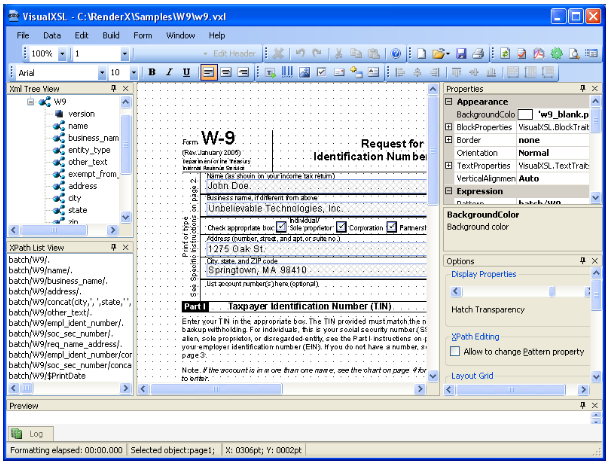
4.4. Main Window and Panes

The *VisualXSL* application has a docking style interface.

You can choose the windows from several predefined Main Window options from the **Window** menu: **All Windows**, **Optimized**, and **Lightweight**. Also using the **Window** menu, you can toggle the appearance of individual panes: **Preview**, **Log**, **Source XML (XML Tree View)**, **XPath List View**, **Properties**, and **Options**. The last window configuration used is saved when *VisualXSL* exits successfully and is restored on the next application startup.

Because the panes are dockable, they can be detached from their position in the *VisualXSL* Main Window by double-clicking them and then moved to elsewhere on the monitor (even outside the *VisualXSL* Main Window) or to a second monitor (if you are using a dual-monitor system).

The **All Windows** combination displays the following panes:



batch/W9/.
batch/W9/name/.
batch/W9/business_name/.
batch/W9/address/.
batch/W9/concat(city,', ',state,' ',
batch/W9/other_text/.
batch/W9/empl_ident_number/.
batch/W9/soc_sec_number/.
batch/W9/req_name_address/.
batch/W9/empl_ident_number/cor
batch/W9/soc_sec_number/conca
batch/W9/$PrintDate

Form **W-9**
(Rev. January 2005)
Department of the Treasury
Internal Revenue Service

Request for Identification Number

Print or type
See Specific Instructions on page 2.

Name (as shown on your income tax return)
John Doe

Business name, if different from above
Unbelievable Technologies, Inc.

Check appropriate box: ☑ Individual/ Sole proprietor ☑ Corporation ☑ Partnersh

Address (number, street, and apt. or suite no.)
1275 Oak St.

City, state, and ZIP code
Springtown, MA 98410

List account number(s) here (optional)

Part I Taxpayer Identification Number (TIN)

Enter your TIN in the appropriate box. The TIN provided must match the n backup withholding. For individuals, this is your social security number (SS alien, sole proprietor, or disregarded entity, see the Part I instructions on p your employer identification number (EIN). If you do not have a number, s page 3.

Note. *If the account is in more than one name, see the chart on page 4 for to enter.*

⊟ **Appearance**	
BackgroundColo	□ **'w9_blank.p**
⊞ BlockProperties	VisualXSL.BlockTrait
⊞ Border	**none**
Orientation	**Normal**
⊞ TextProperties	VisualXSL.TextTrait:
VerticalAlignmen	**Auto**
⊟ **Expression**	
Pattern	**batch/W9**

BackgroundColor
Background color

Display Properties
Hatch Transparency
XPath Editing
□ Allow to change Pattern property
Layout Grid

Formatting elapsed: 00:00.000 | Selected object:page1; | X: 0306pt; Y: 0002pt

Figure 4.3. VisualXSL interface with the W9 sample project opened

- **XML Tree View**, at the upper left, displays the XML schema.
- **XPath List View**, at the lower left, displays a list of the XPath expressions.
- **Layout (Designer Area)**, in the upper middle of the screen, is the primary working area, where all data fields are placed according to the background form (only in **static** sections).
- **Properties** pane, at the upper right, displays the properties of the selected data field object in the properties grid. (Sometimes this pane is referred to as “object properties”.) You can select several objects at once by clicking them with the **Ctrl** button pressed. Only the common properties of the selected objects are displayed. If the values of the properties are the same they are displayed with the current value, otherwise they are shown as blank.
- **Options** pane, at the lower right, shows miscellaneous options.
- **Preview** pane, immediately below **Layout**, shows the PDF document preview.
- **Log** pane, at the bottom of the screen, stores the formatter messages.

> Each of the panes above, except of **Layout**, has a **Pin** (📌) and a **Close** (×) button in its upper-right corner. You can pin dockable windows via **Pin** button. If a window is pinned, it will always stay on the screen, otherwise it will be hidden as soon as it is not focused.

4.5. Menus

Most of the basic commands are placed on the menus for quick access. Almost all of them are also available in the toolbars.

- **File menu**

  The **File** menu contains the following options:

  - **New (Ctrl+N)** - Opens a new project. Before opening a new project, you are prompted to save your changes to the current project.
  - **Open Project** (**Ctrl+O**) **-** Opens system's **open file** dialog, where you can select an existing project to open. Before opening a new project, you are prompted to save your changes to the current project.

- **Open Source XML** (**Ctrl+F**) - Opens a system's **open file** dialog where you can select and add an XML data source to the project.
- **Save** (**Ctrl+S**) - Saves the current project, with changes.
- **Save As** (**Ctrl+Shift+S**) - Saves the current project, with changes, under a new name.
- **Print** (**Ctrl+P**) - Prints the last generated PDF document. If you haven't created a resulting PDF or haven't previewed the result of XSLT, a warning message will be shown.
- **Recent Projects** - Displays a list of recent projects.
- **Exit** - Quits the *VisualXSL* application. You are prompted to save your changes.

- **Data Menu**

  The **Data** menu contains the following options:

  - **Add Text Frame** - Adds a text field to the layout, according to the XML source file. Current XML node's XPath expression is used to get the data from the XML source file. Note, that text frames can get static text as well as XPath.
  - **Add Barcode** - Adds a barcode field to the layout, according to the XML source file. Current XML node's XPath expression is used to get the data from the XML source file. Note, that barcodes can get static text as well as XPath (see detailed description in **Barcode** subchapter, in **Data Fields and Properties** chapter).
  - **Add CheckBox** - Adds a checkbox to the layout. The item's **Test** property (under **Expression** group) can be specified to render the checkbox under only certain conditions. It must have the correct XPath expression in the context of the node set, as determined by the expression from **Pattern** property (see detailed description in **Checkboxes** subchapter, in **Data Fields and Properties** chapter).
  - **Add Image** - Adds an empty image field to the layout. **Note:** You must correct **ImageUrl** to generate images from the XML content. For example, to use the XML element content as the partial name for an image : **XPath** property must be updated as the following : `concat('http://my-intranet/images/', {xpath-value}, '.jpg')` **Image** subchapter, in **Data Fields and Properties** chapter).
  - **Add Address Block** - Starts a wizard that adds an address label to the layout (see detailed description in **Address Block** subchapter, in **Data Fields and Properties** chapter).
  - **Add Paragraph** - Adds an empty paragraph data field to the design layout (see detailed description **Paragraph Block**, in **Data Fields and Properties** section).
  - **Add Table** - Adds a table to the layout (see detailed description **Table**, in **Data Fields and Properties** section).
  - **Add Page Number** - Adds a Page Number to the layout (see detailed description **Page Number**, in **Data Fields and Properties** section).
  - **Add Button** - Adds a Button to the layout (see detailed description **Button**, in **Data Fields and Properties** section).
  - **Add List Box** - Adds a List Box to the layout (see detailed description **List Box**, in **Data Fields and Properties** section).
  - **Add Combo Box** - Adds a Combo Box to the layout (see detailed description **Combo Box**, in **Data Fields and Properties** section).

- **Edit Menu**

  The **Edit** menu contains the following options:

  - **`Insert Symbol`** - Inserts symbols in your project.
  - **`Remove From Layout`** - Removes the selected data field(s) from the layout.
  - **`Add Section`** (**Alt+P**) - Adds a new section to the project.
  - **`Remove Section`** (**Alt+R**) - Removes the current section from the project.
  - **`Edit Section Header`** - Used only for dynamic sections. This option is for editing the static header of the current dynamic section.
  - **`Undo`** (**Ctrl+Z**) - Rolls back the last action.
  - **`Redo`** (**Ctrl+Y**) - Redoes the last undone action.

    For more information about **Undo** and **Redo** operations see Undo/Redo Operations subchapter.
  - **`Cut`** (**Ctrl+X**) - Removes and copies the selection to the clipboard.
  - **`Copy`** (**Ctrl+C**) - Copies the selection to the clipboard.
  - **`Paste`** (**Ctrl+V**) - Inserts the contents of the clipboard at the location of the cursor, and replaces any selected text or objects.
  - **`Edit Table`** - If a table data field is selected, that table's editor opens; if a table editor is opened, that editor closes; if no table is selected or multiple data fields are selected, this option is passive.
  - **`Edit Paragraph`** - If a paragraph data field is selected, that paragraph's editor opens; if a paragraph editor is opened, that editor closes; if no paragraph is selected or multiple data fields are selected, this option is passive.
  - **`Edit Address Block`** - If a address block data field is selected, that address block's editor opens; if a address block's editor is opened, that editor closes; if no address block's is selected or multiple data fields are selected, this option is passive.

- **Build Menu**

  The **Build** menu contains the following options:

  - **Refresh** - Refresh all data fields in the project.
  - **Validate** (**F6**) - checks all XPath expressions in all data fields in the project.
  - **Generate XSL** (**F5**) - Generates an XSL for the current project.
  - **Preview** (**Ctrl+F5**) - Generates preview for current project and shows it in the **Preview** pane.
  - **Create Resulting Document** (**Ctrl+Shift+F5**) - Creates the resulting document from the current project.
  - **Set Output Format** - Set your desired output format from the current project.
  - **`PDF`** - .pdf output file
  - **`PostScript`** - .ps output file
  - **`AFP`** - .afp output file
  - **`XEP Intermediate`** - .xep output file
  - **`SVG`** - .svg output file

  - **`HTML`** - .html output file
  - **XPS** - .xps output file
- **Window Menu**

  The **Windows** menu contains the following options:

  - **All Windows** - Shows all windows. If this option is selected a check sign appears.
  - **Optimized** - **XPath List**, **Options**, **Preview**, and **Log** panes are minimized. If this option is selected a check sign appears.
  - **Lightweight** - **Xml Tree View**, **XPath List**, **Options**, **Preview**, and **Log** panes are minimized. If this option is selected a check sign appears.

    Only one of the above mentioned options can be selected

    The last used window configuration is saved when *VisualXSL* exits, and is restored on the next application startup.
  - **Reset Toolbars** - Toggle set toolbars' deafult positions.
  - **Preview** - Toggle switch for showing/hiding the **Preview** pane.
  - **Log** - Toggle switch for showing/hiding the **Log** pane.
  - **Source XML** - Toggle switch for showing/hiding the **Xml Tree View** pane.
  - **XPath List** - Toggle switch for showing/hiding the **XPath List View** pane.
  - **Properties** - Toggle switch for showing/hiding the **Properties** pane. The **Properties** pane is described in **Data Fields and Properties** chapter.
  - **Options** - Toggle switch for showing/hiding the **Options** pane. The **Options** pane is described at the end of this chapter.
  - **List Open VisualXSL Projects**. Lists all the *VisualXSL* projects that are open. The project that you are currently working on is checked. To work on a different project, click its name.
- **Help Menu**

  The **Help** menu contains the following options:

  - **Contents** (**F1**) - Opens **VisualXSL User Guide**.
  - **About** - Lists general information about the application.

4.6. Toolbars

Most of the basic commands are placed on the toolbars for quick access. Almost all of them are also available in the menus.

- **File operations**
  - **New Project** - creates a new project.
  - **Open** - is a drop-down list with the following options:
    - **Open Project** - loads an existing project.
    - **Open Source XML** - opens and adds an XML data source to the project.

  - **Save** - saves the current project with changes.
  - **Print** - Prints the last generated PDF document. If you haven't created a resulting PDF or haven't previewed the result of XSLT, a warning message will be shown.
- **Data operations**
  - **Add Text Frame** - Adds a text field to the layout, according to the XML source file. Current XML node's XPath expression is used to get the data from the XML source file. Note, that text frames can get static text as well as XPath.
  - **Add Barcode** - Adds a barcode field to the layout, according to the XML source file. Current XML node's XPath expression is used to get the data from the XML source file. Note, that barcodes can get static text as well as XPath (see detailed description in **Barcode** subchapter, in **Data Fields and Properties** chapter).
  - **Add Image** - Adds an empty image field to the layout. **Note:** You must correct **ImageUrl** to generate images from the XML content. For example, to use the XML element content as the partial name for an image : **XPath** property must be updated as the following : `concat('http://my-intranet/images/', {xpath-value}, '.jpg')` **Image** subchapter, in **Data Fields and Properties** chapter).
  - **Add CheckBox** - Adds a checkbox to the layout. The item's **Test** property (under **Expression** group) can be specified to render the checkbox under only certain conditions. It must have the correct XPath expression in the context of the node set, as determined by the expression from **Pattern** property (see detailed description in **Checkboxes** subchapter, in **Data Fields and Properties** chapter).
  - **Add Address Block** - Starts a wizard that adds an address label to the layout (see detailed description in **Address Block** subchapter, in **Data Fields and Properties** chapter).
  - **Add Table** - Adds a table to the layout (see detailed description **Table**, in **Data Fields and Properties** section).
  - **Add Paragraph** - Adds an empty paragraph data field to the design layout (see detailed description **Paragraph Block**, in **Data Fields and Properties** section).
  - **Add Button** - Adds a button to the design layout (see detailed description **Button** , in **Data Fields and Properties** section).
  - **Add Select Box** - Adds a 'select box' to the design layout (see detailed description **Select Box** , in **Data Fields and Properties** section).
  - **Add Combo Box** - Adds a 'combo box' to the design layout (see detailed description **Combo Box** , in **Data Fields and Properties** section).
- **Edit operations**
  - **`Remove From Layout`** (**Del**) - Removes the selected data field(s) from the layout.
  - **`Undo`** (**Ctrl+Z**) - Rolls back the last action.
  - **`Redo`** (**Ctrl+Y**) - Redoes the last undone action.

For more information about **Undo** and **Redo** operations see Undo/Redo Operations subchapter.

- **`Cut`** (**Ctrl+X**) - Removes and copies the selection to the clipboard.
- **`Copy`** (**Ctrl+C**) - Copies the selection to the clipboard.
- **`Paste`** (**Ctrl+V**) - Inserts the contents of the clipboard at the location of the cursor, and replaces any selected text or objects.
- **`Symbol`** - Can include symbols in your project

- **Build operations**
  - **`Refresh Design Layout`** calculates all XPath expressions and refreshes the data fields content on the design layout (where possible).
  - **`Validate Project`** checks all XPath expressions in all data fields in the project.
  - **`Create Resulting Document`** generates a document (PDF, PostSript, AFP, XEP Intermediate, SVG, HTML or XPS) using all data records from the data source.
  - **`Generate XSL Stylesheet`** generates an XSL stylesheet for the current layout. This stylesheet can be used with the *RenderX XEP* formatter to produce PDF files.
  - **`Preview Result of XSLT`** generates a PDF document preview, using the specified number of data records.
  - **`Create Local TranslateProject.xsl`** allows the customer to keep his/her own TranslateProject.XSL in project directory.
- **Selection alignment buttons**
  - **Align Left** aligns the selected data field to the left of the data field.
  - **Align Center** aligns the selected data field to the center of the data field.
  - **Align Right** aligns the selected data field to the right of the data field.
  - **Align Lefts** aligns the selected data fields to the left side of the first selected data field.
  - **Align Centers** aligns the selected data fields to the (horizontal) center of the first selected data field.
  - **Align Rights** aligns the selected data fields to the right side of the first selected data field.
  - **Align Tops** aligns the selected data fields to the top of the first selected data field.
  - **Align Middles** aligns the selected data fields to the middle (vertical center) of the first selected data field.
  - **Align Bottoms** aligns the selected data fields to the bottom of the first selected data field.
  - **Make Same Width** makes selected data fields' widths the same as the width of the first selected data field's.
  - **Make Same Height** makes the selected data fields' heights the same as the height of the first selected data field's.

- **Make Same Size** makes the selected data fields' sizes the same as the size of the first selected data field's.

- **Other operations**
  - 100% ▾ **Scale Layout** scales the layout.
  - 1 ▾ **Choose Section** switches the current section (available for multi-section forms).
  - **Help** opens the *VisualXSL* User Manual.
  - ▾ **Choose Section Template** selects the current template from the list of available templates.

    The **Choose Section Template** option is available only for a dynamic sections.

  - Edit Header **Edit Header** is used to edit the static header of the current template.

    The **Edit Header** option is available only for a dynamic sections.

  - Arial ▾ **Font Family** is used to change the font of the selected text.
  - 10 ▾ **Font Size** is used to change the font size of the selected text.
  - B **Bold** is used to change the font of the selected text to/from bold.
  - I **Italic** is used to change the font of the selected text to/from italic.
  - U **Underline** is used to change the font of the selected text to/from underline.

4.7. Options Pane

The **Options** pane contains the following options.

- **Display Properties**
  - **Hatch Transparency** is a slider for changing the transparency of the background. To make the background more transparent, move the slider to the left.
- **XPath Editing**
  - **Allow changes to Pattern property** specifies whether the Pattern property (under **Expression**, in the **Properties** pane) may be changed. This allows the user to deviate from the supposed/default page pattern.
- **Layout Grid**
  - **Snap to Grid** switches on/off data fields' align to the grid in the layout designer area.

    Snap to grid is used only for static sections.

  - **Grid Size** sets the size of the grid cell in pixels.

    Is active only if **Snap to grid** is switched on. The minimum allowed size is 4 pixels and the maximum allowed sized is 60 pixels.

  - **Draw Grid** switches on/off the grid draw.

If the grid size is too small for the selected scale, the grid will not be displayed.

- **Rendering**
  - **Number of records to preview** enables you to limit the number of records used to generate the preview. It can be useful when the source XML contains a large number of records.
  - **Open last project on start** switches on/off the automatic opening of the last project at the application's next start.
  - **Render form background** switches on/off the generation of the form background.

    Only the **Number of records to preview** and **Render form background** options will take an effect immediately; the other options require application restart.
- **Rasterization**
  - **File Format** is the raster format of the image file.
  - **Color Depth** is the number of bits per pixel in an image format used to rasterize the form background.
  - **Rasterized resolution** (in DPI) specifies the horizontal and vertical resolution of the rasterized image of the form background.

4.8. Undo/Redo Operations

Most actions that are made in **Layout** and **Properties** panes can be rolled back using the **Undo and Redo** commands. Note the following points regarding changes, actions, and operations:

- All changes made to a data field object while it is still selected are considered to be one action, therefore **Undo/Redo** treats them as a single action. For example, if both the size and position are changed while the same data field is selected, **Undo** will roll back both changes. However, if the size is changed for one data field and the position for a second data field, **Undo** will roll back the last action only.

  In *VisualXSL* **Undo** and **Redo** operations are multistep operations. It means that you can roll back and forward the last operations multiple times.
- Operations to add or remove sections **cannot** be undone.

The selected data fields, or a selected whole section, can be copied to the Clipboard, and then pasted one or more times to the same or another window of the application (if more than one project is being edited at the same time).

Make sure that newly pasted data field has the correct XPath expression. To do this, either select **Build | Validate**, or press **F6**, or click the **Validate Project** toolbar button.

Chapter 5. Data Fields and Properties

5.1. Data Field Properties

This chapter describes the data properties (in the **Properties** pane) associated with the various field types that can be added to a *VisualXSL* project from the **Data** menu. The properties are grouped according to **Appearance**,**Custom**, **Data**, **Expression**,**Form Fields**, **Indentation**, **JavaSript**, **Location**, **Padding**, **Size**, and **Spaces**, as described in this chapter.

If a data field has been selected, **the change you make to a property affects only that field**.

When **no** data field has been selected, the change in formatting for a property affects **all** data fields on the section that use the template's formatting for that property. However, a data field which property (for example, font size) has been set individually, is not affected by the change of template formatting for that property.

Many of the fields in the **Properties** panel have drop-down lists, ... buttons, etc. that appear only when the data field is selected in the **Properties** panel.

5.2. Common Data Field Properties

The following data fields, which appear in alphabetical order under **Properties** pane, are used for all data fields.

5.2.1. Appearance

- **BackgroundColor** specifies the color for the background. For example, choosing white makes the area behind the data field white in the resulting PDF.
- **BlockProperties** is a subgroup that contains the following properties:
  - **Alignment** specifies the text alignment. The possible values are:
    - `Inherit`
    - `Left`
    - `Center`
    - `Right`
    - `Justify`
  - **AlignmentLast** specifies the text alignment for the last line of the text block. It has the same set of values as **Alignment**.
  - **LineFeed** corresponds to the XSL-FO **linefeed-treatment** property, which defines how line feed characters are handled. The possible values are :
    - `Inherit` sets this property to the value of the parent container.
    - `TreatAsSpace` forces the formatter to treat line feed characters as a space.
    - `Ignore` lets the formatter ignore any line feed characters.
    - `Preserve` treats all line feed characters as line breaks.

        - **`TreatAsZeroWidthSpace`** forces the formatter to treat line feed characters as a zero-width space.

          displaying this feature in the designer area is not implemented.

    - **WrapOption** specifies whether the text should be wrapped to fit the data field's frame.
- **Border** is a subgroup that is used for specifying the border properties:
    - **BorderColor** specifies the border color.
    - **BorderStyle** specifies the border line style.
    - **BorderWidth** specifies the line width.
- **Orientation** is the block orientation which can be used for rotating data fields. This property corresponds to the XSL-FO reference-orientation **`trait`**.
- **VerticalAlignment** the texts contains the vertical alignment. The possible values are :
    - **`Auto`** sets vertical alignment to the default value (**`Before`**) or to the value of the parent container.
    - **`Before`** sets vertical alignment to the top of the data field.
    - **`Center`** sets vertical alignment to the center of the data field.
    - **`After`** sets vertical alignment to the bottom of the data field.
- **ZOrder** contains the Z-Order.

5.2.2. Custom

The use of the following custom properties requires knowledge in XSL-FO and XSLT.

- **CustomAttribute** is a space-delimited list of new **XSL-FO** styles.
- **CustomProcess** is used to pass a trigger into the TranslateProject.xsl where a user can then override the typical template used to process that element. This allows complete customization for any frame type in the drawing grid.
- **OverriddenAttribute** specifies the **XSL-FO** style(s) to omit.

5.2.3. Data

- **Comment** is used to generate comments in the resulting stylesheet. It is a description for the field. Besides, it can be helpful for navigating the document structure.
- **BorderStyle** specifies the line style of the border.
- **FieldName** specifies the field name.
- **Value** specifies the field value.

5.2.4. Expression

- **Pattern** is the XPath expression that selects an XML branch (node set) used for the data field.

  If you manually modify the XPath expression in the **Pattern** property, be sure that it is correct and evaluates to an XML node set, and that this node set is a descendant of the XML node evaluated with the section **Pattern** property. It is suggested to validate [24] your project to make sure all XPath expressions are correct.
- **Test** contains the XPath expression that defines the conditions for showing or hiding the content.

5.2.5. Form Fields

- **Hidden** contains two logical values: true/false. If *true*, field will **not** be shown on the layout
- **Max Length** which has deafult `' 0 '` value. The maximum allowed count of the entered symbols
- **Multiline** contains two logical values: true/false. If *true* the text field may contain more than one fields
- **Name** contains automatic generated value. **Required unique** property for all fillable fields.
- **Noexport** contains two logical values: true/false. If *true*
- **Password** contains two logical values: true/false. If *true* the text field is generated as a *password* field: instead of symbols, `*` are shown
- **Printable** contains two logical values: true/false. If *true*, the field can be printed
- **Readonly** contains two logical values: true/false. If *true*, no data can be inserted
- **Required** contains two logical values: true/false. If *true*, some data is required to be entered
- **Treat as Field** contains two logical values: true/false. Is editable only for **Text Frames** and **Check Boxes**, as for all other fillable fields it is set **`true`** and is **`Readonly`**.

5.2.6. Indentation

- **First Line Indent** contains the first-line indentation for the data field. It corresponds to the XSL-FO trait `text-indent`.
- **Last Line Indent** contains the last-line indentation for the data field. It corresponds to the XSL-FO trait `last-line-end-indent`.
- **Left Indent** contains the left indentation for the data field. It corresponds to the XSL-FO traits `start-indent`.
- **Right Indent** contains the right indentation for the data field. It corresponds to the XSL-FO traits `end-indent`.

5.2.7. JavaScript

- **On Blur** can write some javascript code to invoke, when the cursor leaves the field
- **On Calculate** This event is defined when a change in a form requires that all fields that have a calculation script attached to them be executed. All fields that depend on the value of the changed field will now be recalculated.
- **On Focus** can write some javascript code to invoke, when a field loses or receives focus.
- **On Format** can write some javascript code to invoke, when format is wrong.
- **On Keystroke**This event occurs whenever a user types a keystroke into a text box or combo box (including cut and paste operations) or selects an item in a combo box list or list box field. A keystroke script may limit the type of keys allowed. For example, a numeric field might only allow numeric characters.
- **On MouseDown** can write some javascript code to invoke, when a mouse button is pressed down in the field area.
- **On MouseEnter** can write some javascript code to invoke, when a mouse pointer enters in the field area.

- **On MouseExit** can write some javascript code to invoke, when a mouse pointer exits in the field area.
- **On MouseUp** can write some javascript code to invoke, when a mouse is pressed up in the field area.
- **On Validate** can write some javascript code to invoke, when when validate.

5.2.8. Location

- **Left** contains the left coordinate, in points.
- **Top** contains the top coordinate, in points.

5.2.9. Padding

- **Bottom Padding** contains the padding of the text from the data field's frame's bottom border, in points.
- **Left Padding** contains the padding of the text from the data field's frame's left border, in points.
- **Right Padding** contains the padding of the text from the data field's frame's right border, in points.
- **Top Padding** contains the padding of the text from the data field's frame's top border, in points.

5.2.10. Size

- **Height** contains the height of the data field's frame. The value for **`Height`** can use the following units of measurement:
  - **`points (pt)`**
  - **`pixels (px)`**
  - **`inches (in)`**
  - **`centimeters (cm)`**
  - **`millimeters (mm)`**

  ℹ If no unit of measurement is specified, the default (**`points`**) is used.

- **Width** contains the width of the data field's frame. The value for the **Width** property can use the same measurements as for the **Height** property.

5.2.11. Spaces

- **Space After The Area** contains the height of the space after the area of the data field's frame, in the specified units. The measurements of **Height** and **Width** properties are used, the default value is `pt`.
- **Space Before The Area** contains the height of the space before the area of the data type, in the specified units. The measurements of **Height** and **Width** properties are used, the default value is `pt`.

5.3. Data Fields That Appear Only When No Editable Item Selected

The following data fields appear only when there is no editable item selected.

- **JavaScript for Document**
  - **Did Print** can write some javascript code(e.g. some alert) to invoke after printing the document.
  - **Did Save** can write some javascript code(e.g. some alert) to invoke after saving the document.
  - **Library** User can write his/her own JavaScript library for the document and use in at any desired place.
  - **Open** can write some javascript code(e.g. some alert) to invoke, when the document is opened.
  - **Will Close** can write some javascript code(e.g. some alert) to invoke, before closing the document.
  - **Will Print** can write some javascript code(e.g. some alert) to invoke, before printing the document.
- **Layout**
  - **BackgroundPath** specifies the path to the background (PDF or image file) that is used.
  - **DynamicFlow** displays type of current section, `True` for dynamic sections, `False` for static sections.
  - **RasterizedPath** specifies the path to the rasterized copy of background image file that has been initially specified in **BackgroundPath** property.
- **Misc**
  - **Treat as field** displays whether the section is dynamic `True` or static `False`. Read only.

5.4. Text Frame

Besides the properties mentioned in Common Data Field Properties, the following additional properties are used for **Text Frames**:

- **TextProperties** is a subgroup that contains properties for formatting text inside the data field:
  - **Font** is a subgroup that specifies the text's font properties.
    - **`Name`** specifies the font type. You can set font type via both typing the font type name into the box and choosing the font type from the drop-down list.
    - **`Size`** specifies the font size. The measurement unit is specified by **Unit**.
    - **`Unit`** specifies the unit for **Size**. It is one of the following:
      - **`Pixel`** - the **Size** measures in pixels.
      - **`Point`** - the **Size** measures is points.
      - **`Inch`** - the **Size** measures in inches.
      - **`Millimeter`** - the **Size** measures in millimeters.
    - **`Bold`** specifies whether the font is bold-faced.
    - **`Italic`** specifies whether the font is italicized.
    - **`Strikeout`** specifies whether the font has a strike-out line through it.
    - **`Underline`** specifies whether the font is underlined.
  - **ForeColor** specifies the text color (the foreground color).
- **Data**
  - **CustomDataFormat** is a custom data format description.

  - **DataFormat** contains XSLT code that defines additional field formatting. For more details, see section **Additional Formatting for Data Fields** , where there is also a table of available codes.

    The application does not provide the list of codes and does not check their validity. If an incorrect value is set for this property, the default code (`0`) is used in the resulting XSL.
  - **DataType** specifies the data type stored in the XML source. This property provides information that is necessary for formatting the text from the XML source. Currently, only the **`Date`** and **`Text`** types are supported.
  - **Text** is a design-time text to be displayed at the layout. It has special meaning for some fields. For more information see below, at **`Image`** and at **`Special Fields`**.
- **Expression**
  - **XPathValue** is an XPath expression that returns a value from a node set defined in the **Pattern** property.

5.5. Barcode

Besides the properties mentioned in Common Data Field Properties, the following additional properties are used for barcodes:

- **Appearance**
  - **Module** is the width of the narrowest stripe in the barcode, in one of the following measures:
    - **`pixels (px)`**
    - **`points (pt)`**
    - **`inches (in)`**
    - **`centimeters (cm)`**
    - **`millimeters (mm)`**

    The default measure is **`pixels (px)`**

    Use this parameter to control the width and height of the barcode.
- **Data**
  - **Type** is the barcode type. Available barcode types are:
    - **`EAN8`**
    - **`EAN13`**
    - **`UPC-A`**
    - **`UPC-E`**
    - **`Interleaved 2 of 5`**
    - **`Code 39`**
    - **`Codabar`**
    - **`Code 128`**
    - **`Australia Post 4-State`**

No validation of an XPath expression result to the barcode specification rules is available.

5.6. Check Box

Besides the properties mentioned in Common Data Field Properties, the following additional properties are used for check boxes:

- **Appearance**
  - **CheckType** specifies the style of the check mark. The possible values are:
    - **`Default`**
    - **`Check`**
    - **`LightCheck`**
    - **`LightCross`**
    - **`Cross`**
    - **`LightWavyCross`**
    - **`WavyCross`**
    - **`Plus`**
    - **`EmptyPlus`**
    - **`Bullet`**
    - **`EmptyBullet`**
    - **`Square`**
    - **`EmptySquare`**
  - **FontSize** specifies the font size of the check mark inside the field.
- **Data**
  - **`BorderStyle`** specifies the style of the check box's border. The possible values are:
    - **`None`**
    - **`Solid`**
    - **`Beveled`**
    - **`Dashed`**
    - **`Inset`**
    - **`Underline`**
  - **`CheckStyle`** specifies the style of the check box. The possible values are:
    - **`None`**
    - **`Check`**
    - **`Circle`**
    - **`Cross`**
    - **`Diamond`**
    - **`Square`**

  - **`Star`**
- **`FieldName`** specifies
- **`Value`** specifies

5.7. Image

Besides the properties mentioned in Common Data Field Properties, the following additional properties are used for images:

- **Data**
  - **ImageUrl** is the URL or the local path to the image file. By default it is a static path to the image.

    To change it, click the ... button, which opens the file browse dialog. If you want to use XPath-Value to extract a part of the URI from XML, use the tag `{xpath-value}` inside the **ImageUrl** property.

    XPathValue, in Expression, by default is **not** used for a newly created field. To extract the image URI from the source XML using XPath, you should manually edit the **ImageUrl** property. For more details, see the following example.
  - **Example**

    Suppose we have employees' photos that are stored in a shared folder of our intranet server. All files are named in the following way: `http://intranet-server.com/photo/`**`XXXX`**`.jpg`, where **XXXX** is a serial number from 1 to 9999.

    Suppose that we also have the following XML fragment describing every employee:

```
<employees>
              <record id="1">
                 <name>John Doe</name>
              </record>
              <record id="2">
                 <name>Adam Smith</name>
              </record>
              </employees>
```

    We should use the following values to render images in our document:

Property	Value
ImageUrl	http://intranet-server.com/photo/{**xpath-value**}.jpg
XPathValue	@id
Pattern	records/record

- **Expression**

- **XPathValue** is an XPath expression that returns a value from a node set defined in the **Pattern** property.

5.8. Special Fields

You can insert special fields into the project. When generating the XSL stylesheet, the special fields are transformed into XSL parameters. This feature enables you to set some values at the XSL transformation stage.

Special fields are described in the `format-plugin.xml` (it is created in the `ProgramFiles/RenderX/VisualXSL/XSL` folder). To insert a special field, drag and drop its XML node from the XML tree view onto the layout. Change the value of the XPathValue property of the created object's properties to the name of the desired field (see the following list). The name should be specified with the preceding symbol **$**.

Field Name	Description	Default Value
PrintDate	Date when the project was last transformed	**`Current date`**
CreateDate	Date when the project was created	**`None`**
AuthorName	Name of the project author	**`None`**
Company-Name	Name of the author’s company	**`None`**

You can also insert a custom special field, which is later transformed into a stylesheet parameter. Set the **XPathValue** property to `$<special-name>`, where `special-name` is CName (it satisfies standard XSL rules of naming variables). Set the default value of the parameter by changing the **Text** property.

5.9. Additional Formatting for Data Fields

Data field contents can be customized by additional formatting. Currently, only different date formats are implemented:

- ISO8601 format (`yyyy-mm-ddThh:mm:ss:mmm`)
- ODBC canonical format (`yyyy-mm-dd hh:mi:ss`)

The date value in source XML file should be properly specified in one of the implemented formats.

Select the field containing the correct date value for formatting application. Switch to its object properties, and choose **`Date`** value for the **DataType** property. Then specify the format code for the **DataFormat** property. All supported format codes are listed in the table below. To refresh the designer area (and see the updated formatting of the data field), click the **Refresh** button.

Code		Description
without century	with century	

1	101	USA mm/dd/yy
2	102	ANSI yy.mm.dd
3	103	British/French dd/mm/yy
4	104	German dd.mm.yy
5	105	Italian dd-mm-yy
6	106	dd mon yy
7	107	Mon dd, yy
8	108	hh:mm:ss
9 or 109		Default + milliseconds mon dd yyyy hh:mi:ss:mmmAM (or PM)
10	110	USA mm-dd-yy
11	111	JAPAN yy/mm/dd
12	112	ISO yymmdd
13 or 113		Europe default + milliseconds dd mon yyyy hh:mm:ss:mmm(24h)
14	114	- hh:mi:ss:mmm(24h)
20 or 120		ODBC canonical yyyy-mm-dd hh:mi:ss(24h)
21 or 121		ODBC canonical (with milliseconds) yyyy-mm-dd hh:mi:ss.mmm(24h)
-	126	ISO8601 yyyy-mm-ddThh:mm:ss:mmm(no spaces)

Milliseconds (if they are specified in the XML data for the field) are not rendered after the layout is refreshed, but are rendered in the resulting PDF document.

5.10. Address Block

The application provides **Address Block Wizard** for adding address labels. The resulting label is aligned vertically to the bottom of the label and consists of four or five address lines. There are three areas in the wizard dialog (see the following figure). The first area, at the left of the dialog, is a tree representing the first element of the XML source file (only the first batch of the XML source is visible in the **XML Tree View**). The second area, at the upper-right of the dialog, is a group of textboxes for the parts of the address. The third area, at the lower-right of the dialog, is a preview for the address label. You can add data from the XML tree to text boxes by double-clicking on the tree node, by dragging nodes into the **Layout**, or by clicking the **Add** button.

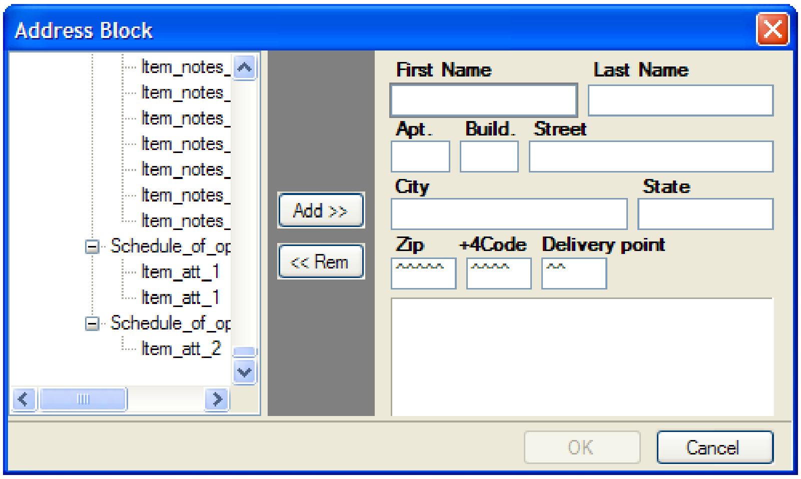


Figure 5.1. Address Block Wizard

If you click the **Add** button, data is added to the highlighted text box. If you add new data to a text box that already contains some data, the new data replaces the old data. To clear the highlighted text box, click the **Remove** button. The data is added from the XML tree in such a way that it is dynamically inserted into the resulting document. Besides, you can type some static data if you need it.

The application forms the PostNet barcode from the three last text boxes (ZipCode, +4Code, Delivery Point). It assumes that all data typed into them are numeric, so make sure that the XML data represents a proper number for a post code.

Address blocks are generated as a special kind of Paragraph Block. This means that they can be edited as both Paragraph and as Address. Editing as Address allows you to visually modify address-related fields, while editing as Paragraph provides with assigning fonts, colors, and customizing order and layout of address fields. The application has possibility for both ways of editing Address blocks.

Double-clicking on Address Frame invokes Address editor to allow changing Address fields.

Procedure 5.1. To invoke Paragraph editor on Address block

1. Create an Address block using the Wizard
2. Close the Wizard to save your initial changes
3. Right-click the Address block on layout
4. Select **Edit Paragraph** pop-up menu item (see Figure 5.2)

5.

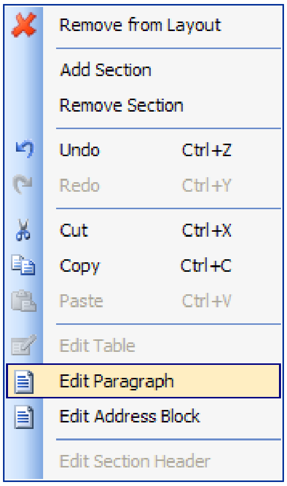
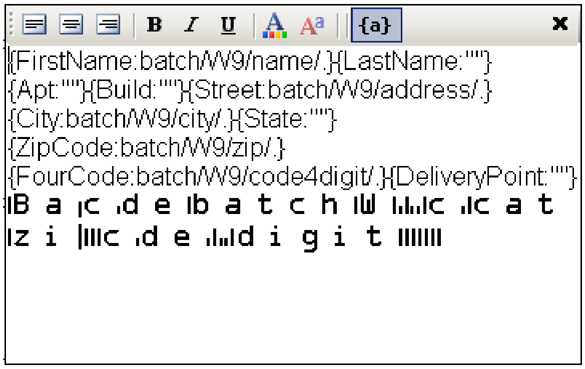


Figure 5.2. Selecting Edit Paragraph option in Address Block Wizard

6. The paragraph editor will open the content of the Address block (see Figure 5.3).



{FirstName:batch/W9/name/.}{LastName:""}
{Apt:""}{Build:""}{Street:batch/W9/address/.}
{City:batch/W9/city/.}{State:""}
{ZipCode:batch/W9/zip/.}
{FourCode:batch/W9/code4digit/.}{DeliveryPoint:""}

Figure 5.3. Editing Paragraph in Address Block Wizard

The fields of wizard block are specified in curl brackets. Before semicolon symbol “:” there is name of field, after that symbol there is content of that filed. To set special font setting, it is important to select the content of filed and from toolbar set the appropriate formatting. It is impossible to see the formatting options while you are editing the address block in wizard. For font options visualization you could press toggle view button “{a}” which is located on paragraph toolbar and could see the view of address label which will be in resulting document.

Currently, the application has no option for specifying the fonts to be used in document formatting. Therefore, to describe the PostNet True Type font, you should manually edit the configuration file for *XEP* and add a section as shown in the following example.

Procedure 5.2. To manually edit the configuration file:

Assume that *XEPWin* is installed in the `C:\Program Files\RenderX\XEPWin` folder.

1. Open the `XEP.xml` file from `xep` subfolder.
2. Find the `font-group` XML element with `@label="Windows TrueType"` attribute.
3. Add the following text inside the found element:

```
<font-family name="IDAutomationPOSTNET" embed="true">
    <font>
        <font-data ttf="IDAutomationPOSTNET.ttf"/>
    </font>
</font-family>
```

This enables the for the *XEP* formatter to use True Type font to render PostNet barcodes.

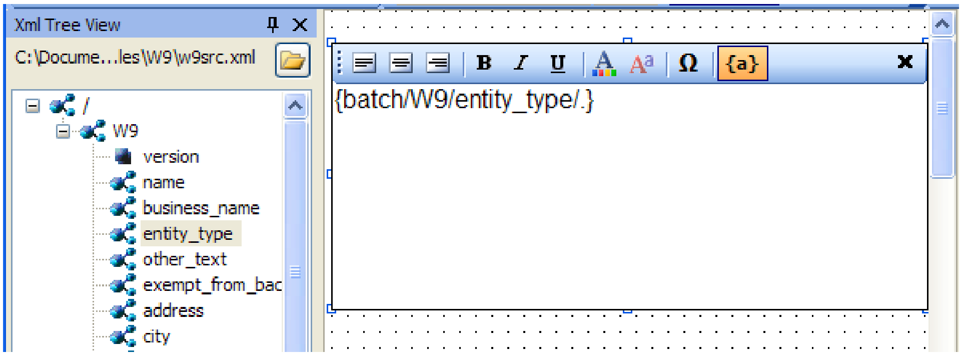
5.11. Paragraph Block

You can put paragraphs with different formatting properties into one block container, or **paragraph block**. An example of such a paragraph block is an address block. Using a paragraph block allows you to prepare text blocks using XML data of variable length.

You cannot leave any data field blank, but at least one block must be filled.

Using a paragraph block also helps when you wish to apply different formatting to different parts of one paragraph.

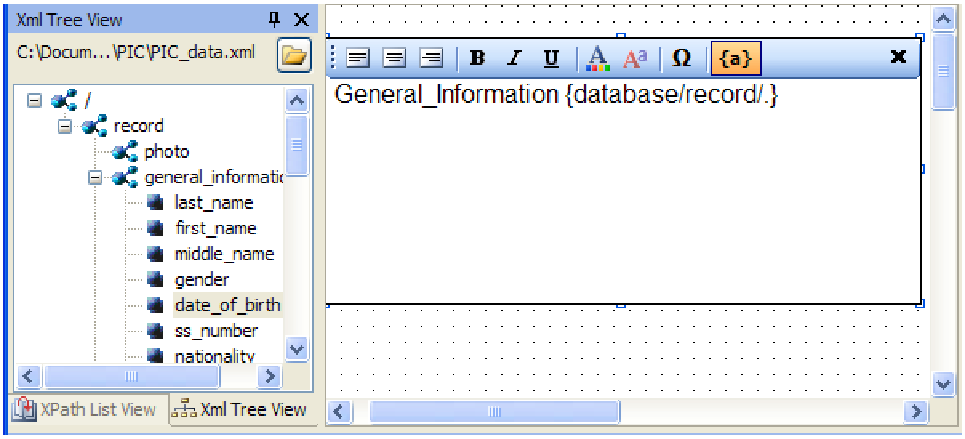
Initially, when you add a paragraph block to the project (from the **Add Paragraph | Data** menu), it has one paragraph. The content of the block is rendered from the selected XML node. You should select **Edit | Edit Paragraph** from the menu or double-click on the **`Paragraph`** block in order to add a new paragraph, or to edit existing ones inside the block. This brings up the following view:



{batch/W9/entity_type/.}

Figure 5.4. Empty paragraph block added to the layout

You can type static text into the paragraph block, drag and drop nodes with data from the XML tree, or type an XPath expression inside curly brackets ({}). To apply formatting to the text, select it, and then click the appropriate toolbar button.



General_Information {database/record/.}

Figure 5.5. Editing the paragraph block - some data fields and static text have been added

To finish editing the paragraph, select **Edit | Finish Editing Paragraph**, or click the **Cross (X)** button.

Before editing a paragraph block, ensure that the entire block on the **Layout** (the **Designer Area**) is visible to you. In some cases, you must scroll the **Layout** manually to see all text being edited.

`undo/redo` is **not** supported while editing the paragraph block.

5.12. Table

The following additional data field is used for tables.

- **Data**
  - **RepeaterPattern** is the context with which the table data will repeat.

    For example, if you have XML as `root/customers` and there are many `/customer` elements below `/customers`, the **`Table`** XPath expression would be `root/customers` and the table **RepeaterPattern** would be `root/customers/customer`.

5.13. PageNumber

Page Number is a Paragraph, which contains a special text, to show current page number in the document.

By default, Page Number is in Page ' i ' of count format, where ' i ' is the current page's number and count is the count of the pages in the document.

5.14. Output Formats

You can get following **Output formats** with *VisualXSL* :

- **PDF** -Portable Document Format
- **PS** - PostScript

- **AFP** - Advanced Function Printing
- **XEP Intermediate**
- **SVG** - Scalable Vector Graphics
- **HTML** - Hyper Text Markup Language
- **XPS** - XML Paper Specification

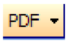
To be able to use those **Output formats** you need:

- a license file allowing to use **PDF, PS, SVG, HTML, XPS**
- a separate license file allowing to use **AFP** (because of Formatter behaves differently seeing AFP backend in the license file - namely, disabling kerning and some other features)

Procedure 5.3. Creating Output formats

The default **Output format** of *VisualXSL* is **PDF**

1. To choose the output format for the document,select **Build | Set Output Formats** from menu or click the `Set Output Format` button:

   .

2. From the drop-down list, select either **PostScript,AFP,XEP Intermediate,SVG,HTML** or **XPS**. (**PPML** is not Supported yet).

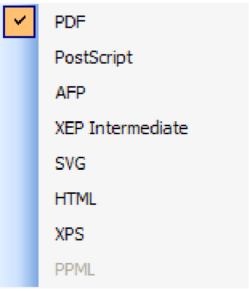


Figure 5.6. Choosing document format

For more information about **Output formats** see RenderX XEP

5.15. PDF Forms

VisualXSL 2.3 and later versions provide a new way of creating fillable Acrobat® forms. From now on *VisualXSL* uses XEP to create fillable forms, which made this process fast and simple. The process of creating fillable forms fully differs from the processes used in previews versions. First, creating fillable form is not a separate process, furthermore, the fillable fields are created like any other field; the only difference is **Treat as Field** property in the **Properties** pane. If **Treat as Field** is set to true, in the resulting PDF the field will be a created as a fillable field, otherwise the field will be a *normal* PDF field.

⚠ Forms are only applicable to PDF output format and will not work for other formats.

⚠ In order to create fillable Forms, *XEP* requires a special license. The license file should be put in `xep` folder in the *XEPWin* installation folder.

ℹ In the 2.3 version *VisualXSL* supports as fillable fields only Text Frames, CheckBoxes, RadioButtons, Buttons, ListBoxes and ComboBoxes. Other frames cannot be created as fillable form fields. Each frame is described above at this section.

ℹ It is no longer required to set PDF background in order to create fillable forms.

5.15.1. Common Properties For PDF Forms

All the frames which can be made fillable PDF fields have several common properties. You can set/view them at the **Properties** pane.

- **Threat As Fields** - is editable only for **Text Frames** and **Check Boxes**, as for all other fillable fields it is set **`true`** and is **`Readonly`**.
- **Name** - **required unique** property for all fillable fields. By default *VisualXSL* generates a unique name for all fields, but you can change **Name** property for fillable fields.

  ℹ Be sure NOT to have multiple fillable fields with the same **Name**. Otherwise, fields will not be generated normally.
- **Readonly** - if *true*, no data can be inserted
- **Required** - if *true*, some data is required to be entered
- **Noexport** - if *true*,
- **Hidden** - if *true*, field will **not** be shown on the layout
- **Printable** - if *true*, the field can be printed

5.15.2. Text Frame

Besides the common properties **Text Frame** has some specific properties.

- **Text** - the default text entered into the field
- **Multiline** - if *true* the text field may contain more than one field
- **Password** - if *true* the text field is generated as a *password* field: instead of symbols, * are shown
- **Max Length** - the maximum allowed count of the entered symbols

5.15.3. Check Box and Radio Button

Besides the common properties **Radio Button** has some specific properties.

- **Group Name** - **required** property to indicate separate Radio Buttons groups. All the radio buttons with the same **Group Names** are in one radio group. *VisualXSL* defaults a unique **Group Name** for Radio Buttons, but you can set your own names.
- **Initially Selected** - indicates if the field is initially selected
- RadioButtons are generated as series of CheckBoxes. You just need to

- Create several CheckBoxes
- In the **Test** properties of each CheckBox give the same left part (e.g. if there are several conditions that compare certain **XPath** expression with different constant values)
- Set the CheckBoxes' **Treat as Field** properties to **`true`**
- Generate the resulting PDF document

See Example of Using Radio Buttons for an example of generating Radio Buttons.

5.15.4. Button

VisualXSL 2.3 supports two types of buttons : Submit and Reset. The button's type is set via **Button Type** property. By default it is set **`Submit`**

Common Properties For Button

All buttons have some common properties.

- **Fields** - space delimited list of affected field names. If empty string is given, all fields are included. The default value is empty string.
- **Text** - Text shown on the button

Submit

Submit Button is used for submitting form data to the server. Besides the common properties, it contains some others:

- **Url** - **Required**, the server URL, to which the data will be sent. By default its value is the inner text of the XML node.
- **Submit Format** - Possible values are : **`XFDF`**, **`FDF`**, **`PDF`**, **`HTML`**. The default value is **`XFDF`**
- **Method** - Possible values are : **`POST`** and **`GET`**. The default value is **`POST`**

Reset

Reset Button is used to reset all the fields values, which names are in **Fields** list

5.15.5. ListBox and ComboBox

Both **ListBox** and **ComboBox** are tools, which allows the user to select one or more values from the given list.

- **Multiselect** - indicates, if multiple values can be selected
- **Values** - list of selectable values, delimited by the given **Separator**. By default its value is the inner text of the XML node.
- **Separator** - a symbol, indicating the separator for **Values** field by default its value is **|**.

Also, **ComboBox** has one more property:

- **Editable** - indicates if a value can be typed, or the value can only be selected from the given list

5.15.6. Example of creating a fillable Acrobat® form

Procedure 5.4. To create a fillable Acrobat® form

Assume you already have some frames on your document and now want to add some fillable ones.

1. Add the frames you wish to make fillable (e.g. Text Frames, CheckBoxes and RadioButtons)
2. Select those frames, which will be fillable fields and navigate to **Properties** pane.
3. Change the **Treat as Field** properties value to **`true`** (by default it is set to **`false`**).
4. Create resulting PDF file

5.16. JavaScript For PDF Forms

VisualXSL 2.3 and later versions support **`JavaScript`** for PDF forms. You can write JavaScript code for fillable PDF fields, for pages and for the PDF document, also you can write your JavaScript library for the PDF document. For fillable fields JavaScript is written in **JavaScript For Document** properties in the **Properties** pane.

5.16.1. JavaScript For Fillable Fields

All the fillable fields (Text Frames, Checkboxes, Radio Buttons, Buttons, Listboxes, ComboBoxes) support JavaScript events. Bellow is the list of supported events.

- **On Blur** - invokes when the cursor leaves the field
- **On Calculate** - This event is defined when a change in a form requires that all fields that have a calculation script attached to them be executed. All fields that depend on the value of the changed field will now be recalculated.
- **On Focus -** invokes when a field loses or receives focus.
- **On Format** - invokes when format is wrong.
- **On Keystroke** - This event occurs whenever a user types a keystroke into a text box or combo box (including cut and paste operations) or selects an item in a combo box list or list box field. A keystroke script may limit the type of keys allowed. For example, a numeric field might only allow numeric characters.
- **On Mouse Down -** invokes when a mouse button is pressed down in the field area.
- **On Mouse Enter** - invokes when a mouse pointer enters in the field area.
- **On Mouse Exit** - invokes when a mouse pointer exits in the field area.
- **On Mouse Up -** invokes when a mouse is pressed up in the field area.
- **On Validate** - invokes when validate.

5.16.2. JavaScript Example For PDF Forms

Procedure 5.5. To create a PDF file with JavaScript

1. Create Project
2. Add a text frame
3. Select the added frame and navigate to **Properties** pane.
4. Write `app.alert("OnBlur is invoked");` JavaScript code into **On Blur** property
5. Create resulting PDF file

5.16.3. JavaScript For Pages

VisualXSL 2.3 supports two JavaScript events for pages. Which can be found among **JavaScript For Pages** properties in the **Properties** pane.

- **Js Open** - invokes when new page is opened.
- **Js Close -** invokes when window is closed.

These are hooks for events of a page coming into or out of the view in the PDF reader. They go transparently to `<xep:page>` and then to Page objects in PDF document for each page created with the page master where the attributes were set.

5.16.4. JavaScript Example For Pages

Procedure 5.6. To create a PDF file with JavaScript for pages

1. Create Project
2. Select the layout and navigate to **Properties** pane.
3. Write `app.alert("Page is Opened");` JavaScript code into **Js Open** property
4. Create resulting PDF file

5.16.5. JavaScript For Document

VisualXSL 2.3 supports several JavaScript events for the document. Which can be found among **JavaScript For Document** properties in the **Properties** pane.

- **Js Open** - invokes when the document is opened.
- **Js Did Print -** invokes after printing the document.
- **Js Did Save -** invokes after saving the document.
- **Js Will Close -** invokes before closing the document.
- **Js Will Print** - invokes before printing the document.

VisualXSL 2.3 also allows the user to write his/her own JavaScript library for the document and use in at any desired place. The library is written in the **Library** property under **JavaScript For Document** property group in the **Properties** pane.

5.16.6. JavaScript For Document Example

Procedure 5.7. To create a PDF file with JavaScript for the document

1. Create Project
2. Select the layout and navigate to **Properties** pane.
3. Write `function f() { app.alert("function f() is called"); }` JavaScript code at **Library** property
4. Write `f();` JavaScript code into the **Js Open** property
5. Create resulting PDF file
6. Open the created PDF file. See the "function f() is called" message.

Chapter 6. Using the VisualXSL Application

6.1. How to Create a New Project

Procedure 6.1. To create a new project:

1. Run a project in *VisualXSL* with one of the above-mentioned methods.
2. If the previous project loaded automatically on startup, press the **New Project** button on the toolbar, or select the **File | New Project** menu item, or press the **Ctrl+N** key combination.
3. To select the XML file (it must be a well-formed XML document), either use the **File | Open Source XML** menu item, or press the **Ctrl+F** key combination. After the XML file is successfully loaded, its schema is shown in the **XML Tree View**.
4. To set the required properties in **Object Properties**, click an empty space of the section in **Layout** and do as follows:
   - Specify the **Pattern** property in the property grid. The **Pattern** property should contain a valid XPath of an XML element.

     The XPath expression is case sensitive.
   - Set the **BackgroundPath** property value to the valid full path of the previously prepared PDF file (which contains the form that you want to fill with data from the XML). It can easily be done by clicking the ... button.
   - You can also set an alternative value for the **RasterizedPath** property. Set it to the valid full path of a previously prepared raster file with a form you want to fill with data from the XML.

     Currently, the application supports the following raster image formats: PNG, TIFF, JPG, and GIF (for more details, see the RenderX XEP On-line Reference).

6.2. How to Create a Multi-Section Project

Procedure 6.2. To create a multi-section project:

1. Create a new project (see **How to Create a New Project** above).
2. Select an XML file either by using the **File | Open Source XML** menu item or by pressing **Ctrl+F**.
3. In the object properties, set the **BackgroundPath** property to a full path of the required multi-page PDF.

   You can click the ... button to use the open file dialog. After the PDF background is selected, the first section is rendered in the layout area and the full path to the rasterized version appears in the object properties in the **RasterizedPath** property. Also, the application creates a PDF for every section of the selected PDF, and produces rasterized versions of them. Rasterized files are generated from the multi-page PDF background file. For each page a separate PDF and PNG files are generated. Files are created in the project folder and named: `PDF File-NameXXXX.pdf` and `PDF File-NameXXXX.png`, where `XXXX` is a section number starting from 0001. These files are set as the backgrounds for the corresponding pages.

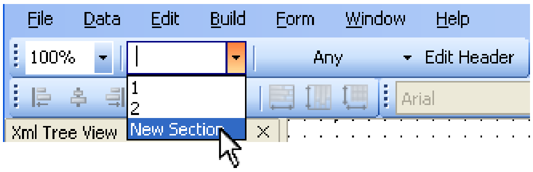


Figure 6.1. Adding a new section to the project

If you do not have a PDF file with a form, but you have a file in a supported graphics format such as PNG, TIFF, JPG, and GIF (for more details, see RenderX XEP On-line Reference), you may manually set its path to the **RasterizedPath** property.

4. Add and adjust the data fields for this section (see, **How to Add Data Fields** below).
5. When all data fields have been added and are properly adjusted, click the toolbar drop-down list with the active section number (see the figure 6.1 above).

   The drop-down list shows the currently available section numbers, and at the bottom of the drop-down list there is a **New Page** item. Click this item if you need to add a new section to the project. Alternatively, you can use the **Edit | Add section** menu item or press the **Alt+P** key combination. To add a new section, you may also enter the next number of the last section's number.
6. To remove the currently selected section, use the **Edit | Remove section** menu command or the **Alt+R** key combination.

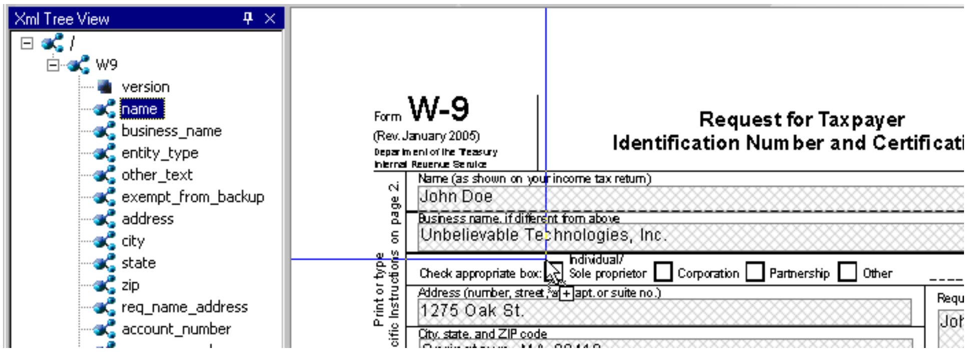
6.3. How to Add Data Fields

After the project is created, you can add, modify, or remove data fields.

Data field is a frame on the **Layout**. It is dynamically rendered in the resulting document by getting data from the XML data file.

Procedure 6.3. To add a new data field:

1. Select a node in the **Xml Tree View**. The data stored in this node is rendered in the resulting form.
2. Click the **Add to Layout** toolbar button to add a field with the selected XML node to the design layout. Alternatively, you can drag a selected tree node and drop it onto the layout. A new data field appears in the form layout. Or, you can drag an item by holding the right mouse button. In that case, after releasing the mouse button in the desired place on the layout, you can choose a data field type from the context menu.



Name (as shown on your income tax return)
John Doe
Business name, if different from above
Unbelievable Technologies, Inc.
Check appropriate box: ☒ Individual/ Sole proprietor □ Corporation □ Partnership □ Other
Address (number, street, and apt. or suite no.)
1275 Oak St.
Requ
Joh
City, state, and ZIP code

Figure 6.2. Dragging XML node onto the layout

3. You can now select the newly added frame by clicking it, and setting its desired position by dragging it over the layout.
4. A data field, when selected, has handles at the corners and at the middle of the sides. You change the location and size of the data field by dragging.
5. Additional properties can be modified for a selected data field in the **Properties pane**.

   After all data fields are added and aligned, you can generate the XSL stylesheet and/or the resulting PDF document.

   During the PDF rendering, fields are filled with data from the XML file. See the detailed properties description in the Data Fields and Properties chapter.

6.4. How to Fill In Preprinted Forms with Data

Procedure 6.4. To fill in preprinted forms with data fields:

1. Create a new project in *VisualXSL*.
2. Scan to a raster image of the preprinted form and set it as the background, or use an existing PDF.
3. Add data fields and all necessary pages.
4. Using **Options** pane, clear the **Render form background** check box.
5. Save the **.VXL** file.
6. Generate your PDF with the data fields only, and print it on your preprinted form(s).

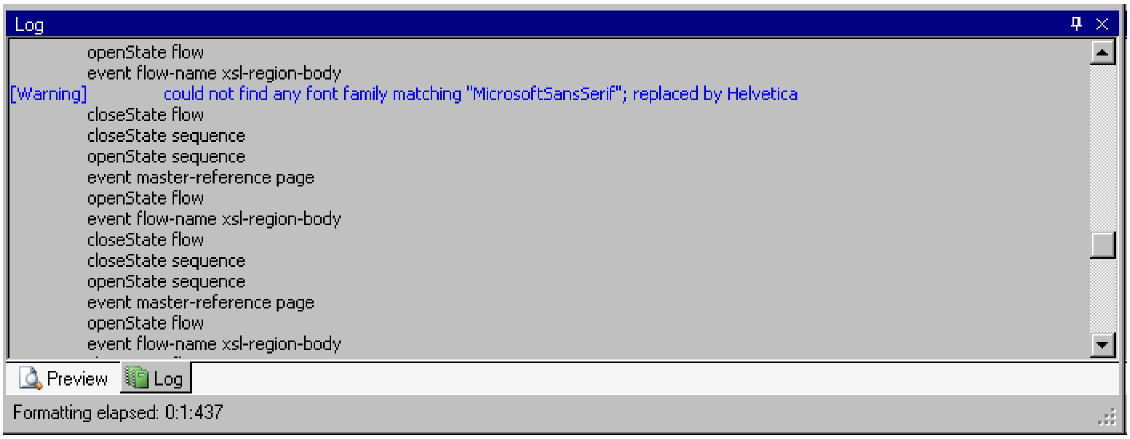
   To restore the background rendering in the destination document, check the **Render Form Background** check box when you are finished.

6.5. How to Use the Log Pane

The **Log Pane** traces the progress of the formatting. It contains all the messages from the *XEP* formatting engine. The messages are highlighted in the following way:

- Errors – red;
- Warnings – blue;
- Other - black.

Some messages contain clickable links (warnings and errors related to incorrect XPath expressions in data fields). You can click those messages to go to the problematic data field.



```
          openState flow
          event flow-name xsl-region-body
[Warning]           could not find any font family matching "MicrosoftSansSerif"; replaced by Helvetica
          closeState flow
          closeState sequence
          openState sequence
          event master-reference page
          openState flow
          event flow-name xsl-region-body
          closeState flow
          closeState sequence
          openState sequence
          event master-reference page
          openState flow
          event flow-name xsl-region-body
```

Formatting elapsed: 0:1:437

Figure 6.3. Warnings are blue in the log pane

It is recommended to look at the log after every document formatting. If the document is not generated successfully, the log can show the reason why.

6.6. How to Create PDF/PostScript Document with VisualXSL

After a project is created and all necessary data fields are placed on the layout, you can create a PDF or a PostScript document.

Procedure 6.5. To create a PDF/PostScript document:

1. Click **Set Output Format** toolbar icon or use the menu command **Build | Set Output Format** and select output format.

   A **File Save** dialog box appears.

2. Click the **Create Resulting Document** toolbar icon or use the menu command **Build | Create PDF**.

   A **File Save** dialog box appears.

3. Type in the file name for your document, set the desired document format (PDF or PostScript), and click **OK**.

   The application automatically rasterizes PDFs into an image when PostScript files are generated and a PDF file is set as the background for the project. This raster image is used as the background in the document.

   Generation of the document can take some time, depending on the number of pages and the PDF background form's complexity.

   If your XML file contains a large number of records, you may get **Not enough memory to format** error. To fix this issue, increase Java memory settings (`-Xms` and `-Xmx` keys) for `Cliser Service` (a part of *XEPWin* package used by *VisualXSL*). For details see the CliserService.exe.config. file installed in the assumed location of `C:\Program Files\RenderX\XEPWin folder`. However, if the number of pages is more than this setting can handle, see below How to Create Large Documents Faster by Using RenderX VDPMil for the solution.

6.7. How to Create Large Documents Faster by Using VDPMill Directly

Creating documents with *VisualXSL* is acceptable if you need only to generate a small number of pages of a simple form (1-100 pages). However if you need to generate thousands of pages of a document, it is much faster to use *RenderX VDPMill* directly.

VDPMill is a complete solution with very high performance rendering of both large print files as well as singular large reports.

VDPMill can generate very large batch print files – hundreds of thousands of pages in a single file. Through the use of a multi-threaded formatting grid for documents, the components of this print file can be formatted simultaneously to meet any performance demands. The *Formatter* layer administrative tools allow the user to add/remove server connections from the grid so all that is required to increase performance is to add additional RenderX XEP server threads. The *Formatter* automatically balances the load across the new server threads. The *Customizer* is fully extendible, allowing the user to modify the print stream as desired before generation of output. This eliminates the need for custom PDF, PostScript and AFP parsers or concatenations. And using the multi-threaded *Generator*, VDPMill can produce simultaneous output documents from the same internal document stream, eliminating the need and inherent deficiencies of trying to convert one output format to another. For example, the user can produce one large batch of invoices as a PostScript or AFP file for print and selected invoices in that batch as PDF or PDF Form for electronic presentment or e-mail.

VDPMill is the perfect application for all your high performance print and large report production, and dynamic and interactive PDF Forms.

For information on RenderX VDPMill, contact RenderX Sales by calling +1.650.327.1000 or via email at sales@renderx.com

6.8. How to Use the Command Line Interface

VisualXSL has a command line interface that gives access to the application functionality from the Windows console.

The syntax to use is:

```
VisualXSL.exe <VisualXSLProject.vxl> <SourceXml.xml> <ResultingFile.pdf | ResultingFile.ps>[options]
```

where

- `<VisualXSLProject.vxl>` is the full path to a *VisualXSL* application previously designed project file.
- `<SourceXml.xml>` is the full path to the source XML file containing data to be used in the transformation.
- `<ResultingFile.pdf | ResultingFile.ps>` is the full path to the resulting PDF or PostScript file name.

  ℹ If the path contains spaces, it should be enclosed with double quotes.

The `[options]` is one of the followings:

- `-?` - displays a message box with the command line syntax help.
- `-l<FULLPATH>` - sets the log file name to store messages of the formatting process, where `<FULLPATH>` is a file name with the full path.
- `-r<NUMBER>` – sets the number of the records to be extracted from the XML file to perform generation of the document, where `<NUMBER>` equals the expected number of records.

Example:

The following command line shows the feature in action:

```
VisualXSL.exe "%ALLUSERSPROFILE%\Documents\VisualXSL
Projects\Samples\W9\W9.vxl" "%ALLUSERSPROFILE%\Documents\VisualXSL
Projects\Samples\W9src.xml" "%ALLUSERSPROFILE%\Documents\VisualXSL
Projects\Samples\W9\W9.pdf" -l"%ALLUSERSPROFILE%\Documents\VisualXSL
Projects\Samples\W9\W9.log" -r5
```

`%ALLUSERSPROFILE%` is a Windows system variable that contains the path to the user's `Documents` folder.

This example uses the `W9.vxl` sample project file with only first five records from `W9src.xml` to produce `W9.pdf`, in the `..\ All Users\Documents\VisualXSL Project\Samples\W9\` folder, and log all the events to `W9.log` file located in the same folder.

In Windows Explorer "Documents" folder may be shown as "Shared Documents".

6.9. How to Add a Table Object

You can add a table object either by using the **Data | Add Table** menu command or by clicking the **Add Table** toolbar icon or right click on XML tree view and select **Add Table**.

To edit a table double-click on it. You can also select it and click the **Edit | Edit Table** menu command or right-click the table and for the context menu click the **Edit Table** command. The **Table Builder** opens and **Table Builder's** toolbar is added, where you can perform the following actions:

- **`Add a table column.`**
- **`Add a table row`**.
- **`Delete a table column`**.
- **`Delete a table row`**.
- **`Delete the content of a cell`**.
- **`Add or delete a table header`**.
- **`Add or delete a table footer`**.

For more detailed information about these actions see Table section.

A table, when selected, has handles at the corners and at the middle of the sides. You change the location and size of a table by dragging.

In the **Properties** pane, you can set the properties of a table, such as:

- **BorderColor**
  - **`Appearance`**
  - **`Border`**
  - **`BorderColor`**
- **Border Style**
  - **`Appearance`**
  - **`Border`**
  - **`BorderStyle`**
- **Border Width**
  - **`Appearance`**
  - **`Border`**
  - **`BorderWidth`**

For more detailed information about **Table Builder** see Table section.

6.10. How to Edit a Field Pattern

To see if editing a field pattern is currently enabled, look in the **Properties** pane's **Expression | Pattern** property. If this field is disabled, the field pattern editing is not currently enabled.

To enable the field pattern editing, go to the **Options** pane's **XPath Editing | Allow Changes to Pattern Property** and select the check box. The **Expression | Pattern** field is now available.

For example, let's concatenate the first name field and the last name field to produce the full name. Assume that the parent node is `parent_name`. Assume that the two name fields are: `first_name`, `last_name`. Then the XPath expression will be: `concat(parent_name/first_name, ' ', parent_name/last_name)`

6.11. How to Create Dynamic Templates

VisualXSL supports a mixture of static and dynamic templates. Dynamic templates can be created in any project, whether it was defined as **Dynamic Document** (in the **New Project** dialog box) or as a **Form**.

The default template in a dynamic document is the **Any** template.

To create additional templates for the project, click the **Choose Section Template** button. From the drop-down list, select either **`Only`**, **`First`**, **`Last`**, or **Any**. If the template does not exist, it is created. This template is the current template for the project. That is, this button can also be used to select the current template.

ℹ For more information, see the Extensible Stylesheet Language (XSL) web site, under the "page-position" paragraph.

VisualXSL knows how many sections are in the project and assigns the templates according to the following rules.

For a single-section dynamic document, the **Only** template is used if it exists, otherwise the Any template is used.

For a multi-section dynamic document:

- The **First** template is used for the first section if this template exists, otherwise the **Any** template is used.
- The **Last** template is used for the last section if this template exists, otherwise the **Any** template is used.
- If there are any intermediate sections (that is, other than the first and the last sections), the **Any** template is used.

6.12. How to use JavaScript in PDF Forms

VisualXSL 2.3 and higher support JavaScript for PDF Forms. There are three types of JavaScripts for a document: JavaScript for a frame, JavaScript for a page and JavaScript for the hole document.

Procedure 6.6. To write JavaScript for a frame

1. Open/Create a *VisualXSL* project
2. Add a **Text Frame**
3. Navigate to **Properties** pane and set **Threat as Field** property to **True**

   JavaScript can be written only for those fields, which **Threat as Field** property is **True**.

4. Navigate to **Mouse Enter** property in the **JavaScript** section of the **Properties** pane. Use ... icon to open *VisualXSL*'s JavaScript window. Usually it is used to paste (**Ctrl** + **V**) the JavaScript written in a JavaScript editor, but this is a simple example and we can write our JavaScript code into the opened window: `app.alert("Mouse entered the field", 2);`
5. Generate resulting document or preview results (PDF)
6. Open the generated PDF document
7. Move the mouse cursor to the field. **Mouse entered the field** message is shown.

6.13. How to use Page Numbering

VisualXL 2.3 and higher versions contain a new field - **Page number**. **Page Number** is a **Paragraph**, which contains a special text, to show current page number in the document.

Procedure 6.7. To add a Page Number

1. Open/Create a new *VisualXSL* project
2. Click **Add Page Number** icon on the Toolbar menu or **Data | Add Page Number** menu item
3. Create resulting document and see page number on each page of the document

By default, **`Page Number`** is in `Page` **`i`** `of` **`count`** format, where **`i`** is the current page's number and **`count`** is the count of the pages in the document.

6.14. How to use Symbols

VisualXSL 2.3 and higher contain a new feature, you can include *symbols* in your project.

Procedure 6.8. To use Symbol

1. Open/create a *VisualXSL* project
2. Add a **Paragraph** frame
3. Click Ω icon on **Toolbars**. **Character Map** window opens.
4. Choose the symbols by double-clicking on them or via **Select** button
5. Click **Copy** button and close the window
6. Open the paragraph and paste (**`Ctrl + V`**) the selected symbols.

   You can use (paste) the copied symbols for any allowed frame.

6.15. How to Use Mvp.Xml EXSLT Module

By default, VisualXSL is using native .NET libraries (MSXML) for XSL transformations during Preview and Generate Document operations. However, this library does not support EXSLT by its own. In order to employ EXSLT features for resulting documents, VisualXSL is capable using alternative library, *Mvp.Xml Project: EXSLT.NET module*.

Mvp.Xml is is a community-developed implementation of the EXSLT extensions to the XSLT 1.0 for the .NET platform. You can download it from its Web site, (http://mvp-xml.sourceforge.net/exslt/. After downloading, just unzip the binaries to VisualXSL folder.

No additional steps required. VisualXSL is looking for Mvsp.Xml.dll in its folder (same as the VisualXSL.exe is located) and if found, uses it via reflection.

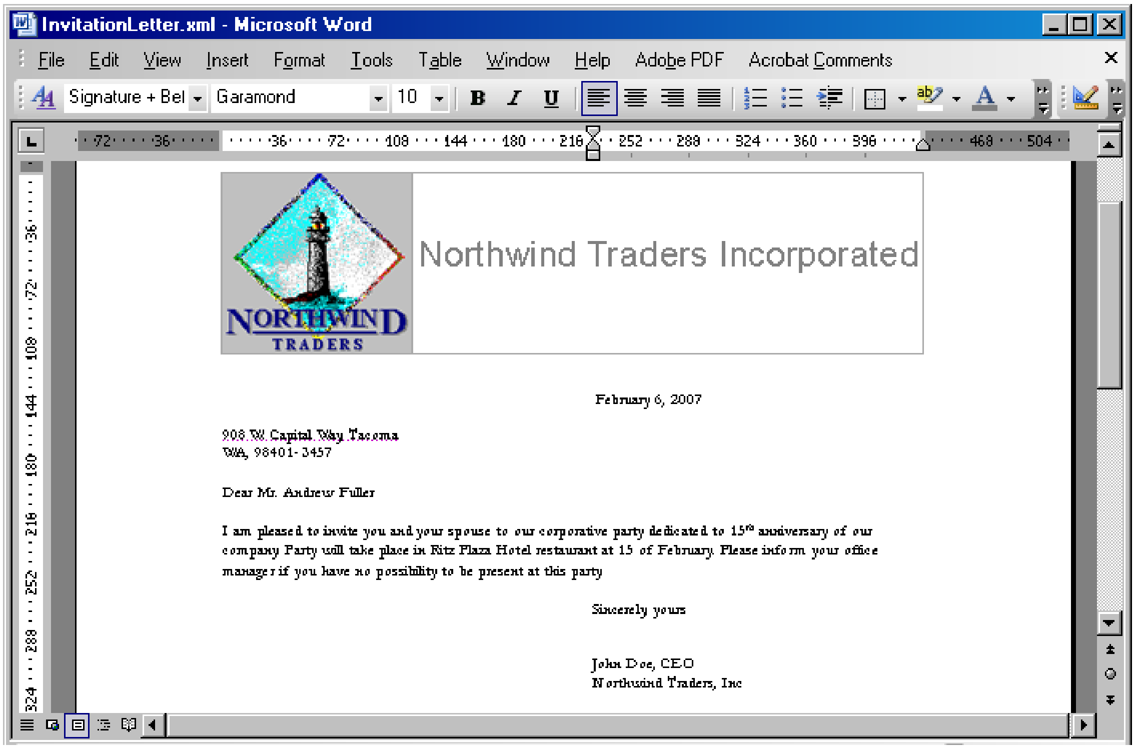
Chapter 7. Examples

7.1. Example of Creating XML Source and PDF Background

For this example *Microsoft Office Professional Edition 2003* (or later) must be installed.

You can use *Microsoft Excel* to prepare an XML source file and *Microsoft Word* to make an invitation letter form.

In this example, you prepare and mail an anniversary party invitations to all company managers named CEO. All employees' information is stored in an Excel spreadsheet (see **`Figure 7.1`** below). The letter should look like the one in the figure below, and you should be able to print it or Email it as an electronic document.



February 6, 2007

908 W. Capital Way Tacoma
WA, 98401-3457

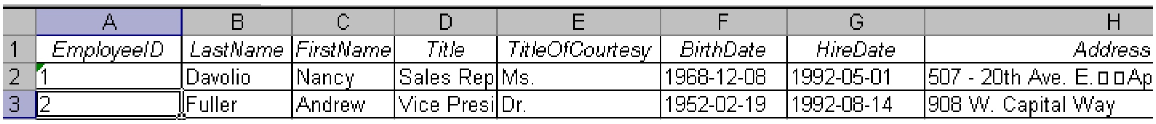
Dear Mr. Andrew Fuller

I am pleased to invite you and your spouse to our corporative party dedicated to 15th anniversary of our company Party will take place in Ritz Plaza Hotel restaurant at 15 of February. Please inform your office manager if you have no possibility to be present at this party

Sincerely yours

John Doe, CEO
Northwind Traders, Inc

Figure 7.1. Invitation letter



	A	B	C	D	E	F	G	H
1	*EmployeeID*	*LastName*	*FirstName*	*Title*	*TitleOfCourtesy*	*BirthDate*	*HireDate*	*Address*
2	1	Davolio	Nancy	Sales Rep	Ms.	1968-12-08	1992-05-01	507 - 20th Ave. E.□□Ap
3	2	Fuller	Andrew	Vice Presi	Dr.	1952-02-19	1992-08-14	908 W. Capital Way

Figure 7.2. Data in the Excel spreadsheet

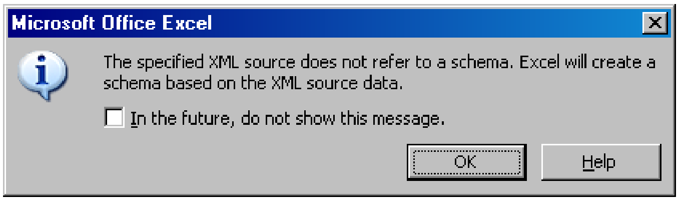
First of all, create the XML source file. You should have an XML schema to be able to map the data to the XML tags and export it to an XML file. The simplest way to do that is to prepare a small XML file describing the common structure of your data.

The XML file should look like the following example (you can omit any elements not necessary for your task). You should write the XML node corresponding to the data record **twice** to make *Microsoft Excel* realize that this node can be replicated many times.

Attach a new XML map by doing the following:

1. Open the **XML Source** tab by selecting **Data | XML | XML Source**.
2. Click the **XML maps** button, which opens the window with the XML map list.
3. Add a new file by clicking the **Add** button and selecting a prepared XML file as an XML source.
4. When a warning message appears saying that no XML schema is referred to this file, click **OK**.

   Excel creates the schema automatically.

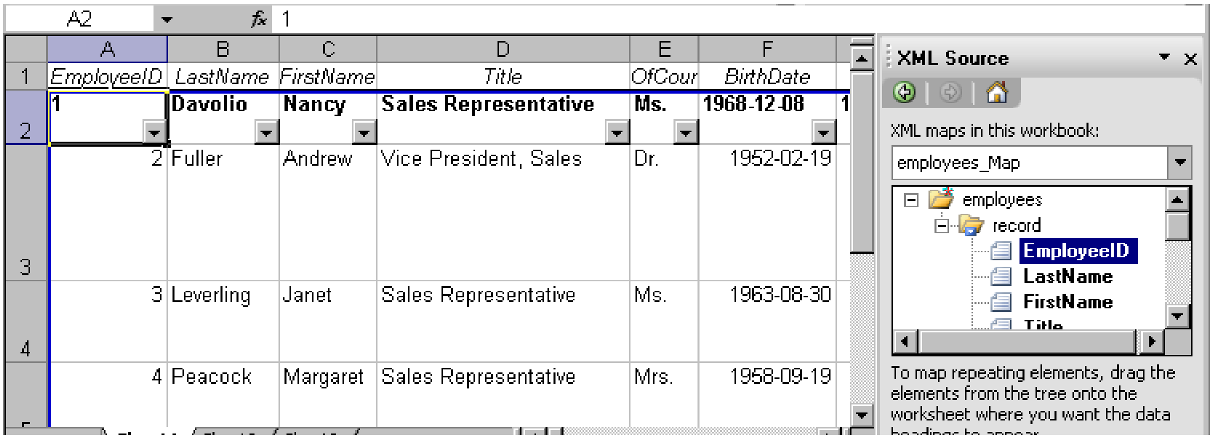


The specified XML source does not refer to a schema. Excel will create a schema based on the XML source data.

□ In the future, do not show this message.

Figure 7.3. Warning that *Microsoft Excel* will create a schema based on selected XML file.

5. Map all XML elements to the spreadsheet data columns, you can see the instructions in the bottom of the **XML Source** tab. After you have finished, you can export the spreadsheet to XML.



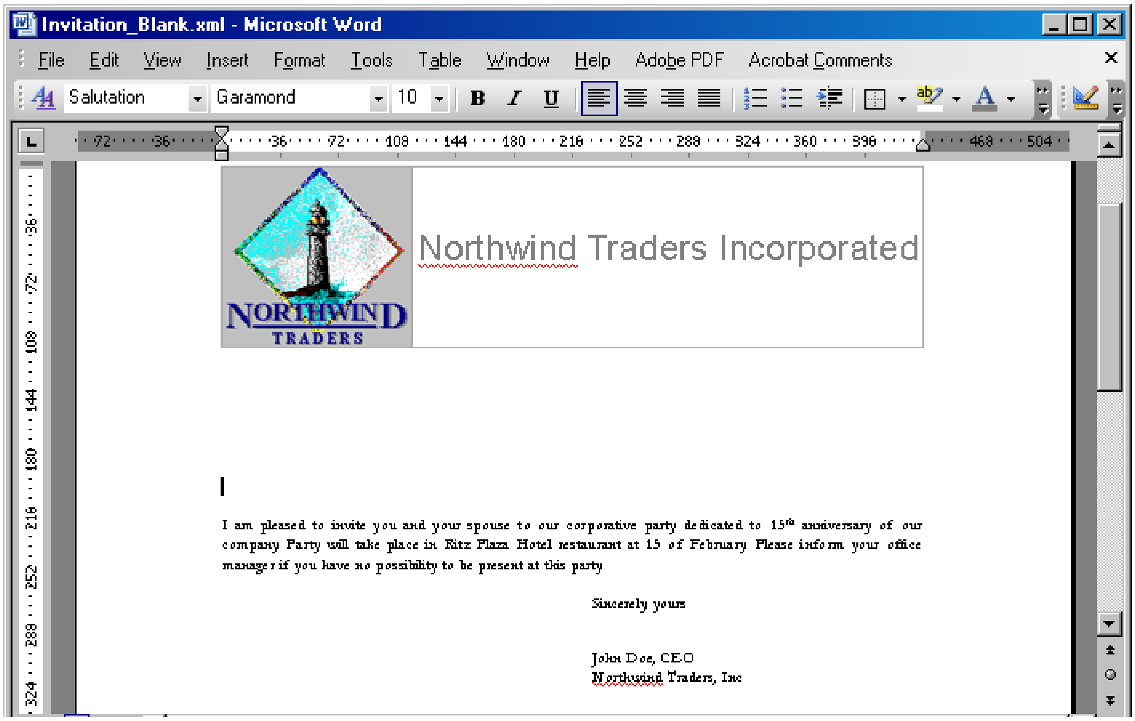
A2 fx 1

	A	B	C	D	E	F
1	EmployeeID	LastName	FirstName	Title	OfCour	BirthDate
2	1	Davolio	Nancy	Sales Representative	Ms.	1968-12-08
3	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19
4	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30
5	4	Peacock	Margaret	Sales Representative	Mrs.	1958-09-19

To map repeating elements, drag the elements from the tree onto the worksheet where you want the data headings to appear.

Figure 7.4. XML elements mapped to *Microsoft Excel* spreadsheet data columns

6. Save your spreadsheet as an XML file to your project folder.
7. To prepare a PDF document that can be used as a background, you can use *Microsoft Word* and create a document with space left for variable data. The document might look like the following `Figure 7.5`. Save it as an XML (WordML) file.



Northwind Traders Incorporated

I am pleased to invite you and your spouse to our corporative party dedicated to 15th anniversary of our company Party will take place in Ritz Plaza Hotel restaurant at 15 of February Please inform your office manager if you have no possibility to be present at this party

Sincerely yours

John Doe, CEO
Northwind Traders, Inc

Figure 7.5. Letter with space left for employee personal data

8. Use *XEP Assistant* and *WordML2FO* stylesheets from the *XEPWin* package to produce a PDF file from your mockup.

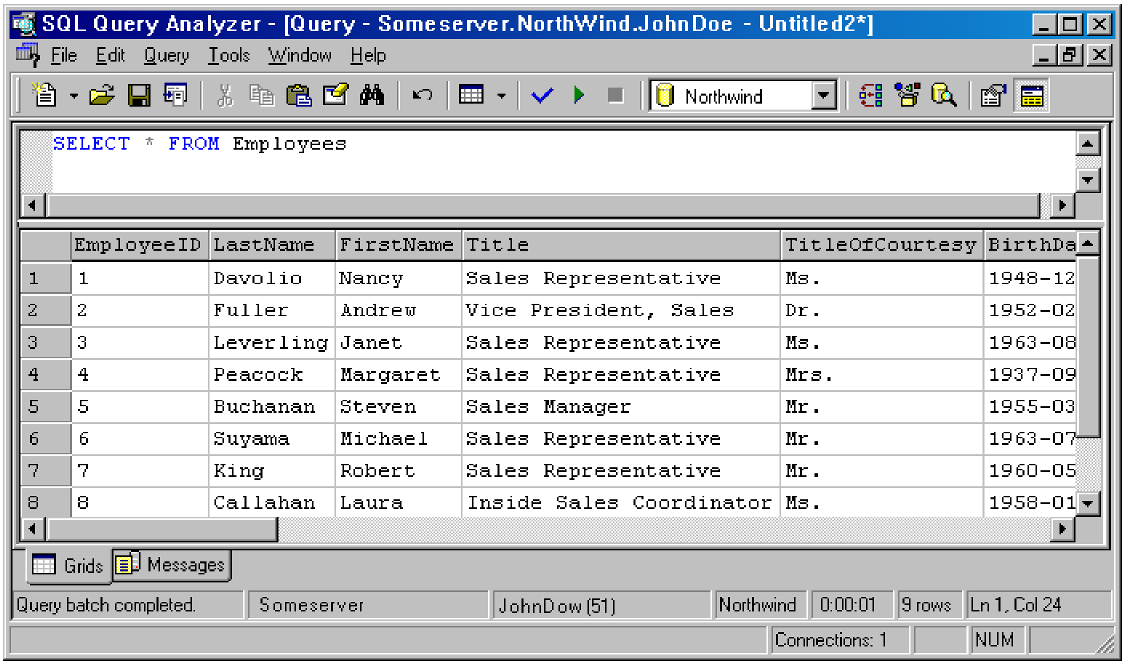
   With the XML source file and the PDF file, you have the files necessary for creating a new project.

7.2. Example of Exporting XML Data File from SQL Server

For this example we use the **`Northwind`** test database and *Query Analyzer* (*Microsoft SQL Server 2000*) for the data conversion.

1. Open *SQL Query Analyzer*(*Microsoft SQL Server 2000*).
2. Connect to the **`Northwind`** database.

   Our goal is to convert data from the **`Employees`** table.



```
SELECT * FROM Employees
```

	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDa
1	1	Davolio	Nancy	Sales Representative	Ms.	1948-12
2	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02
3	3	Leverling	Janet	Sales Representative	Ms.	1963-08
4	4	Peacock	Margaret	Sales Representative	Mrs.	1937-09
5	5	Buchanan	Steven	Sales Manager	Mr.	1955-03
6	6	Suyama	Michael	Sales Representative	Mr.	1963-07
7	7	King	Robert	Sales Representative	Mr.	1960-05
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01

Figure 7.6. Data in the Employees table

3. Assume that the file with the converted data should be placed under `c:\temp\`.

4. Using any text editor (for example **`Notepad`**), create the `template.tpl` conversion template file in this folder.

5. Use the editor to enter the following in this template file:

```
<root>
  <%begindetail%>
    <%insert_data_here%>
  <%enddetail%>
</root>
```

6. Enter the following query into the **`SQL Query Analyzer`** window (see the following figure)

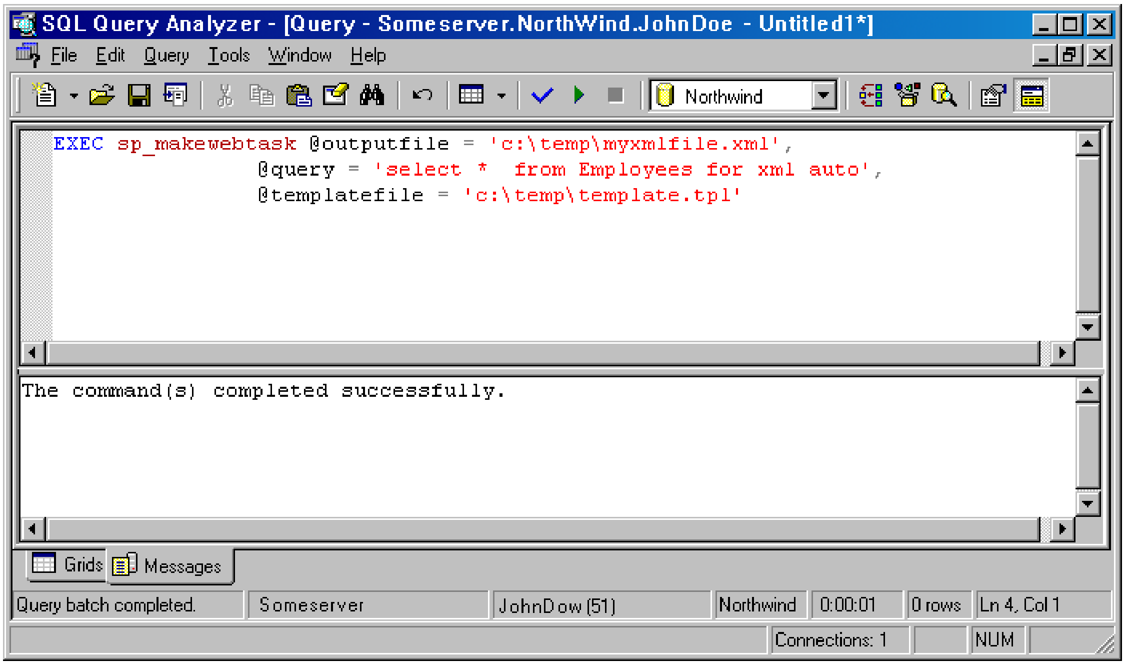
```

EXEC sp_makewebtask @outputfile =
'c:\temp\myxmlfile.xml',
@query = 'select * from Employees for xml auto',
@templatefile = 'c:\temp\template.tpl'
```

where:

- `@outputfile` - the full path to file XML.
- `@query` – the **`sql query`** for the necessary data.
- `@templatefile` - a file template to use for creating the XML file.

7. Run the query by pressing **F5** or by selecting **Query | Execute** from the menu.

The `myxmlfile.xml` file is created in the `c:\temp\` folder.

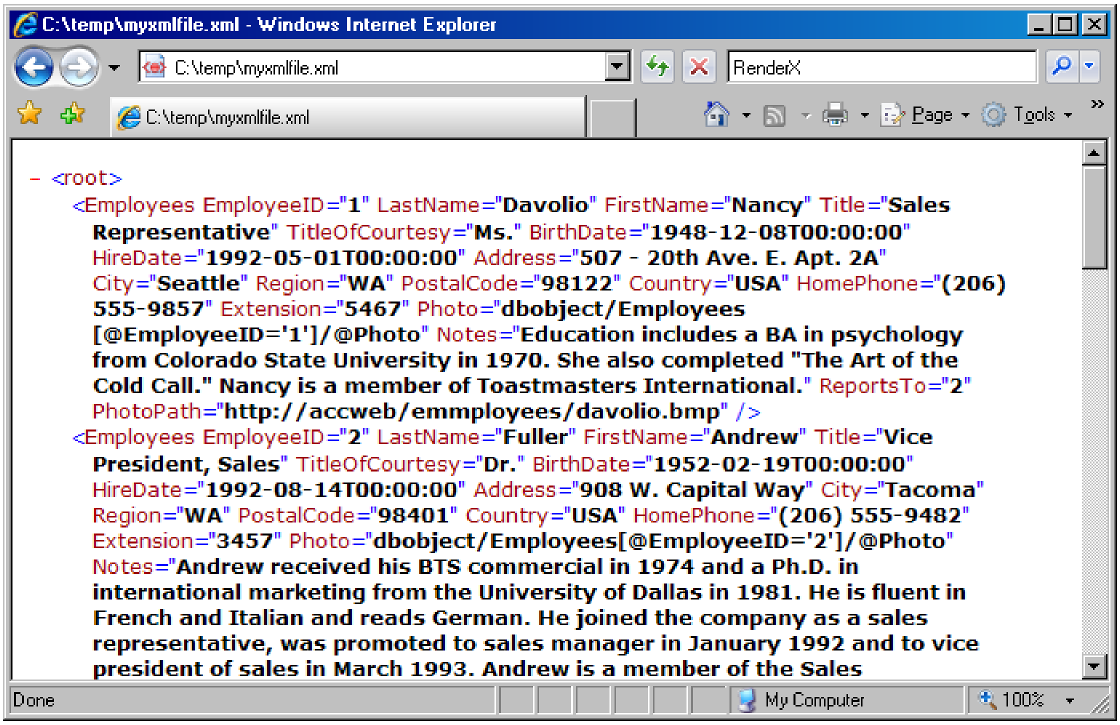


```
EXEC sp_makewebtask @outputfile = 'c:\temp\myxmlfile.xml',
                @query = 'select *  from Employees for xml auto',
                @templatefile = 'c:\temp\template.tpl'
```

```
The command(s) completed successfully.
```

Figure 7.7. Query to export table data

8. The Employees table data is exported to XML format.



```
- <root>
    <Employees EmployeeID="1" LastName="Davolio" FirstName="Nancy" Title="Sales
      Representative" TitleOfCourtesy="Ms." BirthDate="1948-12-08T00:00:00"
      HireDate="1992-05-01T00:00:00" Address="507 - 20th Ave. E. Apt. 2A"
      City="Seattle" Region="WA" PostalCode="98122" Country="USA" HomePhone="(206)
      555-9857" Extension="5467" Photo="dbobject/Employees
      [@EmployeeID='1']/@Photo" Notes="Education includes a BA in psychology
      from Colorado State University in 1970. She also completed "The Art of the
      Cold Call." Nancy is a member of Toastmasters International." ReportsTo="2"
      PhotoPath="http://accweb/emmployees/davolio.bmp" />
    <Employees EmployeeID="2" LastName="Fuller" FirstName="Andrew" Title="Vice
      President, Sales" TitleOfCourtesy="Dr." BirthDate="1952-02-19T00:00:00"
      HireDate="1992-08-14T00:00:00" Address="908 W. Capital Way" City="Tacoma"
      Region="WA" PostalCode="98401" Country="USA" HomePhone="(206) 555-9482"
      Extension="3457" Photo="dbobject/Employees[@EmployeeID='2']/@Photo"
      Notes="Andrew received his BTS commercial in 1974 and a Ph.D. in
      international marketing from the University of Dallas in 1981. He is fluent in
      French and Italian and reads German. He joined the company as a sales
      representative, was promoted to sales manager in January 1992 and to vice
      president of sales in March 1993. Andrew is a member of the Sales
```

Figure 7.8. Resulting XML file

9. Continue to next sample to build a *VisualXSL* document.

7.3. Example of Using Radio Buttons

When creating fillable PDF forms, you may need to be able to add radio buttons in your form. Unlike check boxes, radio buttons are grouped so that only one of them can be selected at a time. Creating radio buttons in *VisualXSL* is similar to creating checkboxes; the only difference is that the **Test** property of these elements must have the same variable name in their **XPath expressions** (but different values to compare with). Another way to create **Radio Buttons** is to give the same **Group Name** for all **Checkboxes**.

Radio Buttons are created only if their **`Treat as Field`** property is set **`true`**.

This example assumes you are starting from an existing project. To start a new project, see Creating a New VisualXSL Project and How to Create a New Project.

Prepare a number of reports in W9 form from existing data (in the `C:\Documents and Settings\All Users\Documents\VisualXSL Projects\W9` folder. A sample PDF file with a W9 form (for the background) and an XML file from the database already exist.) For information about preparing an XML source and PDF background, see Example of Creating XML Source and PDF Background and Example of Exporting XML Data File from SQL Server.

7.4. Example of Building a VisualXSL Project(W9 form)

This example assumes you are starting from an existing project. To start a new project, see Creating a New VisualXSL Project and How to Create a New Project.

The example covers the design and publishing of documents based on the US Government W9 form from existing data in the

`..\All Users\Shared Documents\VisualXSL Projects\samples\W9` folder

or

`..\Users\Public\Public Documents\VisualXSL Projects\Samples\W9` folder.

A sample PDF file with a W9 form (for the background) and an XML file from the database already exist. For information about preparing an XML source and PDF background, see Example of Creating XML Source and PDF Background and in Example of Exporting XML Data File from SQL Server.

Procedure 7.1. To build a VisualXSL project

1. Create a new folder in `C:\Documents and Settings\All Users\Documents\VisualXSL Projects` for the project and call it.
2. Copy the `w9src.xml` and `w9_blank.pdf` files from the `C:\Documents and Settings\All Users\Documents\VisualXSL Projects` to the new folder.
3. Open the `w9src.xml` with an XML Editor or WordPad (not Notepad) to study its content. Note the XML declaration `xml version="1.0" encoding="utf-8"`, a good practice for it to exist, and with the correct encoding value. Note the XML element `batch` at the beginning and `batch` at the end of the file. A top level XML element as noted is required by *VisualXSL* even if only one set of XML children elements exist.

4. Open the `w9_blank.pdf` with Adobe Reader, or equivalent, to become familiar with its content and the various "blank" fields. Close and Exit Adobe Reader.

5. Double-click the *VisualXSL* icon on your desktop.

   The *VisualXSL* application opens.

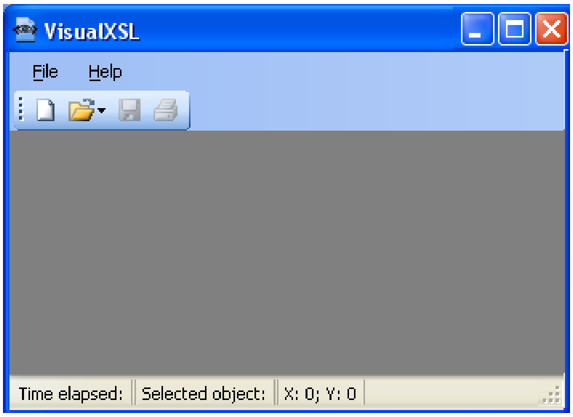
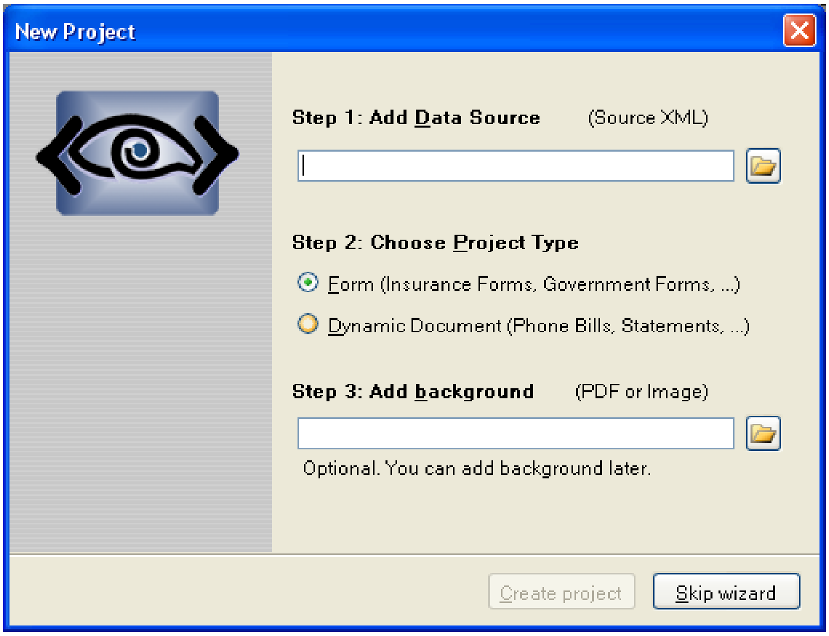


Figure 7.9. Empty VisualXSL application

6. Click the **New Project** button.

   The **New Project** wizard opens.



Step 1: Add Data Source (Source XML)

Step 2: Choose Project Type

● Form (Insurance Forms, Government Forms, ...)

○ Dynamic Document (Phone Bills, Statements, ...)

Step 3: Add background (PDF or Image)

Optional. You can add background later.

Figure 7.10. New Project wizard

7. Add the data source.

   Under **`Step 1 : Add Data Source`** click the **Open** button, navigate to the `..\VisualXSL Projects\Samples\myw9` new folder and select the `w9src.xml` file.

Alternatively, you can skip the wizard, and add the data source later by using the **File | Open Source XML** menu to select the `w9src.xml` file from the newly created folder.

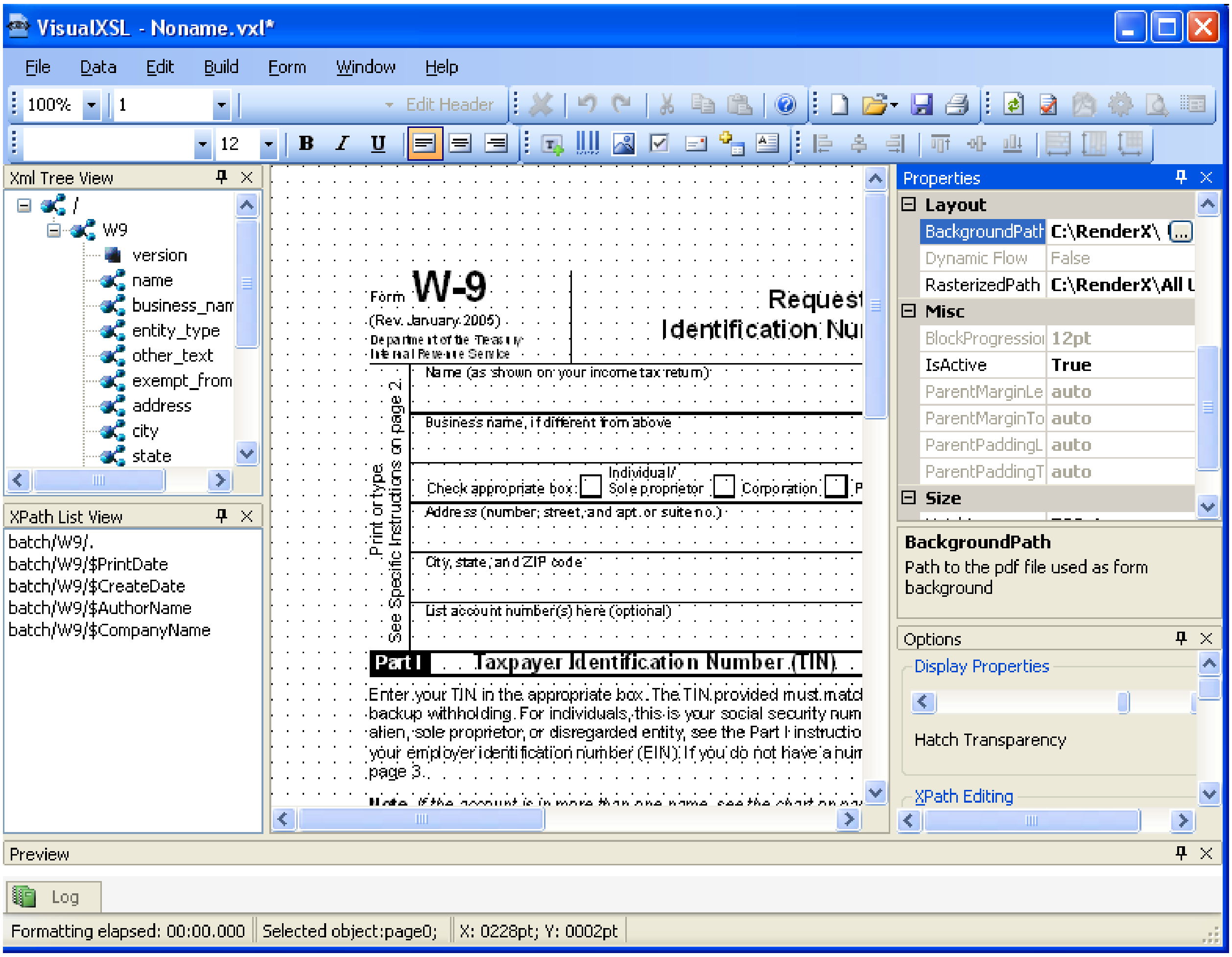
`Step 2 : Choose Project Type` Form (Insurance Forms, Government Forms,...) for this project the default selection is correct because in the next procedure we will choose a PDF background that is a Form.

8. Add the background (PDF or image).

   Under **`Step 3`** click the **Open** button ,navigate to the `..\VisualXSL Projects\Samples\myw9` new folder and select the `w9src.xml` file.

   Note that this step automatically set the **BackgroundPath** and **RasterizedPath** property (under **Properties | Layout**).

9. Click the **Create Project** button. The new project, yet unnamed, opens in *VisualXSL*.


VisualXSL - Noname.vxl*
File
Data
Edit
Build
Form
Window
Help
100%
1
Edit Header
12
Xml Tree View
/
W9
version
name
business_nam
entity_type
other_text
exempt_from
address
city
state
XPath List View
Form W-9
(Rev. January 2005)
Department of the Treasury
Internal Revenue Service
Request
Identification Nu
Name (as shown on your income tax return)
Business name, if different from above
Check appropriate box: Individual/ Sole proprietor Corporation
Address (number, street, and apt. or suite no.)
City, state, and ZIP code
List account number(s) here (optional)
Print or type
See Specific Instructions on page 2.
Part I Taxpayer Identification Number (TIN)
Enter your TIN in the appropriate box. The TIN provided must matc
backup withholding. For individuals, this is your social security num
alien, sole proprietor, or disregarded entity, see the Part I instructio
your employer identification number (EIN). If you do not have a num
page 3.
Properties
Layout
Misc
Size
Options
Display Properties
Hatch Transparency
XPath Editing
Preview
Log

batch/W9/.
batch/W9/$PrintDate
batch/W9/$CreateDate
batch/W9/$AuthorName
batch/W9/$CompanyName

BackgroundPath	C:\RenderX\
Dynamic Flow	False
RasterizedPath	C:\RenderX\All L
BlockProgressio	12pt
IsActive	True
ParentMarginLe	auto
ParentMarginTo	auto
ParentPaddingL	auto
ParentPaddingT	auto

BackgroundPath
Path to the pdf file used as form background

Formatting elapsed: 00:00.000 | Selected object:page0; | X: 0228pt; Y: 0002pt

Figure 7.11. New project page

10. Click **File | Save as**, and save the project as `1stTry.vxl`.

11. Click in the **Layout** (the **Designer**). The page properties are shown in the **Properties** pane.

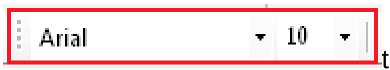
12. If you skipped **`Step 6`**, add the background now. In the **Properties | Layout**, select the **BackgroundPath** property, click the ... button, and open the `w9_blank.pdf` file.

    The application automatically rasterizes PDFs into a PNG image with a filename as noted in the **Properties | Layout|RasterizedPath** property. This Raster image is used as the document background in the **Layout (Designer Area).**

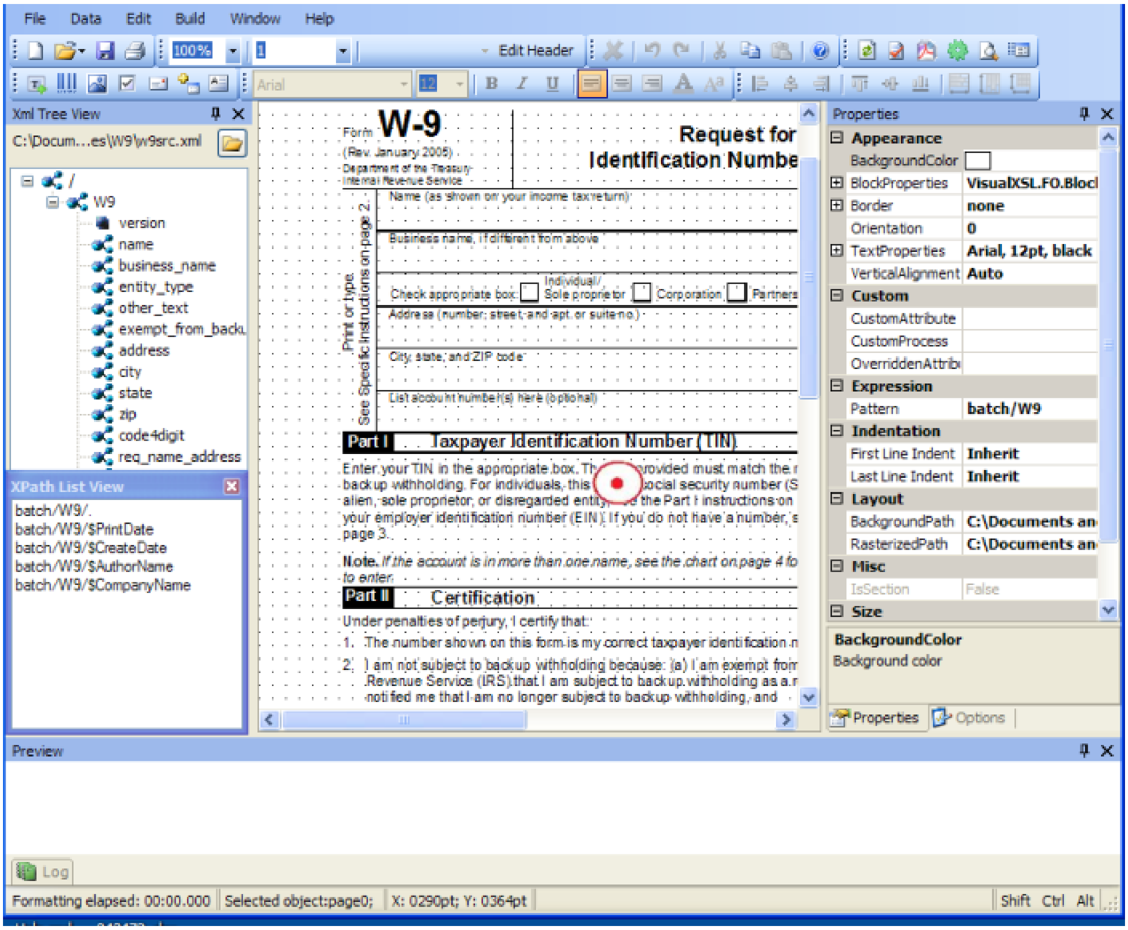
13. In the **XML Tree View** pane, find the **name** node, and drag-and-drop it to the **Layout**. Place it, and resize the **name** data field to fit the template in the background form.

14. Change the selected data field's properties by clicking in the **Properties** pane.

    Expand the **Appearance** property, and then the **TextProperties** property group. You can change the font settings for the selected data field by clicking the ... button, and making desired changes in the `Font` dialog box to **`Arial, 10pt`**.

    Alternatively , you can select the `Font size` through the  Arial 10 toolbar. Because you are going to add data fields with the same formatting, change the section formatting properties (**Appearance | TextProperties | Font**)

    To do so move the cursor to an area in **Layout** away from any fields and select (left mouse button)- see Figure 7.12 below for suggested location shown as ⊙


File
Data
Edit
Build
Window
Help
100%
1
Edit Header
Arial
12
B
I
U
Xml Tree View
C:\Docum…es\W9\w9src.xml
/
W9
version
name
business_name
entity_type
other_text
exempt_from_backu
address
city
state
zip
code4digit
req_name_address
XPath List View
Form W-9
(Rev. January 2005)
Department of the Treasury
Internal Revenue Service
Request for
Identification Numbe
Name (as shown on your income tax return)
Business name, if different from above
Check appropriate box: □ Individual/ Sole proprietor □ Corporation □ Partners
Address (number, street, and apt. or suite no.)
City, state, and ZIP code
List account number(s) here (optional)
Print or type
See Specific Instructions on page 2.
Part I Taxpayer Identification Number (TIN)
Enter your TIN in the appropriate box. Th… provided must match the r
backup withholding. For individuals, this … social security number (S
alien, sole proprietor, or disregarded entity … the Part I instructions on
your employer identification number (EIN). If you do not have a number, s
page 3.
Note. If the account is in more than one name, see the chart on page 4 fo
to enter.
Part II Certification
Under penalties of perjury, I certify that:
1. The number shown on this form is my correct taxpayer identification n
2. I am not subject to backup withholding because: (a) I am exempt fror
Revenue Service (IRS) that I am subject to backup withholding as a r
notified me that I am no longer subject to backup withholding, and
Properties
Options
Preview
Log
Shift
Ctrl
Alt

batch/W9/.
batch/W9/$PrintDate
batch/W9/$CreateDate
batch/W9/$AuthorName
batch/W9/$CompanyName

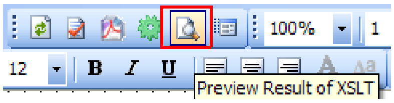
Properties	
⊟ **Appearance**	
BackgroundColor	□
⊞ BlockProperties	**VisualXSL.FO.Blocl**
⊞ Border	**none**
Orientation	**0**
⊞ TextProperties	**Arial, 12pt, black**
VerticalAlignment	**Auto**
⊟ **Custom**	
CustomAttribute	
CustomProcess	
OverriddenAttrib	
⊟ **Expression**	
Pattern	**batch/W9**
⊟ **Indentation**	
First Line Indent	**Inherit**
Last Line Indent	**Inherit**
⊟ **Layout**	
BackgroundPath	**C:\Documents an**
RasterizedPath	**C:\Documents an**
⊟ **Misc**	
IsSection	False
⊟ **Size**	

BackgroundColor
Background color

Formatting elapsed: 00:00.000 | Selected object:page0; | X: 0290pt; Y: 0364pt

Figure 7.12. Text formatting properties of the selected data field

15. To view a preview of the project at this stage, click the **Preview Result of XSLT** button in the toolbar.



The **Log** pane you can monitor the formatting process by tracing messages from the *XEP* formatter. When the *XEP* formatter finishes, the Preview pane shows the resulting document. You may now see how the field is placed on the page.

The Log may report a "[Warning]The project page 2 has no frames. It may be rendered not properly." message. This simply means that your document contains a page on which no XML node is rendered. The original US Government W9 form consist of four pages of which page 2 through 4 consist of instructions.

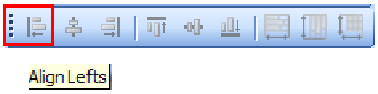
Click 💾 to save the project.

By default, to generate a preview only the first three records from the XML are used. You may change this behavior via the **Options** pane. Be advised that increasing number of records reduces processing of preview, so it is only recommended if your data records vary significantly.

16. Drag-and-drop the rest of the XML nodes to the **Layout** *(note that same are purposely not listed at this time): business_name, address, city, req_name_address, account_number, soc_sec_number* and *empl_ident_number* and resize them to fit the empty spaces on the background.

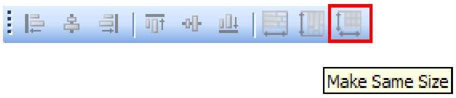
    To align the *Business_Name, Address, City* and *list account number(s)* fields to be left aligned with the *Name* field:

    - Select the *Name* field. While `Ctrl`-key is depressed select the *Business Name, Address, City* and *list account number(s)* fields. Release the `Ctrl`-key. Click on the `Align Lefts` button.

    To size the *Name* and *Business Name* fields to be the same width and height:

    - Select the *Name* field. While `Ctrl`-key is depressed select the *Business Name* field. Release the `Ctrl`-key. Click on the `Make Same Size` button.

17. The address must be correctly formed from the **`city`**, **`state`** and **`ZIP`** values. Select the `city` field and change the **Expression|XPathValue** property to:

    concat (city/.,', ',state/.,' ',zip/.,'-',code4digit/.)

    The XSLT code is generated in the template with the `match="batch/W9"` attribute and data acquisition is rendered by the concat (city/.,', ',state/.,' ',zip/.,'-',code4digit/.)XPath expression

    The `concat` function returns the concatenation of its arguments. The string arguments are separated by a comma. Additional strings, like the comma and space between `city` and `state`, the space between `state zip` ,and the dash (-)between `zip code4digit` are between single (')quotes.

18. `The Social security number` *(soc_sec_number)* data field should be justified. To set the proper spacing between digits, select the *soc_sec_number* field and change the **BlockProperties|AlignmentLast** property value to `justify`.

19. Repeat the procedure above for the *Employer identification number (empl_edent_number)* including changing the **BlockProperties|AlignmentLast** property value to `justify`.

    To view a preview of the project at this stage , click **Preview Result of XSLT** button. Adjust the location(Left and Top) of **all** fields.

    - Select a field and use the Up, Down, Left or Right arrow keys. The field moves several points (px)based on Grid size defined in the **Options** panel, irrespective if **Snap to Grid** is selected or not, or
    - Select a field, and while the `Alt-key` is depressed us the Up, Down, Left or Right arrow keys. You will note that the field moves a fraction (0.75px)irrespective of the **Grid** size defined and if **Snap to Grid** in the **Options** panel is selected or not, or

- Select the field and edit the **Location|Top** and **location|Left** values in the **Options** panel. The field moves irrespective of the **Grid** size defined and if **Snap to Grid** in the **Options** panel is selected or not.

Click 💾 to save the project.

20. Place check marks on the appropriate background check boxes so that they are rendered only under certain conditions, according to the XPath expression results.

    Select the *entity_type* XML node in the **XML Tree View** pane. From the *Data* menu select *Add Checkbox.*

    Select the check murk just placed in **Layout** and drag it into the *Individual/Sole proprietor* check box

    - *Check appropriate box;* ☑ *Individual/Sole proprietor.*

    While still selected view the value displayed in the **expression|test** property. It should read `entity_type/.='individual'`. if not, changed it. This is the XPath expression used as the condition to render the field. That is, the field is rendered only if this condition is true.

21. Repeat the procedure above for the remaining check boxes, setting the **Expression|Test** property accordingly as in:

    `entity_type = 'corporation'`,

    `entity_type = 'partnership'`,

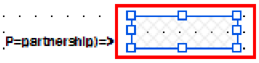
    `entity_type = 'limited'`,

    `entity_type = 'other'`.

    An alternative is to:

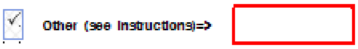
    - Select the first check mark and **`copy`** once and **`past`** four times using one of the actions available.
    - Select and drag each check mark into each check box , and set the **Expression|Test** property accordingly as noted above.

    To view a preview of the project at this stage , click **Preview Result of XSLT** 🔍button.

22. In the **XML Tree View** pane, find the **`other_text`** node, select and drag-and-drop it to the **Layout** here  which is associated with the *Limited liability company* check box.

    While still selected view the value displayed in the **`Expression|XPathValue`** property. It Should read `other_text/`.If not, change it.

    Define a condition when to render the value which is only applicable when the *Limited liability company* check box is checked. Set the **`Expression| Test`** property to `entity_type/.='limited'`. This is the XPath expression used as the condition to render the field. That is, the field is rendered only if this condition is true. If not, change it. This is the XPath expression used as the condition to render the field. That is, the field is rendered only if this condition is true.

23. In the **XML Tree View** pane, find the `other_text/` node, select and drag-and-drop it to the **Layout** here 

which is associated with the *Other* check box. Place it, and resize the **other** data field to fit the template in the background form.

    While still selected view the value displayed in the **`Expression|XPathValue`** property. It Should read `other_text/`.If not,change it.

    Define a condition when to render the value which is only applicable when the *Other* (see instructions) check box is checked. Set the **`Expression| Test`** property to `entity_type/.='other'`. This is the XPath expression used as the condition to render the field. That is, the field is rendered only if this condition is true.

24. Select the **`exempt_from_backup`** XML node in the **XML Tree View** pane. From **`Data`** menu select **`Add Checkbox`**.

    While still selected view the value displayed in the **`Expression| Test`** property. It should read `exempt_from_backup/.='true'`. This is the XPath expression used as the condition to render the field. That is, the field is rendered only if this condition is true.

25. To view a preview of the project at this stage , click **Preview Result of XSLT** button. Adjust the location of **all** fields.

    Click to save the project.

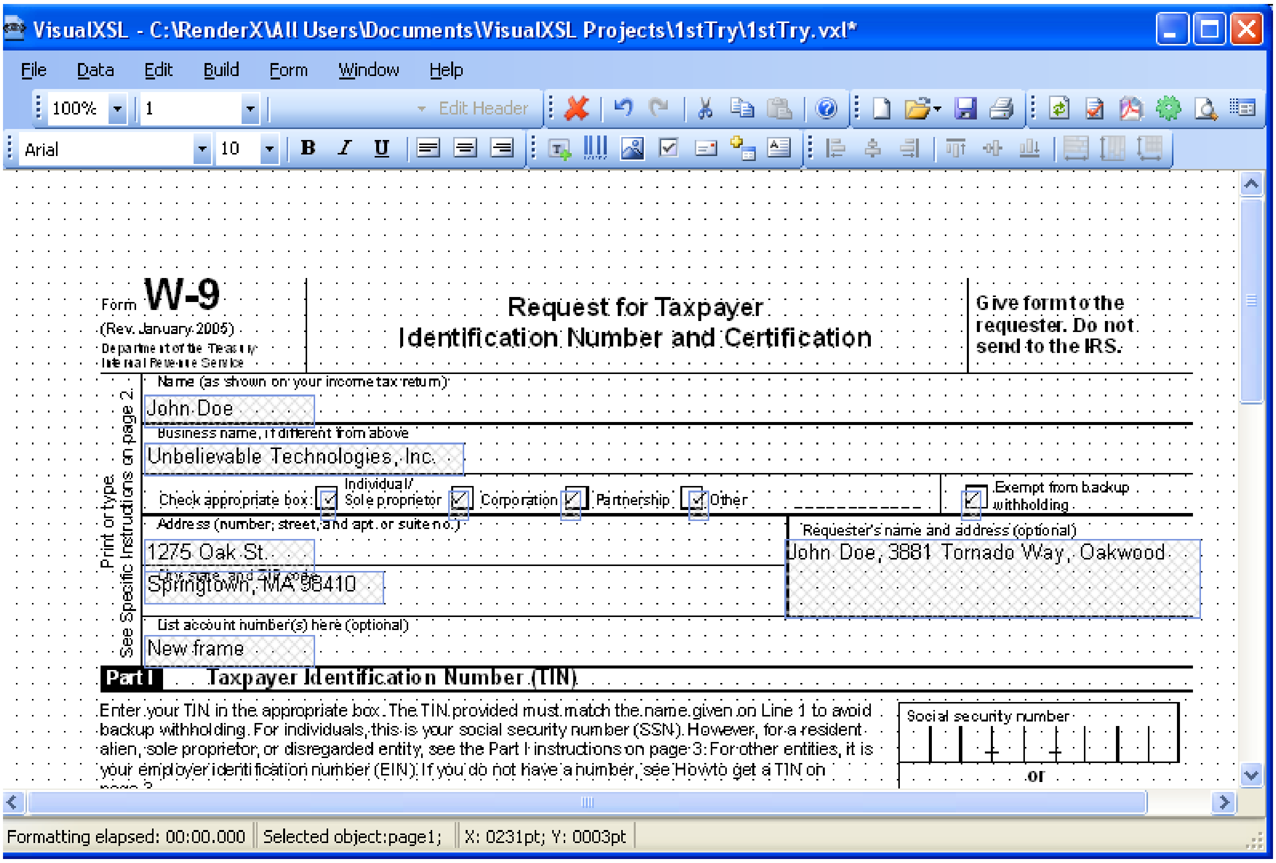
26. After all data fields are added and aligned you can generate the XSL stylesheet and/or the resulting PDF document.

    Click to generate the XSL stylesheet.

    Click to generate the resulting PDF document, enter a filename and save.

    Review the XSL stylsheet and PDF generated.

27. A preliminary view of the project should look similar to that of W9 sample project (except those **`Barcode`** data fields). See **`Figure 7.13`**


VisualXSL - C:\RenderX\All Users\Documents\VisualXSL Projects\1stTry\1stTry.vxl*
File
Data
Edit
Build
Form
Window
Help
100%
1
Edit Header
Arial
10
B
I
U

Form **W-9**
(Rev. January 2005)
Department of the Treasury
Internal Revenue Service

Request for Taxpayer Identification Number and Certification

Give form to the requester. Do not send to the IRS.

Print or type
See Specific Instructions on page 2.

Name (as shown on your income tax return)
John Doe

Business name, if different from above
Unbelievable Technologies, Inc.

Check appropriate box: ☑ Individual/ Sole proprietor ☑ Corporation ☑ Partnership ☑ Other ____________ ☑ Exempt from backup withholding

Address (number, street, and apt. or suite no.)
1275 Oak St.

City, state, and ZIP code
Springtown, MA 98410

Requester's name and address (optional)
John Doe, 3881 Tornado Way, Oakwood

List account number(s) here (optional)
New frame

Part I **Taxpayer Identification Number (TIN)**

Enter your TIN in the appropriate box. The TIN provided must match the name given on Line 1 to avoid backup withholding. For individuals, this is your social security number (SSN). However, for a resident alien, sole proprietor, or disregarded entity, see the Part I instructions on page 3. For other entities, it is your employer identification number (EIN). If you do not have a number, see How to get a TIN on page 3.

Social security number

or

Formatting elapsed: 00:00.000 | Selected object:page1; | X: 0231pt; Y: 0003pt

Figure 7.13. View of unfinished project

Congratulations, you have completed one of many *VisualXSL* projects.

7.5. Example of Building a VisualXSL Project (ICW form)

This example assumes you are starting from an existing project. To start a new project, see Creating a New VisualXSL Project and How to Create a New Project.

The example covers the design and publishing of documents based on the ICW form from existing data in the

`..\All Users\Shared Documents\VisualXSL Projects\samples\ICW` folder

or

`..\Users\Public\Public Documents\VisualXSL Projects\Samples\ICW` folder.

A sample PDF file with a ICW form and an XML file from the database already exist. For information about preparing an XML source and PDF background, see Example of Creating XML Source and PDF Background and in Example of Exporting XML Data File from SQL Server.

Procedure 7.2. To build a Dynamic type of VisualXSL project

1. Create a new folder in `C:\Documents and Settings\All Users\Documents\VisualXSL Projects\my ICW` for the project and call it.

2. Copy the `ICW.xml` and `Images` folder files from the `C:\Documents and Settings\All Users\Documents\VisualXSL Projects` to the new folder.
3. Open the `ICW.xml` with an XML Editor or WordPad or Notepad with "Word Wrap" option enabled to study its content. Note the XML declaration `xml version="1.0" encoding="utf-8"`, a good practice for it to exist, and with the correct encoding value. Note the XML element `batch` at the beginning and `batch` at the end of the file. A top level XML element as noted is required by *VisualXSL* even if only one set of XML children elements exist.
4. Open the `w9_blank.pdf` with Adobe Reader, or equivalent, to become familiar with its content and the various "blank" fields. Close and Exit Adobe Reader.
5. Double-click the *VisualXSL* icon on your desktop.

   The *VisualXSL* application opens.

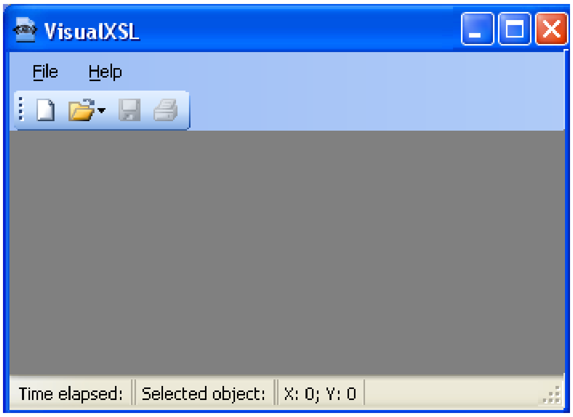
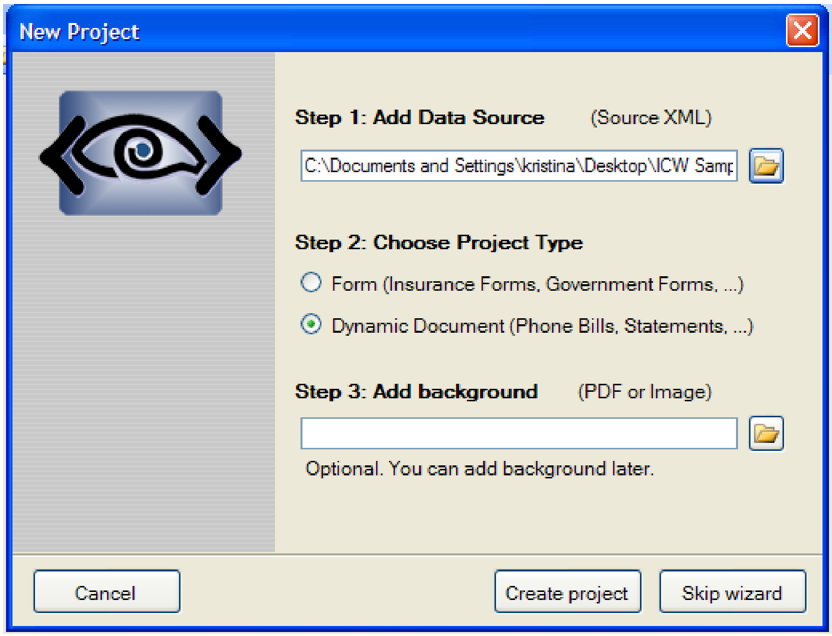


Figure 7.14. Empty VisualXSL application

6. Click the **New Project** 🗋 button.

   The **New Project** wizard opens.



New Project

Step 1: Add Data Source (Source XML)

C:\Documents and Settings\kristina\Desktop\ICW Samp

Step 2: Choose Project Type

○ Form (Insurance Forms, Government Forms, ...)

● Dynamic Document (Phone Bills, Statements, ...)

Step 3: Add background (PDF or Image)

Optional. You can add background later.

Figure 7.15. New Project wizard

7. Add the data source.

   Under `Step 1 : Add Data Source` click the **Open** button, navigate to the `..\VisualXSL Projects\Samples\ICW` new folder and select the `ICW.xml` file.

   Alternatively, you can skip the wizard, and add the data source later by using the **File | Open Source XML** menu to select the `ICW.xml` file from the newly created folder.

   `Step 2 : Choose Project Type` Dynamic document(phone bills, statements)

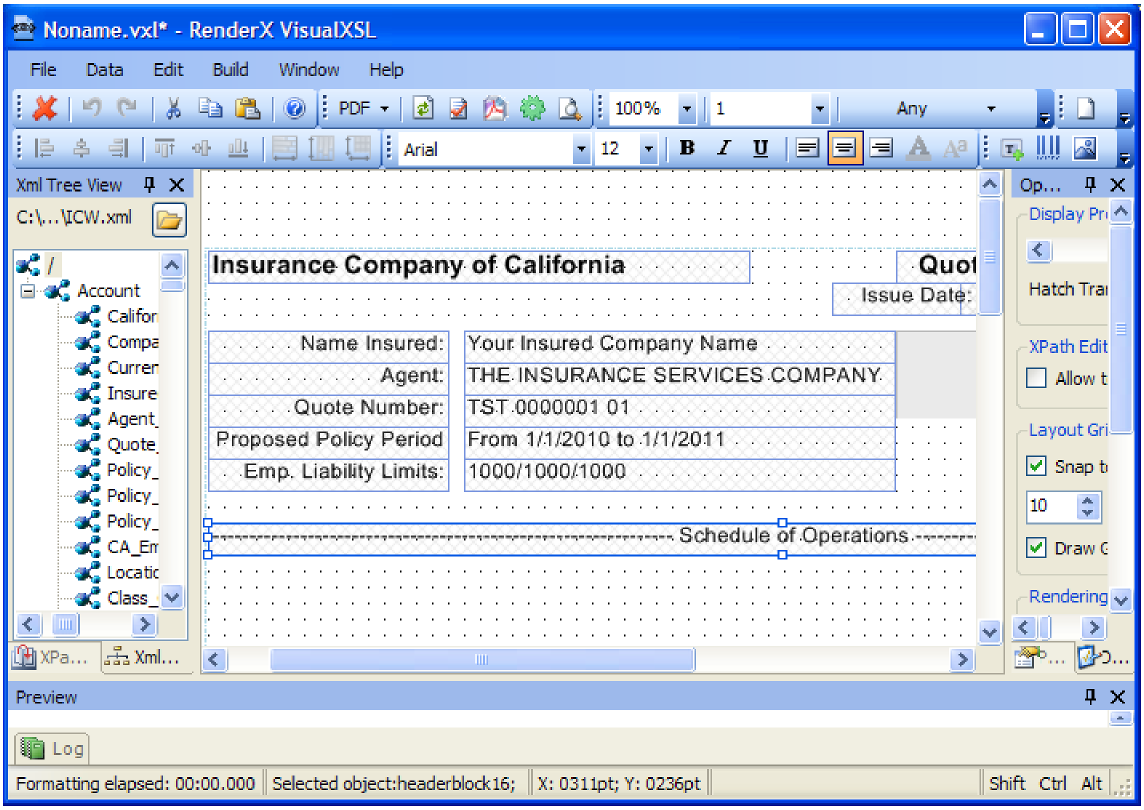
8. Under `Step 3` click the **Open** button ,navigate to the `..\VisualXSL Projects\Samples\ICW` new folder and select the `ICW.xml` file.

   Note that this step automatically set the **BackgroundPath** and **RasterizedPath** property (under **Properties | Layout**).

9. Click the **Create Project** button. The new project, yet unnamed, opens in *VisualXSL*.
10. Click **File | Save as**, and save the project as `2ndTry.vxl`.
11. Click in the **Layout** (the **Designer**). The page properties are shown in the **Properties** pane.
12. To Create a header select `Edit header` Edit Header button in the tool bar.

    You can manually arrange items in the desire place on the field.

13. In the **XML Tree View** pane, find the **name** node, and drag-and-drop it to the **Layout**. Place it, and resize the **name** data field to fit the template in the background form.
14. Change the selected data field's properties by clicking in the **Properties** pane.
15. Add Image(add rx-logo.png from `images` folder),in the **Data** menu select **Add Image** , and in the **Properties** pane, under **Data** select `ImageUrl`.



Insurance Company of California

Quot

Issue Date:

Name Insured:	Your Insured Company Name
Agent:	THE INSURANCE SERVICES COMPANY
Quote Number:	TST 0000001 01
Proposed Policy Period	From 1/1/2010 to 1/1/2011
Emp. Liability Limits:	1000/1000/1000

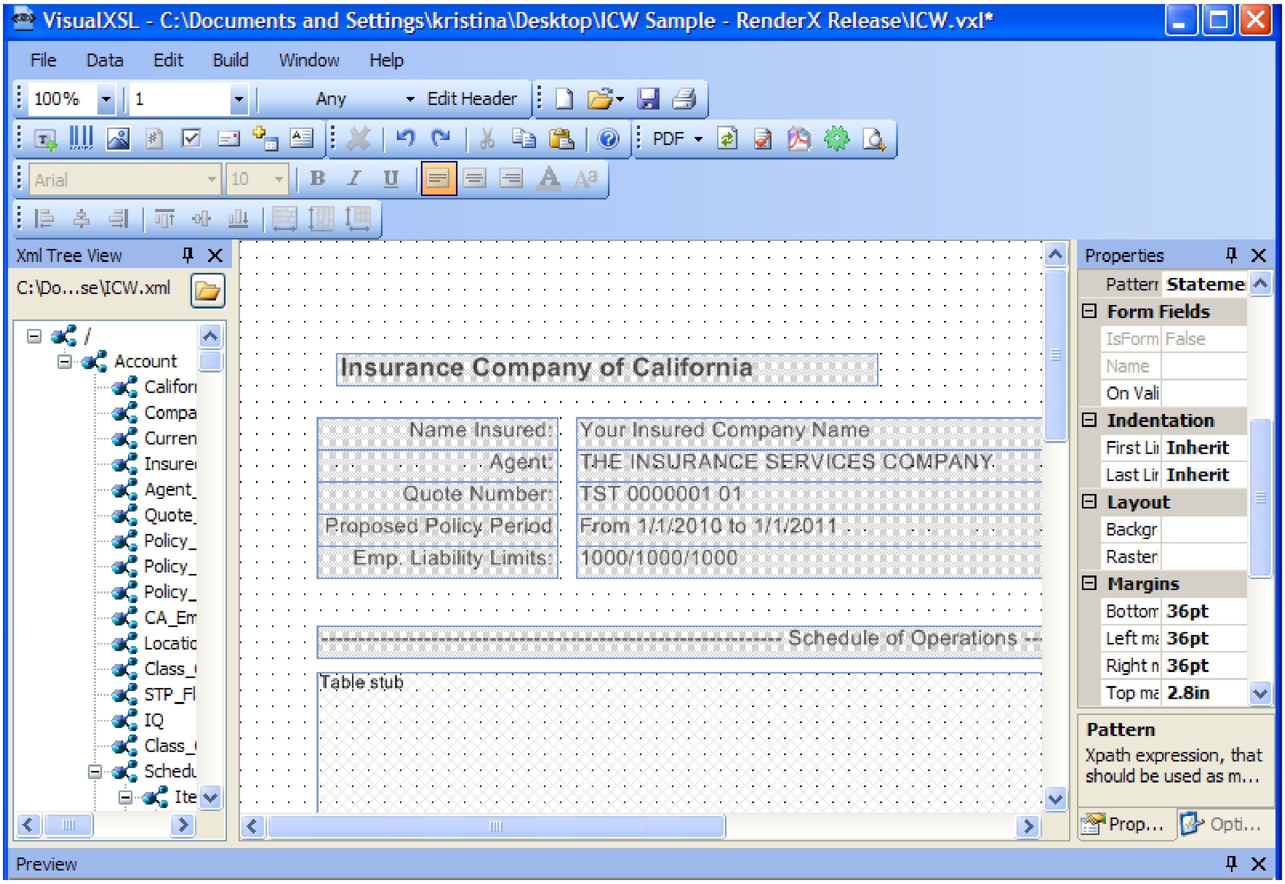
Schedule of Operations

Formatting elapsed: 00:00.000 | Selected object:headerblock16; | X: 0311pt; Y: 0236pt

Figure 7.16. Creating a Header

16. Unselect **Edit Header**.
17. Edit Items.

    The Items will dynamically be arranged by turns on the layout.



VisualXSL - C:\Documents and Settings\kristina\Desktop\ICW Sample - RenderX Release\ICW.vxl*

Insurance Company of California

Name Insured:	Your Insured Company Name
Agent:	THE INSURANCE SERVICES COMPANY
Quote Number:	TST 0000001 01
Proposed Policy Period	From 1/1/2010 to 1/1/2011
Emp. Liability Limits:	1000/1000/1000

-- Schedule of Operations --

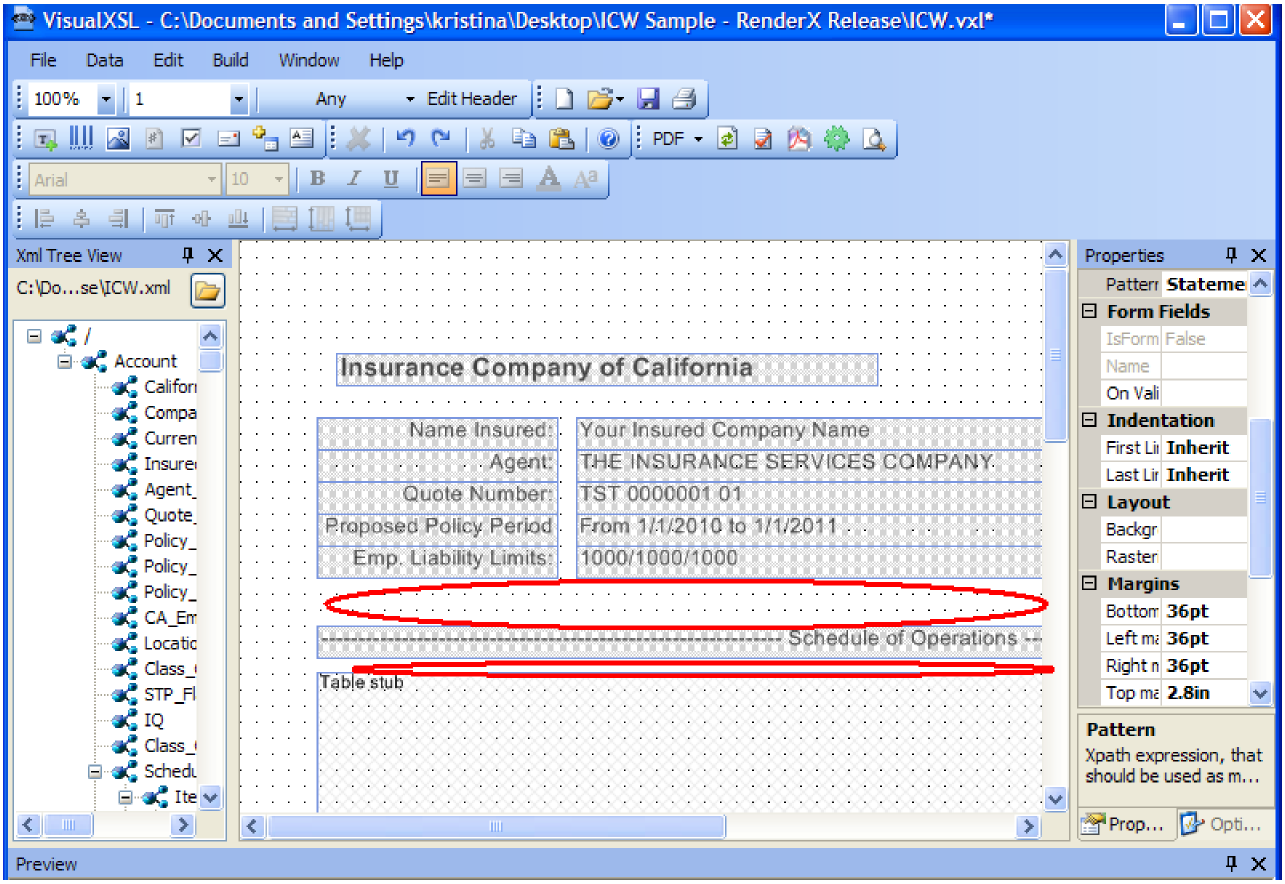
Table stub

Patterr	Stateme
Form Fields	
IsForm	False
Name	
On Vali	
Indentation	
First Lii	Inherit
Last Lir	Inherit
Layout	
Backgr	
Raster	
Margins	
Bottom	36pt
Left ma	36pt
Right n	36pt
Top ma	2.8in

Pattern
Xpath expression, that should be used as m...

Figure 7.17. Dynamic document

18. in the **Properties|Space|Spaces before Area** or **Spaces after Area**



VisualXSL - C:\Documents and Settings\kristina\Desktop\ICW Sample - RenderX Release\ICW.vxl*

Insurance Company of California

Name Insured:	Your Insured Company Name
Agent:	THE INSURANCE SERVICES COMPANY
Quote Number:	TST 0000001 01
Proposed Policy Period	From 1/1/2010 to 1/1/2011
Emp. Liability Limits:	1000/1000/1000

-- Schedule of Operations --

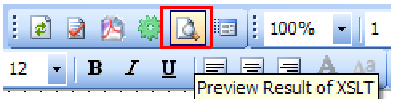
Table stub

Patterr	Stateme
Form Fields	
IsForm	False
Name	
On Vali	
Indentation	
First Lii	Inherit
Last Lir	Inherit
Layout	
Backgr	
Raster	
Margins	
Bottom	36pt
Left ma	36pt
Right n	36pt
Top ma	2.8in

Pattern
Xpath expression, that should be used as m...

Figure 7.18. Spaces between the items

19. To preview the project at this stage, click the **Preview Result of XSLT** button in the toolbar.



The **Log** pane helps you to monitor formatting process by tracing messages from the *XEP* formatter. When the *XEP* formatter finishes, the Preview pane shows the resulting document. You may now see how the field is placed on the page.

Click 💾 to save the project.

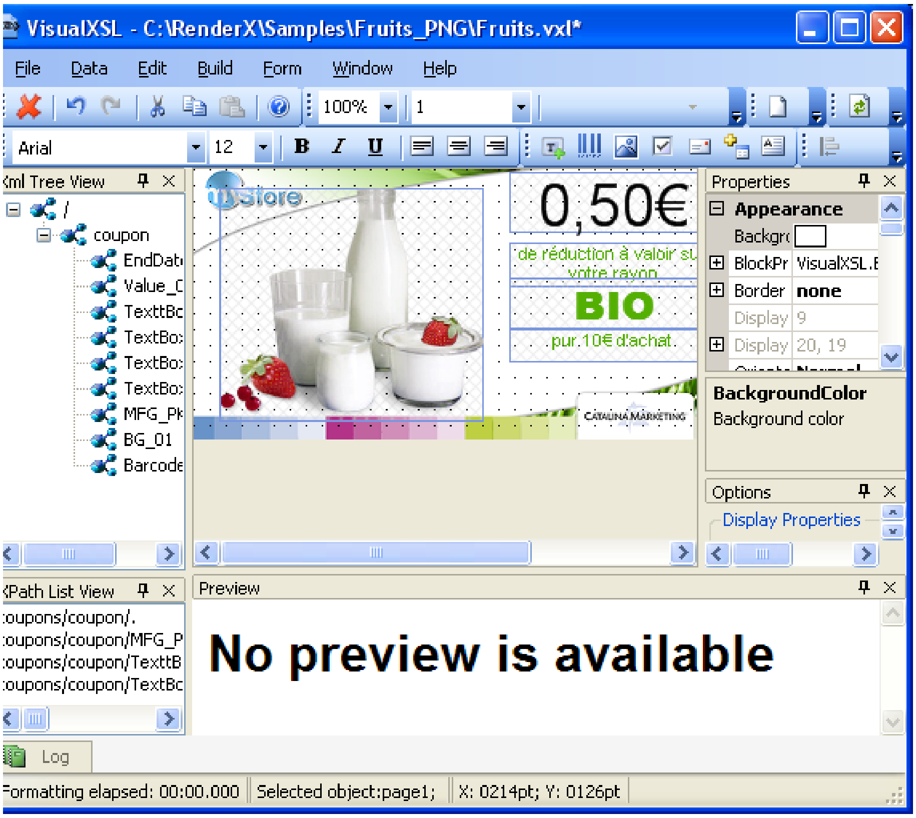
You can find other instructions at [To build a *VisualXSL* project]

7.6. Example of Using XEP and SVG Files As Image

VisualXSL supports using both XEP intermediate format files and SVG graphics files as images. The technique is the same for both XEP intermediate format and SVG files. We illustrate using XEP file; for SVG file the scenario is similar.

Procedure 7.3. To use .XEP file as an image

1. In the `Sample` folder, open the `FRUITS.png` file.



Backgr	□
BlockPr	VisualXSL.E
Border	none
Display	9
Display	20, 19

BackgroundColor
Background color

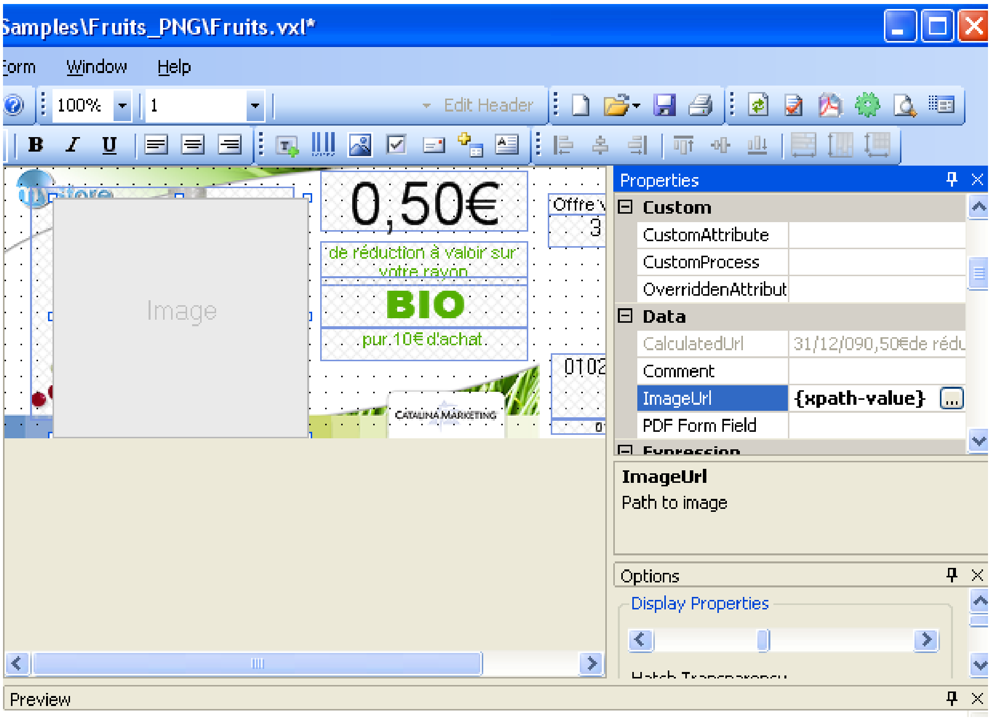
coupons/coupon/.
coupons/coupon/MFG_P
coupons/coupon/TexttB
coupons/coupon/TextBc

No preview is available

Formatting elapsed: 00:00.000 | Selected object:page1; | X: 0214pt; Y: 0126pt

Figure 7.19. Adding an Image

2. In the **Data** menu select **Add Image**, and in the **Properties** pane, under **Data**, select **`ImageUrl`**.



Custom
CustomAttribute
CustomProcess
OverriddenAttribut
Data
CalculatedUrl: 31/12/090,50€de rédu
Comment
ImageUrl: {xpath-value}
PDF Form Field
Expression

ImageUrl
Path to image

Figure 7.20. After Adding an Image and Selecting the `ImageUrl` Field

3. In **`ImageUrl`**, select the ... button; the following dialog box appears.

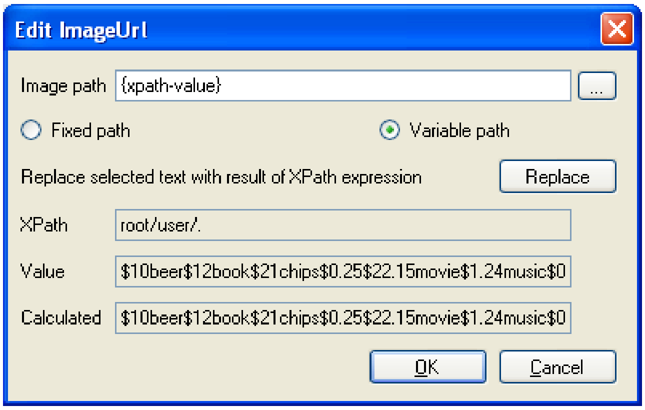


Image path: {xpath-value}

○ Fixed path ● Variable path

Replace selected text with result of XPath expression

XPath: root/user/.

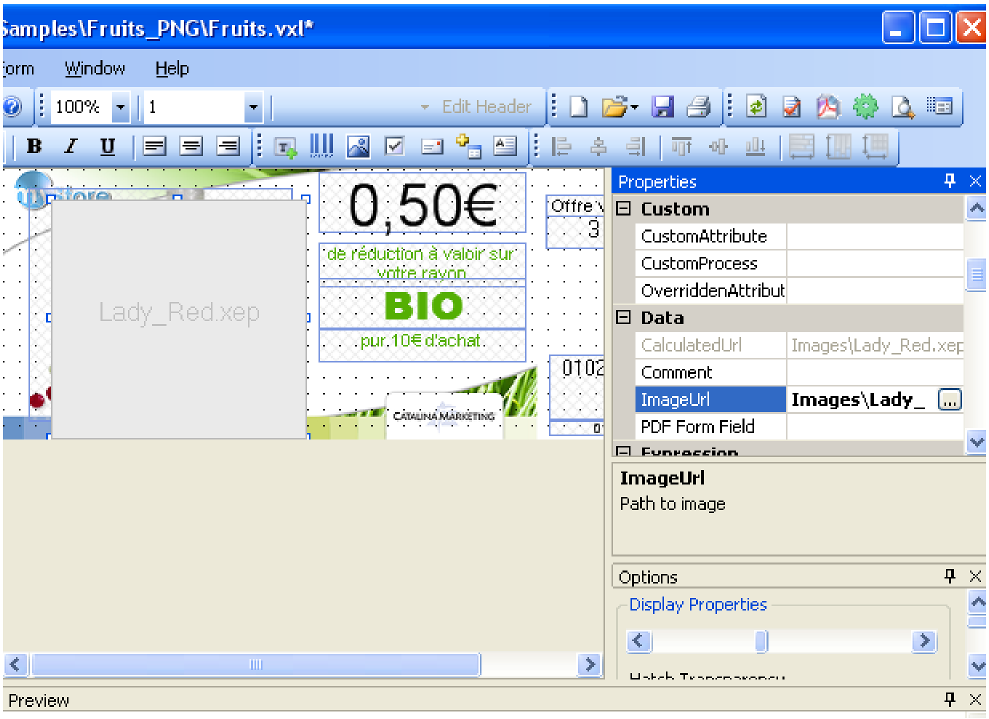
Value: $10beer$12book$21chips$0.25$22.15movie$1.24music$0

Calculated: $10beer$12book$21chips$0.25$22.15movie$1.24music$0

Figure 7.21. Edit ImageUrl dialog box

4. Select **`Fixed Path`** and via **`Image path`** field browse to and select `Fruits_PNG/Images/Lady_Red.xep` file. It may be necessary to change **`Files of type:`** in the file browser window, in order to show .XEP files.

5. In the **Edit ImageUrl** dialog box, click **OK**; the dialog box closes, and the **`Image path`** in the **Edit ImageUrl** window and the **`ImageUrl`** in the **Properties** pane are set `Lady_Red.xep`.



⊟ Custom	
CustomAttribute	
CustomProcess	
OverriddenAttribut	
⊟ Data	
CalculatedUrl	Images\Lady_Red.xep
Comment	
ImageUrl	Images\Lady_
PDF Form Field	
⊟ Expression	

ImageUrl
Path to image

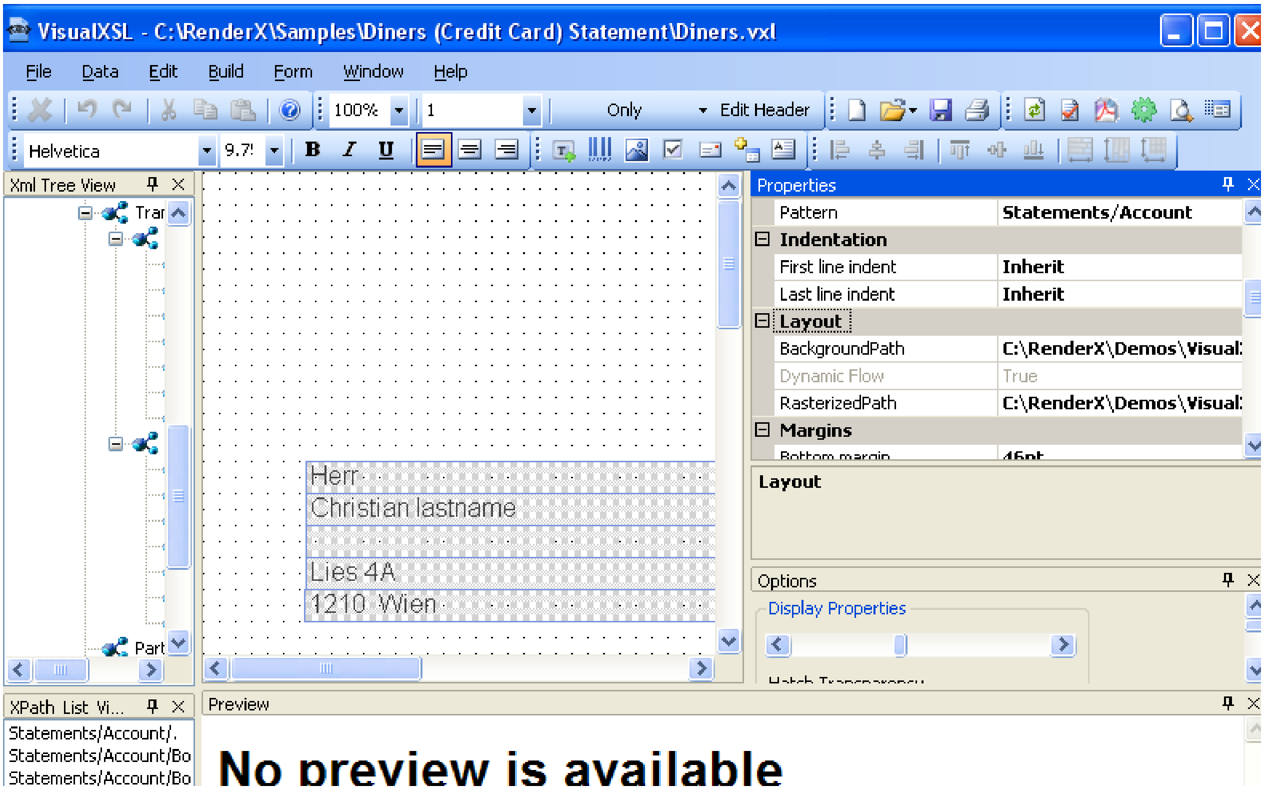
Figure 7.22. Lady_Red.xep has been selected for the new image

6. To see the `Lady_Red.xep` image, in the **Build** menu you can click **Preview**; after completion, `Lady_Red.xep` is visible below in the **Preview** pane.

7.7. Example of Using Templates

For information about dynamic flows, see How to Create Dynamic Templates section.

In the `Samples` folder, open the `Diners (Credit Card) Statement` file.



Properties

Pattern	Statements/Account
⊟ Indentation	
First line indent	Inherit
Last line indent	Inherit
⊟ Layout	
BackgroundPath	C:\RenderX\Demos\Visual:
Dynamic Flow	True
RasterizedPath	C:\RenderX\Demos\Visual:
⊟ Margins	
Bottom margin	46pt

Layout

Figure 7.23. Diners (Credit Card) Statement file

In **Choose Section Template**, the template **`Only`** is shown. note the properties values, in the **Layout** and **Margins** property groups.

In **Choose Section Template**, change from template **`Only`** to **`First`**, **`Last`** and **`Any`**, and notice that the values in the **Properties** pane are changed.

For an example of how the templates work together, see the `Diners_Sample_by_RenderX.pdf` file. The templates that are used for each statement are determined by *VisualXSL*, according to how many sections of data are to be presented for that statement.

7.8. Example of Using Custom Attributes

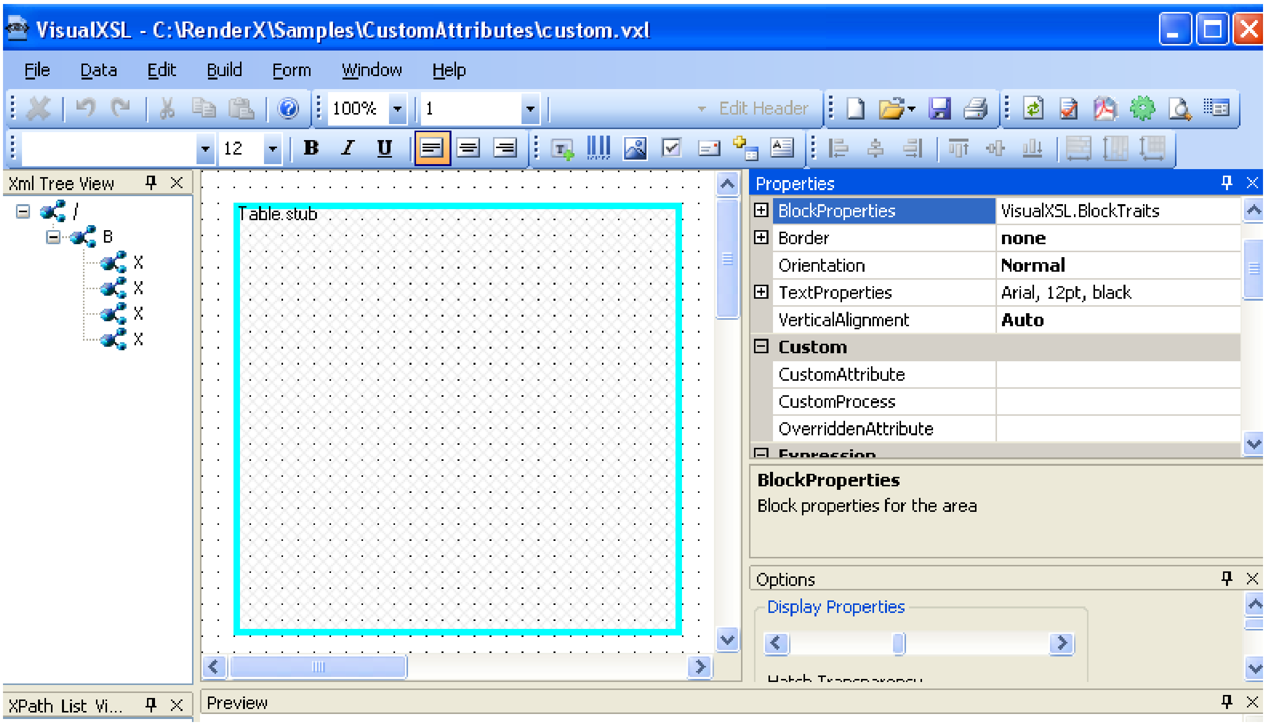
VisualXSL now supports three **`custom attributes`** properties (that is, attributes not yet natively supported by *VisualXSL*) for any object type. These properties (**`CustomAttribute`**, **`OverriddenAttribute`** and **`CustomProcess`**) are described in the Data Fields and Properties chapter.

The following is an example of using **`CustomAttribute`** and **`OverriddenAttribute`**:

Procedure 7.4. To use custom attributes

1. In the `Samples` folder, open the `CustomAttributes` file.

   i All of the borders of the rectangle are the same - **`4pt solid cyan`**. We will use custom properties to access the **FO** style attributes for the individual border sides.



VisualXSL - C:\RenderX\Samples\CustomAttributes\custom.vxl

Properties	
⊞ BlockProperties	VisualXSL.BlockTraits
⊞ Border	none
Orientation	Normal
⊞ TextProperties	Arial, 12pt, black
VerticalAlignment	Auto
⊟ Custom	
CustomAttribute	
CustomProcess	
OverriddenAttribute	

BlockProperties

Block properties for the area

Figure 7.24. CustomAttributes file, with table with cyan border

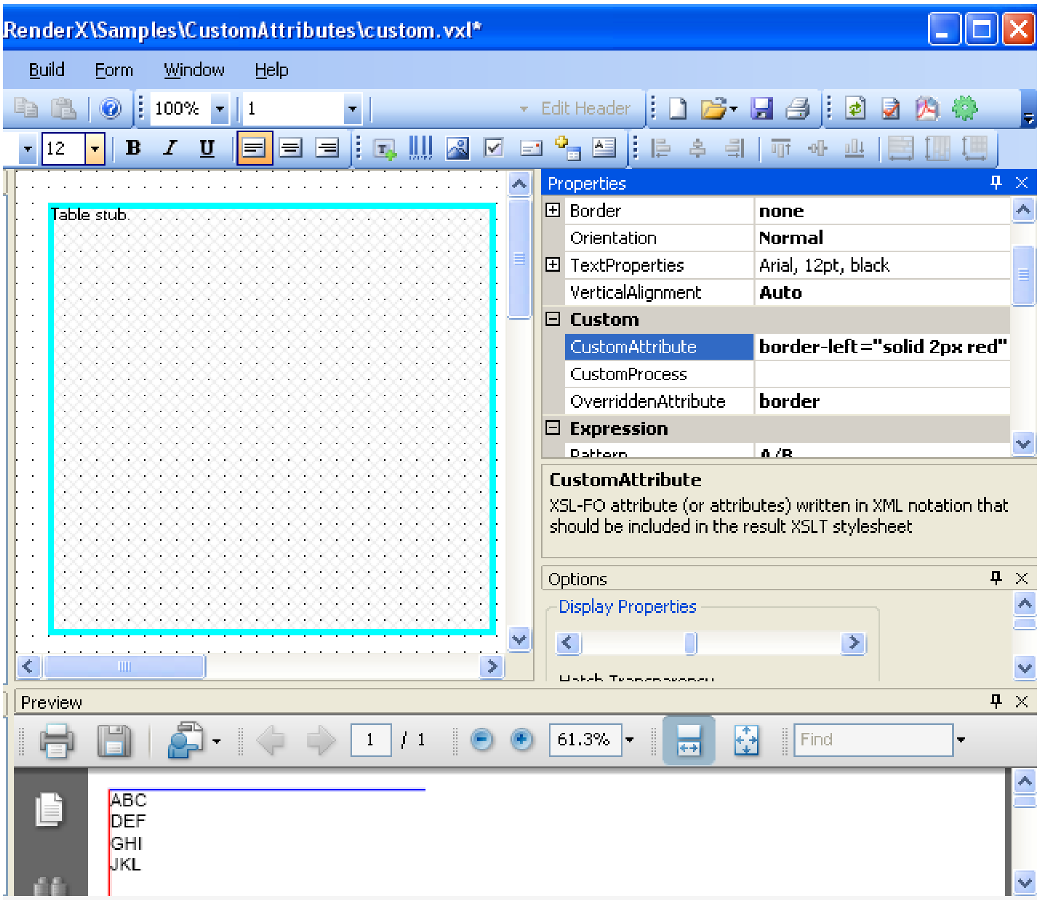
2. In the **Properties** pane, under **Custom**, select **`CustomAttributes`**.

   Change the left border to **`2px red`** and the top border to **`2px blue`** by entering the following value: `border-left="solid 2px red" border-top="solid 2px blue"`. Note the space separating the two values.

3. In the **Properties** pane, under **Custom**, select **`OverriddenAttribute`**.

   Remove the rest of the (`cyan`) border altogether by entering the following value: `border`.

4. Do either **Build Preview** or **Build Create PDF**. The updated frame's left border is red, top is blue and nothing remains of the cyan. The updated frame can be seen in the **Preview** pane.



⊞ Border	none
Orientation	Normal
⊞ TextProperties	Arial, 12pt, black
VerticalAlignment	Auto
⊟ Custom	
CustomAttribute	border-left="solid 2px red"
CustomProcess	
OverriddenAttribute	border
⊟ Expression	
Pattern	A/B

CustomAttribute
XSL-FO attribute (or attributes) written in XML notation that should be included in the result XSLT stylesheet

ABC
DEF
GHI
JKL

Figure 7.25. Preview, showing updated frame, with red left border, and blue top border

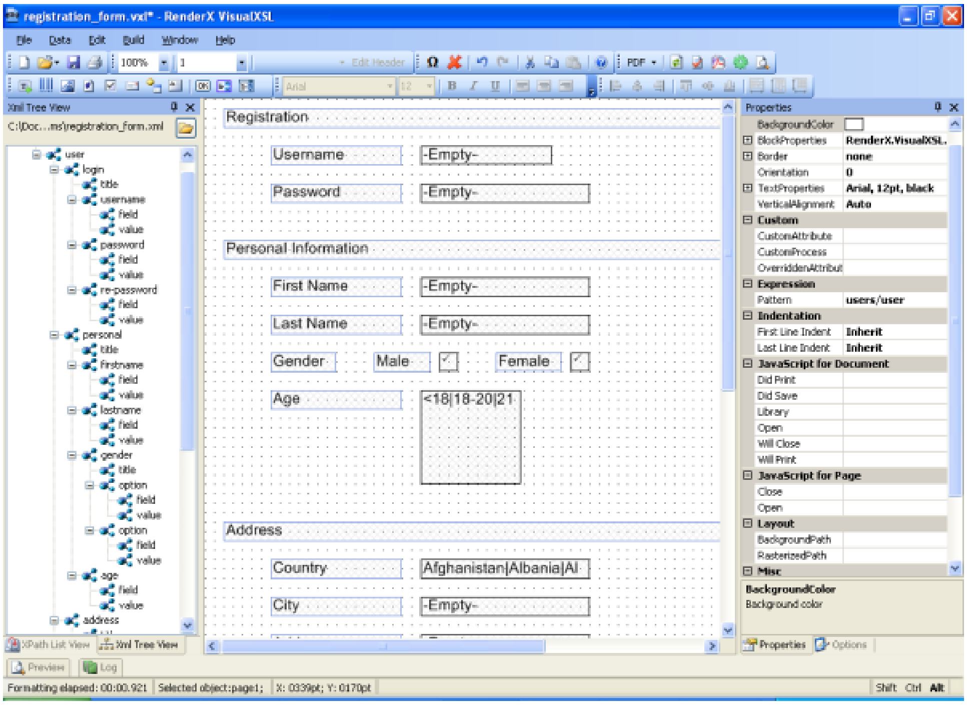
7.9. Example of Creating PDF Form

VisualXSL 2.3 and upper versions provide a new, easy to use method of creating fillable PDF Forms. Now the procedure of creating a PDF Form is the same as creating any PDF document.

This example shows how a simple PDF form can be created and submitted to the server. To submit the form, you need to have access to a server, where the data is submitted and handled.

Procedure 7.5. To create a PDF Form

1. In the `Samples/Registration Form` folder open `registration_form.vxl` file via *VisualXSL*.



Properties

BackgroundColor	□
⊞ BlockProperties	RenderX.VisualXSL.
⊞ Border	none
Orientation	0
⊞ TextProperties	Arial, 12pt, black
VerticalAlignment	Auto
⊟ **Custom**	
CustomAttribute	
CustomProcess	
OverriddenAttribut	
⊟ **Expression**	
Pattern	users/user
⊟ **Indentation**	
First Line Indent	Inherit
Last Line Indent	Inherit
⊟ **JavaScript for Document**	
Did Print	
Did Save	
Library	
Open	
Will Close	
Will Print	
⊟ **JavaScript for Page**	
Close	
Open	
⊟ **Layout**	
BackgroundPath	
RasterizedPath	
⊟ **Misc**	

BackgroundColor
Background color

Formatting elapsed: 00:00.921 | Selected object:page1; | X: 0339pt; Y: 0170pt

Figure 7.26. Registration Form

2. There are several frames, which assemble a simple registration form. The first part has a title called **`Registration`**, which contains two pairs of fields (each pair contains two fields in front each other). In this example all the pairs contain a usual text field and a fillable field (e.g. fillable text frame, checkbox, listbox, ...). Here we will pay attention to the fillable fields; more information about **Text Fields** can be found at 5.4 Text Frame.

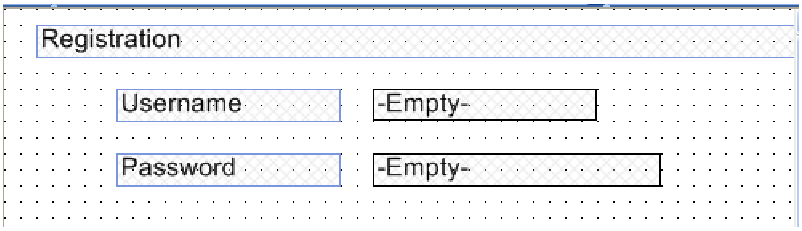
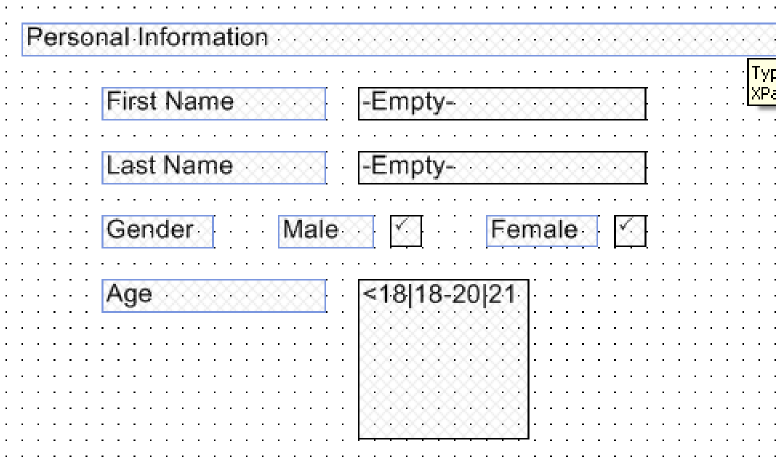


Figure 7.27. Registration Form

3. **`Username`** - fillable text field. To create a fillable text field add a **Text Frame** to the **Layout** and set its **Treat as Field** property to **`true`** (**`Treat as Field`** is located in the **Form Fields** section of the **Properties** pane). *VisualXSL* sets a default value for the **Name** property, but as we need to handle the submitted data on the server, we should give names for the submitting frames. So, the name of the **Username** field is set to **`username`**. For all other form fields we set individual **Name** properties. We can also give the maximum allowed length of the entered text: navigate to the **Properties** pane, **Form Fields** section, **Max Length** property, for this field its value is **15**. Besides, usually **Username** is a required field, to emphasis that, we gave to **Required** property the **`true`** value.

4. **Password** - password field. Password field is created from a fillable text field. To create a **Password**, create a fillable text field, like the **Username** field and set its **Password** property to **true**
5. The second part of the form is called **Personal Data**. The first two pairs are **First Name** and **Last Name**. They are created just like the **Username** field.



Personal Information

First Name -Empty-

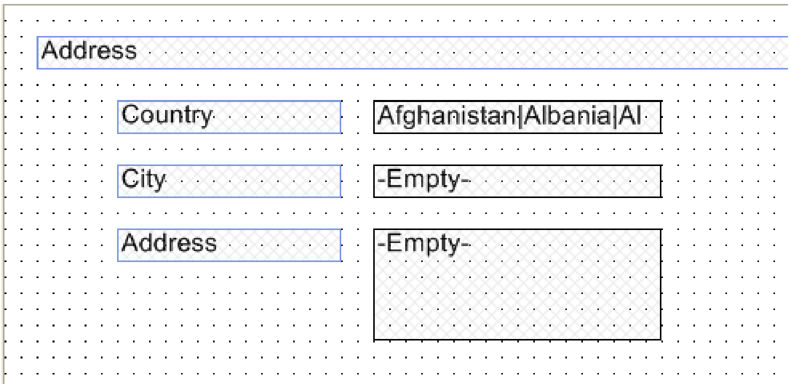
Last Name -Empty-

Gender Male ☑ Female ☑

Age <18|18-20|21

Figure 7.28. Registration Form

6. Then comes **Gender** fields, which is a group of **Radio Buttons**. There are used text frames and two radio buttons. Detailed information about how to create **Radio Buttons** see [7.3. Example of Using Radio Buttons](#).
7. The last pair of this section is **Age**. The user is suggested to select his/her age from the given intervals, which are presented as options of a **List Box**. To create a **List Box** select the XML node from which you wish the options to be generated and click **Add List Box**. The values are taken from the source XML (`registration_form.xml`). *VisualXSL* sets the current XML node's inner text as the **List Box's Values**. Options are generated from the **Values** property on the **Properties** pane, its values are separated by **Separator** property. Here the age intervals are separated by **|** and as the **Separator** property is **|**, we get the desired intervals - **<18**, **18-20**, **21-25**, **26-30**, **31-40**, **>40**.
8. **Address** section contains information about the user's address and consits of three pairs of fields.



Address

Country Afghanistan|Albania|Al

City -Empty-

Address -Empty-

Figure 7.29. Registration Form

Country - **Combo Box** containing a list of countries. To create a **Combo Box** select the XML node from which you wish the options to be generated and click **Add Combo Box**. The values are taken from the source XML (`registration_form.xml`). *VisualXSL* sets the current XML node's inner text as the **Combo Box's Values**. Options are generated from the **Values** property on the **Properties** pane, its values are separated by **Separator** property. Here country names are separated by **|** and as the **Separator** property is **|**, we get the desired intervals - **Afghanistan**, **Albania**, **Algeria**, etc. .

9. **City** - Fillable Text field.
10. **Address** - Fillable Text field, designed for the user to write his/her address (Street, building, apartment, etc.). The only difference from other fillable fields is that it is multiline, that is - it can contain more that one line. To enable this feature set the **Multiline** property in the **Properties** pane **true**.
11. The **Sign** section contains a **Text Frame** and a fillable **Check Box**, that is a **Check Box**, which **Treat as Field** property is set **true**.

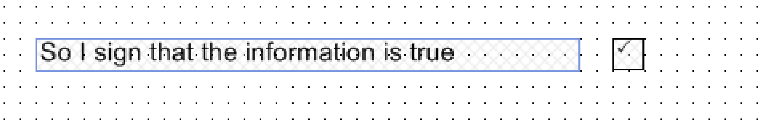


Figure 7.30. Registration Form

12. At the bottom of the form there are 2 **Buttons** - **Reset** and **Submit**.

    Reset - this type of **Button** is used to reset all the fields to their default values. To create a **Reset** button, add a **Button**, in the **Properties** pane set its **Button Type** property **Reset**. **Reset** affects only the fields, which names are included in the **Fields** property (space deliminated list); or if it is left blank, all the fields of the document are affected. As in this example **Fields** property is left blank, all the fields are affected.

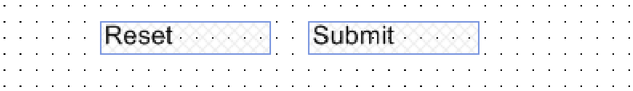
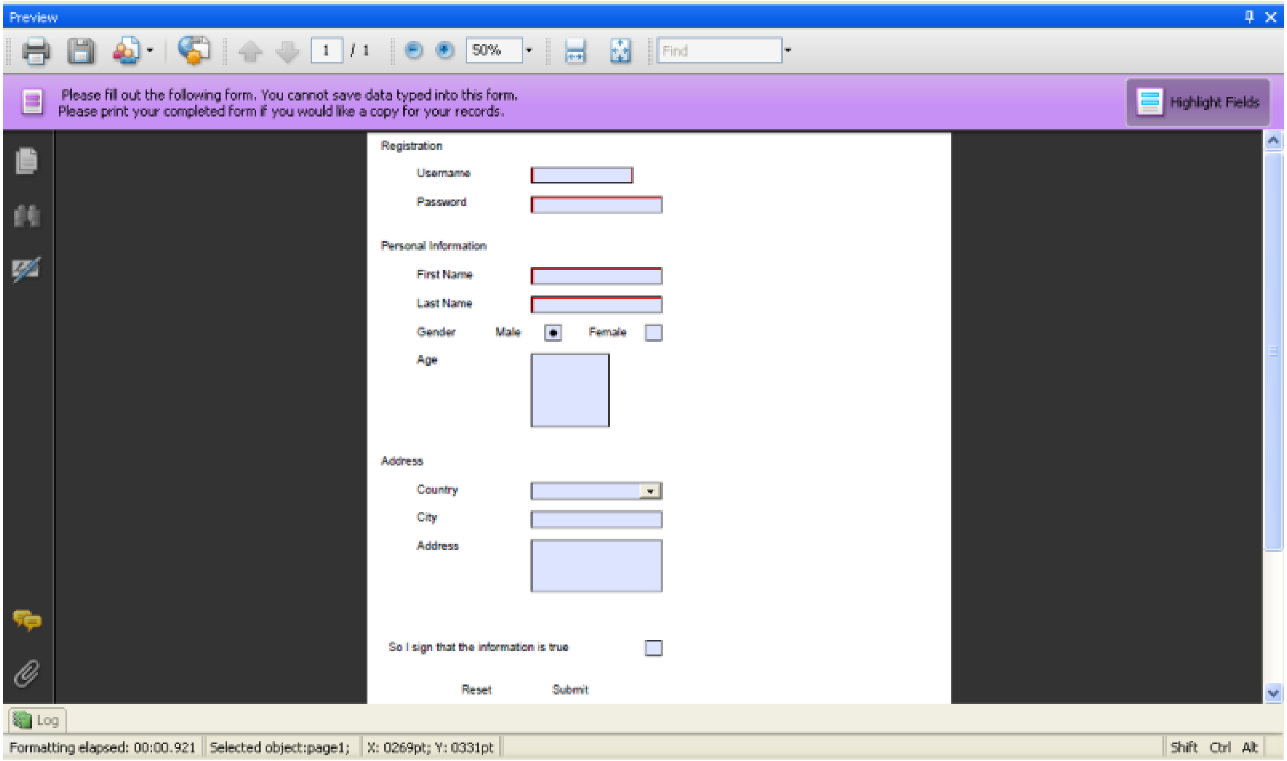


Figure 7.31. Registration Form

13. **Submit** - this type of **Button** is used to submit the form data to the server. To create a **Submit** button, add a **Button**. The server's URL is set in the **URL** property of the **Properties** pane. By default it is the inner text of the XML node from which the button is created. In this example, it is `http://localhost/forms/index.php`, of course you may set your own server. The submittion method is set via **Method** property, its possible values are **POST** and **GET**. Here we use **POST** method. *VisualXSL* supports four formats for submitting: **XFDF**, **FDF**, **PDF** and **HTML**. Here we used **HTML** format. The submitting format is changed from the **Submit Format** property on the **Properties** pane.
14. After creating the form, generate the resulting PDF document, fill the form and click **Submit** button. The form data will be sent to the server, which Url is given to the **Submit Button's Url** property. This sample includes `visualxslforms/index.php` file, which is a simple `PHP` file to show all the data sent to the server. You can copy - paste it to your php server, to see how the sample works.

Adobe Acrobat Reader does **not** read HTML format, and as in this example we use HTML to show the submitted data, you are strongly recommnded to use either **Preview** window or your web browser.



Please fill out the following form. You cannot save data typed into this form.
Please print your completed form if you would like a copy for your records.

Registration

Username

Password

Personal Information

First Name

Last Name

Gender Male ● Female □

Age

Address

Country

City

Address

So I sign that the information is true □

Reset Submit

Formatting elapsed: 00:00.921 | Selected object:page1; | X: 0269pt; Y: 0331pt

Figure 7.32. Registration Form

Glossary

Glossary and Related Material

XEP

RenderX XEP is an XML to PDF (XSL FO) formatter. It takes input in XML, applies an XSL transformation to build XSL Formatting Objects representation, and then formats the Formatting Objects into PDF or PostScript. XEP supports multiple raster and vector graphic formats. Among them is Scalable Vector Graphics (SVG), an XML-based vector graphics representation that is widely used in business applications and for fine typesetting.

XSL-FO

XSL-FO (XSL Formatting Objects) is an XML vocabulary for the formatting of documents. XSL-FO is part of XSL, so that the usual way to produce XSL-FO documents is by transforming XML documents using XSLT. Although the principles behind XSL and CSS (the other Style Sheet Language created by W3C) are quite different, it is planned to align the formatting model between XSL-FO and CSS, so that formatting engines can be based on the same code, both languages can be used to achieve the same results, and the formatted results will look identical.

XSL

XSL (Extensible Style Language) is a Style Sheet Language that can be used for displaying XML documents. Using XSL is two-step process, the first step being a transformation of the XML document using XSLT, and the second step being the rendering of the result of the transformation, which is done using XSL-FO. XSL covers the same application area as CSS but is much more powerful, because the transformation step (using XSLT) can perform arbitrarily complex transformations of the XML document, whereas CSS is not able to make any structural changes to the XML document.

XSLT

XSLT (XSL Transformations) is a specialized Programming Language for transforming XML documents. Although XSLT is part of XSL, and as such intended to be used for transforming XML documents into XSL-FO for presentation purposes, it is not limited to this application area. XSLT uses XML syntax (that is, it is a Programming Language in XML syntax), even though it is based on DSSSL (which uses a Lisp-like syntax). XSLT is particularly interesting in B2B scenarios, where XML documents must be transformed.

.NET Framework

.NET Framework, created by Microsoft, is a software development platform focused on rapid application development, platform independence, and network transparency. .NET is Microsoft's strategic initiative for server and desktop development for the next decade.

Index