WPF Viewer for Reporting Services 2008/2012
User Guide

Table of Content

System Requirements

Product requires the following configuration of software:

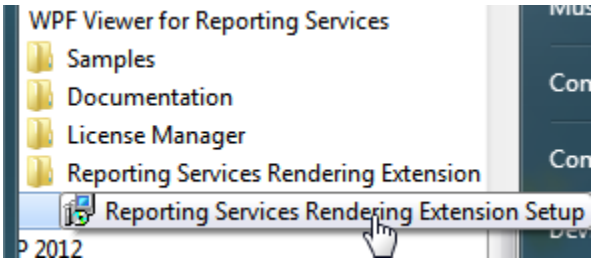
- .NET Framework 4.0.
- Visual Studio 2010/2012
- Microsoft SQL Server Reporting Services 2008/ 2008 R2/ 2012 Standard Edition or higher.
- XAML Rendering Extension installed in the server.

Installing XAML Rendering Extension.

Install PerpetuumSoft Xaml Rendering extension.

> Note: The extension must be installed ONLY for Microsoft SQL Server Reporting Services 2008/2008 R2/2012 Developer, Enterprise or Standard editions. Make sure that your SQL Server edition supports custom rendering extensions.

Run PerpetuumSoft.XamlExtension.msi on the machine where Microsoft SQL Server Reporting Services is installed.



Follow the installer instructions on the screen.

The installer determines your SQL configurations and offers the list of the available Reporting Services instances.

Check/uncheck the instances to install/uninstall Rendering Extension to selected Reporting Services instances and click Apply Changes button.

Restart the affected instances of Reporting Services.

Printing document from WPF Viewer for Reporting Services

There are 2 printing mechanisms available:

1. Print reports as they are represented in the viewer (by default).
2. Print document as it will be represented in the form of exported picture (UseImagePrinting).

```
reportViewer.UseImagePrinting = true;
```

It is possible to set Dpi for pictures (96 dpi is a default value):

```
reportViewer.ImagePrintingDpi = 96;
```

Viewer Setup

To view report in the viewer it is necessary to correctly specify SQL Reporting Services address and user credentials.

```
reportViewer.ReportingServicesUrl = @"http://localhost/ReportServer/ReportExecution2005.asmx";
reportViewer.ReportingServicesCredential = new System.Net.NetworkCredential()
{
    //Write there user credential
    Domain = "",
    UserName = "",
    Password = ""
};
```

```
NOTE: If you will not set ReportingServicesCredential, windows authentication will be used to access reporting services. It does not work if windows authentication is not enabled for the reporting services instance.
```

Reporting Services Timeout

It is possible to set response timeout for Reporting Services using the following property:

ReportingServicesTimeout. Timeout is set in milliseconds .

```
reportViewer.ReportingServicesTimeout = 1000;
```

Adding custom export filters

Third party export filters can be added to the Viewer. They will appear on the toolbar as a separate button.

You need to add the following lines of code to the application:

```
reportViewer.ReportingServicesCustomFilters.Add(new PerpetuumSoft.ReportingServices.V
iewer.WPF.CustomFilterInfo()
{
    DisplayName = "My Filter",
    Name = "XML",
    Image = System.Drawing.Image.FromFile(@"filter.png")
});
```

Commands

Report viewer management commands are available via special Commands property.

To override command you need to inherit from the PerpetuumSoft.ReportingServices.Viewer.WPF.

ReportViewer class and override virtual methods: On<Command_name> and Can<Command_name>.

Example of overridden command ExportToPdf that does nothing:

```
class Class1 : PerpetuumSoft.ReportingServices.Viewer.WPF.ReportViewer
{
    protected override bool CanExportPdf()
    {
        return false;
    }
    protected override void OnExportToPdf()
    {
        //do nothing
    }
}
```

The button is inactive because CanExecute of the ExportToPdf command always gets false.

Automatic report rendering

WPF Viewer for Reporting Services can automatically render documents when report name is changed.

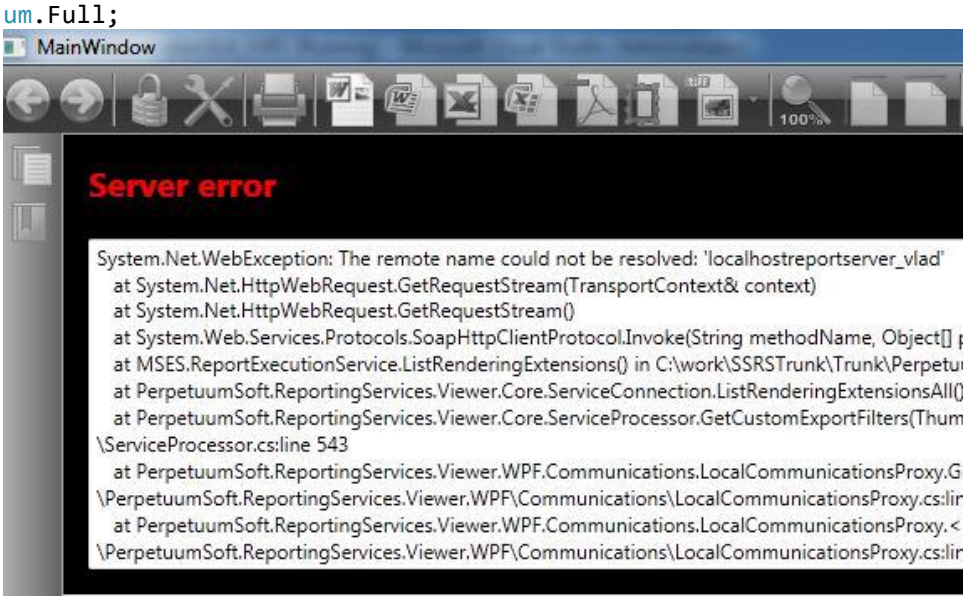
To setup automatic rendering you need to use PerformAutomaticRendering property:

```
reportViewer.PerformAutomaticRendering = true;
```

Report Errors

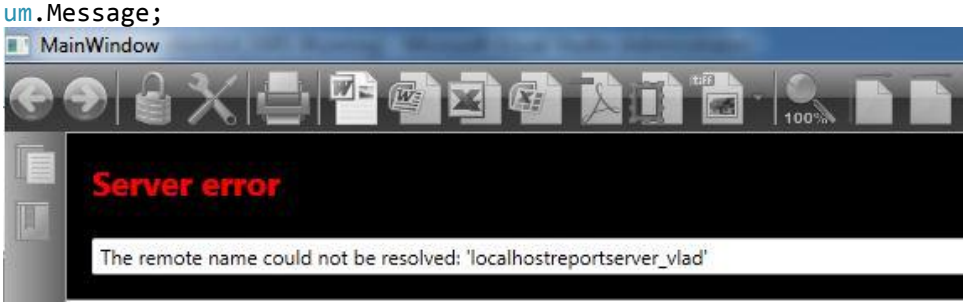
Errors occurred during document rendering can be displayed in the viewer in various ways:

```
reportViewer.DebugMode = PerpetuumSoft.ReportingServices.Viewer.WPF.Utils.DebugModeEnum.Full;
```



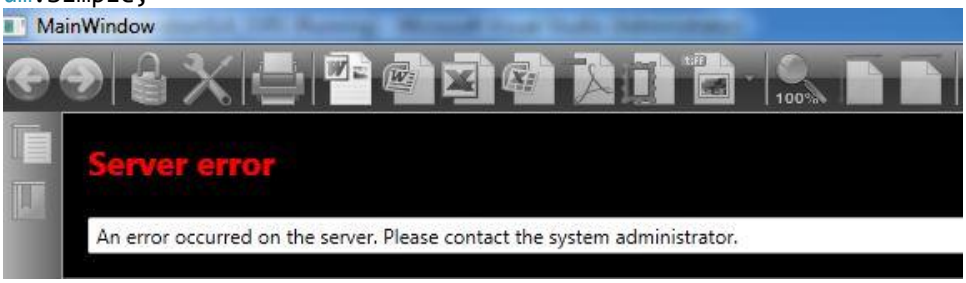
This is a complete error message.

```
reportViewer.DebugMode = PerpetuumSoft.ReportingServices.Viewer.WPF.Utils.DebugModeEnum.Message;
```



This is only an error message without Stack trace.

```
reportViewer.DebugMode = PerpetuumSoft.ReportingServices.Viewer.WPF.Utils.DebugModeEnum.Simple;
```

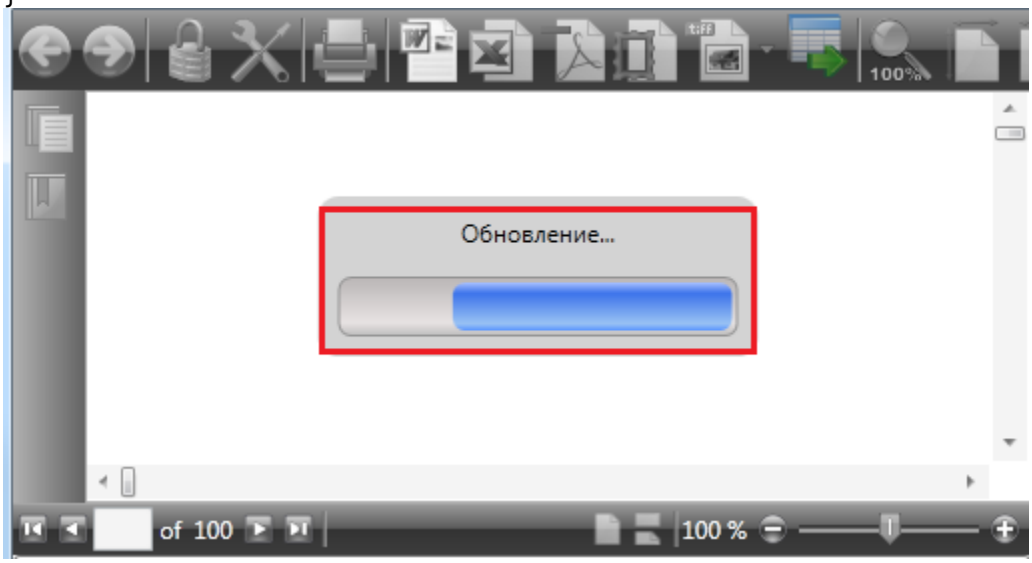


This is a message of an error without specifying what the exact error is. You may use logging mechanism to get detailed information about error occurred.

Localization

It is possible to localize the product:

```
using (var fs = new FileStream(@"ru-RU.xml", FileMode.Open, FileAccess.Read))
{
    reportViewer.SetLocalization(fs);
}
```



You may find English localization file in the Localization subfolder of product installation directory.

Logging

In order to get component's logs, it is necessary to define logger and set it for the viewer via Logger property:

```
reportViewer.Logger = new Logger();
```

You need to implement Ilogger interface to define custom logger:

```
class Logger : PerpetuumSoft.ReportingServices.Viewer.Core.ILogger
{
    public void WriteInfo(string scope, string message)
    {
```

```
            System.Diagnostics.Debug.WriteLine("Info {0} : {1}", scope, message);
        }

        public void WriteError(string scope, Exception error)
        {
            System.Diagnostics.Debug.WriteLine("Error {0} : {1}", scope, error.Message);
        }
    }
```

This implementation of the logger writes data to the Output window of Visual Studio.