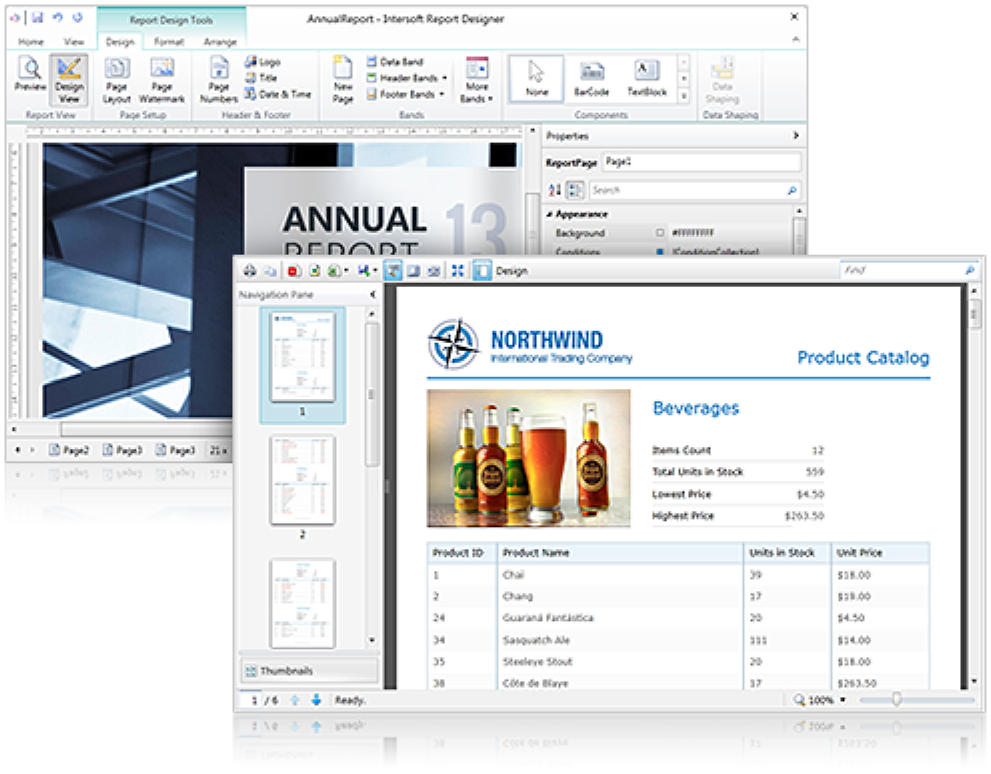
Reporting Controls Overview

ClientUI Reporting is a full-fledged XAML reporting engine which includes a powerful C# style scripting support for the most demanding business reports. Packed along with an intuitive viewer control, ClientUI Reporting offers sophisticated viewing experiences and blazing-fast rendering performance to display advanced business reports without performance bottleneck. The built-in report designer enables you to create a stunning report in significantly shorter time with less effort, thanks to the modern WYSIWYG designer surface and comprehensive design tools.

On this page:



Items Count	12
Total Units in Stock	559
Lowest Price	$4.50
Highest Price	$263.50

Product ID	Product Name	Units in Stock	Unit Price
1	Chai	39	$18.00
2	Chang	17	$19.00
24	Guaraná Fantástica	20	$4.50
34	Sasquatch Ale	111	$14.00
35	Steeleye Stout	20	$18.00
38	Côte de Blaye	17	$263.50

XAML-based Report Engine

Creating a business report using Intersoft reporting tools is definitely easy. It will be very familiar for Silverlight/WPF developers and won't spend too much effort to learn it as it's completely defined using a XAML-like report metadata. Since the reporting XAML conforms to the Visual Studio specification, you get all the productivity features you loved, such as Intellisense, property value auto completion and more.

The following example illustrates a XAML report document.

XAML

```
<ReportDocument Name="Report1" xmlns="http://intersoft.reporting.com/schemas"
ReportUnit="Centimeters">
    <ReportDocument.DataDefinitions>
        ...
    </ReportDocument.DataDefinitions>
    <ReportDocument.Pages>
        <ReportPage Name="Page1" PageHeight="29.7" PageWidth="21" Margin="1,1,1,1">
            <ReportPage.Components>
                <HeaderBand Name="InvoiceHeader" LayoutBox="0,7.4,19,0.8">
                    <HeaderBand.Components>
                        <TextBlock LayoutBox="0,0,2.2,0.8" Text="Product ID"
TextAlignment="Center"/>
                    </HeaderBand.Components>
                </HeaderBand>

                <DataBand Name="InvoiceData" LayoutBox="0,9,19,0.8"
ObjectDataSourceGuid="ab5ee58b31fd440aa34aaec8735abf29">
                    ...
                </DataBand>

                <FooterBand Name="InvoiceFooter" LayoutBox="0,10.6,19,0.8">
                    ...
                </FooterBand>
            <ReportPage.Components>
        </ReportPage>
    </ReportDocument.Pages>
    <ReportDocument.Styles>
        ...
    </ReportDocument.Styles>
</ReportDocument>
```

The XAML report document introduces numerous great benefits, such as it support property value inheritance, an interesting feature introduced in Silverlight/WPF XAML, which enables child elements in report tree to obtain the value of a particular property from parent elements, inheriting the value as it was set in the nearest parent element. To make the report even more neat and simple, it also support style which can be shared by all of the elements inside the report.

Full Client Rendering, Datasource Agnostic

Talking about a report solution, many developers are concerned about the datasource support. Does it support SQL Server? How about Oracle? And so on. Unlike many reporting solutions which are server-based reporting, it's important to understand that the reporting solution fully runs on client-side, from the data source population to the pixel-identical report generation and rendering. This means there are literally no constraints on the datasource, so you can retrieve data from domain service, web service, WCF service, or virtually any kind of objects that can be serialized over the wire.

The independence to the server allows the report engine to perform many tasks by its own in the client-side. There are numerous great benefits associated to this design, such as faster and more responsive user experience in the client-side. And more importantly, the authored report document is truly self-contained, which means that a single report document is all you need to deploy for user consumption. And since the report is a XAML file, you can quickly deploy a report by uploading the file to the website – it's that easy.

C# Style Scripting Engine

One of the key features of Intersoft report engine is the powerful scripting engine. As it is essential to any reporting solution, scripting need to be intuitive to developers and require minimal learning curves. The engine has to be dynamic enough to evaluate complex, multi-line syntax – not only a single line expression – to support the most demanding business report authoring. Actually you can have it all within the ClientUI scripting engine.

As the industry's first report scripting engine, ClientUI scripting engine supports complex C# style statements and syntaxes. Moreover, you have access to full .NET library and classes, you can get or set a property of an object, or even create new objects – all within the report scripting. And the best of all, the script engine performs nearly as fast as the compiled runtime execution – thanks to the full DLR integration and expression binding implemented in the scripting engine.

The following example demonstrates the basic usage of the scripting.

XAML

```
<TextBlock Text="@[Invoice.ShipPostalCode + ' ' + Invoice.ShipCity + ' ' +
Invoice.ShipCountry]"/>
<TextBlock Text="@[string.Format('{0:C2}', (Invoice.UnitPrice * Invoice.Quantity *
(1 - Invoice.Discount)))]"/>
```

In addition, the report engine also supports scripting in the report events which can be authored directly in the report document. This allows you to accomplish dynamic report authoring without requiring code compilation. The following example shows what you can do in the Printing event of a TextBlock using the C# style scripting engine.

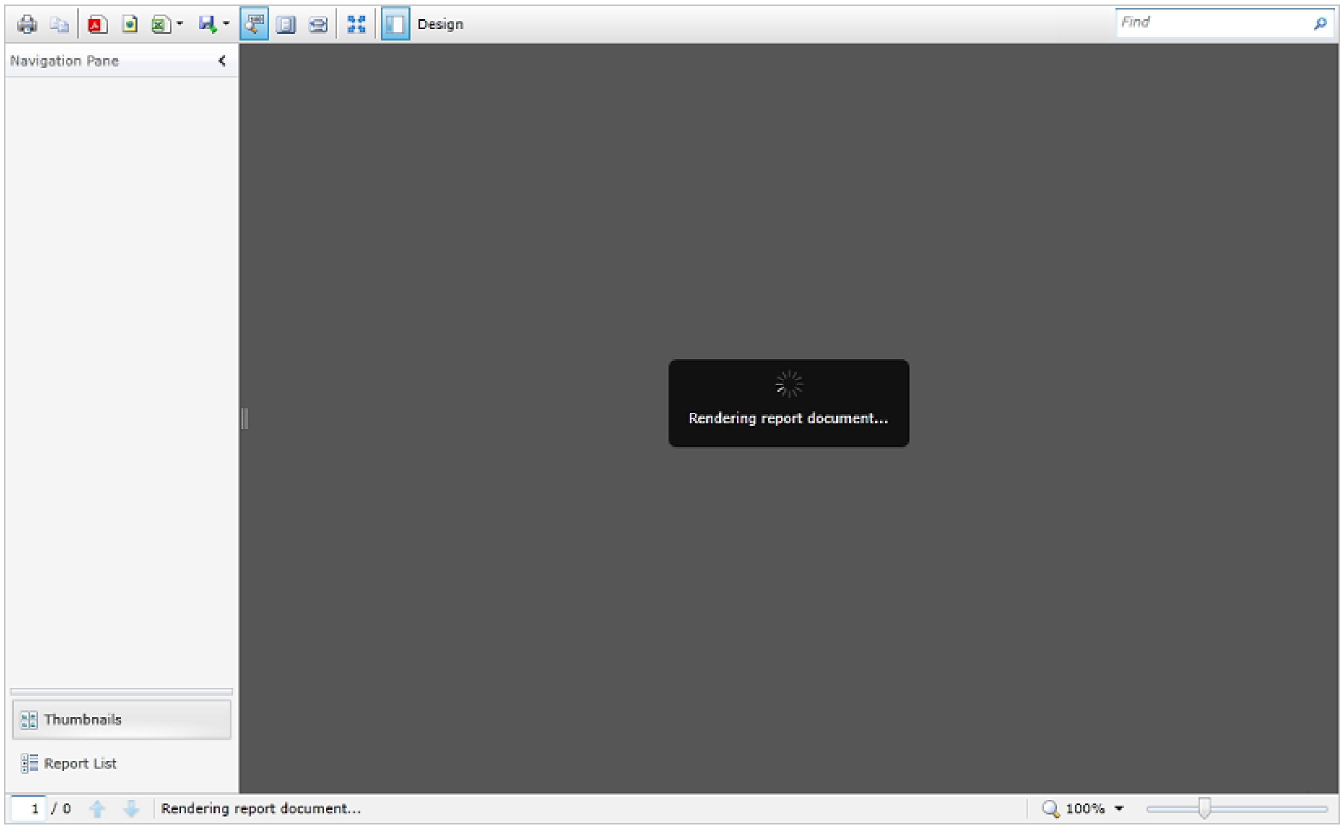
C#

```
// TextBlock_Printing(object sender, EventArgs e)

var totalAmount = Math.Ceiling(Order.Order_Details.Qty *
Order.Order_Details.UnitPrice);
var highestAmount = Math.Max(totalAmount, AllOrderAmount);

if (highestAmount > Customer.CreditLimit)
{
    e.Text = "Over limit";
    sender.Foreground = Colors.Red;
}
else
{
    e.Text = "Good";
    sender.Foreground = Colors.Black;
}
```

Enterprise-Grade Report Viewer

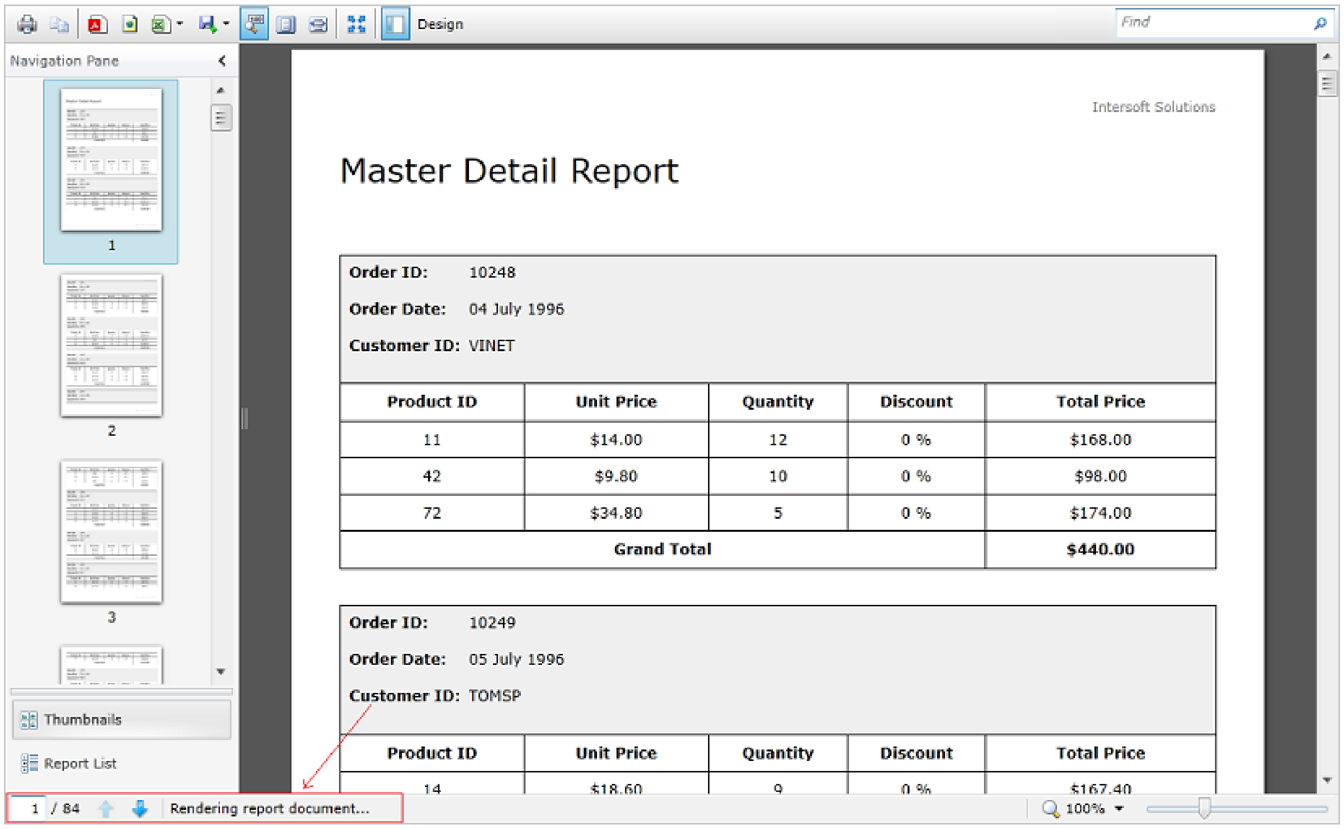
Of course, the report document is useless without the viewer that can render it. You can easily render the report document using ClientUI ReportViewer, an intuitive viewer control which is built on top of ClientUI architecture. What makes it truly set apart is its unique capability in handling large report document without performance bottleneck, thanks to the bare-metal virtualization implementation. It employs the most advanced rendering technique that prevent application from being freezed when rendering a lot of report pages.

In general, report rendering can be done in two ways, which can be set easily through the LoadingMode property. By default, all of pages will be rendered when report is loaded. Then busy indicator will be shown, prevent the report viewer from being accessed, until report is completely rendered. This is the mode that implemented in most reporting solutions in the market.



Freezing the report viewer while the report is being processed may sound just fine for reports with only a handful of pages. The main problem here is you may not have ideas how many pages that your report may eventually generate. It could be dozens of pages, or hundreds of pages, depending on the business data size. In this case, users will notice the significant slowness of the report generation, and worse, thinking that the application has crashed after waiting for minutes.

This is where the innovative background report generation comes to rescue. ClientUI reporting uses a very sophisticated techniques that allow the report generation to be done in the background, while updating the user interface without sacrificing performance. This makes the user interface truly responsive and thus translates to great user experiences. With this mode, users can view the report as soon as the first page is rendered, and scroll down as more pages are rendered in real-time. See the screenshot below.



Intersoft Solutions

Master Detail Report

Order ID: 10248
Order Date: 04 July 1996
Customer ID: VINET

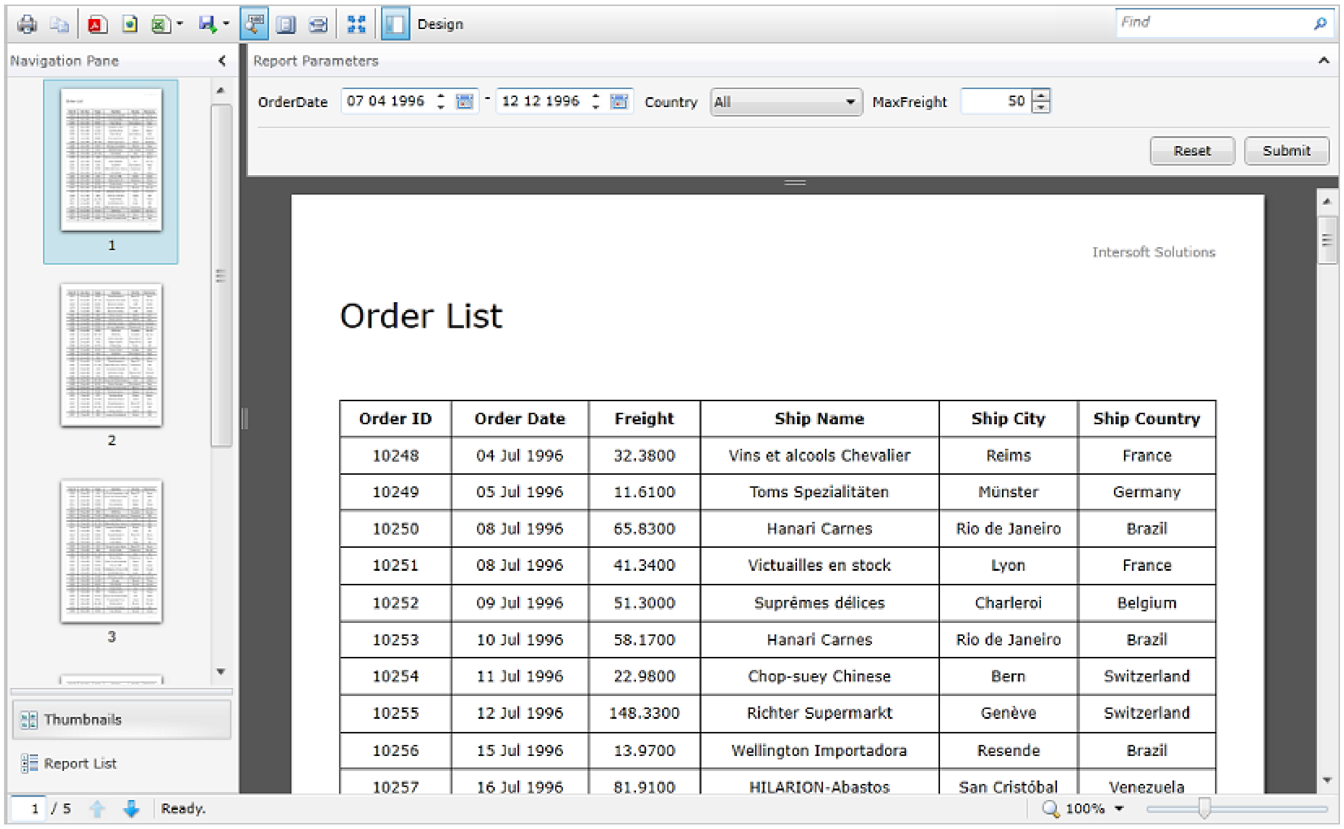
Product ID	Unit Price	Quantity	Discount	Total Price
11	$14.00	12	0 %	$168.00
42	$9.80	10	0 %	$98.00
72	$34.80	5	0 %	$174.00
Grand Total				**$440.00**

Order ID: 10249
Order Date: 05 July 1996
Customer ID: TOMSP

Product ID	Unit Price	Quantity	Discount	Total Price
14	$18.60	9	0 %	$167.40

1 / 84 Rendering report document...

Not only displaying report from the supplied data, it's also possible to display a report based on user input at run-time. ClientUI ReportViewer has provide a mechanism to generate and display the parameter dialog based on the report document. It's really not an easy task to create the parameter dialog that handle different parameter type for each report you have manually.



OrderDate 07 04 1996 - 12 12 1996 Country All MaxFreight 50

Intersoft Solutions

Order List

Order ID	Order Date	Freight	Ship Name	Ship City	Ship Country
10248	04 Jul 1996	32.3800	Vins et alcools Chevalier	Reims	France
10249	05 Jul 1996	11.6100	Toms Spezialitäten	Münster	Germany
10250	08 Jul 1996	65.8300	Hanari Carnes	Rio de Janeiro	Brazil
10251	08 Jul 1996	41.3400	Victuailles en stock	Lyon	France
10252	09 Jul 1996	51.3000	Suprêmes délices	Charleroi	Belgium
10253	10 Jul 1996	58.1700	Hanari Carnes	Rio de Janeiro	Brazil
10254	11 Jul 1996	22.9800	Chop-suey Chinese	Bern	Switzerland
10255	12 Jul 1996	148.3300	Richter Supermarkt	Genève	Switzerland
10256	15 Jul 1996	13.9700	Wellington Importadora	Resende	Brazil
10257	16 Jul 1996	81.9100	HILARION-Abastos	San Cristóbal	Venezuela

As you can see in the figure above, the report viewer also implement the sophisticated viewing features already invented in ClientUI's document viewer, such as printing, precise text selection, copy to clipboard, zooming, and searching. On the left panel, users can easily preview pages in a thumbnail form and navigate throughout the report pages. Actually the thumbnail will be rendered exactly same with the real page's appearance.

In addition, ReportViewer also provides a mechanism to display a report list which can be retrieved through many ways, such as directories lookup, from XML file, and etc. It also can be easily customized to overcome some specific scenario.

Modular and Extensible Architecture

ClientUI report engine is shipped with various built-in report controls, which generally used in most of line-of-business scenarios. Unlike other reporting solutions, Intersoft report engine is designed with highly modular pattern, instead of being bundled in a huge code base. You only need to include what you need, so you can maintain the solutions to be lightweight. Built with extensible capability, it's also possible for you to add a custom report control to overcome your business specific scenarios.

Learn More

This topic provides an overview of the ClientUI Reporting that covers it's uniqueness and key features. To get a clear explanation on ClientUI

Reporting, you can explore it through the following links.

- ClientUI Reporting Fundamentals
- Creating Reports
- Viewing Reports
- End-User Report Designer

Related Topics

- Windowing Controls Overview
- UI Controls Overview
- Toolbar and Menu Controls Overview
- Scheduling Controls Overview
- Ribbon Controls Overview

19 related results