ComponentOne

FlexReport for UWP

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

FlexReport for UWP Overview

ComponentOne Studio introduces **FlexReport for UWP** that adds reporting capabilities to your UWP applications. **FlexReport for UWP** is a comprehensive reporting tool which provides complete reporting solution - from building complex reports to previewing and exporting. A rich object model, previewing pane, high quality rendering, modern user interfaces in its previewing control, and ease of use make **FlexReport** a must have control for advanced as well as basic level report designers.

FlexReport for UWP uses same overall approach as **FlexReport for WinForms** – the report is generated locally within a UWP application (as opposed to being generated on a server). The application is self-contained, so you don't have to write a separate web service to provide the reports.

Help with UWP Edition

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with UWP Edition.

FlexReport Key Features

The key features of **FlexReport for UWP** are as follows:

- **Light-weight and Fast**
  FlexReport is light-weight and fast in particular for smaller reports.

- **Enhanced Rendering**
  FlexReport uses modern rendering with DirectWrite/DirectX technology to draw and generate high performance report content. This makes text, shapes and borders rendering better and increases the accuracy along with the quality.

- **Preview control with Modern UI**
  FlexViewer control has report previewing capabilities and can load and view FlexReport. It allows you to navigate through the report pages, change the page settings before printing reports, print reports, and export report to multiple formats.

- **Supported Data Providers**
  FlexReport for UWP currently supports the following data sources:
  - SQLite
  - An object in an external DLL supporting IC1FlexReportExternalRecordset
  - An object that supports either IC1FlexReportRecordset or IList (needs to be assigned to DataSource.RecordSet in code)

- **Exporting Capabilities**
  FlexReport for UWP can be exported to PDF, HTML, DOCX, RTF, XLSX, TIFF, BMP, PNG, JPEG and GIF formats.

Changes in Public Properties and Methods

Certain properties and methods are changed in the UWP version of FlexReport to make its use easier:

- The type of C1FlexReport.BasePath and C1FlexReport.DefaultBasePath properties in the UWP version is StorageFolder.
- Method overloads C1FlexReport.Save(...), C1FlexReport.Load(...) were added that accept a StorageFile as the argument.

Limitations of FlexReport

- **Loading a Report**
  - **FlexReport for UWP** cannot directly load legacy C1Report report definition files (.xml). To use such report definitions, you need to first convert them to the FlexReport format (e.g. using the FlexReport designer app for WinForms), and then use the converted .flxr report definitions instead.
- **Features not supported till now**
  - Metafiles (if a report uses a metafile as an image, it will not be displayed).
  - Chart Field, RTFField and Legacy Field having Field.RTF=True.

- **Data Providers not supported**
  - As there is no ADO.NET in UWP, the following data providers are not supported in UWP:
    - OLEDB
    - ODBC
    - SQLServerCe3_5
    - SQLServerCe4_0
    - XMLFile data provider.

- **Printing a Report**
  - The C1FlexReport.Print(...) methods are not supported. Instead, C1DocumentSource.ShowPrintUIAsync() method is provided.

FlexReport Dissection

It is important to know about the components and controls shipped along with FlexReport, the object model of FlexReport and sections or bands available in FlexReport, before you start exploring and working with FlexReport control. The following sections walk you through these details.

Components and Controls

FlexReport consists of the following assemblies:

C1.UWP.FlexReport dll

It provides the report generating and rendering functionality through the following component:

- **C1FlexReport:**
  The C1FlexReport is a report generating component that generates band-oriented reports. You can render reports directly to the preview control, or export them to various portable formats (including PDF, XLSX, HTML, and DOCX).

C1.UWP.FlexViewer dll

It includes the following report viewing/UI components:

- **C1FlexViewer:**
  The C1FlexViewer control is a full-featured report and document viewing control with a rich UI providing access to the various capabilities such as zooming, exporting, and printing the report.
- **C1FlexViewerPane:**
  The C1FlexViewerPane is a bare-bones preview control that allows programmatic manipulation of the view such as zooming, rotating and so on.

Included Applications

In addition to the reporting components and controls, FlexReport also includes the following stand-alone application:

- **C1FlexReportDesigner**
  C1FlexReportDesigner is a desktop application, used to create and edit **C1FlexReport** report definition files (.FLXR). It can also convert legacy **C1Report** report definitions (.XML) to the new .FLXR format.
  C1FlexReportDesigner can design reports compatible with both UWP and WinForms versions of the product. The designer app comes shipped in two flavors - **C1FlexReportDesigner.4.exe,** built for 'Any CPU' target and will run in 64-bit mode on a x64 system, and **C1FlexReportDesigner32.4.exe,** built for x86 target, and will always run in 32 bit mode. You can find the application located at the following path:
  C:\Program Files (x86)\ComponentOne\Apps\v4.0

Note that **C1FlexReportDesigner** application is installed with ComponentOne Studio WinForms Edition and this directory reflects the default installation path installed with the WinForms Edition.

Object Model Summary

FlexReport has a rich object model. The objects, collections, and the associated properties and methods together provide an ease and flexibility in generating FlexReport. The following table lists objects and their main properties and methods:

C1FlexReport
Properties: Credential, DataSource, Document, FileName, Layout, MaxPages, OnClose, OnError, OnOpen, Page,

Parameters, ReportDefinition, ReportInfo, ReportName, Sections **Methods:** Evaluate, Execute, GetReportList, Load, Render, Save, Clear
Layout
Properties: Width, MarginLeft, MarginTop, MarginRight, MarginBottom, PaperSize, Orientation, Columns, ColumnLayout, PageHeader, PageFooter
DataSource
Properties: CalculatedFields, ConnectionString, Filter, RecordSource, SortDefinitions
DataSourceCollection
Properties: Report **Methods:** Add, RemoveAt
SortDefinition
Properties: Direction, Expression, Owner
SortDefinitionCollection
Properties: Owner, Report
CalculatedField
Properties: DataSource, Expression, Type
CalculatedFieldCollection
Properties: Owner, Report
Group
Properties: GroupBy, KeepTogether, SectionHeader, SectionFooter, Sort, SortExpression
GroupCollection
Properties: Report **Methods:** Add, Clear, RemoveAt
ReportParameter
Properties: AllowedValuesDefinition, DisplayText, ParentReport **Methods:** SetName
ReportParameterCollection
Properties: Item, Report **Methods:** InsertItem, RemoveItem, SetItem
AllowedValuesDefinition
Properties: Binding, Values **Methods:** AssignFrom
Section
Properties: Calculated, Fields, Height, SplitBehavior, SubSections
SectionCollection
Properties: Detail, Footer, Header, PageFooter, PageHeader
SubSection

Properties: Calculated, Fields, Height, ParentReport, ParentSection, SplitBehavior, Visible
SubSectionCollection
Properties: Report **Methods:** Add, Remove, RemoveAt
FieldBase
Properties: Anchor, Height, ForcePageBreak, MarginBottom, MarginLeft, MarginRight, MarginTop, Section, SplitHorzBehavior, SplitVertBehavior
FieldCollection
Methods: Add, Remove, RemoveAt
BarCodeField
Properties: BarCode, BarCodeOptions, Font, Text
CheckBoxField
Properties: CheckAlign, CheckMark, Text, ThreeState, Value
DataField
Properties: Calculated, Name, Type, Value
ImageField
Properties: Picture, PictureAlign, PictureScale
ShapeField
Properties: Line, Shape, ShapeBackground, ShapeType
SubreportField
Properties: ParameterValues, Subreport, SubreportFilter
TextField
Properties: Format, Text
VisualReportObject
Properties: Background, Border, BordersSplitHorzMode, BordersSplitVertMode, OutlineLabel
BehaviorOptions
Properties: AllowHorizontalSplitting, EnableAggregatesOnReportFields, IgnoreInvisibleFieldsInGrowShrinkSections **Methods:** AssignFrom, Reset

Sections of FlexReport

Each FlexReport includes at least the following five mandatory sections:

Section	Description
Detail	The Detail section contains fields that are rendered once for each record in the source recordset.
Header	The Report Header section is rendered at the beginning of the report.

Section	Description
Footer	The Report Footer section is rendered at the end of the report.
Page Header	The Page Header section is rendered at the top of every page (except optionally for pages that contain the Report Header).
Page Footer	The Page Footer section is rendered at the bottom of every page.

Group Header and a **Group Footer** are two additional sections for each group. For example, a report with 3 grouping levels will have 11 sections.

Each section consists of sub-sections, where the actual report content is shown. A section always contains at least one sub-section. Additional sub-sections can be added to enhance functionality, e.g. the visibility of a sub-section can be toggled by a script depending on some condition, and so on.

> Note that sections can be made invisible, but they cannot be added or removed, except by adding or removing groups.

You can find the details of all the sections in FlexReport Section.

FlexReport Quick Start

Although you can use C1FlexReport in various scenarios, on the desktop and the Web, the basic sequence in most cases probably remains the same as discussed below:

1. **Create a report definition**

   Report Definition can be created using the **FlexReportDesigner** application. Reports can be designed from scratch, converted from legacy C1Report report definitions, or imported from existing Microsoft Access Reports and Crystal Reports. You can also do it through code using the rich object model provided by C1FlexReport. The result of this step is a .FLXR file containing the report definition.

2. **Load the report definition into the C1FlexReport component**

   Make the .FLXR report definition file created in step 1 available to your app's code, as a resource file or as an embedded resource. At runtime, use any of the Load or LoadAsync method overloads of C1FlexReport, to load the report definition into the C1FlexReport component.

3. **Render the report**

   To preview the report, add a **C1FlexViewer** control to your app, and at runtime assign the **C1FlexReport** component to the viewer's DocumentSource property. This will generate the report and show it in **C1FlexViewer** control.

The detailed steps in the following topics will show you how to create a report definition, load and render the report in the **FlexViewer** control.

Step 1 of 2: Creating a Report Definition

Create a report definition using the **FlexReportDesigner** application or code. You can simply load an existing definition and render it in the **FlexViewer** control. The easiest way to create a report definition is to use the stand-alone **C1FlexReportDesigner** desktop application that ships with **FlexReport**.

The **C1FlexReportDesigner.4.exe** for 64 bit platform and **C1FlexReportDesigner32.4.exe** for 32 bit platform are located in **C:\Program Files (x86)\ComponentOne\Apps\v4.0** folder on your computer.

You can create a Report Definition using **FlexReportDesigner** app available with **ComponentOne Studio WinForms Edition**. The steps to create a Report Definition are as follows:

1. Run the **FlexReportDesigner** app and select **New** from the **File** Menu.
2. Click **New Report** drop down in the **Reports** tab located on the extreme left of designer and select **Report Wizard**.
3. Select **SQLite Data Provider** from the **Data provider** dropdown and click the ellipsis button next to the **Connection string** textbox to select the **C1NWind.db** file.
4. Select a table from **Data source** tab and Click **Next**. In our case, we have selected **Products** table.
5. Select the fields, layout, and style for the report after connecting the reports to a data source, give a suitable Title to the Report, and click **Finish**.

Note that the path to **C1NWind.db** is specified using the **?(SpecialFolder.SystemDefault)** syntax. FlexReport connection strings support a special syntax that points to the Application.Current.LocalFolder in case of FlexReport for UWP. With this syntax, the connection string in the report can be specified as: **Data Source=?(SpecialFolder.SystemDefault)\C1NWind.db**

Step 2 of 2: Loading and Rendering the Report

1. Create a new UWP project and select **Blank App** (Universal Windows) in Visual Studio.
2. Add the **C1FlexViewer** control to the main page.
3. Add a reference to **C1.UWP.FlexReport**.
4. Download the FlexReport.SQLite project and add it to your solution.
   Also, if UWP Edition is installed on your system, you can find FlexReport.SQLite project in **Documents\ComponentOne Samples\UWP\C1.UWP.FlexReport\CS** folder.
5. Rebuild your project after adding the project.
6. Right-click your existing project in **Solution Explorer** and Select **Add|Reference...**. The **Reference Manager** opens.
7. Select **Projects|Solution** from the left pane and then select **FlexReport.SQLite** and click **OK**. This will add the **FlexReport.SQLite.dll** to the References folder in your project.

   > **Note:** Adding **FlexReport.SQLite project** as a reference to your app's project is required for FlexReport to be able to use SQLite. **FlexReport.SQLite** works as a wrapper that allows the FlexReport assembly to avoid having a hard reference to SQLite.

8. Add the report definition file created using the designer in Step 1 of 2: Creating a Report Definition to the **Assets** folder of your project and set its **Build Action** property to **Content**.
9. Add **C1NWind.mdb** database to the **Assets** folder, set its **Build Action** property to **Content**.
10. Add the following **Page_Loaded** event in the code view to load and render the report in FlexViewer control:

Visual Basic

```
Private Sub Page_Loaded(sender As Object, e As RoutedEventArgs)
	' Copy the SQLite database file from app's Assets to LocalFolder - in a
.FLXR report def,
	' it can be referenced as ?(SpecialFolder.SystemDefault):
	' Data Source=?(SpecialFolder.SystemDefault)\C1NWind.db
	' When designing the report, ?(SpecialFolder.SystemDefault) refers to
Environment.SpecialFolder.MyDocuments, so you can
	' put the report database file in your MyDocuments folder to
conveniently design and test run your reports:
		Dim dbPath = Path.Combine(ApplicationData.Current.LocalFolder.Path,
"C1NWind.db")
		If Not File.Exists(dbPath) Then
			File.Copy("Assets\C1NWind.db", dbPath)
		End If

		' Create and load the report:
		Dim report As New C1FlexReport()
		Using fs As Stream = File.OpenRead("Assets/ProductsUWP.flxr")
			report.Load(fs, "ProductList")
		End Using

		' Assign the report to the viewer's DocumentSource - it will be
generated automatically
		' (asynchronously by default) when the viewer shows it:
		Me.flexViewer.DocumentSource = report
End Sub
```

- **C#**

```
private async void Page_Loaded(object sender, RoutedEventArgs e)
{
// Copy the SQLite database file from app's Assets to LocalFolder-in .FLXR report definition
    var dbPath = Path.Combine(ApplicationData.Current.LocalFolder.Path, "C1NWind.db");
    if (!File.Exists(dbPath))
        File.Copy(@"Assets\C1NWind.db", dbPath);

// Create and load the report:
    C1FlexReport report = new C1FlexReport();
    using (Stream fs = File.OpenRead("Assets/ProductsUWP.flxr"))
        report.Load(fs, "ProductList");

// Assign the report to the viewer's DocumentSource - it will be generated automatically
```

```
// (asynchronously by default) when the viewer shows it:
    this.flexViewer.DocumentSource = report;
}
```

Working with FlexReport

While FlexReport for UWP can be used in various scenarios, the key steps relevant in most of those scenarios are:

1. **Creating a report definition** - This can be done using the **C1FlexReportDesigner** desktop application. The .FLXR report definition file created by the designer needs to be made available to the application's runtime, so that it can be loaded into the **C1FlexReport** component. Alternatively, the report definition can be created completely at runtime in code, using the rich object model of the **C1FlexReport** component.
2. **Providing data for the report** - The report definition must be created keeping in mind the data sources available to the UWP version. At runtime, once the report definition has been loaded into the **C1FlexReport** component, the data must be accessible so the report can generate. Note that the report is generated completely on the client. See below for details on the available data sources.
3. **Render and output the report** - Typically, the report will be generated for one of the following targets:
    - **Preview:** Simply assigning the **C1FlexReport** to the C1FlexViewer.DocumentSource property will generate the report and show it in the FlexViewer. Also, the UI of the FlexViewer control allows to print or export the report interactively.
    - **Print:** The report can be rendered and then printed directly using the C1DocumentSource.ShowPrintUIAsync method.
    - **Export:** The report can be rendered directly into one of the supported formats such as PDF, HTML etc. using C1FlexReport.RenderToFilter, C1FlexReport.RenderToFilterAsync or C1FlexReport.RenderToFilterAsyncEx method.

Data Binding in FlexReport

In addition to a report definition, **FlexReport** needs the actual data to create the report. In most cases, the data comes from a database, but there are other options. The following topics explore how to retrieve data from SQLite and other sources.

Data Binding using SQLite

FlexReport supports data binding using SQLite. SQLite is a software library and widely used SQL database engine. It does not have a separate server process so it can read and write directly to different files in ordinary disks.

For binding data using SQLite, you are required to use FlexReport.SQLite project, which is available in **Documents\ComponentOne Samples\UWP\C1.UWP.FlexReport\CS** folder. You need to add FlexReport.SQLite project to the solution of your app's project, as it is required for FlexReport to be able to use SQLite.

The following code illustrates the use of SQLite for data binding:

1. Add the following code to create a database connection:

Visual Basic

```
' Copy the SQLite database file from app's Assets to LocalFolder -
in a .FLXR report definition
Dim dbPath = Path.Combine(ApplicationData.Current.LocalFolder.Path,
"C1NWind.db")
If Not File.Exists(dbPath) Then
        File.Copy("Assets\C1NWind.db", dbPath)
End If
```

C#

```
// Copy the SQLite database file from app's Assets to LocalFolder -
```

```
in a .FLXR report definition
var dbPath = Path.Combine(ApplicationData.Current.LocalFolder.Path,
"C1NWind.db");
if (!File.Exists(dbPath))
  File.Copy(@ "Assets\C1NWind.db", dbPath);
```

2. Add the following code to create and load a report:

Visual Basic

```
' Create and load the report:
Dim report As New C1FlexReport()
Using fs As Stream = File.OpenRead("Assets/ProductsUWP.flxr")
        report.Load(fs, "ProductList")
End Using
```

C#

```
// Create and load the report:
C1FlexReport report = new C1FlexReport();
using(Stream fs = File.OpenRead("Assets/ProductsUWP.flxr"))
report.Load(fs, "ProductList");
```

3. Render the report in FlexViewer control using the following code:

Visual Basic

```
Me.flexViewer.DocumentSource = report
```

C#

```
this.flexViewer.DocumentSource = report;
```

Data Binding using External Objects

You can easily use external objects for data binding in FlexReport. Here, we discuss data binding using Open Data Protocol (OData) client library. The OData allows you to access data in the same style as in Representational State Transfer (REST) resources. The use of Simple.OData.Client library for data binding is illustrated below.

Create a Report Definition

Create a report definition using code to bind the data using OData client library.

1. Add the following namespace in code view:
   using Simple.OData.Client;
2. Create an object of FlexReport using the following code:

Visual Basic

```
Dim _report As New C1FlexReport()
```

C#

```
C1FlexReport _report = new C1FlexReport();
```

3. Add following code to request data from OData service:

Visual Basic

```
' request data from OData service
Dim client = New ODataClient(ODataUri)
' select all categries and products of each category
Dim categories = (Await client.[For](Of Category)
().Expand(Function(x) New From { _
        x.Products _
}).FindEntriesAsync()).ToList()
Dim products = (From c In categoriesFrom p In c.ProductsNew With { _
        Key .CategoryID = c.ID, _
        Key .CategoryName = c.Name, _
        Key .ID = p.ID, _
        Key .Name = p.Name, _
        Key .Description = p.Description, _
        Key .ReleaseDate = p.ReleaseDate, _
        Key .DiscontinuedDate = p.DiscontinuedDate, _
        Key .Rating = p.Rating, _
        Key .Price = p.Price _
}).ToList()
```

C#

```
// request data from OData service
var client = new ODataClient(ODataUri);
// select all categries and products of each category
var categories = (await client.For<Category>().Expand(x => new {
x.Products }).FindEntriesAsync()).ToList();
var products = (
from c in categories
from p in c.Products
select new
{
    CategoryID = c.ID,
    CategoryName = c.Name,
    ID = p.ID,
    Name = p.Name,
    Description = p.Description,
    ReleaseDate = p.ReleaseDate,
    DiscontinuedDate = p.DiscontinuedDate,
    Rating = p.Rating,
    Price = p.Price,
}).ToList();
```

4. Add a new folder named Resources to your application and add a report to it. In our case, we are using Reports.flxr report.

5. Load the report definition from Resources folder using the following code:

Visual Basic

```
' load report definition from resources
Dim asm As Assembly = GetType(MainPage).GetTypeInfo().Assembly
Using stream As Stream =
asm.GetManifestResourceStream("Binding.Resources.Reports.flxr")
        _report.Load(stream, "Products")
End Using

' assign dataset to the report
_report.DataSource.Recordset = products
```

C#

```
// load report definition from resources
Assembly asm = typeof(MainPage).GetTypeInfo().Assembly;
using (Stream stream =
asm.GetManifestResourceStream("Binding.Resources.Reports.flxr"))
_report.Load(stream, "Products");

// assign dataset to the report
_report.DataSource.Recordset = products;
```

6. Use the following code to build your report and view it in FlexViewer control after loading the report definition:

Visual Basic

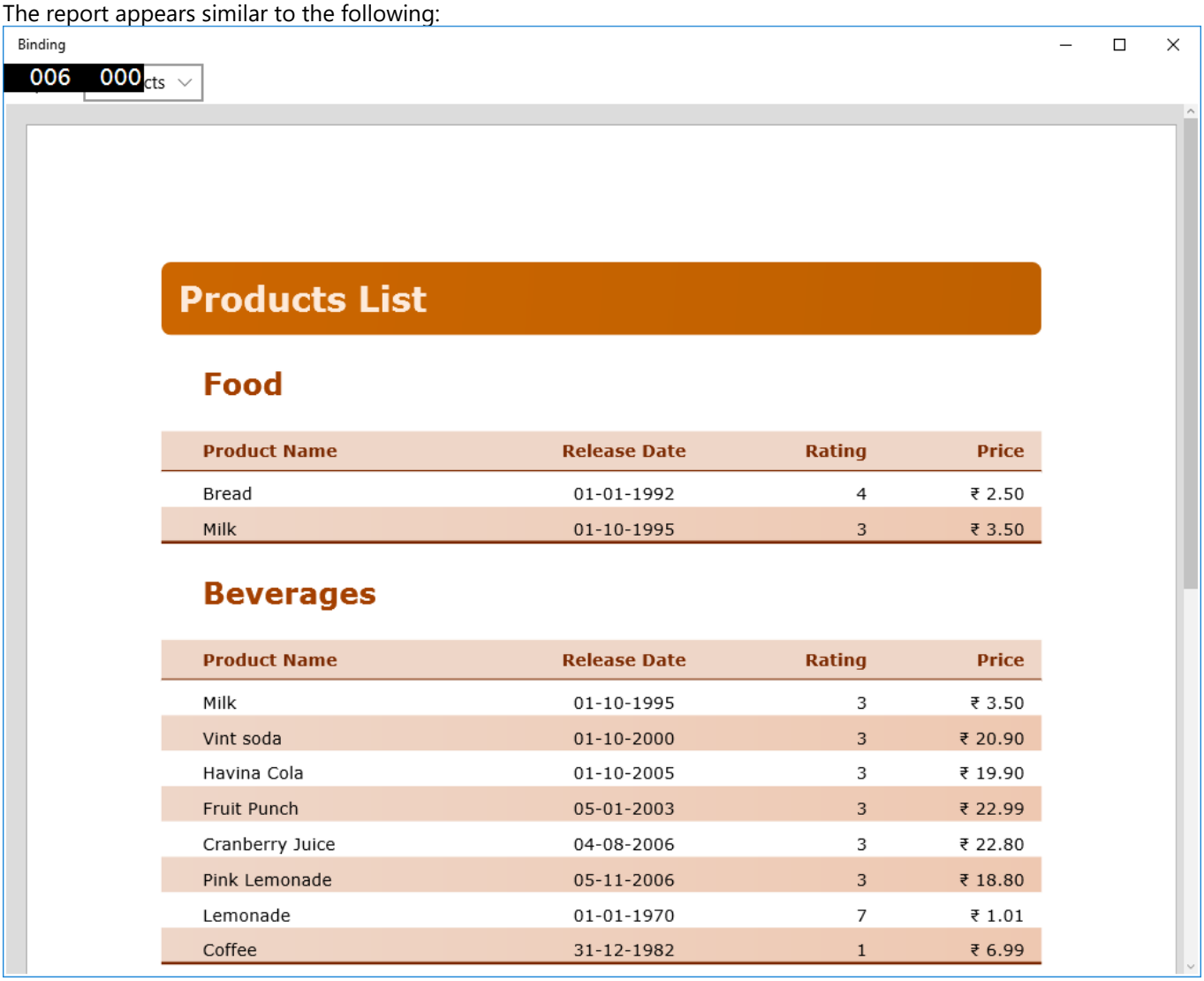
```
Try
        ' build report
        prMain.IsActive = True
        Await BuildProductsReport()
        prMain.IsActive = False

        ' assign report to the preview pane
        flxViewer.DocumentSource = Nothing
        flxViewer.DocumentSource = _report
Catch ex As Exception
        Dim md As New MessageDialog(String.Format("Failed to show ""
{0}"" report, error:" & vbCr & vbLf & "{1}", reportName,
ex.Message))
        Await md.ShowAsync()
End Try
```

C#

```
try
{
    // build report
    prMain.IsActive = true;
    await BuildProductsReport();
    prMain.IsActive = false;

    // assign report to the preview pane
```

```
    flxViewer.DocumentSource = null;
    flxViewer.DocumentSource = _report;
}
catch (Exception ex)
{
    MessageDialog md = new MessageDialog(string.Format("Failed to
show \"{0}\" report, error:\r\n{1}", reportName, ex.Message));
    await md.ShowAsync();
}
```

The report appears similar to the following:



Exporting

FlexReport allows you to export reports to different files and distribute them electronically. It supports the following export formats and respective export filters for exporting the report:

Format	Description
RTF (*.rtf)	RtfFilter export filter is used to export the reports into RTF streams or files.
Microsoft Excel	XlsFilter export filter is used to export the reports into XLSX/XLS streams or files.

Format	Description
(*.xlsx/*.xls)	
TIFF (*.tiff), BMP, PNG, JPEG, GIF images	TiffFilter, BmpFilter, PngFilter, JpegFilter, and GifFilter export filters are used to export the reports into different image format filles or streams.
PDF (*.pdf)	PdfFilter export filter is used to export the reports into PDF streams or files.
HTML(*.html)	HtmlFilter export filter is used to export the reports into HTML streams or files.

Exporting a report to XLSX format

You can export the report created in Quick Start section to **XLSX** format using the following steps.

1. Add the following namespace in the code view.

Visual Basic

```
Imports C1.Xaml.Document.Export
```

C#

```
using C1.Xaml.Document.Export;
```

2. Add the following code to export the report to **XLSX** format using **XlsFilter** class.
Note that the same class can be used to export the report to **XLS** format.

Visual Basic

```
' request target file from the user
Dim fileSavePicker As New FileSavePicker()
fileSavePicker.FileTypeChoices.Add("XLSX files", New String()
{".xlsx"})
Dim storageFile As StorageFile = Await
fileSavePicker.PickSaveFileAsync()
If storageFile Is Nothing Then
        ' export cancelled by the user
        Return
End If
' initialize XlsFilter
Dim filter As New XlsFilter()
filter.UseZipForMultipleFiles = True
filter.StorageFile = storageFile

' render report to the filter
Await report.RenderToFilterAsync(filter)

' launch the exported file
Await Windows.System.Launcher.LaunchFileAsync(storageFile)
```

- **C#**

```
// request target file from the user
```

```
FileSavePicker fileSavePicker = new FileSavePicker();
fileSavePicker.FileTypeChoices.Add("XLSX files", new string[] { ".xlsx" });
StorageFile storageFile = await fileSavePicker.PickSaveFileAsync();
if (storageFile == null)
    // export cancelled by the user
    return;

// initialize XlsFilter
XlsFilter filter = new XlsFilter();
filter.UseZipForMultipleFiles = true;
filter.StorageFile = storageFile;

// render report to the filter
await report.RenderToFilterAsync(filter);

// launch the exported file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
```

Similarly, you can export your reports to RTF, HTML, and PDF formats.

Exporting a report to an image file format

The above code can be used for exporting a report to an image file but exporting a multi-paged report to an image file only exports the first page of the report at a time as the image format filters do not directly support a multiple paged report in a single file. However, it is possible to generate multiple image files corresponding to each page of a report in a ZIP file. The following code uses one of the image format filter class, JpegFilter, to export the multi-paged report to JPEG format and creates a single ZIP file of the exported images.

Visual Basic

```
' request target file from the user
Dim fileSavePicker As New FileSavePicker()
fileSavePicker.FileTypeChoices.Add("ZIP files", New String() {".zip"})
Dim storageFile As StorageFile = Await
fileSavePicker.PickSaveFileAsync()
If storageFile Is Nothing Then
        ' export cancelled by the user
        Return
End If

' initialize JpegFilter
Dim filter As New JpegFilter()
filter.UseZipForMultipleFiles = True
filter.StorageFile = storageFile

' render report to the filter
Await report.RenderToFilterAsync(filter)

' launch the exported file
Await Windows.System.Launcher.LaunchFileAsync(storageFile)
```

C#

```
  // request target file from the user
  FileSavePicker fileSavePicker = new FileSavePicker();
  fileSavePicker.FileTypeChoices.Add("ZIP files", new string[] {
```

```
    ".zip"
});
StorageFile storageFile = await fileSavePicker.PickSaveFileAsync();
if (storageFile == null)
// export cancelled by the user
    return;

// initialize JpegFilter
JpegFilter filter = new JpegFilter();
filter.UseZipForMultipleFiles = true;
filter.StorageFile = storageFile;

// render report to the filter
await report.RenderToFilterAsync(filter);

// launch the exported file
await Windows.System.Launcher.LaunchFileAsync(storageFile);
```

Printing

FlexReport allows you to print a report using ShowPrintUIAsync method of C1DocumentSource class. The following code implements the **ShowPrintUIAsync** method for printing a report. The example uses the sample created in Quick Start.

Visual Basic

```
' show print UI
report.ShowPrintUIAsync()
```

- **C#**

```
// show print UI
report.ShowPrintUIAsync();
```

Besides using **ShowPrintUIAsync** method, you can also print a report using FlexViewer UI. The FlexViewer control provides an option to print a report directly from its UI using the Print icon. On clicking the icon, you are provided with the standard printer settings, such as selecting a printer and selecting orientation, size, and number of pages to print.

About FlexReportDesigner

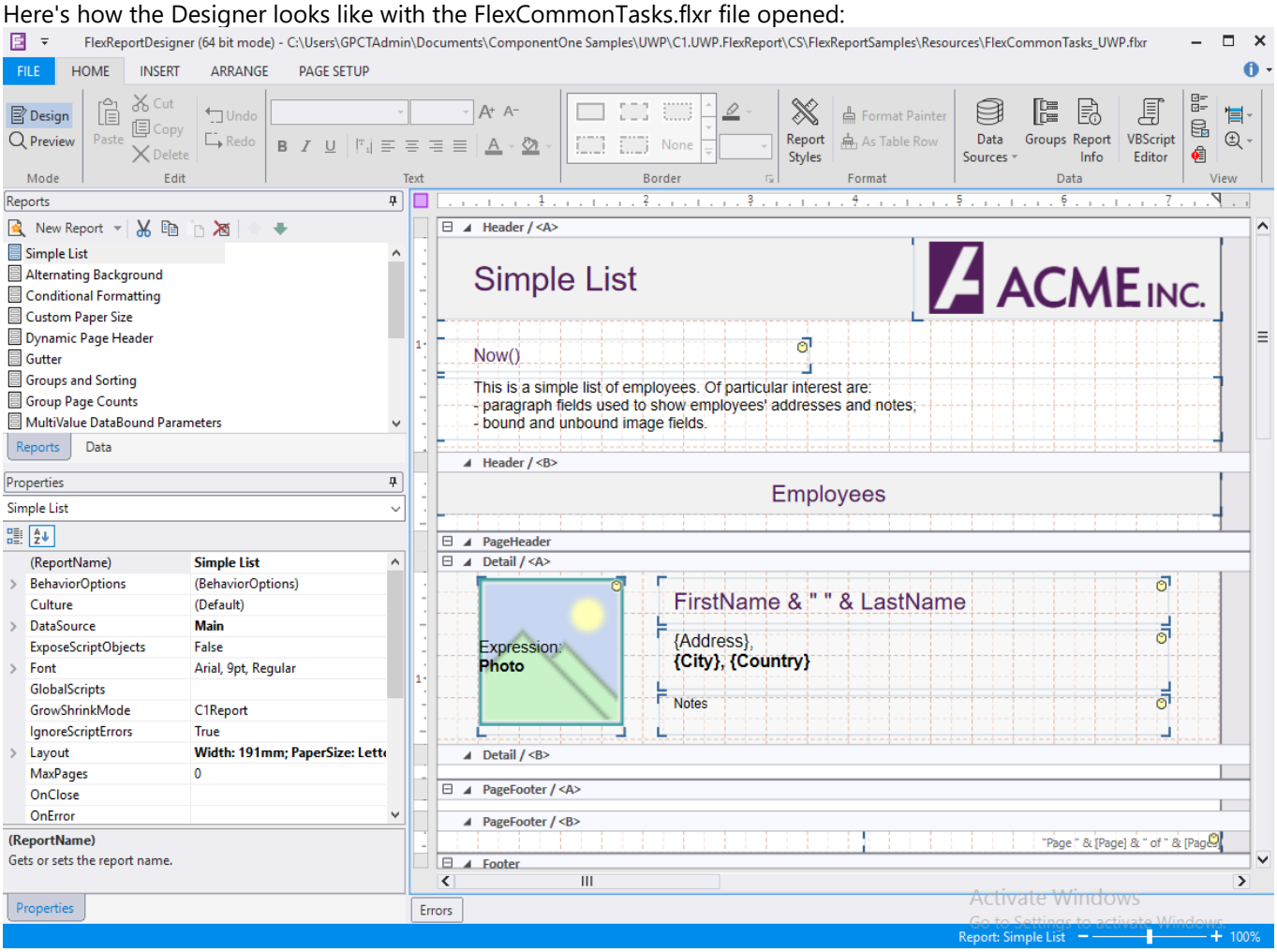
The **FlexReportDesigner** application is a tool used for creating and editing C1FlexReport report definition files. The Designer allows you to create, edit, load, and save files with .flxr extension. It also allows you to import C1Report (.xml) and report definitions from Microsoft Access files (.mdb) and Crystal Reports (.rpt).

To run the Designer, double-click the **C1FlexReportDesigner.4.exe** (for 64-bit platform) or **C1FlexReportDesigner32.4.exe** (for 32-bit platform) file located by default in the following path for .NET 4.0:

- **C:\Program Files (x86)\ComponentOne\Apps\v4.0** for 64 bit platform
- **C:\Program Files\ComponentOne\Apps\v4.0** for 32 bit platform

Note that the given location reflects the default installation path. It might differ if you make changes to the installation path.

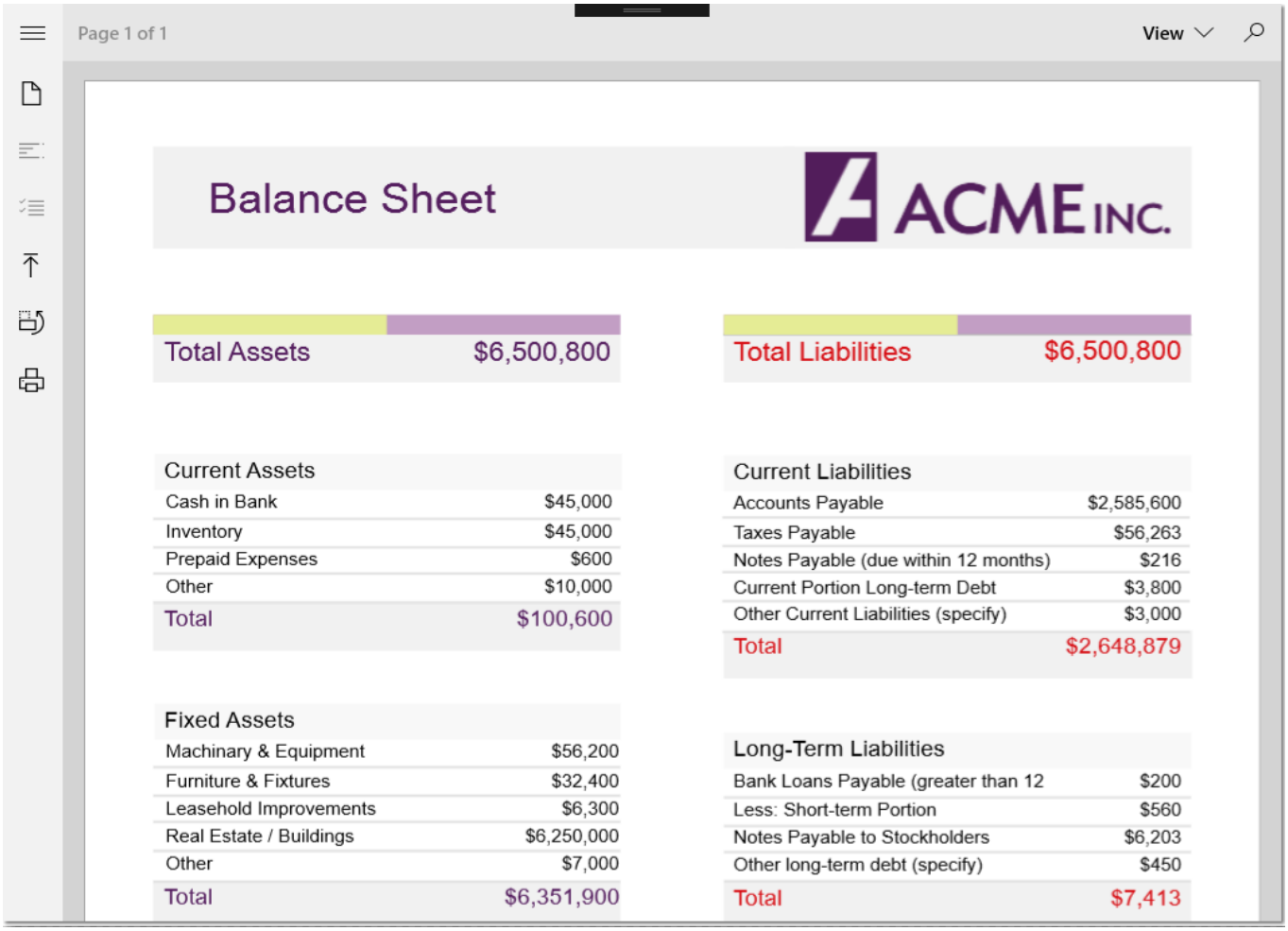
Here's how the Designer looks like with the FlexCommonTasks.flxr file opened:



You can find the detailed description about the components of the Designer window in FlexReportDesigner application for WinForms.

FlexViewer for UWP

FlexViewer is a previewing control which can be used to view FlexReport and PDF documents. The **FlexViewer** control comes with a modern, interactive and user-friendly User Interface(UI). Using **FlexViewer** control, you can navigate through the report pages using Page navigation option and jump to a specific page by entering the page number in Page textbox.



You can easily export a report to multiple formats from within the **FlexViewer** control. You can also change the page settings of the report by clicking on the Page Settings icon from the **FlexViewer** control. You can also use the **FlexViewer** control to print your reports using the Print icon which provides the standard printer settings.

FlexViewer Key Features

The key features of FlexViewer are as follows:

- **Modern User-friendly UI**
  FlexViewer has an interactive and user friendly UI that helps preview different document types such as FlexReport, SSRS, and PDF document. It follows standard practices used to design UWP Apps and its design makes it well suited for any device – Windows 10 PC, Windows Phone etc.

- **Page Navigation**
  Page navigation is available at the top of the FlexViewer control, with which you can navigate through the report pages and if you want to jump to a specific page number then you can type the page number in the page number textbox.

- **View Modes**
  FlexViewer supports different views of the reports/documents:
  - Continuous - Shows the pages in continuity.
  - Actual Size – Shows the pages in their actual size.
  - Page Width - Fits the page to the width of the preview window.
  - Whole Page - Fits the whole page in the preview window.
  - Rotate Clockwise - Rotates the view clockwise.
  - Rotate Counter-Clockwise - Rotates the view counter-clockwise.
  - One Page – Shows report pages page by page in a single page view.
  - Facing Pages – Shows report pages side by side.
  - Two Pages – Shows Two page view of pages.
  - Four pages – Shows pages in 4x4 mode.
  - Eight pages – Shows pages in 8x8 mode.

- **Use/Reset Parameters**
  The FlexViewer control adds interactivity, by letting you enter data parameters you require to display in your report. The control supports String, Boolean, Date, Integer and Float type parameters.

- **View Thumbnails and Hierarchy**
  FlexViewer allows you to view the thumbnails of the report pages and if your report contains Document Map, FlexViewer includes the Outlines panel, which helps you choose the required location to jump to.

- **Page Settings**
  The FlexViewer control allows you to change the page settings according to your requirements before printing your reports. You can just click on Page Settings icon in the left panel and set the following options:
  - Orientation - Potrait/Landscape
  - Size
  - Margins - Top margin, Bottom margin, Left margin, and Right margin

- **Print**
  FlexViewer allows you to print your reports using the Print icon, which provides the standard printer settings.

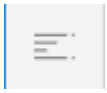
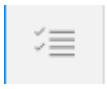
- **Export**
  FlexViewer allows you to export your reports and documents to various formats, such as HTML, PDF, RTF, GIF, JPEG, PNG, BMP, TIFF, Open XML Excel, and Open XML Word. You can also choose to open the exported document automatically after export.

- **RightToLeft**
  Apart from all the above features which are directly accessible from the UI, the FlexViewer allows you to set the direction of FlexViewer Tool panel by setting the FlowDirection property to RightToLeft or LeftToRight.

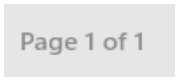
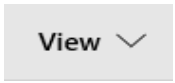
FlexViewer Toolbar

The toolbar appears on the left side of the **FlexViewer** control. It consists of the following command buttons:

Command Button	Command Button Name	Description
	Tools	Shows the list of tools available in the **FlexViewer** control

	Thumbnails	Displays thumbnails of all the pages available in the displayed report
	Outlines	Displays the outline of report pages
	Parameters	Displays the parameters set for the report pages
	Export	Allows user to export the report to different formats
	Page Settings	Allows you to set the orientation, size, and margins of the report pages
	Print	Allows user to print a report

Few other commands are also available in **FlexViewer** control which are visible at the top of the control.

Command Button	Command Button Name	Description
	Pages	Displays the current page number and the total number of pages in a report
	View	Allows you to view the report pages in different views
	Search	Allows you to search text in report

Rotate View of Reports

FlexViewer provides you the flexibility to rotate the view of reports to different angles according to your requirements. To rotate view of a report to various degrees of rotation, you can set the RotateView property of C1FlexViewer class. The RotateView property accepts the following values from the FlexViewerRotateView enum describing the rotation angle of the view:

- **NoRotation:** Rotation is not applied to the view.
- **Rotation180:** Allows rotation of the view by 180 degrees.
- **Rotation90Clockwise:** Allows rotation of the view by 90 degrees in clockwise direction.
- **Rotation90CounterClockwise:** Allows rotation of the view by 90 degrees in counter-clockwise direction.

Rotate View of Report at Runtime

You can rotate the view of a report at runtime by selecting one of the rotate views from the **View** dropdown list on the top right corner of the **FlexViewer** control.

Rotate View of Report Programmatically

To rotate view of a report, you can use FlexViewerRotateView enum to rotate view of a report. The following code illustrates the use of **FlexViewerRotateView** enum:

Visual Basic

```
flxViewer.RotateView =
C1.Xaml.FlexViewer.FlexViewerRotateView.Rotation90Clockwise
```

- **C#**

```
flxViewer.RotateView = C1.Xaml.FlexViewer.FlexViewerRotateView.Rotation90Clockwise;
```

Binding FlexReport with FlexViewer

To render a report, you need to load the report first. Once the report definition has been created, a data source is defined, and the report definition is loaded, you can render the report to the **FlexViewer** control.

To preview the report in **FlexViewer** control, use the following code:

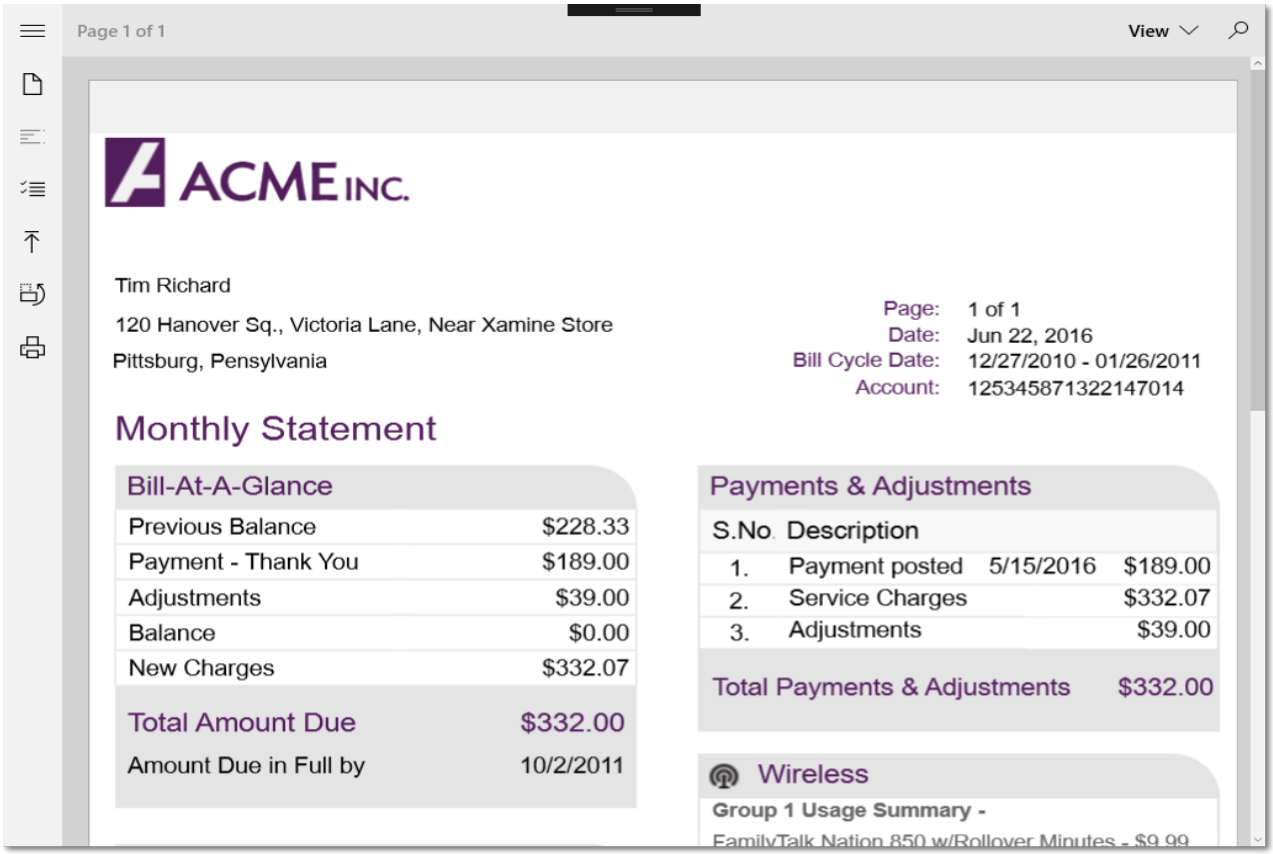
Visual Basic

```
Try
    ' load from resource stream
    Using stream As Stream =
asm.GetManifestResourceStream("BindingApp.Resources.TelephoneBillReport.flxr")

    rpt.Load(stream, "TelephoneBillReport")
End Using
Catch ex As Exception
```

```
        Dim md As New MessageDialog(String.Format("Failed to Load Report",
rpt.ReportName, ex.Message), "Error")
        Await md.ShowAsync()
        Return
End Try

Flxviewer.DocumentSource = rpt
```

C#

```
try
{
 // load from resource stream
 using (Stream stream =
asm.GetManifestResourceStream("BindingApp.Resources.TelephoneBillReport.flxr"))

 rpt.Load(stream, "TelephoneBillReport");
}
catch (Exception ex)
{
 MessageDialog md = new MessageDialog(string.Format("Failed to Load Report",
rpt.ReportName, ex.Message), "Error");
 await md.ShowAsync();
 return;
}

Flxviewer.DocumentSource = rpt;
```



FlexReport Samples

With the **C1Studio** installer, you get FlexReport samples that help you understand the implementation of the product. The C# and VB samples are available at the default installation folder - **Documents\ComponentOne Samples\UWP\C1.UWP.FlexReport.**

The list of C# samples available at the default installation location are as follows:

Sample	Description
Binding	This sample uses the Simple.OData.Client library which is installed automatically as a Nuget package.
FlexReportSamples	This sample allows to select a report from a .flxr FlexReport report definition file, or pick a report from the list of predefined reports, and generates the report. The report then can be exported to any of the supported external formats.
FlexReportExplorer	The application features industry standard reports for domains like Finance, Medical, Enterprise and Telecom. It demonstrates the major features of FlexReport for UWP.
FlexPdfViewer	This sample loads a PDF document into C1PdfDocumentSource, then uses FlexViewer control to show the document in UWP application.

The list of available VB samples are as follows:

Sample	Description
Binding	This sample uses the Simple.OData.Client library which is installed automatically as a Nuget package.
FlexReportSamples	This sample allows to select a report from a .flxr FlexReport report definition file, or pick a report from the list of predefined reports, and generates the report. The report then can be exported to any of the supported external formats.
FlexReportExplorer	The application features industry standard reports for domains like Finance, Medical, Enterprise and Telecom. It demonstrates the major features of FlexReport for UWP.
FlexPdfViewer	This sample loads a PDF document into C1PdfDocumentSource, then uses FlexViewer control to show the document in UWP application.