///////Export enabling code

1)public static class ReportViewerExtensions

 {

 public static void SetExportFormatVisibility(this ReportViewer viewer, ReportViewerExportFormat format, bool isVisible)

 {

 string formatName = format.ToString();

 const System.Reflection.BindingFlags Flags = System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Public | System.Reflection.BindingFlags.Instance;

 System.Reflection.FieldInfo m\_previewService = viewer.LocalReport.GetType().GetField("m\_previewService", Flags);

 System.Reflection.MethodInfo ListRenderingExtensions = m\_previewService.FieldType.GetMethod("ListRenderingExtensions", Flags);

 object previewServiceInstance = m\_previewService.GetValue(viewer.LocalReport);

 IList extensions = (IList)ListRenderingExtensions.Invoke(previewServiceInstance, null);

 System.Reflection.PropertyInfo name = extensions[0].GetType().GetProperty("Name", Flags);

 //object extension = null;

 foreach (var ext in extensions)

 {

 if ((string.Compare(name.GetValue(ext, null).ToString(), formatName, true) == 0))

 {

 System.Reflection.FieldInfo m\_isVisible = ext.GetType().GetField("m\_isVisible", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance);

 System.Reflection.FieldInfo m\_isExposedExternally = ext.GetType().GetField("m\_isExposedExternally", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance);

 m\_isVisible.SetValue(ext, isVisible);

 m\_isExposedExternally.SetValue(ext, isVisible);

 break;

 }

 }

 }

 }

 public enum ReportViewerExportFormat

 {

 Excel,

 PDF

 }

2) ReportViewer1.SetExportFormatVisibility(ReportViewerExportFormat.PDF, false);

The above first program code has SetExportFormatVisibility extension method which helps in the extension. The second program code has ReportViewerExportFormat for specifying the available export formats.

/////Print report enabling code

 Report.ReportPath = "C:\InspectionReport.rdlc"

 Report.DataSources.Add(New Microsoft.Reporting.WinForms.ReportDataSource \_

 ("InspectionReportDS\_Inspection\_Report", Me.InspectionReportDS.Inspection\_Report))

 PrintReport.Export(Report)

 PrintReport.m\_currentPageIndex = 0

 PrintReport.Print()

that calls the following class

Imports System.IO

Imports System.Data

Imports System.Text

Imports System.Drawing.Imaging

Imports System.Drawing.Printing

Imports System.Collections.Generic

Imports Microsoft.Reporting.WinForms

Public Class Reporting

 Implements IDisposable

 Public m\_currentPageIndex As Integer

 Public m\_streams As IList(Of Stream)

 Public Function CreateStream(ByVal name As String, \_

 ByVal fileNameExtension As String, \_

 ByVal encoding As Encoding, ByVal mimeType As String, \_

 ByVal willSeek As Boolean) As Stream

 Dim stream As Stream = \_

 New FileStream("C:\" + name + "." + fileNameExtension, FileMode.Create)

 m\_streams.Add(stream)

 Return stream

 End Function

 Public Sub Export(ByVal report As LocalReport)

 Dim deviceInfo As String = \_

 "<DeviceInfo>" + \_

 " <OutputFormat>EMF</OutputFormat>" + \_

 " <PageWidth>8.5in</PageWidth>" + \_

 " <PageHeight>11in</PageHeight>" + \_

 " <MarginTop>0.25in</MarginTop>" + \_

 " <MarginLeft>0.25in</MarginLeft>" + \_

 " <MarginRight>0.25in</MarginRight>" + \_

 " <MarginBottom>0.25in</MarginBottom>" + \_

 "</DeviceInfo>"

 Dim warnings() As Warning = Nothing

 m\_streams = New List(Of Stream)()

 '' report.Render("Image", deviceInfo, AddressOf CreateStream, warnings)

 Try

 report.Render("Image", deviceInfo, AddressOf CreateStream, warnings)

 Catch e As System.Exception

 Dim inner As Exception = e.InnerException

 While Not (inner Is Nothing)

 MsgBox(inner.Message)

 inner = inner.InnerException

 End While

 End Try

 ' Dim stream As Stream

 ' For Each stream In m\_streams

 'Stream.Position = 0

 ' Next

 End Sub

 Public Sub PrintPage(ByVal sender As Object, ByVal ev As PrintPageEventArgs)

 ' Dim pageImage As New Metafile(m\_streams(m\_currentPageIndex))

 Dim pageImage As New Metafile(m\_streams(m\_currentPageIndex))

 ev.Graphics.DrawImage(pageImage, ev.PageBounds)

 m\_currentPageIndex += 1

 ev.HasMorePages = (m\_currentPageIndex < m\_streams.Count)

 End Sub

 Public Sub Print()

 ' Const printerName As String = "Microsoft Office Document Image Writer"

 Const printerName As String = "Laser on Tiger"

 If m\_streams Is Nothing OrElse m\_streams.Count = 0 Then

 Return

 End If

 Dim printDoc As New PrintDocument()

 ' printDoc.PrinterSettings..PrinterName = printerName

 If Not printDoc.PrinterSettings.IsValid Then

 Dim msg As String = String.Format( \_

 "Can't find printer ""{0}"".", printerName)

 Console.WriteLine(msg)

 Return

 End If

 AddHandler printDoc.PrintPage, AddressOf PrintPage

 printDoc.Print()

 End Sub

 Public Overloads Sub Dispose() Implements IDisposable.Dispose

 If Not (m\_streams Is Nothing) Then

 Dim stream As Stream

 For Each stream In m\_streams

 stream.Close()

 Next

 m\_streams = Nothing

 End If

 End Sub

End Class