



PrintWideHTML Utility v2.0 Release Notes

Solution to Print Wide HTML Pages

AASoftech Inc.

<http://www.aasoftech.com>

sales@aasoftech.com

June 6, 2006

Table of Contents

| | |
|---|-------------|
| Chapter 1: General Overview | 1-1 |
| Overview:..... | 1-1 |
| Background:..... | 1-1 |
| Print Wide HTML page just like Print Excel:..... | 1-2 |
| Advantages:..... | 1-2 |
| “PrintWideHTML” Solution Path: | 1-3 |
| Chapter 2: Convert to PDF User Interface | 2-4 |
| Description..... | 2-4 |
| Input File: | 2-4 |
| Output File:..... | 2-4 |
| Page Type:..... | 2-5 |
| Page Orientation:..... | 2-5 |
| Custom Page Width:..... | 2-6 |
| Custom Page Height:..... | 2-6 |
| Tile: | 2-6 |
| Tile Page Type:..... | 2-6 |
| Tile Page Orientation:..... | 2-6 |
| Chapter 3: Print User Interface | 3-7 |
| Description..... | 3-7 |
| Input File: | 3-7 |
| Custom Page Width:..... | 3-7 |
| Custom Page Height:..... | 3-8 |
| Left Header Text: | 3-8 |
| Right Header Text: | 3-8 |
| Left Footer Text:..... | 3-8 |
| Rightt Footer Text: | 3-8 |
| Tile: | 3-8 |
| Chapter 4: API Interface (User’s Manual)..... | 4-10 |
| Interfacing with Print Function:..... | 4-10 |

| | |
|---|-------------|
| Interfacing with the PDF conversion system: | 4-11 |
| Return codes..... | 4-13 |
| Interfacing with Visual Basic | 4-14 |
| Chapter 5: API Examples | 5-16 |
| VB PDF Example | 5-16 |
| VB Print Example | 5-17 |
| VB.Net PDF Example | 5-25 |
| VB.Net Print Example | 5-39 |

Chapter 1: General Overview

Overview:

Your employees and customers require access to the information they need. When they need it on the web, it is no longer acceptable to change data layout just because you can't print it from your browser.

We reengineer browser print streams so you don't need to worry about the data layout any more.

Background:

Problem 1:

"I need to print html pages that contain tabular data. The data is organized in a specific number of columns (25), so when I get to preview the printing area, only the portion visible on the screen is displayed and printed. So the table gets cut in the middle or something and a large number of columns is not displayed. Is there anyway to fix this? I tried reducing the text size to smallest but some columns still doesn't get displayed. I also tried reducing margins from page set up and changing the orientation to landscape and can't get the whole table printed on A4 paper size. Any ideas?" (user note in the message board.)

Problem 2:

"Wide html report will not print completely"

"I have a web-based report that is very wide and views in the browser via the horizontal scroll bar. If I print the page using window.print() only about 2/3 of the report's width prints even though I have the opportunity to tell my printer to scale the page down.

I'm having to resort to taking screenshots of the thing and paste the images together in Word.

Is there any way to get the whole thing?" (user note in the message board.)

Problem 3:

"Internet Explorer Cannot Print Large Images

<http://support.microsoft.com/?kbid=317978>

SYMPTOMS

When you try to print a large image file by using the Enhanced Metafile Format (EMF), the temporary EMF file may not be generated, the print job may fail, and you may receive the following error message:

Explorer

This program has performed an illegal operation and will be shut down.

[If the problem persists, contact the program vendor.](http://support.microsoft.com/?kbid=317978)” (Note from Microsoft site <http://support.microsoft.com/?kbid=317978>)

Problem 4:

“We provide an HTML page created from an Excel file that contains a large matrix of products and their suitability in various other server products. We use Excel to allow us to format this complicated table with enough cells to be readable. When save the Excel as HTML we can't print the HTML file and we miss a lot of columns.” (user note in the message board.)

There are a lot of cases in the HTML report in which you need to display many data fields, just like in excel. IE browser doesn't have the capability to print wide pages. To turn around the problem, a lot of reporting utilities attempt to export data to Excel and print the data in Excel. Although this is a turn-around solution, it does allow the user to change data before printing. Data security is nonexistent when the user gets data in excel format.

Print Wide HTML page just like Print Excel:

"PrintWideHTML" can print an HTML page in a similar fashion as Excel prints a worksheet. AASoftech provides this utility to convert large and wide HTMLs to PDF format or to print it directly (in version 2.0). The PDF can be tiled in one-to-one proportion, and the printout can output the HTML in multiple pages. The print layout, just like Excel, separates the result in several pages.

Advantages:

Using "PrintWideHTML"

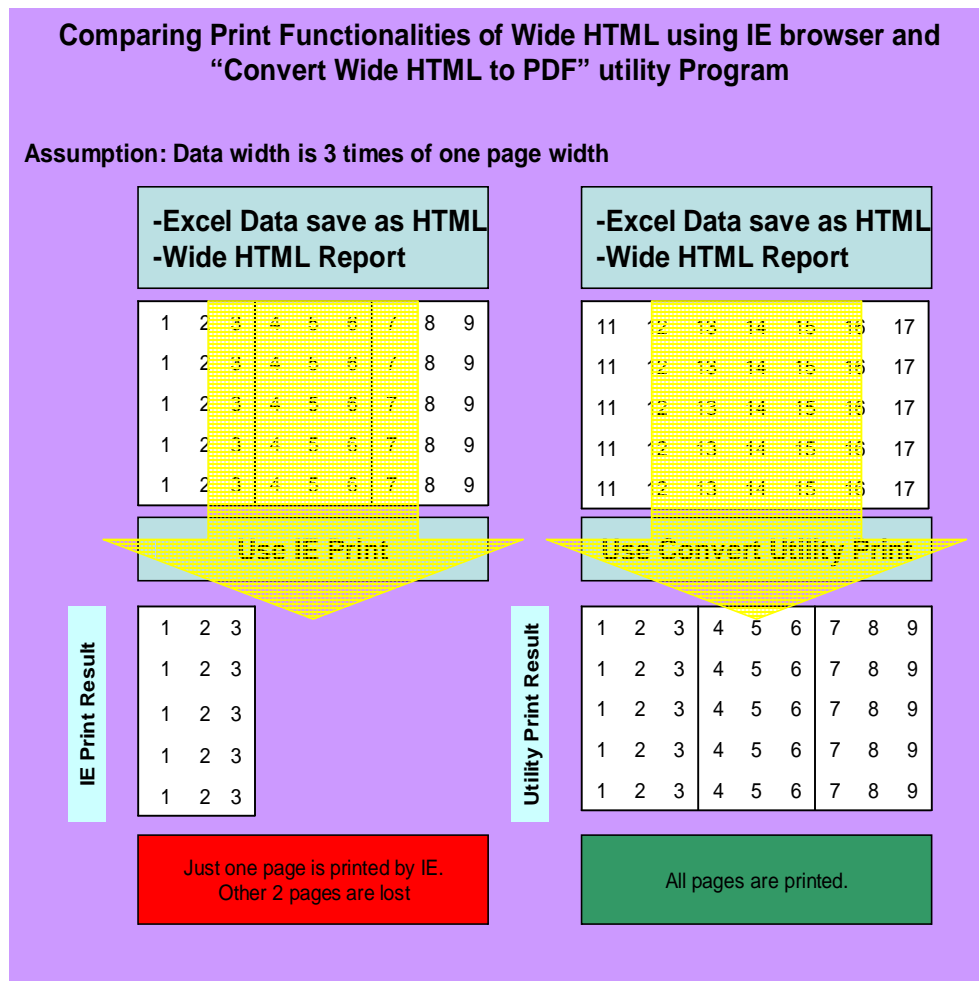
- No need to export report data to excel for printing.
- No security breaks (read-only access to data).
- No limitation in width and height of HTML page.

- Simulates Excel Print.

"PrintWideHTML" Solution Path:

"PrintWideHTML" can print a wide HTML file.

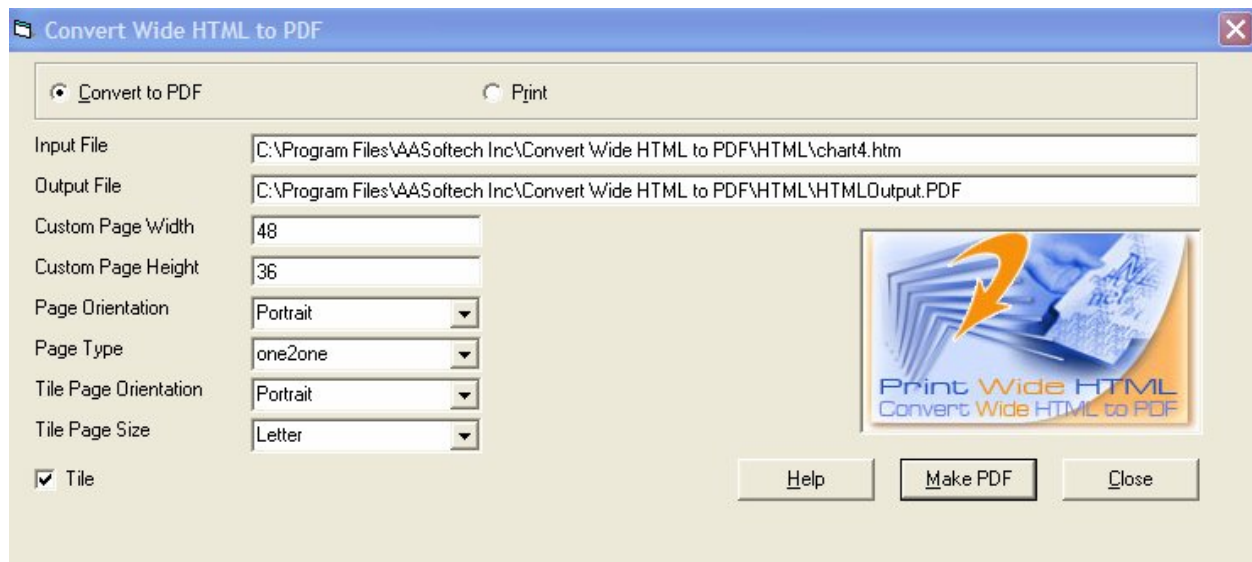
"PrintWideHTML" can also create a PDF from an HTML file. The PDF result can be printed all in one page or be printed tiled like in Print Excel.



Chapter 2: Convert to PDF User Interface

Description

The following screen describes the user interface of PrintWideHTML:



You can either use the above interface or can create your own interface. To use your customized interface you can look at Chapter 3 API Interface section.

Input File:

The completely qualified url to the file on disk or web.

Example 1: C:\Release\Chart\chart.htm

Example 2: <http://www.cnn.com>

Output File:

The complete filename of the PDF file being generated page. The directory should already exist in your computer.

Example: C:\Temp\HTMLOutput.PDF

Page Type:

A string representing one of the available page types from the list that follows:

Example: One2one

Page Types:

| Paramete | Comments |
|----------|---|
| one2one | uses the dimensions of the original web page as the output dimensions |
| Custom | uses values supplied in customPageWidth and customPageHeight |
| A0 | |
| A1 | |
| A2 | |
| A3 | |
| A4 | |
| A5 | |
| A6 | |
| B5 | |
| Letter | |
| Legal | |
| Ledger | |
| p11x17 | |

Page Orientation:

Orientation of the page: Portrait or Landscape.

Example: Landscape

Custom Page Width:

String representing a custom page width in inches.

Example: 48

Custom Page Height:

String representing a custom page height in inches.

Example: 36

Tile:

Boolean value used to set tiling status of the final printout to on or off. If this value is set to 1 the values in the [Custom Page Width](#) and [Custom Page Height](#) will be used as the size of the desired output.

Tile Page Type:

The size of paper used for the tiles. This sheet will be combined with other sheet tiles to make up the final printout. This value is only applicable if tile is set to true.

Tile Page Orientation:

Orientation of the paper for tiling: Portrait or Landscape. This value is only applicable if tile is set to true.

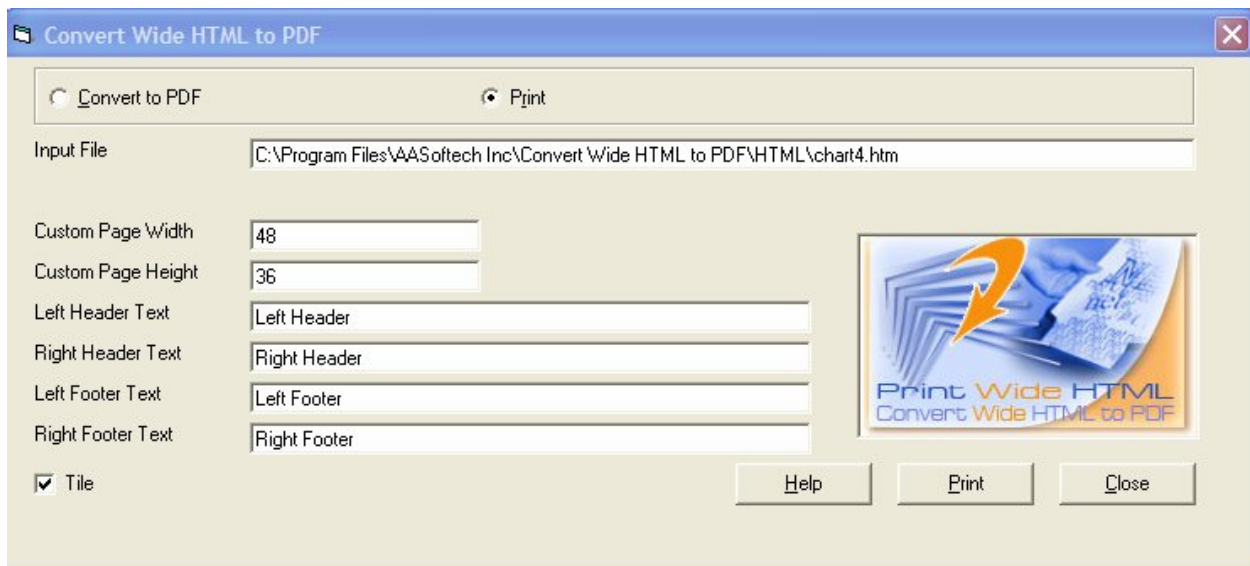
Example: Landscape

Note: The [Page Type](#) must be set to "custom" in order for these values to apply.

Chapter 3: **Print** User Interface

Description

The following screen describes the “Print” user interface of PrintWideHTML:



You can either use the above interface or can create your own interface. To use your customized interface, you can look at Chapter 4 API Interface section.

Input File:

The completely qualified url to the file on disk or web.

Example 1: C:\Release\Chart\chart.htm

Example 2: http://www.cnn.com

Custom Page Width:

String representing a custom page width in inches.

Example: 48

Custom Page Height:

String representing a custom page height in inches.

Example: 36

Left Header Text:

String representing a Header txt displayed on the left side.

Example: Header Text1

Right Header Text:

String representing a Header txt displayed on the right side.

Example: Header Text2

Left Footer Text:

String representing a Footer txt displayed on the left side.

Example: Footer Text3

Right Footer Text:

String representing a Footer txt displayed on the right side.

Example: Footer Text4

Tile:

Boolean value used to set to on or off the tiling status of the final printout. If this value is set to 1, the values in the [Custom Page Width](#) and [Custom Page Height](#) will be used as the size of the desired output.

Chapter 4: API Interface (User's Manual)

Interfacing with Print Function:

The interface to Print is `PrintHTML` function. The interface is handled through one call that is prototyped as follows:

```
int PrintHTML ();
```

The print method starts the entire print job. It will launch the ability to pick a printer; however, if the `PageSetup` was called prior to this, the printer chosen and its values will be the default values for the printer-picking dialog. This function is required to have a `returCode` integer parameter passed in.

```
int PageSetup ();
```

The printer chosen and its values will be the default values for the printer picking dialog.

`url (Input HTML File):`

The completely qualified url to the file on disk or web.

Example 1: C:\Release\Chart\chart.htm

Example 2: http://www.cnn.com

`customPageWidth (Custom Page Width):`

String representing a custom page width in inches.

Example: 48

`customPageHeight (Custom Page Height):`

String representing a custom page height in inches.

Example: 36

Tile (Tile):

Boolean value used to set tiling status of the final printout to on or off. If this value is set to 1 the values in the customPageWidth and customPageHeight will be used as the size of the desired output.

Interfacing with the PDF conversion system:

The main interface to the conversion processes the **HTMLToPDF** function. The interface is handled through one call that is prototyped as follows:

```
int HTMLToPDF (char *url, char *outputPDFFileName, char *pageType, char
*pageOrientation, char *customPageWidth, char *customPageHeight, bool tile, char
*TilePageType, char *TilePageOrientation);
```

url (Input HTML File):

The completely qualified url to the file on disk or web.

Example 1: C:\Release\Chart\chart.htm

Example 2: http://www.cnn.com

outputPDFFileName (Output PDF File):

The complete filename of the PDF file being generated page

Example: C:\Temp\HTMLOutput.PDF

pageType (Page Type):

A string representing one of the available page types from the list that follows:

Example: Onetoone

Page Types:

| Paramete | Comments |
|----------|---|
| one2one | uses the dimensions of the original web page as the output dimensions |
| Custom | uses values supplied in customPageWidth and customPageHeight |
| A0 | |
| A1 | |
| A2 | |
| A3 | |
| A4 | |
| A5 | |
| A6 | |
| B5 | |
| Letter | |
| Legal | |
| Ledger | |
| p11x17 | |

pageOrientation (Page Orientation):

Orientation of the page: Portrait or Landscape.

Example: Landscape

customPageWidth (Custom Page Width):

String representing a custom page width in inches.

Example: 48

customPageHeight (Custom Page Height):

String representing a custom page height in inches.

Example: 36

Tile (Tile):

Boolean value used to set tiling status of the final printout to on or off. If this value is set to 1 the values in the customPageWidth and customPageHeight will be used as the size of the desired output.

TilePageType (Tile Page Size):

The size of the paper used for the tiles. This sheet will be combined with other sheet tiles to make up the final printout. This value is only applicable if tile is set to true.

TilePageOrientation (Tile Page Orientation):

Orientation of the paper for tiling: Portrait or Landscape. This value is only applicable if tile is set to true.

Example: Landscape

Note: The pageType must be set to "custom" in order for these values to apply.

Return codes

The return codes generated by the [HTMLToPDF](#) process are as follows:

| Return Code | Description |
|-------------|--|
| 0 | PDF file generated successfully without errors. |
| 1 | The module could not be initialized properly. A component may be missing if this error occurs. |
| 2 | An invalid URL was passed into the conversion system. |

| | |
|---|---|
| 3 | A disk write error occurred. |
| 4 | ATLCOM error. One of the required ATL or COM objects could not be instantiated. |

Return code Helper Functions:

The size of the paper to use for the tiles. This sheet will be combined with other sheets tile to make up the final printout. This value is only applicable if tile is set to true.

```
int GetLastErrorNumber(int *errorNumber);
```

This function returns the error code of the last error that was generated. This value is the same as the return code of [HTMLToPDF](#).

```
int GetLastErrorString(char *errorMessage);
```

This function returns a more detailed description of the error that occurred.

Interfacing with Visual Basic

Interfacing with the [HTMLToPDF](#) conversion process using Visual Basic .NET involves the following:

- Importing [System.Runtime.InteropServices](#)
- Declaring the functions exposed in the DLL as follows:

```
<DllImport(PATHTODLL, CallingConvention:=CallingConvention.StdCall)> _
    Private Shared Function HTMLToPDF ( _
        ByVal url As String, _
        ByVal outputFileName As String, _
        ByVal pageType As String, _
        ByVal pageOrientation As String, _
        ByVal customWidth As String, _
        ByVal customHeight As String, _
        ByVal tile As Integer, _
```

```
ByVal tileSize As String, _  
ByVal tilePageOrientation As String) As Integer
```

```
< DllImport (PATH TODLL, CallingConvention:=CallingConvention.StdCall)> _  
Private Shared Sub GetLastErrorNumber (ByRef errorNumber As Integer)  
End Sub
```

```
< DllImport (PATH TODLL, CallingConvention:=CallingConvention.StdCall)> _  
Private Shared Sub GetLastErrorString (ByRef errorNumber As String)  
End Sub
```

Please refer to the examples folder in the distribution for an example project which demonstrates the use of the conversion system within a Visual Basic application.

Chapter 5: API Examples

VB PDF Example

The following is a VB example used to call [HTMLToPDF](#) API function:

```
Set htmltopdf = New htmltopdf.CIHTMLTOPDF

'      // setup all the variables required
htmltopdf.outputFileName = txtOutput.Text
htmltopdf.pageOrientation = cmbPageOrientation.Text
htmltopdf.pageType = cmbPageType.Text
htmltopdf.tile = chkTile.Value
htmltopdf.tilePageType = cmbTilePageSize.Text
htmltopdf.tilePageOrientation = cmbTilePageOrientation.Text
htmltopdf.url = txtInput.Text
htmltopdf.customPageHeight = txtPageHeight.Text
htmltopdf.customPageWidth = txtPageWidth.Text
'      // run the conversion
retVal = 0
htmltopdf.Render (retVal)

If retVal <> 0 Then
    errorMessage = ""
    htmltopdf.GetLastErrorMessage (errorMessage)
    lblMessage.Caption = "Error: " & errorMessage
Else
    lblMessage.Caption = "Output PDF file " & txtOutput.Text & " has been created at "
& Now()
    Screen.MousePointer = vbDefault
    MsgBox "PDF file is saved at " & txtOutput.Text
End If
```

Set obj = Nothing

VB Print Example

```
Set HTMLToPDF = New HTMLToPDF.CIHTMLTOPDF
retVal = 0
HTMLToPDF.url = txtInput.Text
HTMLToPDF.customPageHeight = txtPageHeight.Text
HTMLToPDF.customPageWidth = txtPageWidth.Text
HTMLToPDF.SetFooterLeft txtFooterLeft.Text
HTMLToPDF.SetFooterRight txtFooterRight.Text
HTMLToPDF.SetHeaderLeft txtHeaderLeft.Text
HTMLToPDF.SetHeaderRight txtHeaderRight.Text
```

```
' call the printer function to print the document
HTMLToPDF.PrintHTML (retVal)
```

```
If retVal <> 0 Then
    errorMessage = ""
    errorNumber = 0
    HTMLToPDF.GetLastErrorMessage (errorMessage)
    lblMessage.Caption = "Error: " & errorMessage
    HTMLToPDF.GetLastError (errorNumber)
    If errorNumber = 2 Then
        MsgBox "Bad Address (URL)"
    End If
Else
    lblMessage.Caption = "HTML file " & txtOutput.Text & " has been printed at " &
Now()
    Screen.MousePointer = vbDefault
    MsgBox "HTML file is printed "
End If
```

C# PDF Example

The following is a C# example used to call [HTMLToPDF](#) API function:

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Data.OleDb;
using HTMLToPDF;

namespace CSharp_Test_Harness
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        internal System.Windows.Forms.PictureBox PictureBox2;
        internal System.Windows.Forms.TextBox url;
        internal System.Windows.Forms.Button MakePDF;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();
        }
    }
}
```

```

        //
        // TODO: Add any constructor code after InitializeComponent call
        //
    }

```

```

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if (components != null)
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
    }

```

#region Windows Form Designer generated code

```

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>

```

```

    private void InitializeComponent()
    {
        this.PictureBox2 = new System.Windows.Forms.PictureBox();
        this.url = new System.Windows.Forms.TextBox();
        this.MakePDF = new System.Windows.Forms.Button();
        this.SuspendLayout();
        //

```

```

// PictureBox2
//
this.PictureBox2.BackColor =
System.Drawing.SystemColors.ControlDark;
this.PictureBox2.Location = new System.Drawing.Point(32, 56);
this.PictureBox2.Name = "PictureBox2";
this.PictureBox2.Size = new System.Drawing.Size(464, 8);
this.PictureBox2.TabIndex = 28;
this.PictureBox2.TabStop = false;
//
// url
//
this.url.Location = new System.Drawing.Point(48, 9);
this.url.Name = "url";
this.url.Size = new System.Drawing.Size(320, 20);
this.url.TabIndex = 22;
this.url.Text = "C:\\Softech\\bin\\Release\\chart4\\chart4.htm";
//
// MakePDF
//
this.MakePDF.Location = new System.Drawing.Point(384, 9);
this.MakePDF.Name = "MakePDF";
this.MakePDF.Size = new System.Drawing.Size(96, 24);
this.MakePDF.TabIndex = 21;
this.MakePDF.Text = "Make PDF";
this.MakePDF.Click += new
System.EventHandler(this.MakePDF_Click);
//
// Form1
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(520, 86);
this.Controls.Add(this.PictureBox2);

```

```

        this.Controls.Add(this.url);
        this.Controls.Add(this.MakePDF);
        this.Name = "Form1";
        this.Text = "CSharp Test Harness";
        this.Load += new System.EventHandler(this.Form1_Load);
        this.ResumeLayout(false);
    }
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void MakePDF_Click(object sender, System.EventArgs e)
{
    // get access to the class that will do the work
    HTMLToPDF.CIHTMLTOPDFClass htmltopdf = new
HTMLToPDF.CIHTMLTOPDFClass();

    // setup all the variables required
    htmltopdf.outputFileName = "chart5.pdf";
    htmltopdf.pageOrientation = "Landscape";
    htmltopdf.pageType = "custom";
    htmltopdf.tile = 1;
    htmltopdf.tilePageType = "Letter";
    htmltopdf.tilePageOrientation = "Landscape";
}

```

```
htmltopdf.url = url.Text;
htmltopdf.customPageHeight = "42";
htmltopdf.customPageWidth = "36";

// run the conversion

int retVal = 0;

htmltopdf.Render(ref retVal);

if(retVal!=0)
{
string errorMessage = "";

htmltopdf.GetLastErrorMessage(ref errorMessage);

}

}

private void Form1_Load(object sender, System.EventArgs e)
{
}
}
}
```

C# Print Example

The following is a C# example used to call [PrintHTML](#) API function:

```
private void Print_Click(object sender, System.EventArgs e)
{
    // set the default return value to no error
    int retVal = 0;

    // assign the URL to be printed
    htmltopdf.url = url.Text;

    // check to see if the user wants a tile or just a regular output
    if( TilePrinting.Checked == true)
        htmltopdf.pageType = "custom";
    else
        htmltopdf.pageType = "Letter";

    // get the custom width and height from the edit boxes on the Form
    htmltopdf.customPageWidth = customWidth.Text;
    htmltopdf.customPageHeight = customHeight.Text;

    // turns the header and footer visibility off
    // htmltopdf.SetHeaderFooterVisibility(0);

    // turn on visibility of all the header and footers
    htmltopdf.SetHeaderFooterVisibility(15);

    // set each of the header and footer strings
    htmltopdf.SetFooterLeft("Footer Left");
    htmltopdf.SetFooterRight("Footer Right");
    htmltopdf.SetHeaderLeft("Header Left");
    htmltopdf.SetHeaderRight("Header Right");
}
```

```
// call the printer function to print the document
htmltopdf.PrintHTML(ref retVal);
```

```
if(retVal!=0)
{
    int errorNumber = 0;
    htmltopdf.GetLastError(ref errorNumber);
    if(errorNumber==2)
        MessageBox.Show("Error","Bad URL");
    else
        MessageBox.Show("Error","Error occured");
}
}
```

```
private void MakePDF_Click(object sender, System.EventArgs e)
```

```
{
    // get access to the class that will do the work
```

```
String outputFileName = "c:\\chart.pdf";
```

```
// setup all the variables required
```

```
htmltopdf.outputFileName = outputFileName;
```

```
htmltopdf.pageOrientation = "Landscape";
```

```
htmltopdf.pageType = "custom";
```

```
htmltopdf.tile = 1;
```

```
htmltopdf.tilePageType = "Letter";
```

```
htmltopdf.tilePageOrientation = "Landscape";
```

```
htmltopdf.url = url.Text;
```

```
htmltopdf.customPageHeight = "42";
```

```

htmltopdf.customPageWidth = "36";

// run the conversion

int retVal = 0;

htmltopdf.Render(ref retVal);

if(retVal!=0)
{
    int errorNumber = 0;
    htmltopdf.GetLastError(ref errorNumber);
    if(errorNumber==2)
        MessageBox.Show("Error","Bad URL");
    else
        MessageBox.Show("Error","Error occured");
}
else
{
    System.Diagnostics.Process.Start(outputFileName);
}
}

```

VB.Net PDF Example

The following is a VB.Net example used to call [HTMLToPDF](#) API function:

Imports System.Runtime.InteropServices

```
Public Class HTMLtoPDF_TestHarness
```

```
Inherits System.Windows.Forms.Form
```

```
Public Const PATHTODLL As String = "iHTMLCapture.dll"
```

```
#Region " Windows Form Designer generated code "
```

```
Public Sub New()
```

```
MyBase.New()
```

```
'This call is required by the Windows Form Designer.
```

```
InitializeComponent()
```

```
'Add any initialization after the InitializeComponent() call
```

```
End Sub
```

```
'Form overrides dispose to clean up the component list.
```

```
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
If disposing Then
```

```
    If Not (components Is Nothing) Then
```

```
        components.Dispose()
```

```
    End If
```

```
End If
```

```
MyBase.Dispose(disposing)
```

```
End Sub
```

```
'Required by the Windows Form Designer
```

```
Private components As System.ComponentModel.IContainer
```

'NOTE: The following procedure is required by the Windows Form Designer

'It can be modified using the Windows Form Designer.

'Do not modify it using the code editor.

Friend WithEvents MakePDF As System.Windows.Forms.Button

Friend WithEvents url As System.Windows.Forms.TextBox

Friend WithEvents PortraitLandscape As System.Windows.Forms.ComboBox

Friend WithEvents pageType As System.Windows.Forms.ComboBox

Friend WithEvents Label1 As System.Windows.Forms.Label

Friend WithEvents Label2 As System.Windows.Forms.Label

Friend WithEvents customPageWidth As System.Windows.Forms.Label

Friend WithEvents Label3 As System.Windows.Forms.Label

Friend WithEvents TileCheckBox As System.Windows.Forms.CheckBox

Friend WithEvents TileCustomPageWidth As System.Windows.Forms.TextBox

Friend WithEvents TileCustomPageHeight As System.Windows.Forms.TextBox

Friend WithEvents TilePageSize As System.Windows.Forms.ComboBox

Friend WithEvents Label4 As System.Windows.Forms.Label

Friend WithEvents Label5 As System.Windows.Forms.Label

Friend WithEvents TilePageOrientation As System.Windows.Forms.ComboBox

Friend WithEvents Label6 As System.Windows.Forms.Label

Friend WithEvents Label7 As System.Windows.Forms.Label

Friend WithEvents txtOutput As System.Windows.Forms.TextBox

Friend WithEvents Button1 As System.Windows.Forms.Button

<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()

Me.MakePDF = New System.Windows.Forms.Button

Me.url = New System.Windows.Forms.TextBox

Me.PortraitLandscape = New System.Windows.Forms.ComboBox

Me.pageType = New System.Windows.Forms.ComboBox

Me.Label1 = New System.Windows.Forms.Label

Me.Label2 = New System.Windows.Forms.Label

Me.TileCustomPageWidth = New System.Windows.Forms.TextBox

Me.TileCustomPageHeight = New System.Windows.Forms.TextBox

Me.customPageWidth = New System.Windows.Forms.Label

```

Me.Label3 = New System.Windows.Forms.Label
Me.TileCheckBox = New System.Windows.Forms.CheckBox
Me.TilePageSize = New System.Windows.Forms.ComboBox
Me.Label4 = New System.Windows.Forms.Label
Me.Label5 = New System.Windows.Forms.Label
Me.TilePageOrientation = New System.Windows.Forms.ComboBox
Me.Label6 = New System.Windows.Forms.Label
Me.Label7 = New System.Windows.Forms.Label
Me.txtOutput = New System.Windows.Forms.TextBox
Me.Button1 = New System.Windows.Forms.Button
Me.SuspendLayout()
'
'MakePDF
'
Me.MakePDF.Location = New System.Drawing.Point(296, 344)
Me.MakePDF.Name = "MakePDF"
Me.MakePDF.Size = New System.Drawing.Size(96, 24)
Me.MakePDF.TabIndex = 10
Me.MakePDF.Text = "&Make PDF"
'
'url
'
Me.url.Location = New System.Drawing.Point(32, 32)
Me.url.Name = "url"
Me.url.Size = New System.Drawing.Size(376, 20)
Me.url.TabIndex = 1
Me.url.Text = ""
'
'PortraitLandscape
'
Me.PortraitLandscape.Items.AddRange(New Object() {"Landscape", "Portrait"})
Me.PortraitLandscape.Location = New System.Drawing.Point(32, 184)
Me.PortraitLandscape.Name = "PortraitLandscape"

```

```

Me.PortraitLandscape.Size = New System.Drawing.Size(128, 21)
Me.PortraitLandscape.TabIndex = 3
Me.PortraitLandscape.Text = "Portrait"
'
'pageType
'
Me.pageType.Items.AddRange(New Object() {"one2one", "custom", "A0", "A1",
"A2", "A3", "A4", "A5", "A6", "B5", "Letter", "Legal", "Ledger", "p11x17"})
Me.pageType.Location = New System.Drawing.Point(32, 240)
Me.pageType.Name = "pageType"
Me.pageType.Size = New System.Drawing.Size(128, 21)
Me.pageType.TabIndex = 6
Me.pageType.Text = "custom"
'
'Label1
'
Me.Label1.Location = New System.Drawing.Point(32, 160)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(104, 24)
Me.Label1.TabIndex = 10
Me.Label1.Text = "Page Orientation"
'
'Label2
'
Me.Label2.Location = New System.Drawing.Point(32, 216)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(104, 24)
Me.Label2.TabIndex = 11
Me.Label2.Text = "Page Type"
'
'TileCustomPageWidth
'
Me.TileCustomPageWidth.Location = New System.Drawing.Point(192, 184)

```

```

Me.TileCustomPageWidth.Name = "TileCustomPageWidth"
Me.TileCustomPageWidth.Size = New System.Drawing.Size(72, 20)
Me.TileCustomPageWidth.TabIndex = 4
Me.TileCustomPageWidth.Text = "48"
'
'TileCustomPageHeight
'
Me.TileCustomPageHeight.Location = New System.Drawing.Point(336, 184)
Me.TileCustomPageHeight.Name = "TileCustomPageHeight"
Me.TileCustomPageHeight.Size = New System.Drawing.Size(72, 20)
Me.TileCustomPageHeight.TabIndex = 5
Me.TileCustomPageHeight.Text = "36"
'
'customPageWidth
'
Me.customPageWidth.Location = New System.Drawing.Point(192, 160)
Me.customPageWidth.Name = "customPageWidth"
Me.customPageWidth.Size = New System.Drawing.Size(112, 16)
Me.customPageWidth.TabIndex = 14
Me.customPageWidth.Text = "Custom Page Width"
'
'Label3
'
Me.Label3.Location = New System.Drawing.Point(336, 160)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(112, 16)
Me.Label3.TabIndex = 15
Me.Label3.Text = "Custom Page Height"
'
'TileCheckBox
'
Me.TileCheckBox.Location = New System.Drawing.Point(192, 240)
Me.TileCheckBox.Name = "TileCheckBox"

```

```

Me.TileCheckBox.Size = New System.Drawing.Size(80, 24)
Me.TileCheckBox.TabIndex = 7
Me.TileCheckBox.Text = "Tile"
'
'TilePageSize
'
Me.TilePageSize.Items.AddRange(New Object() {"A0", "A1", "A2", "A3", "A4", "A5",
"A6", "B5", "Letter", "Legal", "Ledger", "p11x17"})
Me.TilePageSize.Location = New System.Drawing.Point(336, 240)
Me.TilePageSize.Name = "TilePageSize"
Me.TilePageSize.Size = New System.Drawing.Size(128, 21)
Me.TilePageSize.TabIndex = 8
Me.TilePageSize.Text = "Letter"
'
'Label4
'
Me.Label4.Location = New System.Drawing.Point(336, 216)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(112, 16)
Me.Label4.TabIndex = 18
Me.Label4.Text = "Tile Page Size"
'
'Label5
'
Me.Label5.Location = New System.Drawing.Point(336, 280)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(120, 24)
Me.Label5.TabIndex = 20
Me.Label5.Text = "Tile Page Orientation"
'
'TilePageOrientation
'
Me.TilePageOrientation.Items.AddRange(New Object() {"Landscape", "Portrait"})

```

Me.TilePageOrientation.Location = New System.Drawing.Point(336, 304)

Me.TilePageOrientation.Name = "TilePageOrientation"

Me.TilePageOrientation.Size = New System.Drawing.Size(128, 21)

Me.TilePageOrientation.TabIndex = 9

Me.TilePageOrientation.Text = "Portrait"

,

'Label6

,

Me.Label6.Location = New System.Drawing.Point(32, 8)

Me.Label6.Name = "Label6"

Me.Label6.Size = New System.Drawing.Size(104, 16)

Me.Label6.TabIndex = 21

Me.Label6.Text = "Input HTML File"

,

'Label7

,

Me.Label7.Location = New System.Drawing.Point(32, 72)

Me.Label7.Name = "Label7"

Me.Label7.Size = New System.Drawing.Size(104, 24)

Me.Label7.TabIndex = 22

Me.Label7.Text = "Output PDF File"

,

'txtOutput

,

Me.txtOutput.Location = New System.Drawing.Point(32, 96)

Me.txtOutput.Name = "txtOutput"

Me.txtOutput.Size = New System.Drawing.Size(376, 20)

Me.txtOutput.TabIndex = 2

Me.txtOutput.Text = "C:\Temp\HTMLOutput.PDF"

,

'Button1

,

Me.Button1.Location = New System.Drawing.Point(408, 344)

```
Me.Button1.Name = "Button1"  
Me.Button1.Size = New System.Drawing.Size(96, 24)  
Me.Button1.TabIndex = 11  
Me.Button1.Text = "&Close"  
,  
'HTMLtoPDF_TestHarness'  
,  
  
Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)  
Me.ClientSize = New System.Drawing.Size(512, 382)  
Me.Controls.Add(Me.Button1)  
Me.Controls.Add(Me.txtOutput)  
Me.Controls.Add(Me.Label7)  
Me.Controls.Add(Me.Label6)  
Me.Controls.Add(Me.Label5)  
Me.Controls.Add(Me.TilePageOrientation)  
Me.Controls.Add(Me.Label4)  
Me.Controls.Add(Me.TilePageSize)  
Me.Controls.Add(Me.TileCheckBox)  
Me.Controls.Add(Me.Label3)  
Me.Controls.Add(Me.customPageWidth)  
Me.Controls.Add(Me.TileCustomPageHeight)  
Me.Controls.Add(Me.TileCustomPageWidth)  
Me.Controls.Add(Me.Label2)  
Me.Controls.Add(Me.Label1)  
Me.Controls.Add(Me.pageType)  
Me.Controls.Add(Me.PortraitLandscape)  
Me.Controls.Add(Me.url)  
Me.Controls.Add(Me.MakePDF)  
Me.Name = "HTMLtoPDF_TestHarness"  
Me.Text = "HTML to PDF Test Harness Application"  
Me.ResumeLayout(False)
```

End Sub

#End Region

' Define the three access methods of the DLL

```
<DllImport(PATHTODLL, CallingConvention:=CallingConvention.StdCall)> _  
Private Shared Function HTMLtoPDF( _  
    ByVal url As String, _  
    ByVal outputFileName As String, _  
    ByVal pageType As String, _  
    ByVal pageOrientation As String, _  
    ByVal customWidth As String, _  
    ByVal customHeight As String, _  
    ByVal tile As Integer, _  
    ByVal tileSize As String, _  
    ByVal tilePageOrientation As String) As Integer
```

End Function

```
<DllImport(PATHTODLL, CallingConvention:=CallingConvention.StdCall)> _  
    Private Shared Sub GetLastErrorNumber(ByRef errorNumber As Integer)  
End Sub
```

```
<DllImport(PATHTODLL, CallingConvention:=CallingConvention.StdCall)> _  
    Private Shared Sub GetLastErrorString(ByRef errorNumber As String)  
End Sub
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load
```

End Sub

Private Sub MakePDF_Click(**ByVal** sender **As** System.Object, **ByVal** e **As** System.EventArgs) **Handles** MakePDF.Click

Dim tilePageSz **As** String

Dim tilePageOrient **As** String

Dim tile **As** Integer

tilePageSz = TilePageSize.Text

tilePageOrient = tilePageOrientation.Text

tile = TileCheckBox.CheckState

Dim murl **As** String

murl = url.Text()

Dim outputFile **As** String = txtOutput.Text

' note * if paper size is "custom" values are active in customPageHeight
' and customPageWidth

' note 2 if paper size is One2One the size of the web page is used
' as the paper size

' Page Types:

' one2one uses dimensions in pixels of web page

' custom uses values supplied in customPageWidth and
customPageHeight

' A0

' A1

' A2

' A3

' A4

```
' A5  
' A6  
' B5  
' Letter  
' Legal  
' Ledger  
' p11x17
```

```
Dim paperSize As String = pageType.Text()
```

```
Dim paperOrientation As String = PortraitLandscape.Text()
```

```
' custom page width output size
```

```
Dim customPageWidth As String = TileCustomPageWidth.Text
```

```
' vustom page height output size
```

```
Dim customPageHeight As String = TileCustomPageHeight.Text
```

```
Dim returnCode As Integer
```

```
Dim errorNumber As Integer
```

```
Dim errorMessage As String
```

```
If (url.Text() = "") Then
```

```
    MsgBox("Please Enter the Input HTML File path.")
```

```
    Exit Sub
```

```
End If
```

```
If (outputFile.ToString() = "") Then
```

```
    MsgBox("Please Enter the Output PDF File path.")
```

```
    Exit Sub
```

```
End If
```

```
' call the DLL
```

```
returnCode = HTMLtoPDF(murl, outputFile, paperSize, paperOrientation,  
customPageWidth, customPageHeight, tile, tilePageSz, tilePageOrient)
```

```

If (returnCode <> 0) Then
    ' Process error here
    GetLastErrorNumber(errorNumber)
    GetLastErrorString(errorMessage)
    MsgBox(errorMessage)

Else
    ' Launch the adobe reader
    System.Diagnostics.Process.Start(outputFile)

End If

```

```
End Sub
```

```
Private Sub ProcessFileAsPDF(ByVal fileName As String, ByVal chartNumber As Integer)
```

```

    Dim murl As String
    murl = fileName
    Dim outputFile As String = "C:\Softech\bin\Examples\output\chart" +
    Str$(chartNumber) + ".pdf"
    Dim paperSize As String = pageType.Text()
    Dim paperOrientation As String = PortraitLandscape.Text()
    ' custom page width output size
    ' custom page width output size
    Dim customPageWidth As String = TileCustomPageWidth.Text
    ' vustom page height output size
    Dim customPageHeight As String = TileCustomPageHeight.Text
    Dim returnCode As Integer
    Dim errorNumber As Integer

```

```

Dim errorMessage As String

Dim tilePageSz As String
Dim tilePageOrient As String
Dim tile As Integer

tilePageSz = TilePageSize.Text
tilePageOrient = TilePageOrientation.Text
tile = TileCheckBox.CheckState

' call the DLL
returnCode = HTMLtoPDF(murl, outputFile, paperSize, paperOrientation,
customPageWidth, customPageHeight, tile, tilePageSz, tilePageOrient)

If (returnCode <> 0) Then
    ' Process error here
    GetLastErrorNumber(errorNumber)
    GetLastErrorString(errorMessage)
    MsgBox(errorMessage)

Else
    ' Launch the adobe reader
    System.Diagnostics.Process.Start(outputFile)

End If

End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Close()
End Sub

```

```
Private Sub TileCheckBox_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TileCheckBox.CheckedChanged
```

```
End Sub
```

```
End Class
```

VB.Net Print Example

The following is a VB.Net example used to call `PrintHTML` API function:

```
Private Sub Print_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Print.Click
```

```
    ' set the default return value to no error
```

```
    Dim retVal As Int32
```

```
    retVal = 0
```

```
    '           // assign the URL to be printed
```

```
    htmltopdf.url = url.Text
```

```
    ' // check to see if the user wants a tile or just a regular output
```

```
    If (TilePrinting.Checked = True) Then
```

```
        htmltopdf.pageType = "custom"
```

```
    Else
```

```
        htmltopdf.pageType = "Letter"
```

```
    End If
```

```
    ' get the custom width and height from the edit boxes on the Form
```

```
    htmltopdf.customPageWidth = customWidth.Text
```

```
    htmltopdf.customPageHeight = customHeight.Text
```

```
    ' turns the header and footer visibility off
```

```
    ' htmltopdf.SetHeaderFooterVisibility(0);
```

```
' turn on visibility of all the header and footers
```

```
htmltopdf.SetHeaderFooterVisibility(15)
```

```
' set each of the header and footer strings
```

```
htmltopdf.SetFooterLeft("Footer Left")
```

```
htmltopdf.SetFooterRight("Footer Right")
```

```
htmltopdf.SetHeaderLeft("Header Left")
```

```
htmltopdf.SetHeaderRight("Header Right")
```

```
' call the printer function to print the document
```

```
htmltopdf.PrintHTML(retVal)
```

```
End Sub
```

```
Private Sub PrinterSetup_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles PrinterSetup.Click
```

```
Dim retVal As Integer
```

```
htmltopdf.PageSetup(retVal)
```

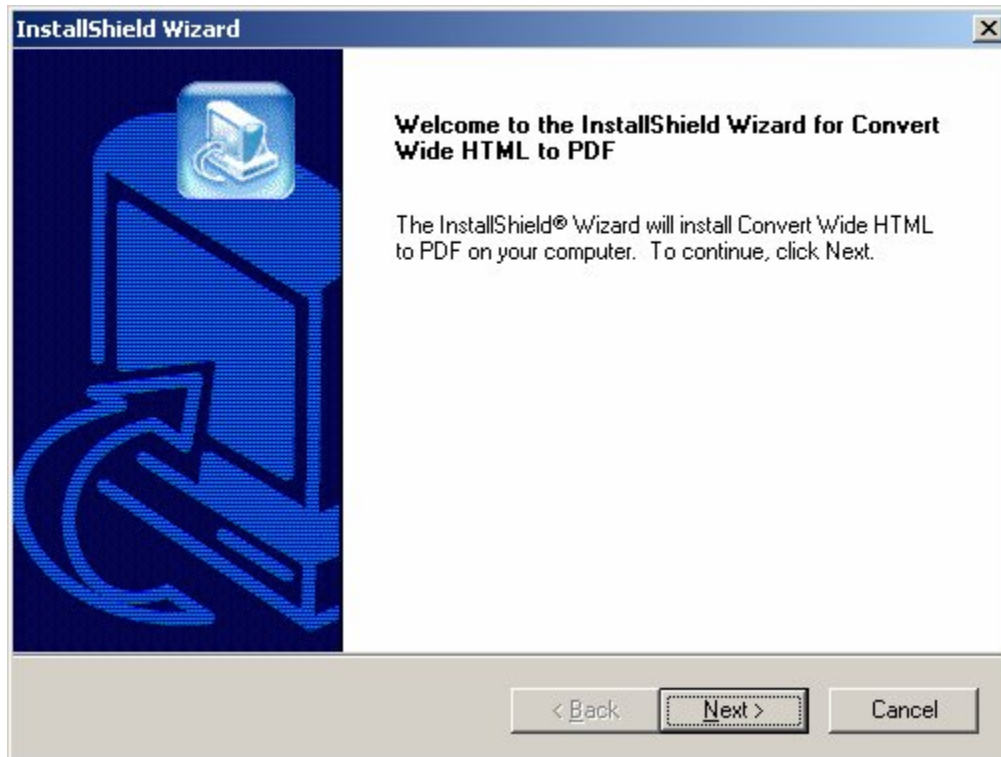
```
End Sub
```

Installation Guide

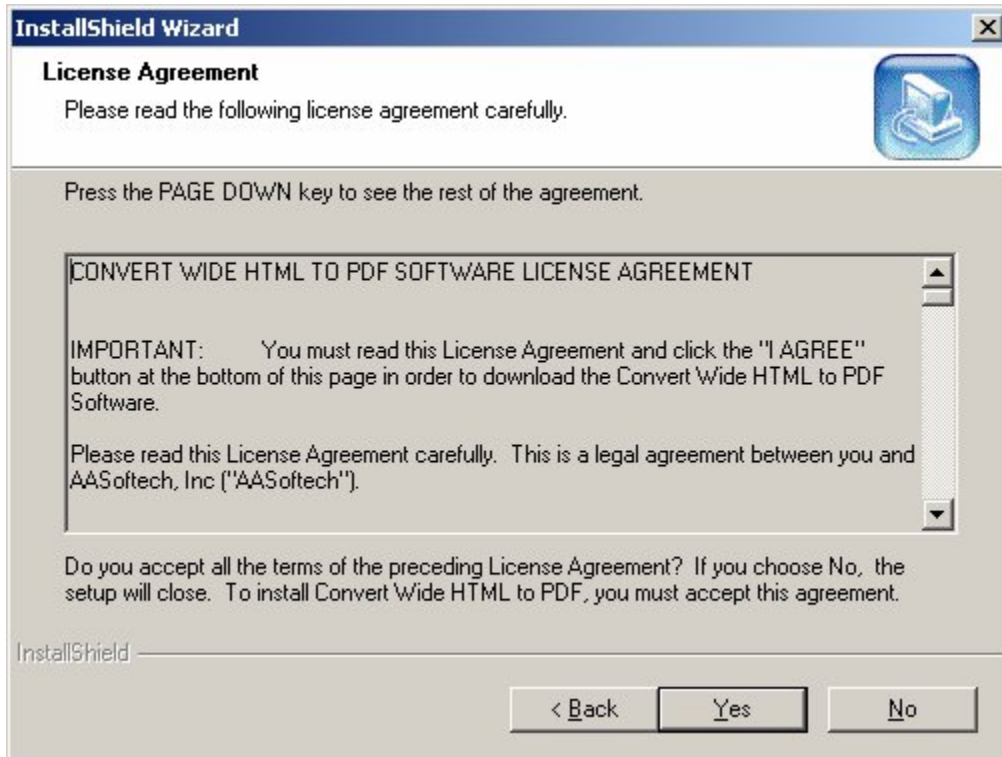
Application can be installed on XP, Windows 2000, Windows 2003, Windows 2005. It can be used as desktop and server-side component should you purchase the right license. Following is a step by step guide line to install the software::

Unzip the file.

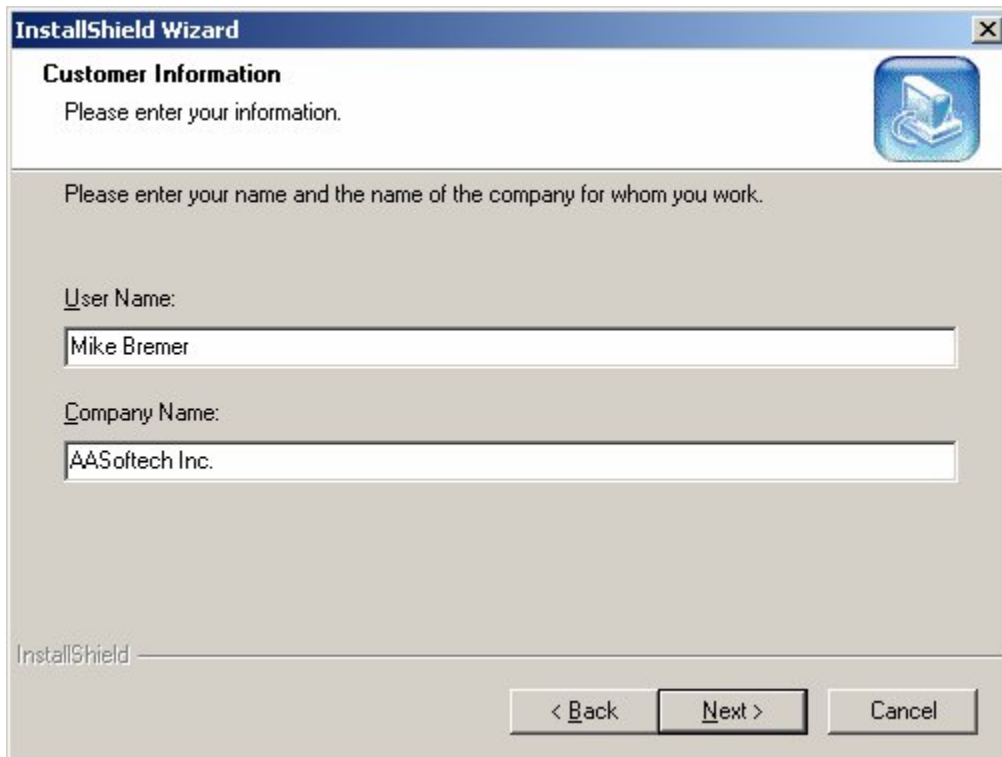
Run the MSI File. You will see the following wizard.



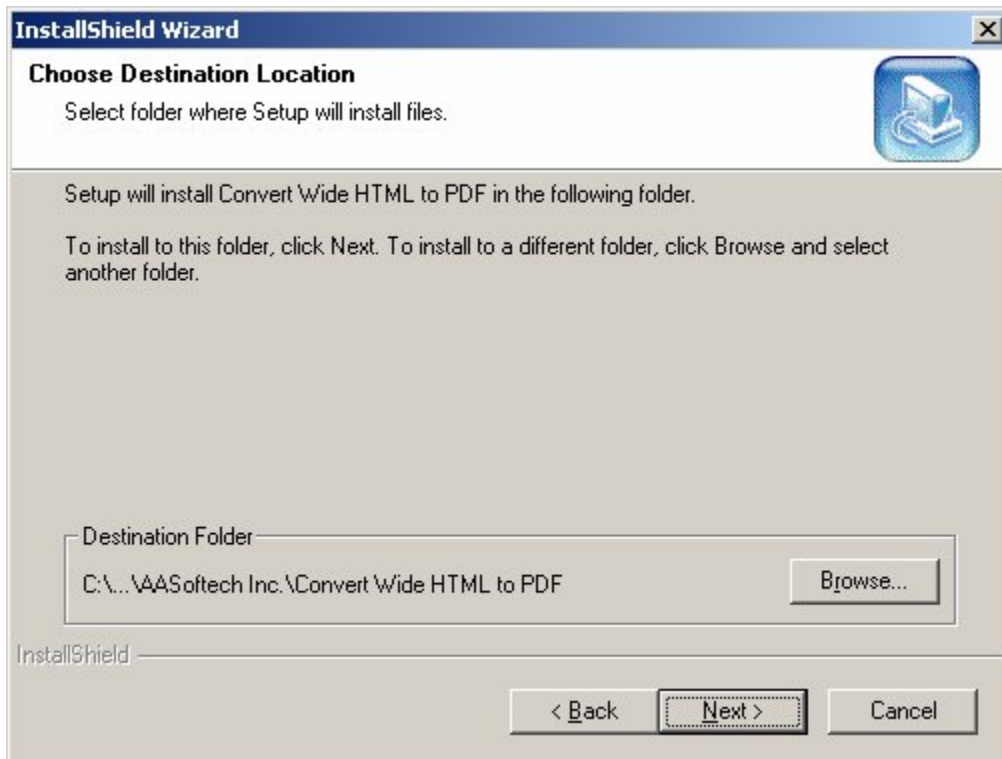
Follow the installation Wizard.



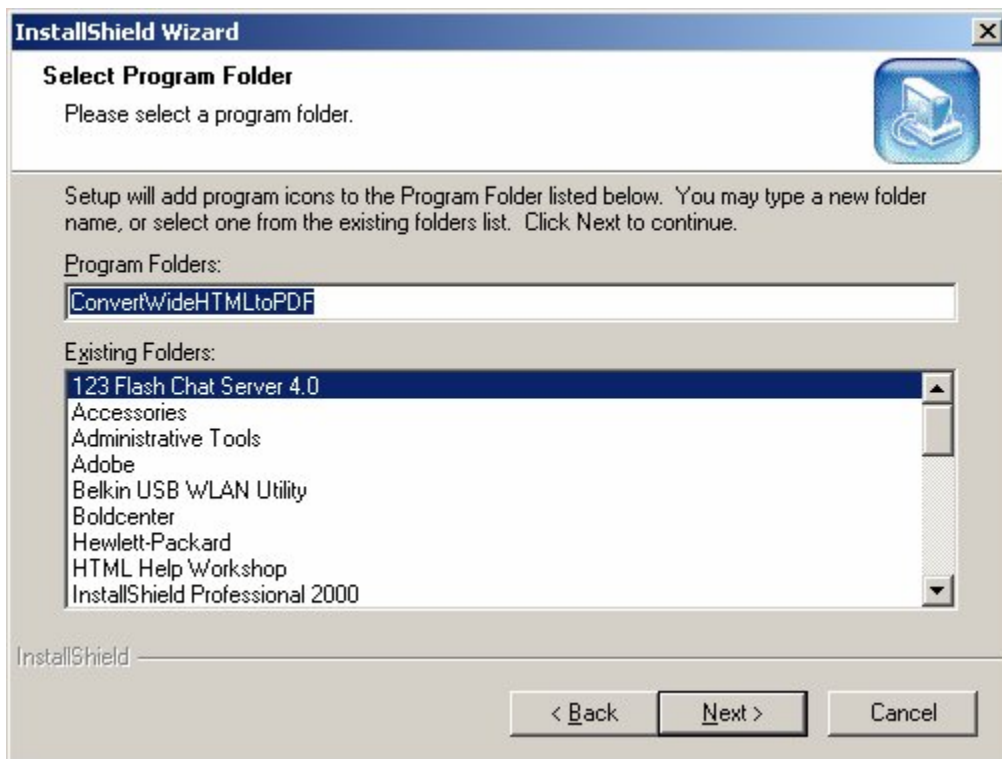
Read and Enter "Yes" for the license agreement if you agree.



Write User Name and Company Name and click on "Next".



Click “Next” if you agree using default Destination Folder.



Click “Next” if you agree with default “Program Folder”.

After the installation is finished you can run the program from “All Programs” Menu Bar.