White Paper

BarTender's ActiveX Automation Interface

Controlling BarTender using Programming Languages not in the ".NET" Family

SEAGULL SCIENTIFIC

Contents

Introduction to Automated Label Printing with BarTender

When your label and envelope printing needs are complex or label-printing performance is mission-critical for your company, the need for user-intervention in the label-printing process can be a significant roadblock to productivity. This is especially true because labels are usually needed in an automated, software-driven, process such as order fulfillment, inventory control, returns processing, and complaint handling.

Unfortunately, many of these software applications have no built-in label printing functions. Other applications have very simple label printing functions, but do not support the concept of variable data being read from multiple data sources or input by end-users in response to print-time prompts.

BarTender's support for ActiveX Automation allows developers to enhance their Windows applications with code that instructs BarTender to automatically perform all of the data-retrieval, user-prompting, and label printing tasks available when using BarTender "stand-alone." It's as you put BarTender's label printing functions inside of your other program.

Automation using C# and VB.Net

If you are using a .NET programming language, such as C# or VB.NET, we recommend that you use one of BarTender .NET SDKs instead of directly accessing the ActiveX Automation interface. The SDKs provide you with prewritten procedures that access and control the ActiveX Automation functions for you. Although your programs will still be controlling BarTender using ActiveX at a low level, use of the SDKs significantly reduces the amount of programming and debugging required. (See the "BarTender's .NET SDKs" white paper for more details.)

Integration on Non-Windows Platforms

For information on integrating BarTender's label printing functions into non-Windows applications, please see *BarTender Automation in Non-Windows Environments* below.

What is ActiveX Automation?

Overview

ActiveX Automation, also known as COM (Common Object Model) or simply Automation, is a Microsoft standard for interaction between Windows programs. The standard enables one application to use the functions of another application in such a smoothly integrated way that the two programs appear to be a single program. The former application is known as the *client* and the latter is called the *server*.

ActiveX Clients and Servers

The client application can be any Windows application that is capable of issuing commands in any of the COM-supporting programming or scripting languages, including:

- Visual Basic, VBA (Visual Basic for Applications), and VBScript (Visual Basic Script)
- Visual C, Visual C++, and other versions of C for Windows
- Java, Visual J++, Visual J#, JavaScript, and JScript
- Any language for which there is an ActiveX scripting engine that runs in the Windows Scripting Host, including PERL, Python, and REXX

The server application can be any Windows application that has "exposed" one or more programming objects, along with the *properties* and *methods* (that is, functions) of the objects, in an API (Application Programming Interface).

Support for VB.NET and C#

Although VB.NET and C# both support ActiveX Automation, it is much easier to use the BarTender .NET SDKs when using those languages. Please see the "BarTender's .NET SDKs" white paper for more details.

Objects, Properties, and Methods

Typically, the root-level object exposed by the server application is called "Application," and a client application launches the server application by "creating" an instance of the Application object. The client application then interacts with the server application by reading and writing the server application's properties and invoking its methods. It also creates and manipulates other kinds of objects made available by the server application's API.

The kinds of objects made available will vary, of course, with the kind of services provided by the server application. A database application would typically enable the creation of table, row, and column objects, among others. An e-mail application would provide message, subject line, and sender objects, among others.

Besides being objects in themselves, rows and columns are parts of tables, and a subject line is a part of an e-mail message. This illustrates an important aspect of ActiveX Automation: the objects in a well-designed API have a hierarchical relation. Some of them are properties of parent objects; and, in turn, typically some of a child-object's properties are themselves objects. Not all properties, however, are also objects: a column object in a database table would typically have a *datatype* property that is not itself an object.

A *method* is a function of an object; that is, it is some action that the object can perform on itself or on some other object. For example, a column object would typically have a *multiply* method by which it could multiply every one of its cells by some value. And, of course, a message object in an e-mail application would need a *send* method to send itself.

Note: In addition to the Windows Scripting Host, full-featured web browsers such as Internet Explorer contain their own script-running engines. Hence, COM-compatible server applications, like BarTender, can be run from within web pages.

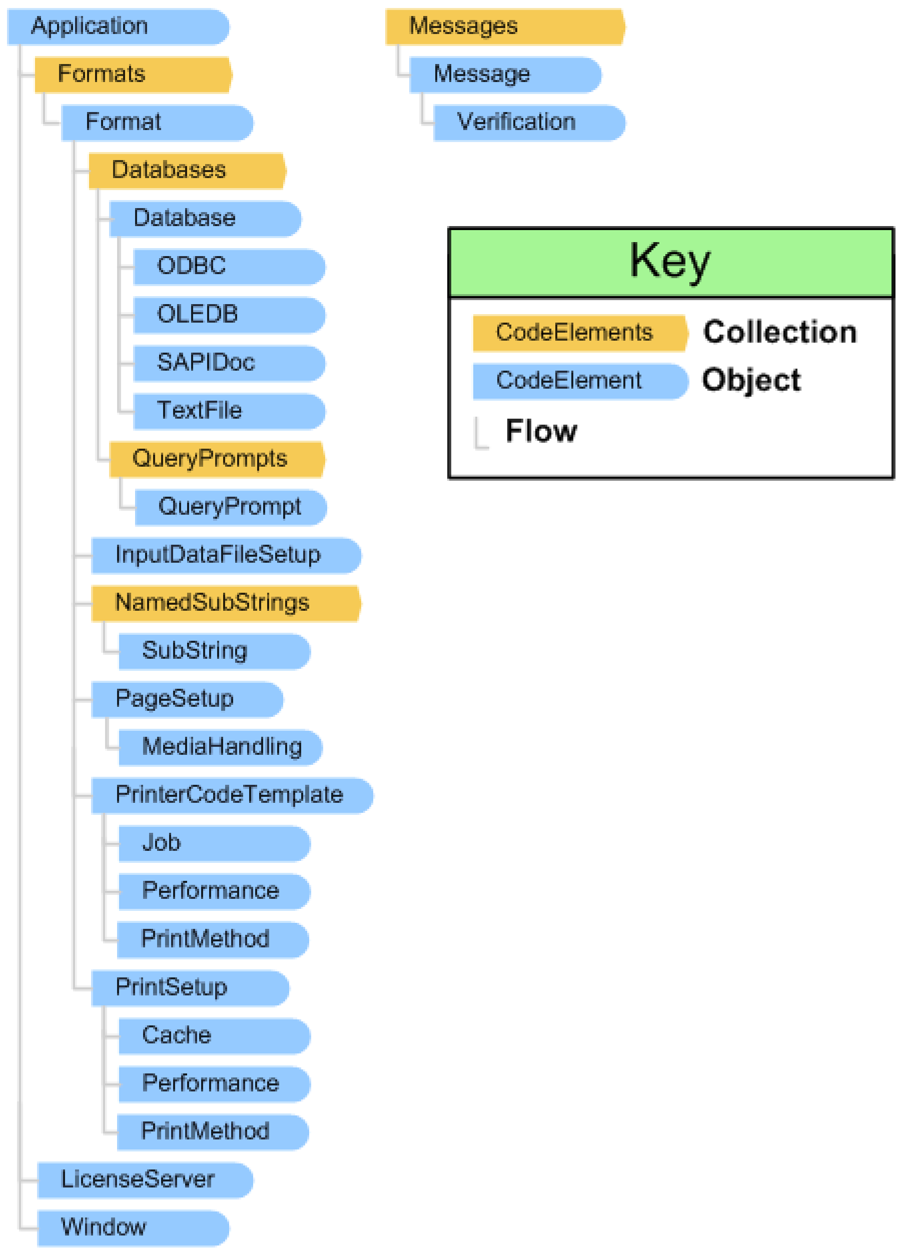
BarTender's ActiveX API

Overview

BarTender's API makes all of BarTender's printing and dynamic data-retrieval features available to client applications. These features include:

- Loading multiple instances of BarTender for improved performance.
- Specifying particular label formats for each print job.
- Specifying the printer(s) to be used.
- Specifying the number of identical copies, and/or a set of serialized labels, to be printed.
- Identifying the data source and the *type* of database (text file, ODBC, OLE DB, or SAP IDoc) for each named field on the label.
- Providing subsidiary information needed for data retrieval, such as the name of the IDoc Configuration file, the text of a SQL query, and the delimitation type of a text file.
- Creating a print-time prompt for a print job and the default reply to the prompt.
- Establishing communication with the Seagull License Server.

A detailed reference for the BarTender API is in BarTender's online help. The following is an overview of the hierarchy of objects.



Example of Automating BarTender with C#

This section explains how to create a simple C# application that uses BarTender's ActiveX Automation interface. These instructions assume basic familiarity with Microsoft's Visual Studio .NET.

A Quick Note About BarTender's .NET SDKs

Although our example below of how to programmatically control BarTender uses C# as the programming language, when you are using C# or VB.NET, it is much easier to use the BarTender .NET SDKs instead of directly using the ActiveX Automation Interface. Please see the "BarTender's .NET SDKs" white paper for more details.

Preliminaries

1. If you are not familiar with substrings and share/names in BarTender text objects, consult BarTender's online help about these topics.

2. Create a label format with a text object containing two substrings.

3. Assign "testing" as a share/name to one of the substrings.

4. Verify that you can print the format.

5. Save this format as "test.btw" to the root of your C drive.

Creating an Application

Specifying a Project Type

Open Visual Studio and select C# for the **Project Type**. When prompted, select a project type and choose **Windows Application** from the Templates. A blank form will appear.

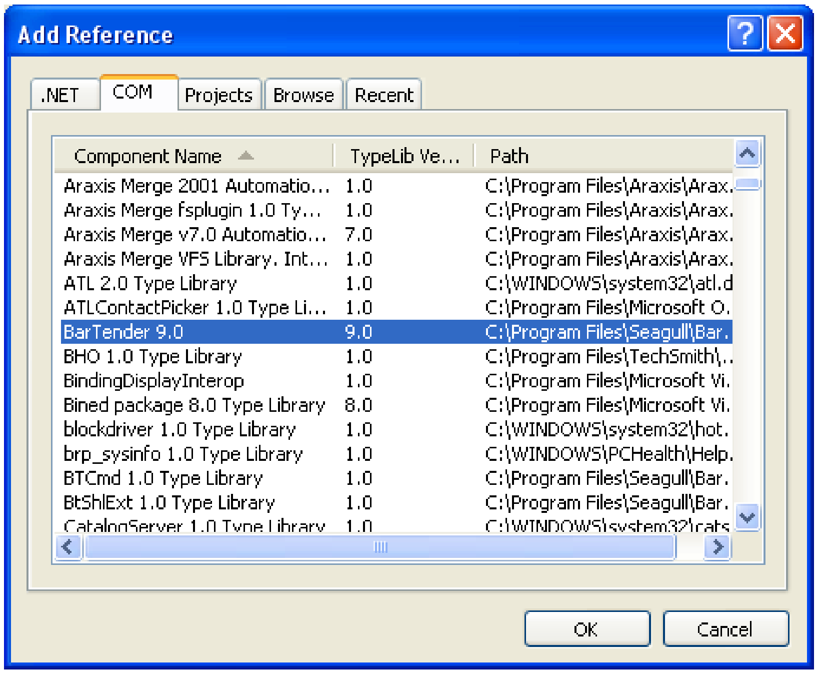
Referencing BarTender

1. Open the **Project** menu and select **Add References.**

2. Click on the **COM** tab.

3. Scroll down and click on **BarTender 9.0** so that it is highlighted.

4. Click the **Select** button and then click **OK**. (This tells C# where to look for BarTender and what objects, methods, and properties it supports.)

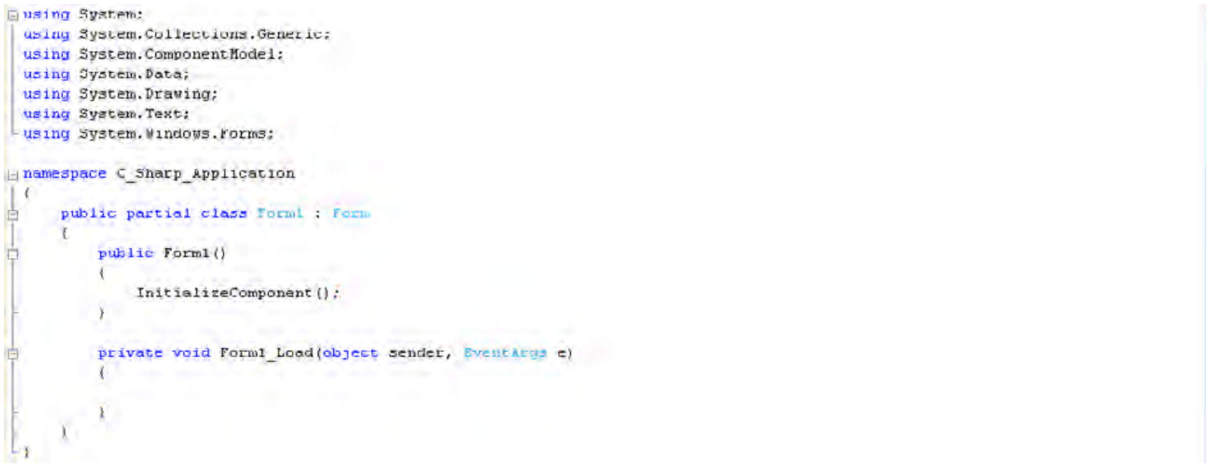


Component Name	TypeLib Ve...	Path
Araxis Merge 2001 Automatio...	1.0	C:\Program Files\Araxis\Arax.
Araxis Merge fsplugin 1.0 Ty...	1.0	C:\Program Files\Araxis\Arax.
Araxis Merge v7.0 Automatio...	7.0	C:\Program Files\Araxis\Arax.
Araxis Merge VFS Library. Int...	1.0	C:\Program Files\Araxis\Arax.
ATL 2.0 Type Library	1.0	C:\WINDOWS\system32\atl.d
ATLContactPicker 1.0 Type Li...	1.0	C:\Program Files\Microsoft O.
BarTender 9.0	9.0	C:\Program Files\Seagull\Bar.
BHO 1.0 Type Library	1.0	C:\Program Files\TechSmith\..
BindingDisplayInterop	1.0	C:\Program Files\Microsoft Vi.
Bined package 8.0 Type Library	8.0	C:\Program Files\Microsoft Vi.
blockdriver 1.0 Type Library	1.0	C:\WINDOWS\system32\hot.
brp_sysinfo 1.0 Type Library	1.0	C:\WINDOWS\PCHealth\Help.
BTCmd 1.0 Type Library	1.0	C:\Program Files\Seagull\Bar.
BtShlExt 1.0 Type Library	1.0	C:\Program Files\Seagull\Bar.
CatalogServer 1.0 Type Library	1.0	C:\WINDOWS\system32\cats

Defining a Variable for BarTender, the Label Format and any Messages Created During the Printing Process

Assign names to BarTender and the label format so that your program can reference them.

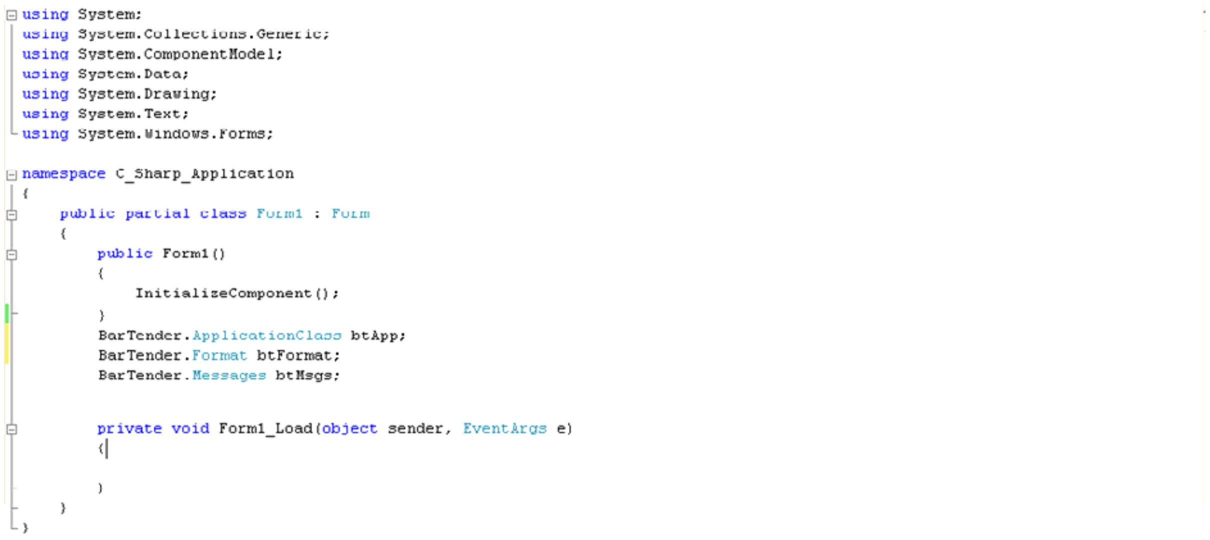
1. Double-click on the form and scroll down to the Form1 events.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace C_Sharp_Application
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

2. Type

```
BarTender.ApplicationClass btApp;

BarTender.Format btFormat;

BarTender.Messages btMsgs;
```



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace C_Sharp_Application
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        BarTender.ApplicationClass btApp;
        BarTender.Format btFormat;
        BarTender.Messages btMsgs;

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Create an Instance of BarTender

Typically, you will want to start BarTender when the application that is going to use it starts and close BarTender when the application closes. This will provide the fastest response time when BarTender is asked to do something by the application. To accomplish this, start BarTender in the **Form Load** method.

Inside the `Form1_Load()` method, type:

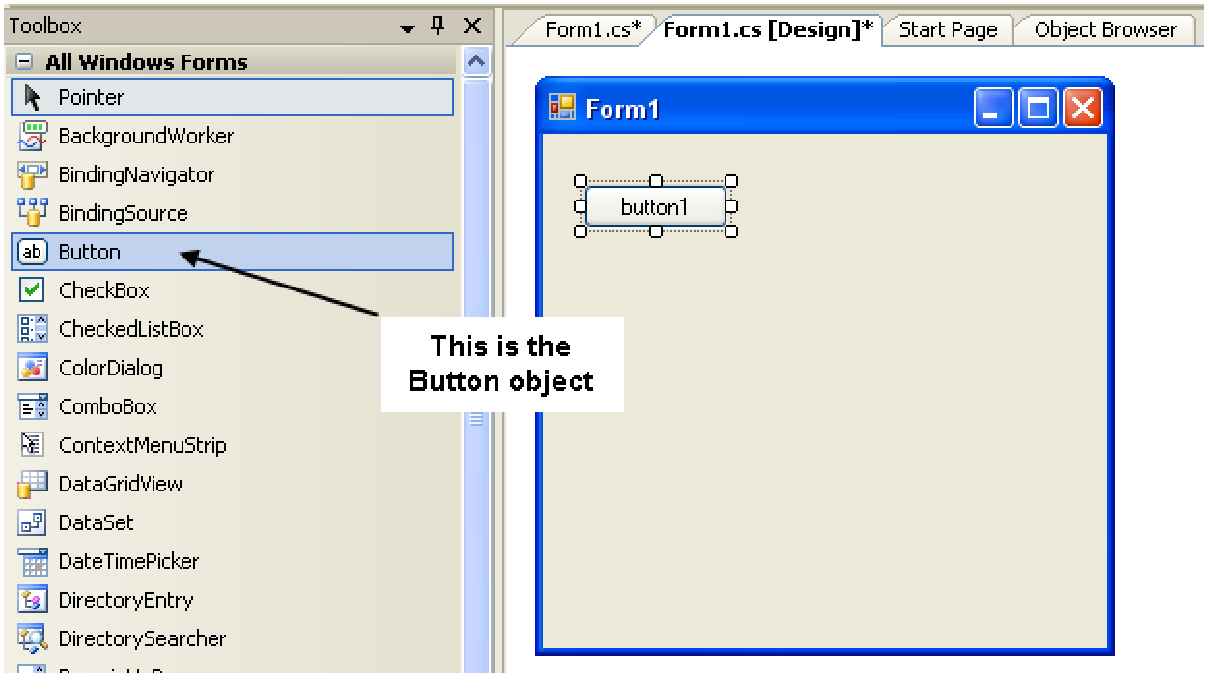
```
btApp = new BarTender.ApplicationClass();
```

This line of code will start an instance of BarTender and assign it the name **`btApp`**. Now when your application is started it will immediately start BarTender and every time you reference **`btApp`** you will be referencing that particular instance of BarTender. By default BarTender will run invisibly.

Load the Label Format

In the example application, the click of a button will tell BarTender to load a specified label format.

1. Open the **Toolbox** and drag a button object to the form. You should now see a button appear on the form that says **Button1** as shown here.



2. Double-click on the button. This will create a method that will run in response to a user clicking the button when your application is running.

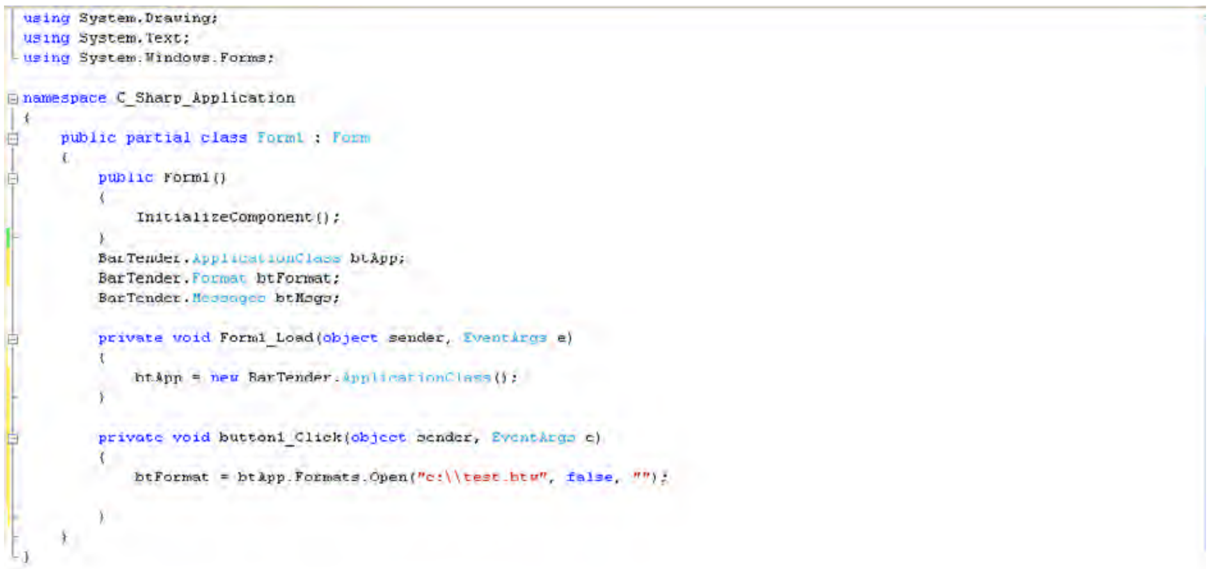
3. Inside the `button1_Click()` method, type

```
btFormat = btApp.Formats.Open("c:\\test.btw", false, "");
```

The whole method should now look like this:

```
private void button1_Click(object sender, EventArgs e)
{

btFormat = btApp.Formats.Open("c:\\test.btw", false, "");
}
```

This tells BarTender to open up the label format test.btw from your **C:** drive. It also assigns that format to the variable **`btFormat`**. This will also not close out any formats that are open and will use the default printer on the system.



```
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace C_Sharp_Application
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        BarTender.ApplicationClass btApp;
        BarTender.Format btFormat;
        BarTender.Messages btMsgs;

        private void Form1_Load(object sender, EventArgs e)
        {
            btApp = new BarTender.ApplicationClass();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            btFormat = btApp.Formats.Open("c:\\test.btw", false, "");

        }
    }
}
```

Printing Out Your Label Format with Desired Text

Now add code that assigns data to the named substring:

1. On the next line type

   ```
   btFormat.SetNamedSubStringValue("testing", "My string");
   ```

   This will send the words "My string" to the text object with the **`testing`** substring.

2. Then underneath that line of code type

   ```
   btFormat.Print("Job1",  true, -1, out btMsgs);
   ```

   This instructs BarTender to print a label named Job1, then to wait indefinitely for the job to complete, and return any messages created during the print job.

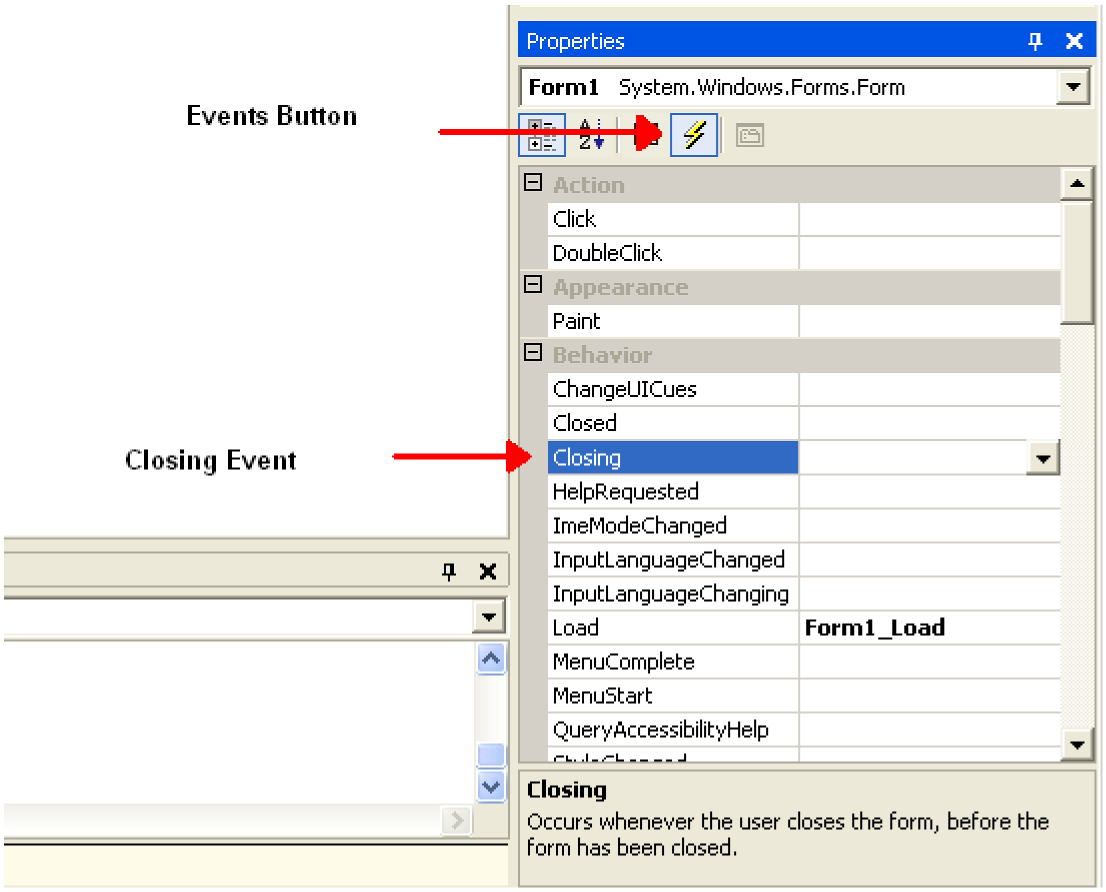
Closing the Label Format and BarTender

1. On the next line type

   ```
   btFormat.Close(BarTender.BTSaveOptions.btSaveChanges);
   ```

This instructs BarTender to save the label format before closing. (This line will not close BarTender.)

2. Click the **Form1.cs [Design]** tab.
3. In the **Properties** of the form click on the **Events** button
4. Double click on the **Closing** event
5. Select **(Form1 Events)** from the object drop down list.
6. This will create the first and last lines of the method that will run when your application closes.

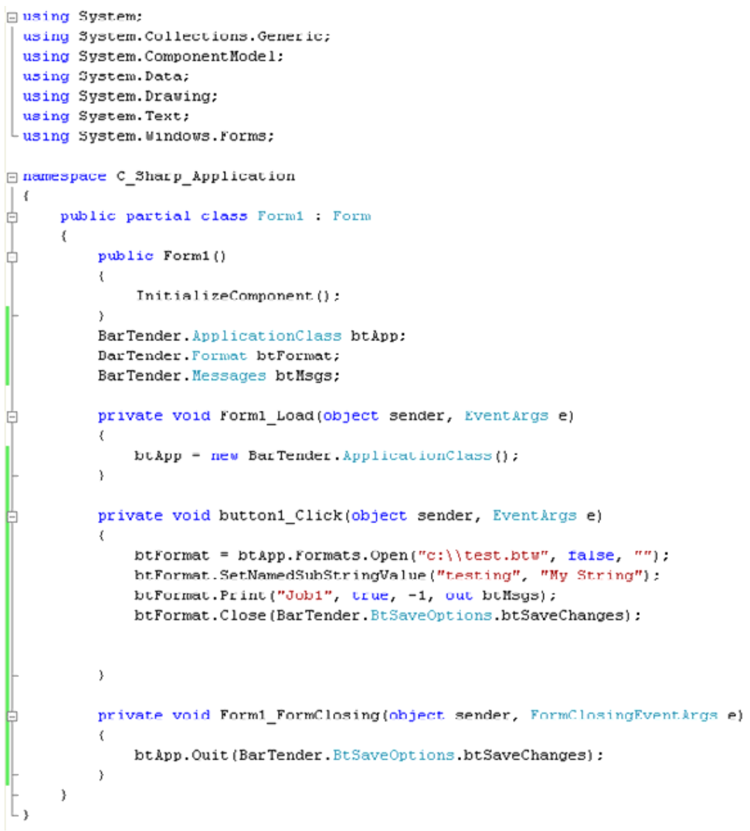


Closing
Occurs whenever the user closes the form, before the form has been closed.

7. Inside the "{" and "}" for the `Form1_Closing()` method, type

   `btApp.Quit(BarTender.BtSaveOptions.btNoNotSaveChanges);`

   This line will then tell BarTender to unload when the application closes, assuring that there won't be invisible instances of BarTender running when the application is closed.

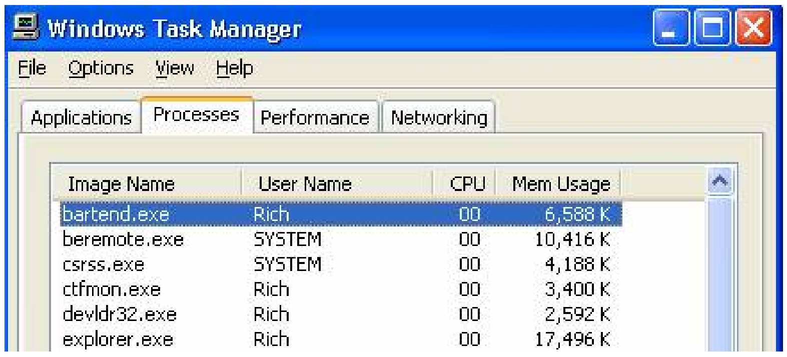


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace C_Sharp_Application
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        BarTender.ApplicationClass btApp;
        BarTender.Format btFormat;
        BarTender.Messages btMsgs;

        private void Form1_Load(object sender, EventArgs e)
        {
            btApp = new BarTender.ApplicationClass();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            btFormat = btApp.Formats.Open("c:\\test.btw", false, "");
            btFormat.SetNamedSubStringValue("testing", "My String");
            btFormat.Print("Job1", true, -1, out btMsgs);
            btFormat.Close(BarTender.BtSaveOptions.btSaveChanges);


        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            btApp.Quit(BarTender.BtSaveOptions.btSaveChanges);
        }
    }
}
```

Testing the Application

1. To test the application, press the F5 key to compile and run the application. The application with its button should appear, but you will not see any visible signs of BarTender.

2. To verify that BarTender was loaded, press the CTRL+ALT+DEL buttons to open the **Windows Task Manager**. You should see BarTender running under the **Processes** tab.



```
Image Name      User Name     CPU   Mem Usage
bartend.exe     Rich          00     6,588 K
beremote.exe    SYSTEM        00    10,416 K
csrss.exe       SYSTEM        00     4,188 K
ctfmon.exe      Rich          00     3,400 K
devldr32.exe    Rich          00     2,592 K
explorer.exe    Rich          00    17,496 K
```

3. Click the application's button and your label format should then print out with the text, "My string" on it.

4. Close the application. To verify that BarTender also closes use the **Windows Task Manager** as in step 2.

BarTender Automation in Non-Windows Environments

There are some situations in which ActiveX automation is not possible or not cost-effective. This includes situations in which

- the application that needs to control BarTender is on a non-Windows platform, such as UNIX or AS/400,
- it is easier for the other program to export data to a file than it is for it to issue ActiveX commands, or
- you do not have access to the other program's source code.

For these situations, Seagull Scientific, Inc. provides a powerful enterprise-integration utility called Commander that enables you to perform automatic label-printing using BarTender in response to triggering events, such as the appearance of a file in a Commander-monitored directory. See the Seagull white paper *Commander* for more information. Commander is included without charge in the Enterprise Edition of BarTender.

Available Seagull White Papers

General White Papers

- The Advantage of Drivers by Seagull
- Choosing the Right BarTender Edition
- Label System Security

Companion Applications

- Printer Maestro, Part 1: Enterprise Print Management
- BarTender Security Center
- BarTender Web Print Server

Recent Upgrades

- What's New in the Latest BarTender

Integration White Papers

- Integration Overview
- Commander
- Commander Examples
- BarTender's .NET SDKs
- BarTender's ActiveX Automation Interface
- Exporting Printer Code Templates
- Using BarTender with Terminal Services and Citrix MetaFrame
- XML Integration with Oracle's WMS and MSCA

Integration With SAP

- SAP Integration Methods
- Reading SAP IDocs
- SAP Auto Infrastructure Integration with BarTender

Miscellaneous White Papers

- BarTender Enterprise Licensing
- Printing International Characters Using BarTender
- BarTender Software Activation
- Using BarTender's Application Identifier Wizard
- Optimizing Label Printing Performance
- Status Monitor Overview
- Silent Install

For downloadable versions, visit:

www.seagullscientific.com/aspx/whitepapers.aspx

SEAGULL SCIENTIFIC
www.seagullscientific.com