# *Getting Started with ActiveX Automation*

## *Understanding ActiveX Automation*

## *and Steps for Adding BarTender to a Visual Basic Project*

*Seagull Scientific, Inc*
*1616 148th Ave. S.E.*
*Bellevue, WA 98007-6848*
*USA*

*TEL: 425-641-1408*
*FAX: 425-641-1599*
*Sales@SeagullScientific.com*
*www.SeagullScientific.com*

***Europe***
*C/ Velázquez, 15-6º-Izq.*
*28001 Madrid*
*Spain*

*TEL: +34 91 435 2525*
*FAX: +34 91 578 0506*
*eurosales@SeagullScientific.com*

In North America:
Seagull Scientific, Inc
1616 148th Ave. S.E.
Bellevue, WA 98007-6848
USA

In Europe:
C/ Velázquez, 15-6º -Izq.
28001 Madrid
Spain

October, 2003

Version: 2003-10-28 11:01

# 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, these types of software applications have no built-in label-printing functions at all, or only very simple functions that do not allow for the retrieval of variable data from multiple data sources or for the occasional print-time prompting of end-users.

BarTender's ActiveX Automation solves this problem by enabling Windows applications to perform automatically all of the data-retrieval, user-prompting, and label printing tasks of BarTender itself.

**Note**: Integration of BarTender can be extended to non-Windows applications. 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++, 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).

## 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.

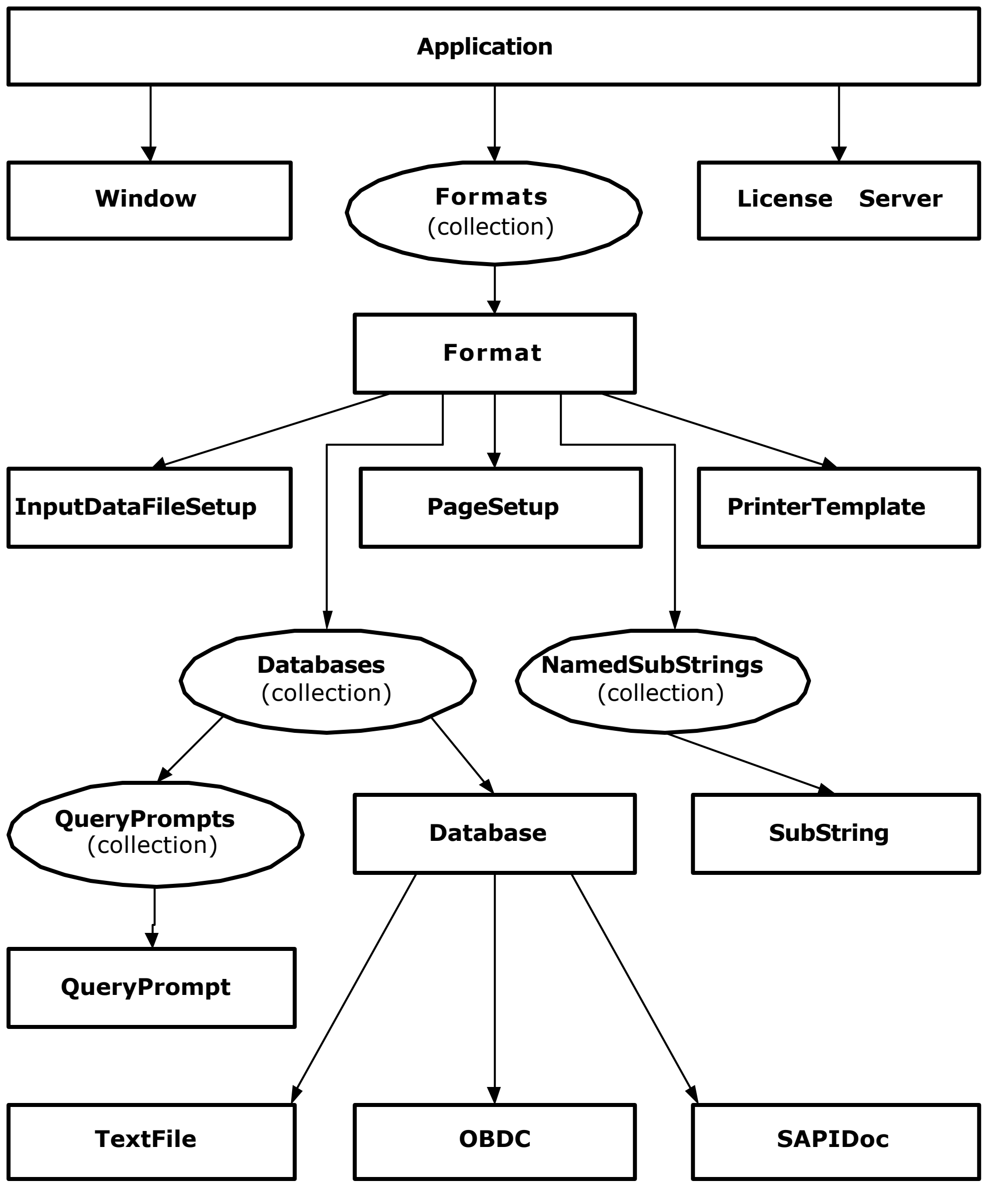
# BarTender’s ActiveX API

## 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, 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 schema provides an overview of the hierarchy of objects.



**Automation Object Model**

# Example of Automating BarTender with Visual Basic

This section explains how to create a simple Visual Basic application that uses BarTender's ActiveX Automation interface. These instructions assume basic familiarity with the Visual Basic 6.0 IDE.

## Preliminaries

1. If you are not familiar with sub-strings and share/names in BarTender text objects, consult BarTender's online help about these topics.
2. Create a label format with a text object containg two sub-strings.
3. Assign "testing" as a share/name to one of the sub-strings.
4. Verify that you can print the format.
5. Save this format as test.btw to the root of your C drive.

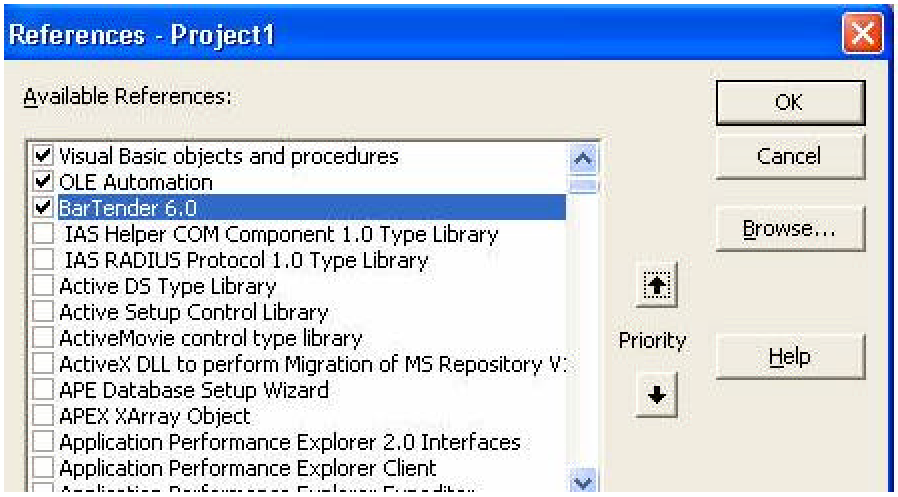
## Creating an Application

### Specifying a Project Type

Open Visual Basic 6.0. If prompted to select a project type, choose **Standard EXE**. A blank form will appear.

### Referencing BarTender

1. Open the **Project** menu and select **References**.
2. In the **References** dialog box, check the **BarTender 6.0** check box and click **OK**. (This tells Visual Basic where to look for BarTender and what objects, methods, and properties it supports.)



References - Project1

Available References:

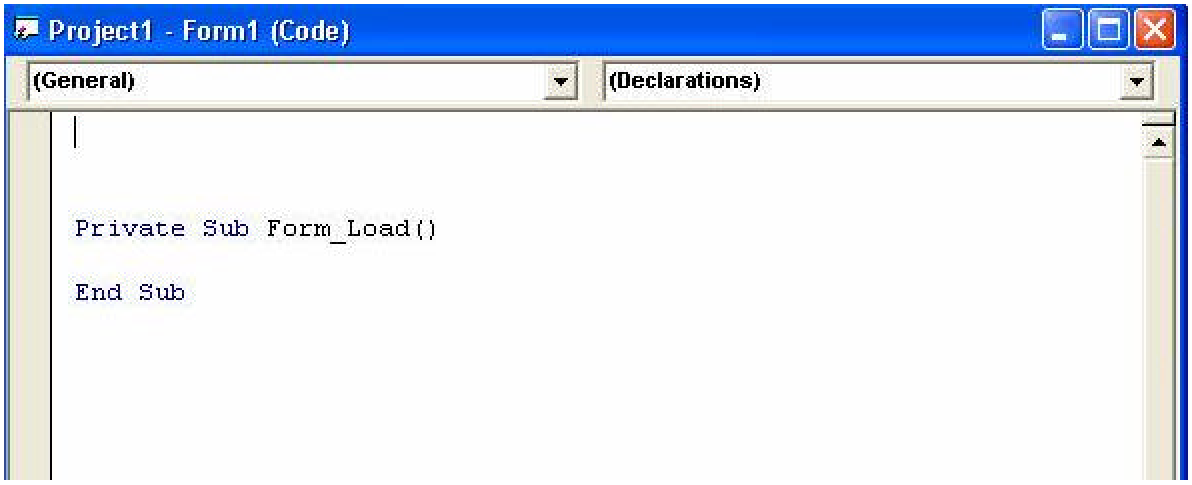
- [✓] Visual Basic objects and procedures
- [✓] OLE Automation
- [✓] BarTender 6.0
- [ ] IAS Helper COM Component 1.0 Type Library
- [ ] IAS RADIUS Protocol 1.0 Type Library
- [ ] Active DS Type Library
- [ ] Active Setup Control Library
- [ ] ActiveMovie control type library
- [ ] ActiveX DLL to perform Migration of MS Repository V:
- [ ] APE Database Setup Wizard
- [ ] APEX XArray Object
- [ ] Application Performance Explorer 2.0 Interfaces
- [ ] Application Performance Explorer Client

## Defining a Variable for BarTender and the Label Format

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

1. Double-click the form and click in the upper left corner.

2. Press **Return** a few times until you are in the **General** / **Declarations** section.



Project1 - Form1 (Code)

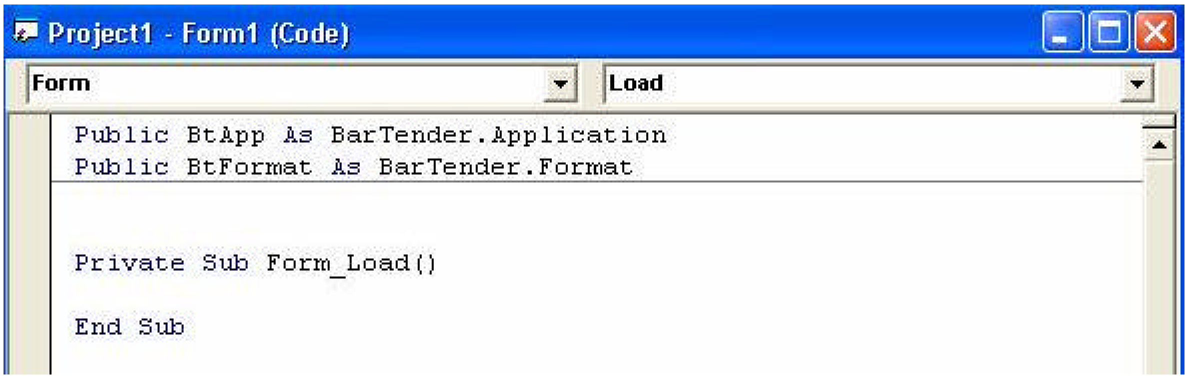
(General) (Declarations)

```
Private Sub Form_Load()

End Sub
```

3. Type:

   **`Public BtApp as BarTender.Application`**

```
Public BtFormat as BarTender.Format
```



```
Public BtApp As BarTender.Application
Public BtFormat As BarTender.Format


Private Sub Form_Load()

End Sub
```

## 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** sub-routine.

On a line between **`Private Sub Form_Load()`** and **`End Sub`** lines, type:

```
Set BtApp = CreateObject("BarTender.Application")
```

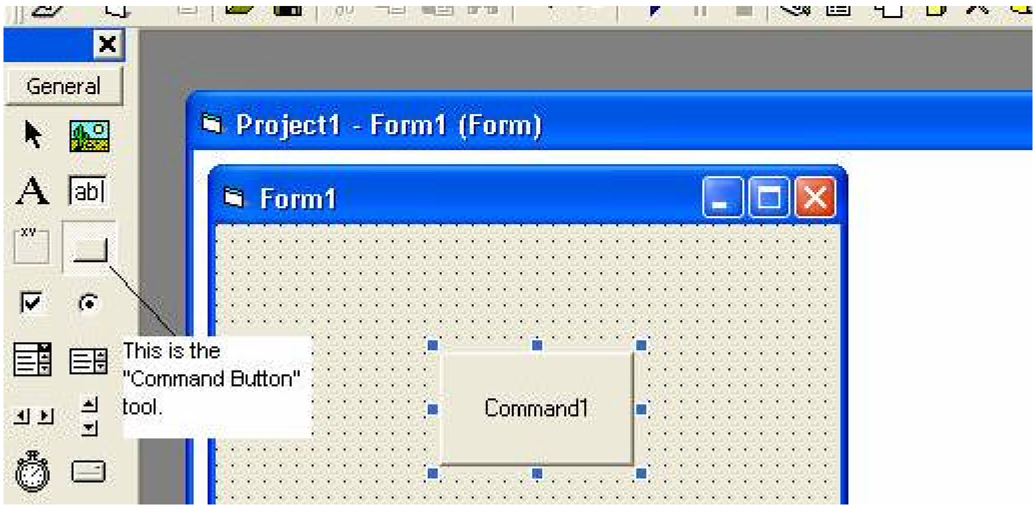
This line of code will start an instance of BarTender and assign it the name **`BtApp`**. Now when your application is started it will immediately start BarTender and every time you reference **`BtApp`** you'll be referencing that particular instance of BarTender. By default BarTender will running invisibly.

## Load the Label Format

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

1. Click on the **Command Button** tool from the controls tool bar.

2. Place the cursor in the dialog form that opens and, while holding down the left mouse button, drag the mouse to the point where you want the opposite corner of the button. You should now see a button appear on the form that says **Command1** as shown here.

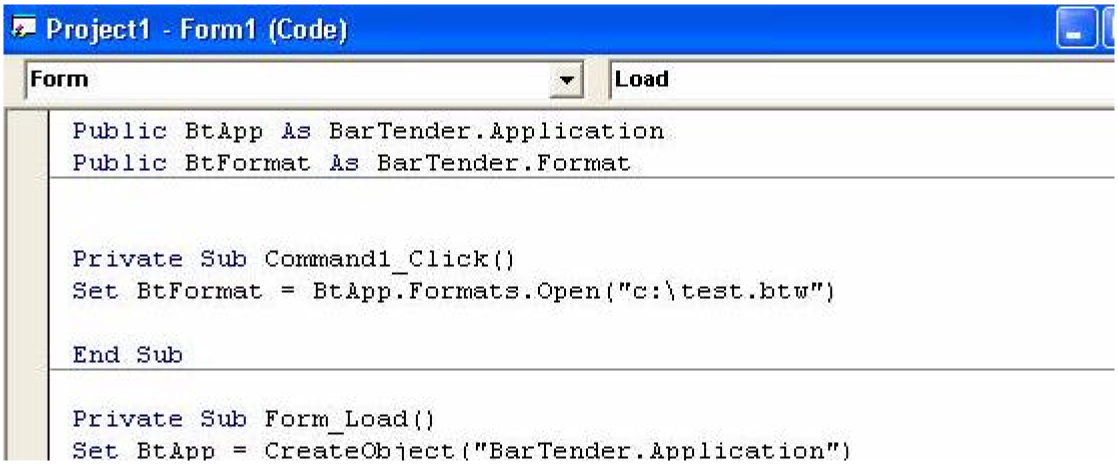


3. Double-click on the button. This will create the first and last lines of the sub-routine that will run in response to an end-user clicking the button when your application is running.

4. On a line between **`Private Sub Command1_Click ()`** and **`End Sub`** lines, type:

   **`Set BtFormat = BtApp.Formats.Open("c:\test.btw")`**

   This tells BarTender to open up the label format test.btw from your c drive, and it assigns that format to the variable **`BtFormat`**.



```
Public BtApp As BarTender.Application
Public BtFormat As BarTender.Format


Private Sub Command1_Click()
Set BtFormat = BtApp.Formats.Open("c:\test.btw")

End Sub

Private Sub Form_Load()
Set BtApp = CreateObject("BarTender.Application")
```

## Printing Out Your Label Format with Desired Text

Now add code that assigns data to the named sub-string:

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** sub-string.

2. Then underneath that line of code enter the following code.

   ```
   BtFormat.PrintOut
   ```

   This instructs BarTender to print label.

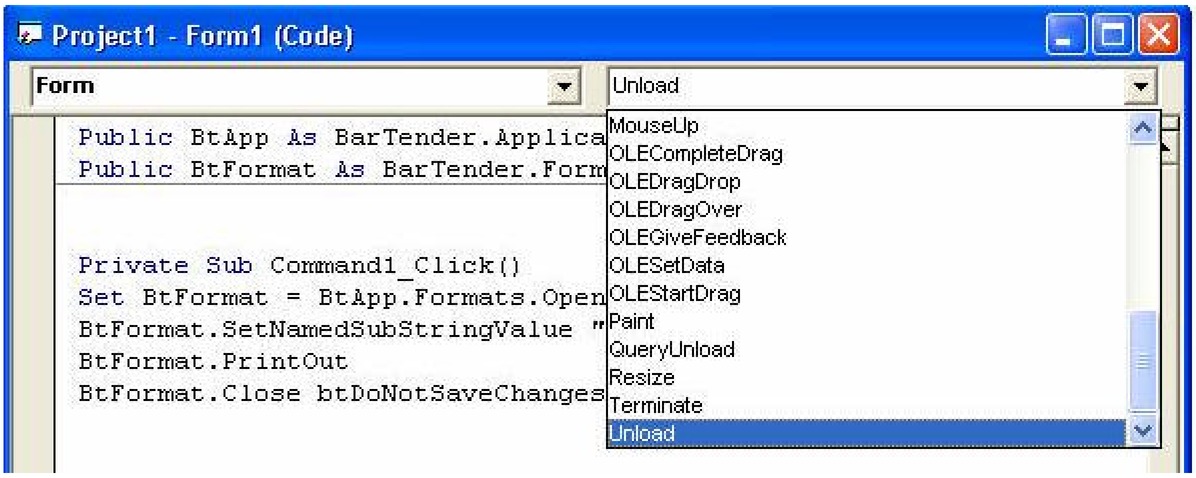
## Closing the Label Format and BarTender

1. On the next line type:

   ```
   BtFormat.Close btDoNotSaveChanges
   ```

   This instructs BarTender to close the *label format* without saving changes. (This line will not close BarTender.)

2. Select **Form** from the object drop down list.
3. Select **Unload** from the events drop down list. This will create the first and last lines of the unload sub-routine.

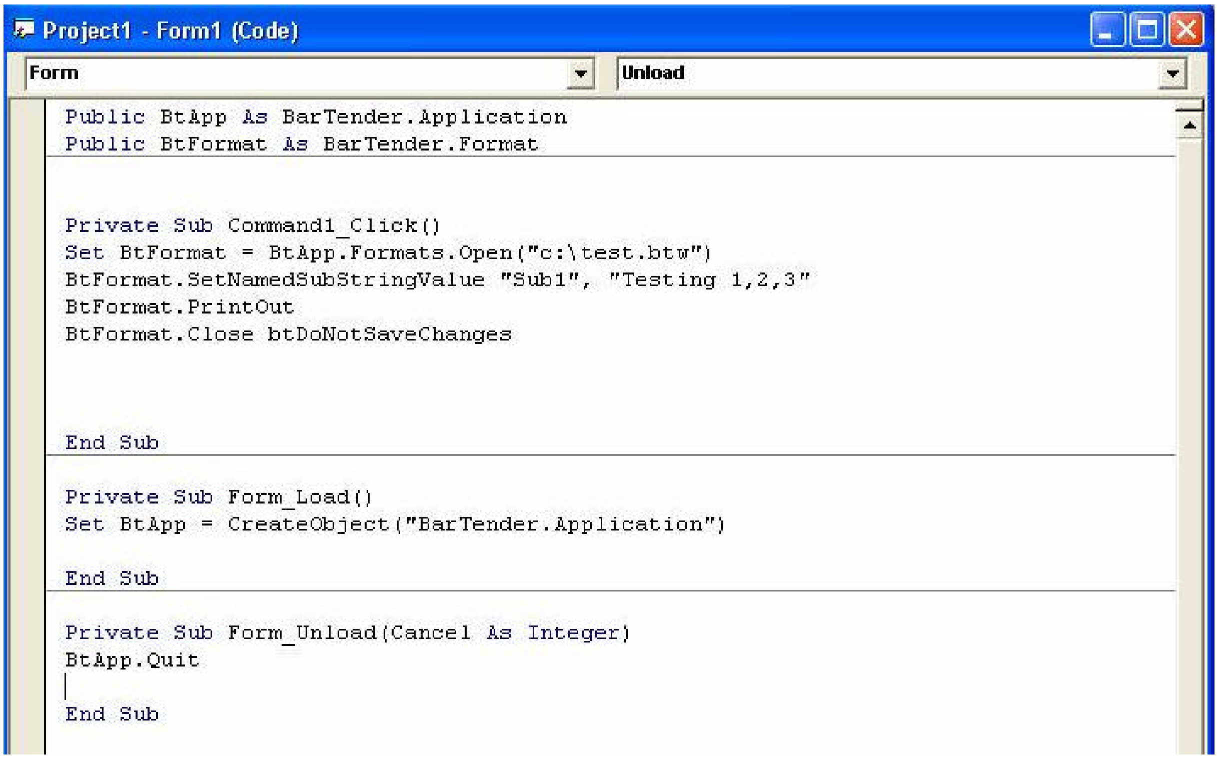


```
Public BtApp As BarTender.Applica
Public BtFormat As BarTender.Form


Private Sub Command1_Click()
Set BtFormat = BtApp.Formats.Open
BtFormat.SetNamedSubStringValue "
BtFormat.PrintOut
BtFormat.Close btDoNotSaveChanges
```

4. On a line between Private Sub Form_Unload(Cancel As Integer)and End Sub, type:

   **`BtApp.Quit`**

   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.

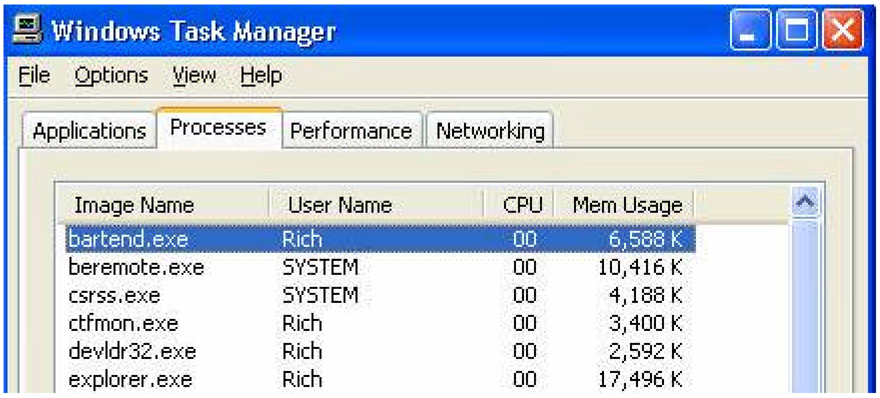


```
Public BtApp As BarTender.Application
Public BtFormat As BarTender.Format


Private Sub Command1_Click()
Set BtFormat = BtApp.Formats.Open("c:\test.btw")
BtFormat.SetNamedSubStringValue "Sub1", "Testing 1,2,3"
BtFormat.PrintOut
BtFormat.Close btDoNotSaveChanges



End Sub

Private Sub Form_Load()
Set BtApp = CreateObject("BarTender.Application")

End Sub

Private Sub Form_Unload(Cancel As Integer)
BtApp.Quit

End Sub
```

## Testing the Application

1. To test the application, press the F5 key. 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 usr the 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 don't 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 Enterprise Integration Utility*** for more information. Commander is included without charge in the Enterprise Edition of BarTender.