WEB-BASED APPROACH FOR CLIENT SERIAL-PORT COMMUNICATION USING ASP.NET

Abdullah M. Al Fararjeh, Edward Jaser
Information Technology Centre, Royal Scientific Society, Al Jubaiha 11941, Amman, Jordan
fararjeh@rss.gov.jo, ejaser@rss.gov.jo

ABSTRACT

It is popular to develop a stand-alone application to access serial-port devices through ready programming interfaces in popular programming languages because the application and the device exist in one box. However, the web-based application architecture is different where it is required, for some applications, to integrate a functionality of client serial port communication (e.g. accessing customer display devices connected to client computers). We introduce in this paper, to the best of our knowledge, a new approach to perform serial-port communication on client devices from a dynamic web application implemented using ASP.NET technology. The proposed approach constructs specialized component to be installed into both the web server and the client. The client component is responsible for dealing with security restrictions safely to access client's resources through a web application. As a case study, we discuss customer display device plugged into the client. Those devices are controlled through a web application to print data-driven values from database.

Keywords: *Serial Port Communication, Customer Display Device, Web Application, ASP.NET, Windows Control Library, Microsoft .Net Framework*

[1] INTRODUCTION

Web applications have become widely used in e-business solutions. The web-based application is installed in the server but the stand-alone application must be installed in all clients. This main characteristic implies that requirement changes will be only maintained in the server if the application is developed based on a web technology.

Many business components are easily developed in the stand-alone applications because of the nature of application where the application is executed in the client itself and it can control the client resources. The web application, however, is requested from the client and it is executed in the server-side and this application is restricted in accessing the client resources due to security.

The approach studies the feasibility of implementing a web-based application to control serial-port devices plugged in the clients. Specifically, the purpose of our approach is to show how to use a customer display device through a web application where the device is plugged in one of the serial ports on the computer client. The approach comes with a new technique of implementing .NET Windows Control inside ASP.NET web application. The component may be implemented in different techniques such as Java Applet and ActiveX control. Java Applet is developed using a technology outside Microsoft .Net technology and ActiveX is not usually safe and the browser automatically blocks it unless the end-user enables its execution [8]. Our approach proposes a more safe methodology when executed in the client-side. Additionally, the component and its web application are developed through Microsoft .Net technology.

[2] WHAT IS SERIAL PORT COMMUNICATION?

Serial port is a physical connector to allow data transmission asynchronously in or out one bit at a time [2] and [3]. The personal computers usually contain two serial ports to communicate with many peripherals such as serial mice, dial-up modems, bar code scanners, and customer display units. Operating systems usually use a symbolic name to refer to the serial ports of a computer. For example, the Microsoft MS-DOS and Windows environments refer to serial ports as COM ports: COM1, COM2…etc.

The serial port connectors are usually referred as DB-9 or DB-25. D represents the shape of the connector if placed vertically [2]. The number 9/25 indicates the number of pins found on the connector. DB9 Serial connections are now commonly found on modern PCs where DB25 is commonly found on older computers. Figure 1 shows the two types of serial ports.

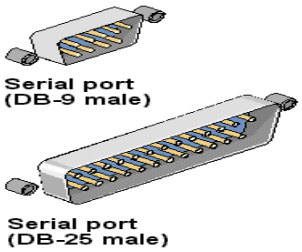

Figure 1: Serial Port Types

[3] WHAT IS CUSTOMER DISPLAY DEVICE?

The customer display device is a terminal to present the purchase price and the amount of change to customers. This device is a part of hardware equipments of Point of Sale System (POS) to show the price of purchase transaction at checkout counter in a retail shop. The device is commonly connected to personal computers through a local serial port.

[4] WEB-BASED APPROACH FOR SERIAL-PORT COMMUNICATION

The functionality is easily developed in a stand-alone application. Our target is transforming this stand-alone application functionality into a web interactive one. The approach proposes hosting windows control library into a web form. The development output of this control a Dynamic Link Library (DLL) file that will be hosted in the root of the web application and the web form calls the DLL control through the object tag [1], [7] and [8]. Of course we need a dynamic interaction between the windows control and the web form, and then the object tag can pass values from a web form into the windows control DLL through the parameter tag contained inside object tag as illustrated in figure 2.

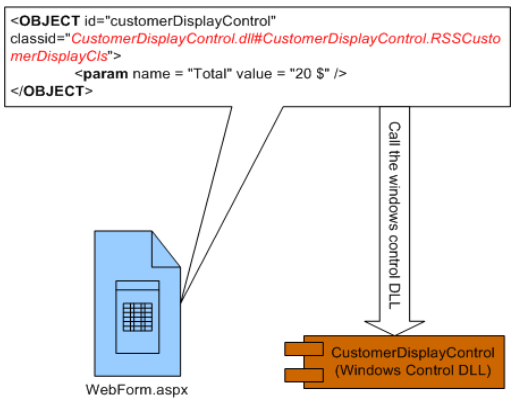

Figure 2: The interaction between a web form and Customer Display control

The proposed approach needs the .Net Framework installed in the client because the execution of windows control code depends on the existence of this framework in the same box [7]. In fact, the mechanism of running of a windows control library is similar to a Java Applet hosted in a web application. The execution of any Java Applet requires a web browser and Java Virtual Machine (JVM) installed in the client [8]. But, the execution of any windows control hosted in the web form requires a web browser and .Net Framework. We assume the web application is developed using ASP.NET 1.1, the right choice is to eliminate Java Applet solution to offer a compatible solution based on Microsoft technology. The .Net Framework is currently packed with Windows Server 2003 and the other windows environments get the framework through windows automatic update tool. Thus the .Net Framework is indirectly embedded in the client computer and the proposed solution is feasible to be applied.

Windows Forms controls are better than ActiveX controls from a security standpoint. Every Web-based ActiveX control comes with a security warning that users must accept or deny no matter what the control does [8]. Then, this approach will adopt on hosting windows form control to perform serial port communication on client because it is the quickest and safest solution based on the .Net Framework.

The .Net Framework 1.1 does not provide an API to control devices plugged on the personal computer through a serial port (COM port) [4]. It is possible to do serial port communication through MSComm control (MSCOMM32.OCX) when come with Visual Basic 6.0 [4] and [13]. MSComm control is an ActiveX control that is will be hosted inside windows operating system (windows\system32) [5] and the web application will control serial-port devices through a licensed and registered control hosted in the client and the windows control will be the safest intermediate layer between the web application and MSComm control. Because the nature of MSCOMM32.OCX is not compatible with the .Net Framework, the .Net application could not call MSCOMM32.OCX directly. So our approach will convert this OCX into DLL to be referenced inside the new developed windows control. The MSComm control will be converted into Interop.MSCommLib.dll file through Microsoft .NET Framework Type Library to Assembly Converter Tool (tlbimp.exe). The interrelation between component parts to manage serial port

communication inside a windows form control hosted in the web application is shown in figure 3:

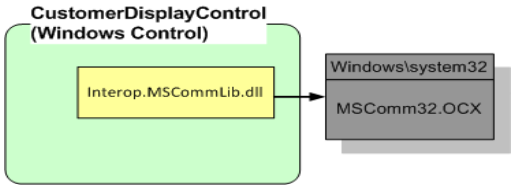

Figure 3: Customer Display Control Parts

When the client loads the web form, the client browser will load the hosted windows control automatically. Then, if the control commands are safe the client will appear and executed silently. Otherwise, the control will disappear and the control execution will be prohibited. In our case, the component will communicate with one of the client resources and this is a potential dangerous. To solve this issue, the approach will install permanently the component output (DLL files) in the client machine. After that, our solution will assign enough permission to run our windows control through a web application [10]. The approach implements an automatic setup to install the component output files into the client (CustomerDisplayControl.dll, Interop.MSCommLib.dll and MSCOMM32.OCX files). The setup wizard will install MSCOMM32.OCX into windows\system32 path and register it. The other assembly files will be installed in a location to be administered and executed by the client's .Net Framework. The approach proposes putting those files inside the Global Assembly Cache (GAC) which is a shared area of memory for storing all the assemblies of all .NET applications running on a given computer [6] and [9]. Thus residing component assemblies in the GAC enables all applications using them. GACUTIL.EXE is a .Net tool used to add an assembly into GAC. To add an assembly into GAC, such assemblies must be strong named and this implies that the approach will sign the two assemblies (CustomerDisplayControl.dll, Interop.MSCommLib.dll). The approach signs the two assemblies (CustomerDisplayControl.dll, Interop.MSCommLib.dll) using the .Net framework signing tool, SN.EXE [9].

In the case of Interop.MSCommLib.dll assembly, the assembly is already compiled and we don't have the source code and, then, the approach will convert this executable file into Microsoft Intermediate Language (MSIL), which is compatible with .Net Framework, using ILASM.EXE and then the signed DLL file will be obtained again. Figure 4 shows the steps to sign Interop.MSCommLib.dll assembly.

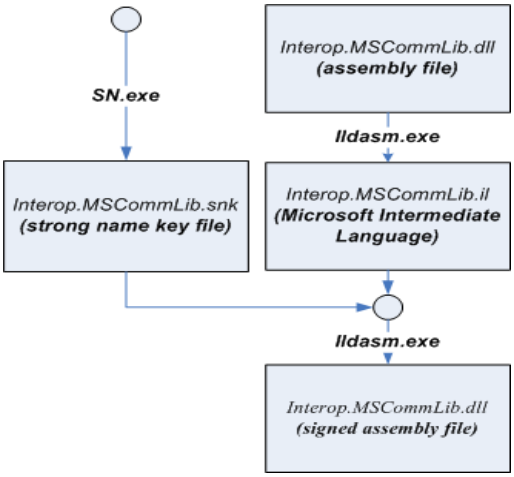

Figure 4: Signing Mechanism of MS Communication Assembly

In the case of CustomerDisplayControl.dll, the source code will be compiled with strong name key file to generate a signed key file. Figure 5 shows the mechanism to sign Customer Display Control assembly.

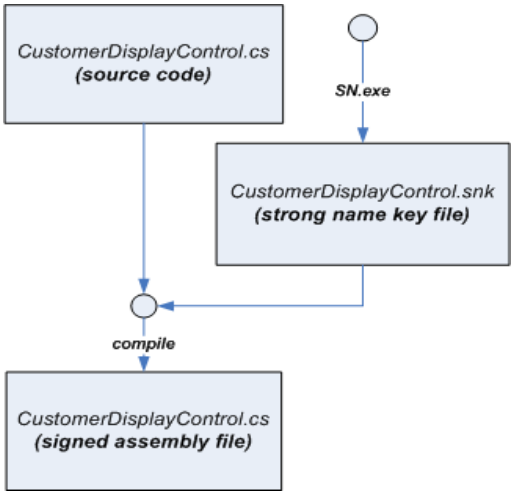

Figure 5: Signing Mechanism of Customer Display Control Assembly

The component's signed-assemblies are ready to be added in dot net framework GAC in the client. But the component setup includes a configuration utility to grant full trust permission for the component assembly. The permission policy is just assigned specifically for our component DLLs and this guarantees the highest security level on client resources [11] and [12].

The control will work jointly with Interop.MSCommLib.dll to communicate with

serial-port devices. The component implements a procedure to find dynamically all COM ports available in the client. In addition, the component implements another procedure to transmit data through serial port and the component receives a sequence of characters as a parameter and then the string are converted into serial bytes to be transmitted through a serial port. If the serial port is connected with a customer display unit, the device will present a message visually for the customer.

As explained, the approach supposes installing the component in both client and server sides. The component is easily integrated with any ASP.NET web application and it can receive data-driven values saved into databases. Figure 6 shows our approach platform.

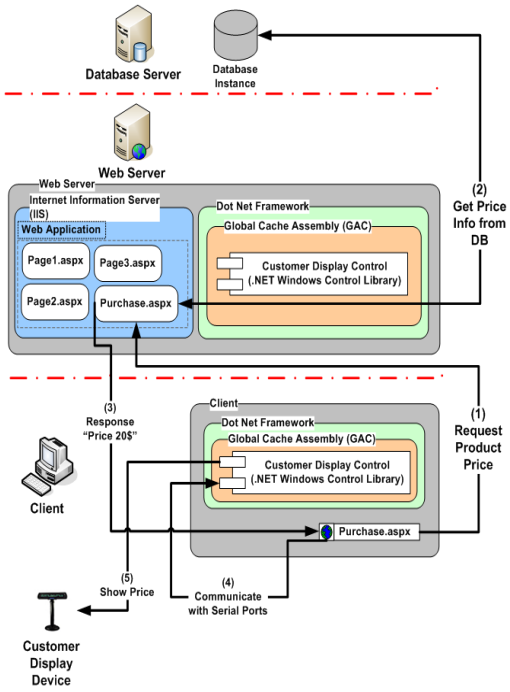

Figure 6: The approach platform

[5] CONCLUSION

It is possible to implement functionality inside our web applications to access serial-port devices in the client. We can implement a Microsoft .Net Control to perform client serial-port communication through a web application developed using ASP.NET.

The approach is feasible to be executed in the client because it depends on .NET Framework which is commonly downloaded through automatic windows update. Additionally, the approach guarantees safe solution to enable a bridge communication between server and client. Then, the approach installs a component in the client to pass security policies and perform serial-port communication on client devices.

REFERENCES

[1] Mridula, P. and Essam, A., *ASP.NET Bible*, Delhi Press, New Delhi, 2002.

[2] Wikipedia Website, *Serial Port*, http://en.wikipedia.org/wiki/Serial_port, date visited: May 14, 2008

[3] Computer Hope.com Website, *Computer Hardware Information about the serial port / com port*, http://www.computerhope.com/help/serial.htm, date visited: May 14, 2008

[4] Noah C., *Serial COM Simply in C#*, http://www.devhood.com/tutorials/tutorial_details.aspx?tutorial_id=320, date visited: May 14, 2008

[5] Ranjan D., *Serial Communication in C#*, http://www.codeproject.com/KB/WCF/SerialComm.aspx, date visited: May 14, 2008

[6] Jeremiah T., *Demystifying the .NET Global Assembly Cache*, http://www.codeproject.com/KB/dotnet/demystifygac.aspx, date visited: May 14, 2008

[7] Tim S., *Hosting a Windows Control in a Web Form*, http://aspnet.4guysfromrolla.com/articles/052604-1.aspx, date visited: May 14, 2008

[8] Thiru T., *Hosting .NET Windows Forms Controls in IE*, http://www.15seconds.com/issue/030610.htm, date visited: May 14, 2008

[9] Michele L., *10 Steps to a Successful Versioning and Deployment Strategy for .NET*, http://www.15seconds.com/issue/041103.htm , date visited: May 14, 2008

[10] Chirs S., *Increasing Permissions for Web-Deployed Windows Forms Applications*, http://msdn.microsoft.com/en-us/library/ms951290.aspx, date visited: May 14, 2008

[11] Meier J., Alex M. and others, *Chapter 8 – Code Access Security in Practice*, http://msdn.microsoft.com/en-us/library/aa302424.aspx, date visited: May 14, 2008

[12] Meier J., Alex M. and others, *Chapter 9 – Using Code Access Security with ASP.NET*, http://msdn.microsoft.com/en-us/library/aa302425.aspx, date visited: May 14, 2008

[13] Yes Telecom Website, *MSCOMM32.OCX MSComm Control*, http://www.yes-tele.com/mscomm.html, date visited: May 14, 2008