

Internet Payment Gateway

.NET API Developer Guide v3.0



Table of Contents

Overview	3
Disclaimer.....	3
Confidentiality.....	3
Document Purpose / Intended Audience	3
Related Documents.....	3
Introduction	4
1.1 Minimum System Requirements	5
Installation Overview.....	5
2 Installation.....	6
2.1 Webpay Specific Information and Files.....	6
2.2 Webpay.NET Client API Reference	6
2.3 Error/Exception Handling Overview	7
3 The Webpay Test Application	8
3.1 Running the Test Application	8
3.2 Supplying Input to the Test Application.....	9
3.3 Analysis of the Response.....	10
4 Configuring Your Development Environment	10
5 Sample Applications	12
5.1 Webpay.NET ASP Sample Web Application	12
5.2 Testing with Visual Studio's Development Server	14
5.3 Deploying to IIS	15
5.4 VB Sample Application.....	25
6 Troubleshooting	26
6.1 Network Errors	26
6.2 Unexpected Errors While Testing	26
6.3 Common errors and solutions	26
6.4 Switching on debugging	27
7 Additional Information	28
7.1 Security	28
7.2 Performance.....	28
7.3 Connectivity.....	28
7.4 Test/Live Merchant Status	28
7.5 Session and Concurrency Issues.....	28
8 Technical Support	29
8.1 IPG Web Site	29
8.2 Contacting St. George Bank	29
9 Appendix	30
9.1 Appendix A – Files included in this Developer Kit.....	30
9.2 Appendix B – Glossary.....	30
9.3 Appendix C – Advanced Configuration Options.....	30
9.4 Appendix D - Security Considerations for older systems.....	32
Windows 2000 Server.....	32
Windows Server 2003.....	32
9.5 Appendix E – Windows 2003 Certificate Store Permissions	34

Overview

Disclaimer

This document, including attachments, is not for general circulation, but for distribution to a limited number of specific corporations and individuals that are known to St. George Bank ("Company"). It is in draft form and its contents are subject to change, including changes of substance. Any person choosing to act on information contained in this document is advised to verify the information.

Confidentiality

By accepting this document, the recipient agrees to keep the information contained in it permanently confidential. This document is intended for use by the recipient only and may not be copied, reproduced or distributed to others at any time without the prior written consent of the Company. It cannot be used for any purpose except that for which it is given to you i.e. as user documentation.

If you do not agree with any of the above conditions, you must return this document immediately.

Document Purpose / Intended Audience

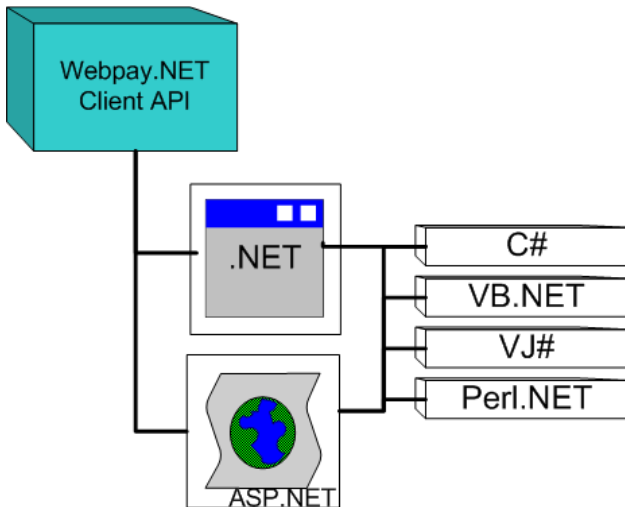
This guide is one component of the St. George Bank Transaction Server document set. It intended to act a reference point for developers seeking to implement the St. George Bank transaction functionality into merchant web applications. It provides specific technical information about the typical deployment of the St. George Bank .NET API and delivers valuable insights into more user-specific deployments. It is assumed that the parties implementing this API will be experienced in implementing web applications in a .Net development environment, and this document is written principally toward parties with this expertise

Related Documents

- **St. George Bank Generic API Developer Guide**
 - **St. George Bank Win32 Developer Guide**
 - **St. George Bank Java Developer Guide**
 - **St. George Bank Perl Developer Guide**
 - **St. George Bank Linux Developer Guide**
 - **St. George Bank PHP Developer Guide**

Introduction

The Webpay.NET API is a Windows-based dynamic link library containing methods for use with the *Webpay Transaction Server (WTS)*. The Webpay.NET Client API enables web site and application developers to efficiently add credit card transaction processing capabilities to their products.



The Webpay.NET Client API can be used in Windows-based applications using any language supported by the .NET Framework 3.5 (SP1). This document should be used in conjunction with the St. George Generic API Developer guide document, which includes details on parameters required for several transaction types including:

- **PURCHASE**
- **REFUND**
- **PREAUTH**
- **COMPLETION**
- **STATUS**

This document also includes common configuration settings, test applications and protocols, and

sample code that will assist developers in incorporating the API into their chosen application.

As additional interfaces and transaction types become available, St.George Bank will issue merchants and developers with the relevant information regarding the new parameters and other settings needed to adopt these transaction types.

This document should be used in conjunction with the appropriate *Transaction Specification* document to suit the transaction types required for your implementation.

This document also includes common configuration settings, test applications and protocols, and sample code that will assist developers in incorporating the chosen API into their chosen application.

Communication between the Webpay API and Webpay Transaction Server Gateway uses SSL connectivity through server addresses and defined ports.

NOTE: If your application is located behind a firewall or proxy server, you should ensure that the relevant addresses and ports are not being filtered or blocked.

It is important to note that the Webpay Transaction Server utilises two distinct transaction processing environments. These environments are defined as **LIVE** and **TEST**. The **LIVE** environment connects directly to the live banking processing system and performs real-time transactions. The **TEST** environment is a simulated environment controlled within the WTS that effectively mimics the live processing environment; **TEST** is principally used for integration testing and training.

Using the parameters provided by your St.George Bank representative you must define and test your application and API integration against the **TEST** environment to ensure compliance.

Minimum System Requirements

The following minimum system configuration is required for implementation of the Webpay.Net API.

Operating System	For desktop (standalone) applications: Windows XP/Vista/7, Windows 2000, Windows Server 2003, Windows Server 2008 For website or client/server applications: Windows 2000 Server, Windows 2003 Server, Windows 2008 Server
.Net Framework	3.5 SP1
Processor	Pentium 4
Processor Speed	3 Ghz
Memory	1 Gb
Compiler (for sample applications)	Visual Studio .NET 2008

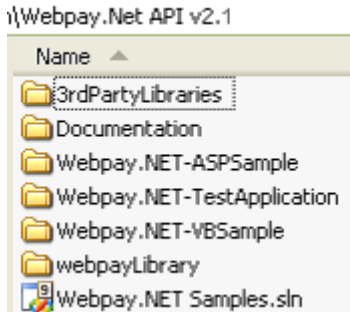
Installation Overview

The Webpay.Net API ships as a zipped archive. It contains the key Webpay Library (webpay.dll) and sample code illustrating how to use the Webpay.Net API in a C# Web application and a VB stand alone application. It is not necessary to install the Webpay.NET Client API on the machine you are deploying to. Using Visual Studio 2008, you can 'publish' the sample projects, and your final application onto the deployment machine.

Installation

Extract the supplied zip archive into your .NET project directory. The files will be placed in a directory called: Webpay.Net API v3.0

where 3.0 represents the version number of the release.



The API includes the core Webpay dynamic linked library (webpay.dll), a precompiled Test Application and two sample applications.

The Test Application is a precompiled VB application and is stored in the following directory.

- Webpay.NET-TestApplication

The two sample applications are stored in the following directories:

- Webpay.Net-ASPSample
- Webpay.Net-VBSample

Clicking on the Webpay.NET Samples.sln file will load these two samples into Visual Studio 2008.

Information on how to use the Test Application and the Sample Code is described in detail below.

Webpay Specific Information and Files

Your Webpay representative will have supplied you with information and files specific to the Webpay Gateway you will be using, including:

- The address of the Webpay Gateway
- The Test (and Live) Gateway port numbers
- Your Webpay Client ID
- A Webpay client certificate file (e.g., "net.pfx") appropriate for your Webpay gateway and the password protecting this certificate file

NOTE: certificates must be in the PKCS#12 format.

NOTE: For security purposes, the certificate file will be provided to you separately.

You must place the Webpay client certificate file somewhere on your machine so that the Webpay client software can access it. It is recommended that you place the file in the same folder with all your other Webpay client files or with other files that make up your own application.

Webpay.NET Client API Reference

Detailed documentation on the classes, functions, and methods that the Webpay.NET Client API provides is included in HTML, XML and CHM formats.

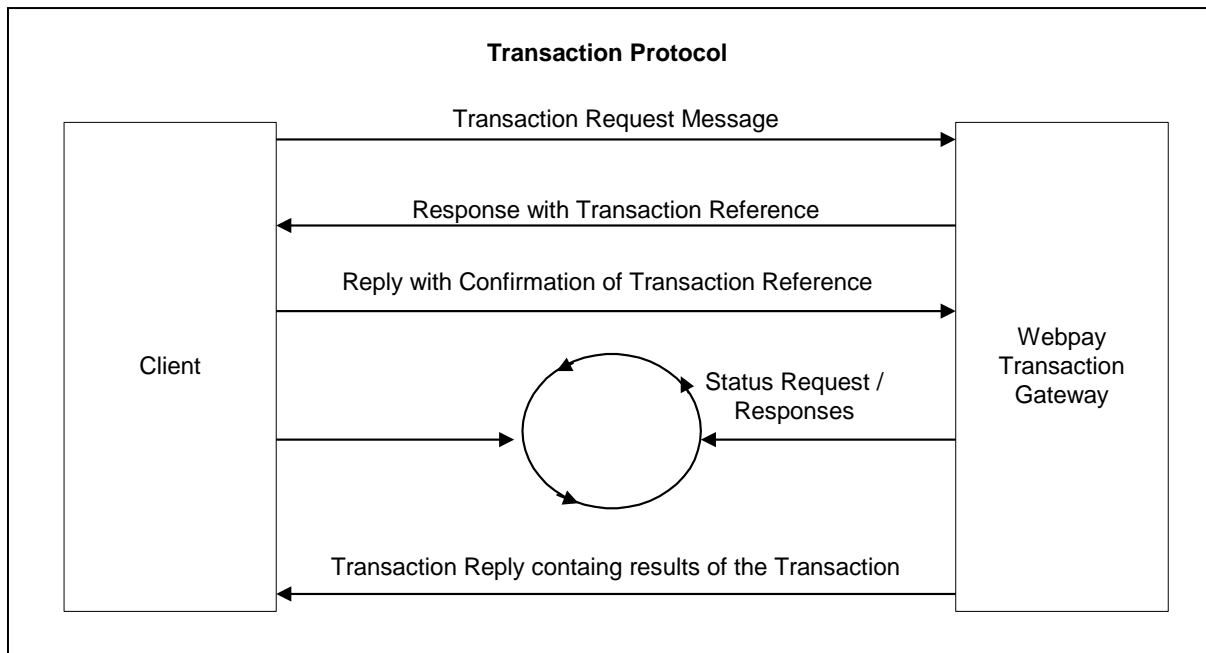
For information on Credit Card transactions and parameters, see the separate *Creditcard Transaction Specifications* document.

Error/Exception Handling Overview

The Webpay .NET API handles all errors internally. Any errors encountered during processing cause the transaction to fail. Error messages are viewed by investigating the ERROR field.

If a transaction attempt fails and a Transaction Reference has been returned by the Webpay Servers, the execute method will automatically query the status up to three more times. Beyond this, execute fails, and the transaction reference is available to the application. The merchant's application should be enabled to query the status of such a transaction, and be able to do so at any time. Both sample applications provided include status checking logic.

The following diagram depicts the transaction protocol implemented.



The Webpay Test Application

The Webpay.NET API comes with a precompiled Test Application. You should run this test program to:

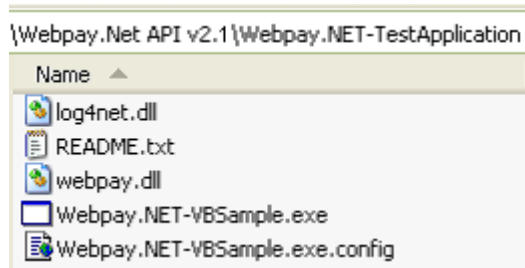
- Ensure there are no connectivity problems between your machine and the payment gateway
- Understand the basic workings of the Webpay Core Library

See the troubleshooting section of this guide for more information.

Running the Test Application

The Test Application is a precompiled VB application and is stored in the following location.

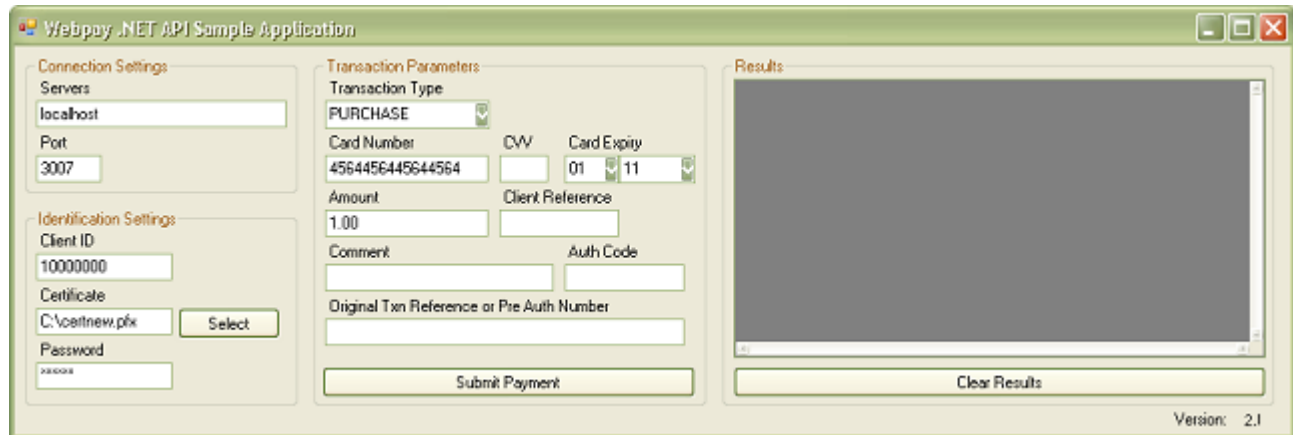
- Webpay.NET-TestApplication\ Webpay.NET-VBSample.exe



The application is launched by simply clicking on the executable file:

- Webpay.NET-VBSample.exe

The following screen shot shows the Test Application.



Supplying Input to the Test Application

The Test Application's interface consists of 3 groups of input boxes, and a text area which will display the results of the transactions. In order to process a transaction, you will need to set the both **Connection Settings** and **Identification Settings** to values specified by your Service Provider.

Typically, the hard-coded values you will need to change are set using the following functions:

Parameter	Description
Servers	The IP address or URL of the Webpay gateway server. Multiple gateways may be separated by commas.
Port	The designated port number for the Webpay gateway. The port number used will be determined on whether the client is connecting to the live or test environment.
Client ID	The Client ID that you have been designated.
Certificate	The name and path and of the certificate file. The full path should be used.
Password	The password that protects the certificate

The screenshot shows two sections: 'Connection Settings' and 'Identification Settings'. In 'Connection Settings', 'Servers' is set to 'localhost' and 'Port' is '3007'. In 'Identification Settings', 'Client ID' is '10000000', 'Certificate' is 'C:\certnew.pfx' with a 'Select' button, and 'Password' is masked with 'xxxxxx'.

The default values in the **Transaction Parameters** section should be sufficient to test basic setup and connectivity.

Now simply click the Submit Payment button to securely send the transaction details through to the Payment Gateway.

The screenshot shows the 'Transaction Parameters' section. 'Transaction Type' is 'PURCHASE'. 'Card Number' is '4564456445644564', 'CVV' is empty, and 'Card Expiry' is '01/11'. 'Amount' is '1.00' and 'Client Reference' is empty. There are empty fields for 'Comment' and 'Auth Code'. 'Original Txn Reference or Pre Auth Number' is empty. A 'Submit Payment' button is at the bottom.

NOTE: The test may take a little time to establish the first SSL connection but subsequent transactions will complete more quickly. After completing a transaction, output similar to the following should appear in the "Results" window:

The screenshot shows the full application window titled 'Webpay .NET API Sample Application'. It includes the 'Connection Settings', 'Identification Settings', and 'Transaction Parameters' sections on the left. The 'Results' section on the right displays the following transaction details:

```

=====
Result           : 0
Response Code    : 00
Error Message    :
Response Text    : APPROVED (TEST TRANSACTION ONLY)
Authorization Code : 496232
Pre Auth Number  :
Settlement Date  : 2009-10-07 00:00:00
Currency         :
Original Txn Ref :
Auth Num         :
CCV Response     :
=====
  
```

At the bottom right, it says 'Version: 2.1' and there is a 'Clear Results' button.

Analysis of the Response

The output of the test program should be similar to ...

- Response Code = "00"
- Response Test = "Approved"
- Error Message = ""
- Transaction Reference = 2230000012345678

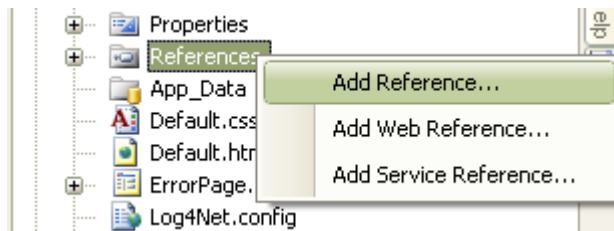
Response Code	A two digit code used to determine success or failure. The code number matches the response text field. "00", "08" or "77" designate an Approved transaction. Response codes are contained in The <i>Transactions Specifications</i> documents that accompany this kit. A response code < 0 means that a request was not processed properly and should be tried again. Users will need to consult the logs to determine what the error was that caused the transaction to fail.
Response Text	A meaningful text message that explains the response code
Error Message	Any errors received
Transaction reference	A unique reference number generated for each transaction.

Configuring Your Development Environment

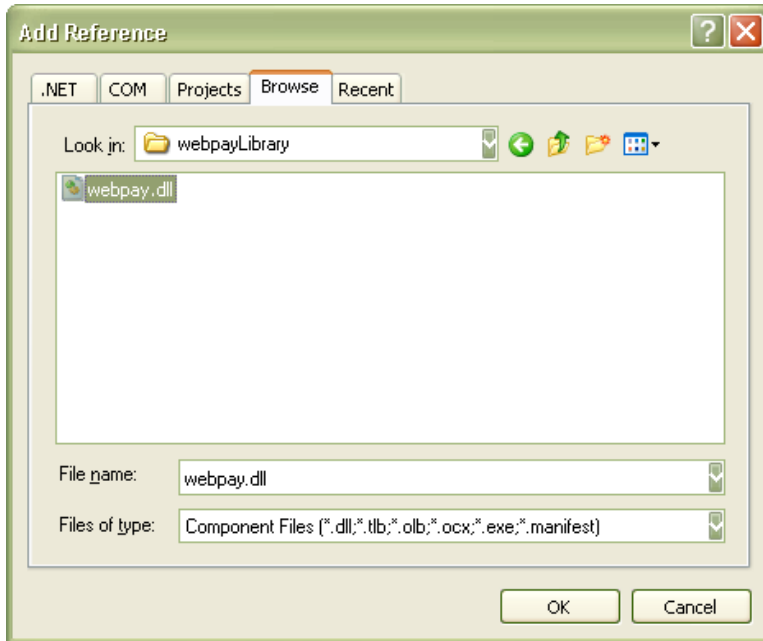
When creating a new project you will need to add a reference to the Webpay.dll in your Visual Studio project. Follow the steps below to add a reference to the webpay.dll file:

NOTE: This task already been performed for both Sample Applications

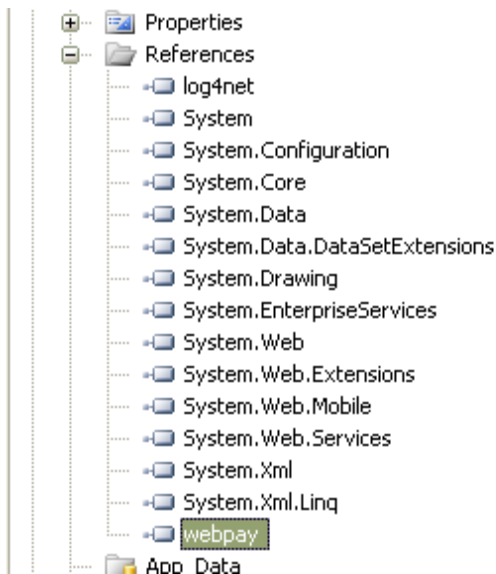
1. Within the project you are adding the reference to, right click on "References" and select "Add Reference"



2. Click the “Browse” tab, and select the webpay.dll file from the webpayLibrary folder which is located in the root directory of the API installation. Click OK to add the reference to your project.



3. If you expand the “References” node of your project in the Solution Explorer, you will see a new reference called “webpay”.

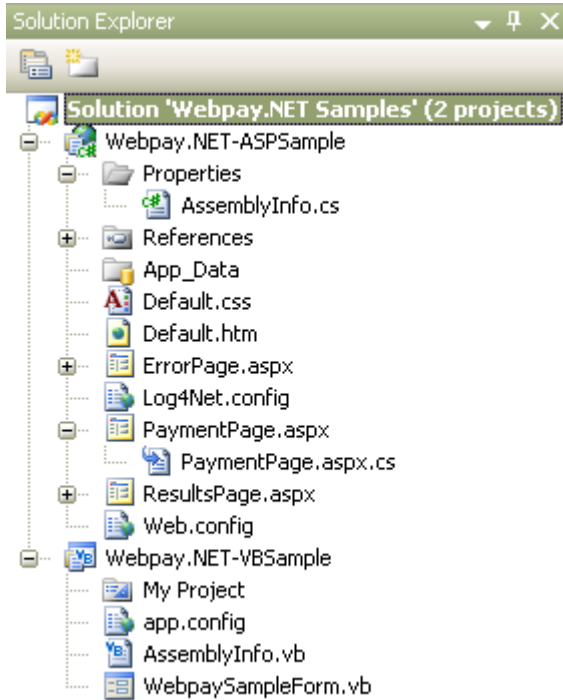


Your application will now be able to reference the Webpay functions.

Sample Applications

The samples included here illustrate two common implementation types using the most basic scenarios possible and contain several hard-coded values. They are intended only to demonstrate the basic concepts involved in using the Webpay.NET Client API in a manner that is relevant to the ways in which the API is normally used. Your production application will need to collect user or other input, and log the transaction responses to a database and/or display them to users.

The two sample applications included in the API package, work with Visual Studio 2008.



The Sample Solutions can be loaded by opening the following file with Visual Studio:

Webpay.NET Samples.sln

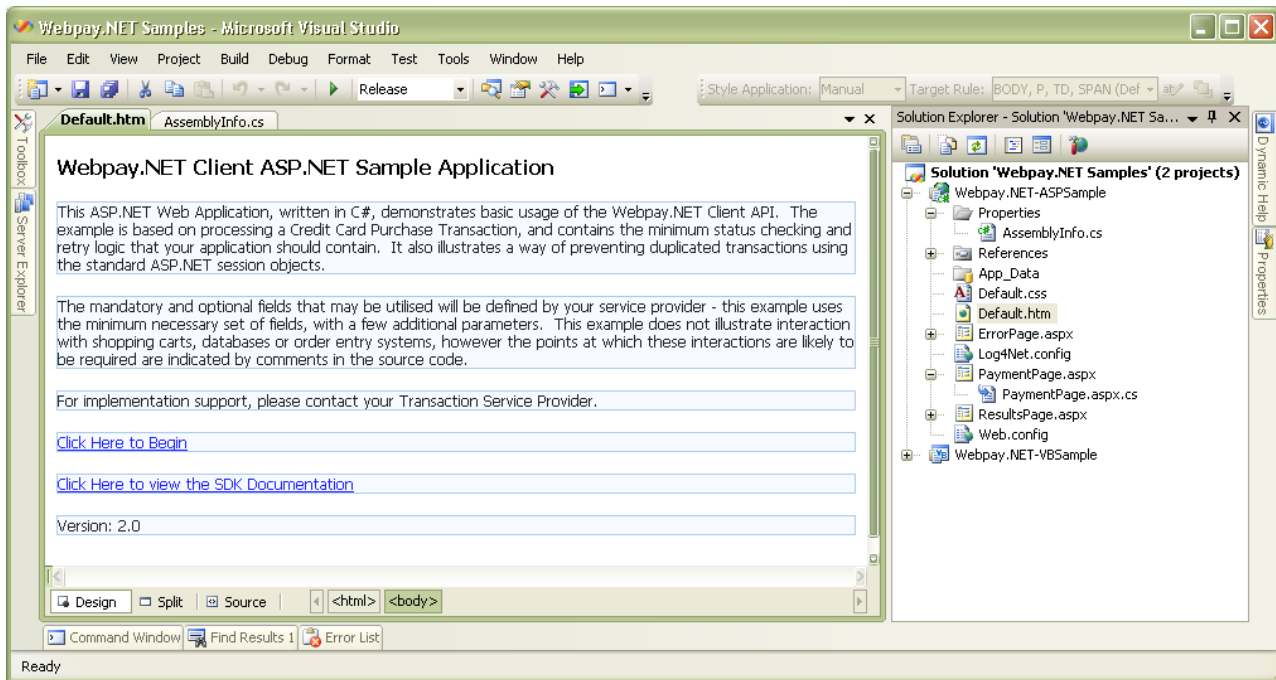
The first of these applications is an ASP.NET Web Application, the second is a Windows Forms Test application. These are both described in detail below.

Webpay.NET ASP Sample Web Application

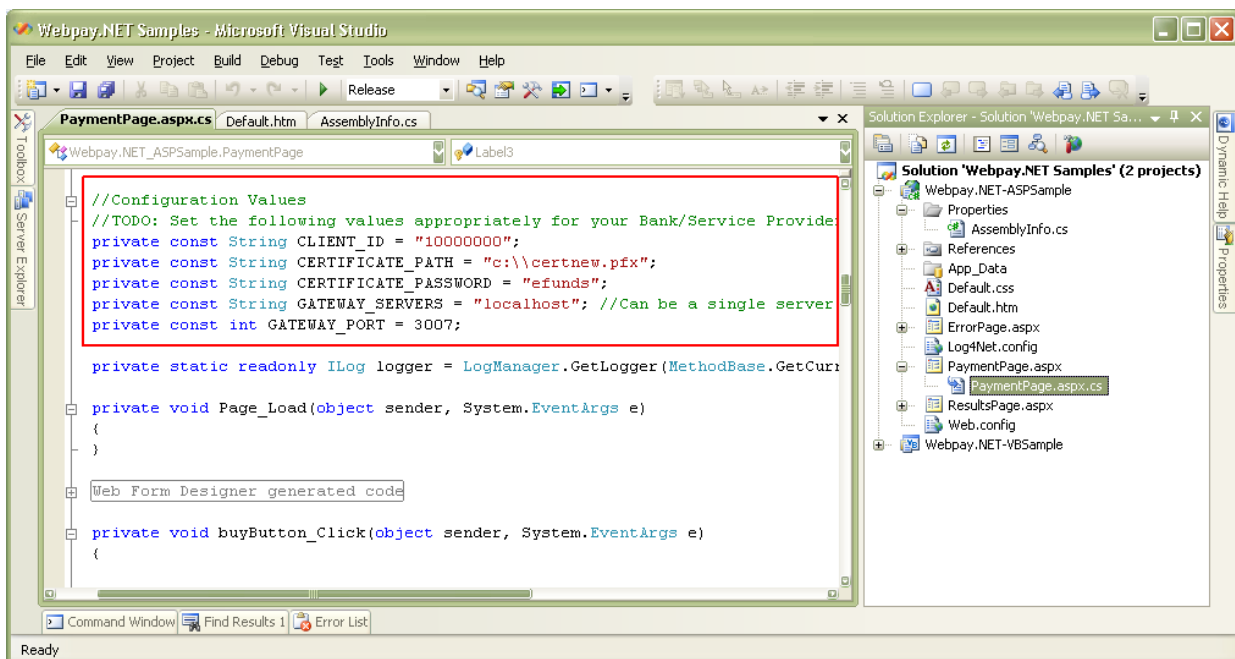
This application, written in C#, demonstrates how to implement the Webpay.NET Client API in a web application, and includes all necessary transaction status checking logic. Also included is the user session tracking logic that is essential for preventing duplicated transactions and ensuring that a response will be available to the user if a page fails to load due to network or other issues.

The application can run under the ASP.NET Development server, or it can be Published to, and run on an IIS Server.

The project consists of a basic HTML start page, which has links to the Payments Page, and to the installed API documentation in HTML format, and three ASP.NET pages used for processing and display.



The C# code has been implemented with the aim of being as straightforward as possible, and as such contains many hard-coded parameters, which your application may (depending on how flexible an application you require), load either from a configuration file or a database. These parameters include the Service Provider's server address, port settings, certificate location, certificate password and your Client ID. **NOTE: The connection and authentication settings are located in the PaymentPage.aspx.cs file. These values must be changed to your site specific values, you must edit this file and recompile the project.**

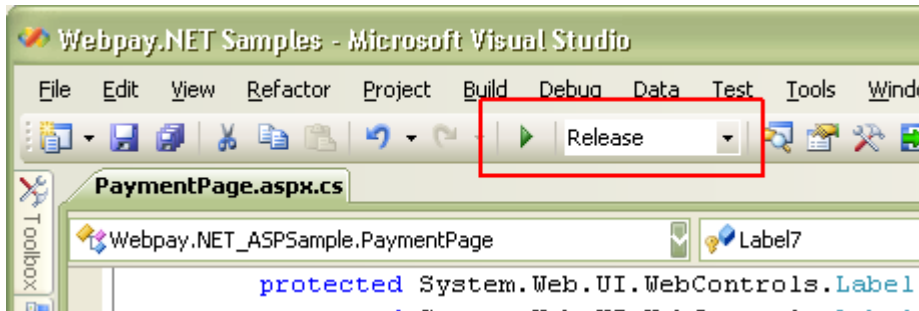


Testing with Visual Studio's Development Server

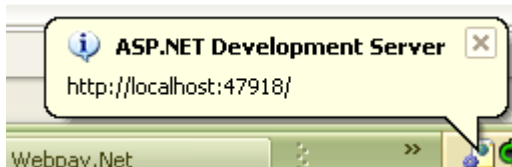
The simplest way of testing the Webpay.NET ASP Sample Web Application is to use the Visual Studio ASP.NET Development Server.

Step 1. Ensure you have updated PaymentPage.aspx.cs with your customised settings (as outlined above).

Step 2. With the PaymentPage.aspx.cs loaded, select 'Release' from the drop down box in the 'standard toolbar' and then click on the 'play' button, to launch the application.



You will see a pop up message in your system tray indicating that your development server is starting.



Your default web browser will then launch and automatically load the sample application:

<http://localhost:47918/PaymentPage.aspx>

Webpay.NET Client ASP.NET Sample Application....

Please fill in the Payment details below to complete your purchase

5 Things, Total Cost: \$ 59.00

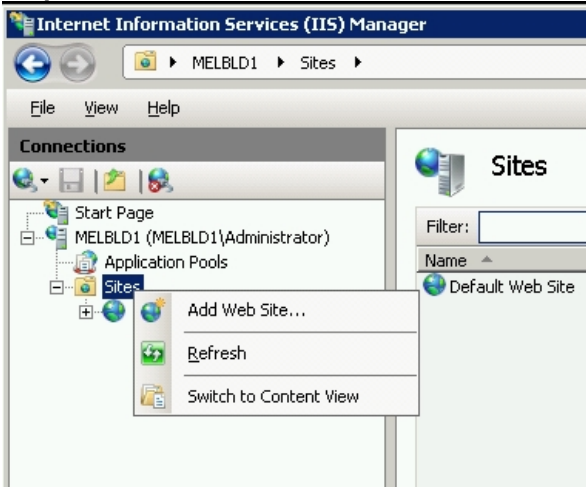
Name on Card	<input type="text"/>
Card Number	<input type="text" value="4564456445644564"/>
CVC2	<input type="text"/>
Expiry Date	<input type="text" value="October"/> <input type="text" value="2015"/>

Press the "Buy Now!" Button to test the application.

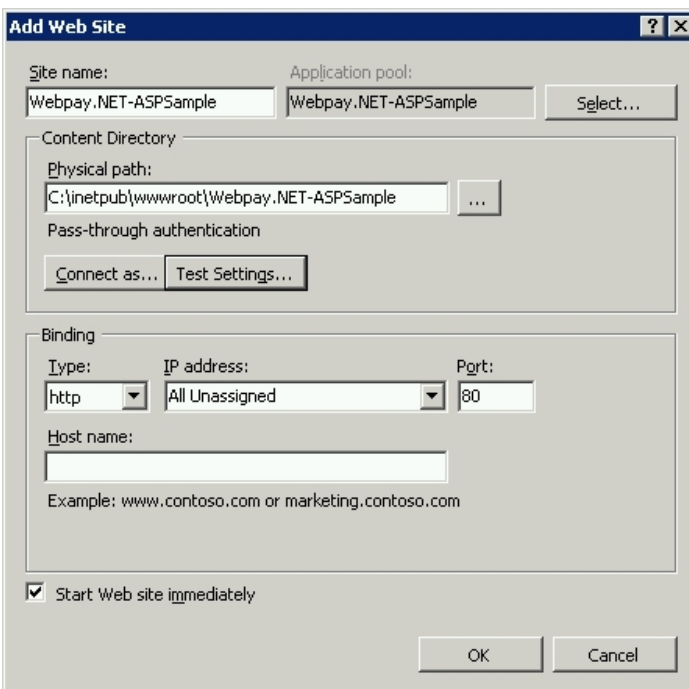
Deploying to IIS

The following section outlines how to deploy the sample application to an IIS Server.

Step 1. Create a new Web Site via the Internet Information Servers (IIS) Manager

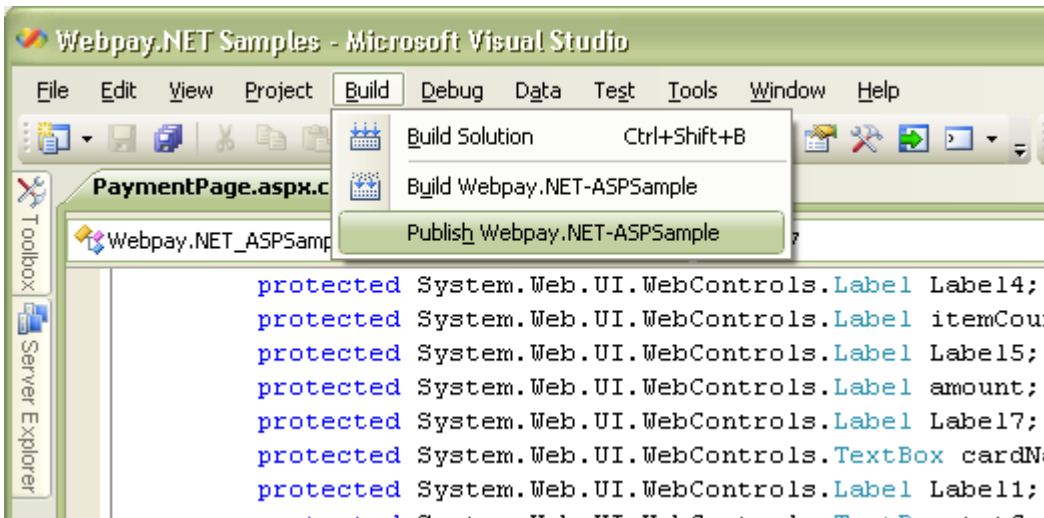


Provide some basic details to setup the Web Site. For the purposes of this example we have used a Site name of: Webpay.NET-ASPSample and have set the Port Binding to Port 80. You can customise these settings to your requirements.

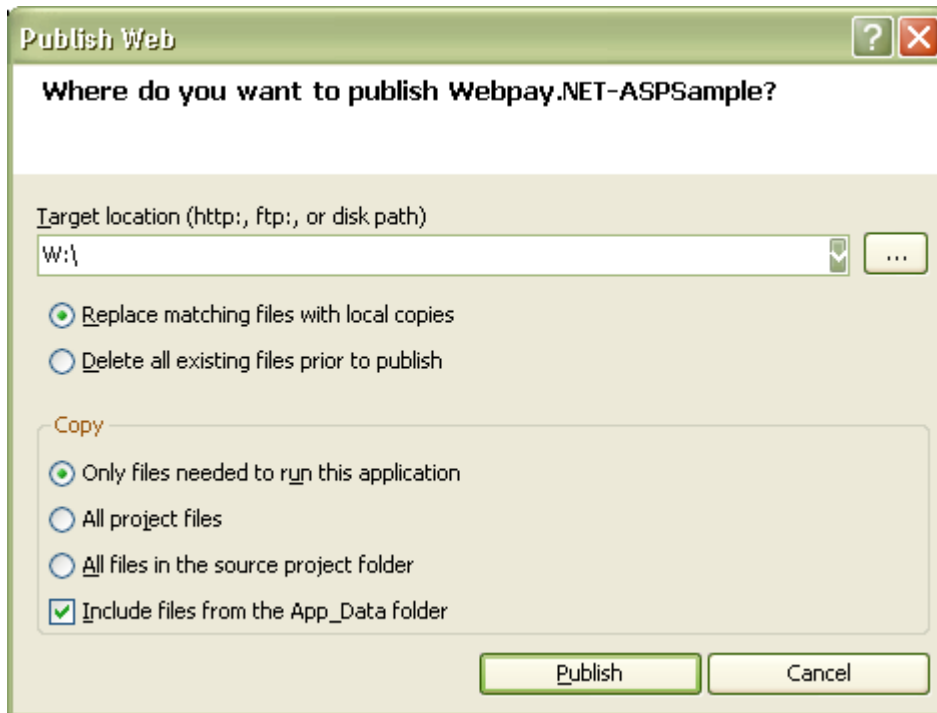


Step 2. Publish the sample application to the new web site.

- Ensure the Webpay.Net-ASPSample is selected in the Solution Explorer Window
- From the Build Menu select Publish Webpay.NET-ASPSample



- Select the location where to publish the Sample application.
In this example we have simply shared the folder defined as the wwwroot ('physical path') for our website on our IIS server. This is shown in the screen shot below as mapped network drive w:\ Please customise these publishing arrangements to satisfy your own requirements and standards.

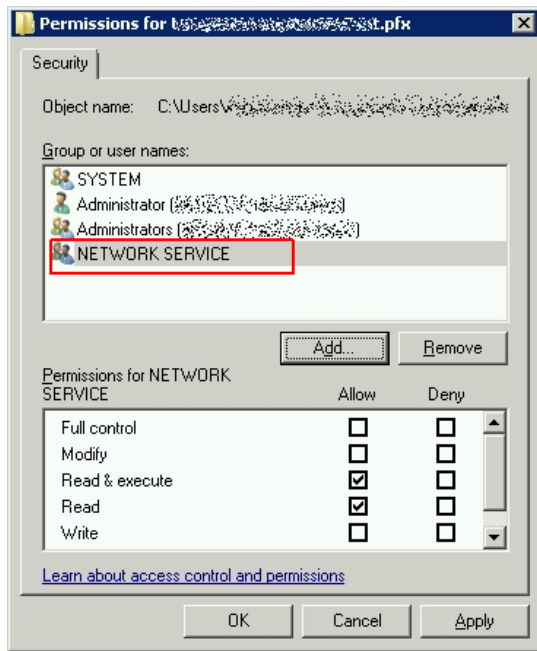


Step 3. Supply the appropriate permissions to the Certificate file (PFX).

You will have been supplied with a PFX certificate that is used to secure communications from your server to the Payment Gateway. This certificate needs to be copied onto the system and provided with the appropriate access permissions.

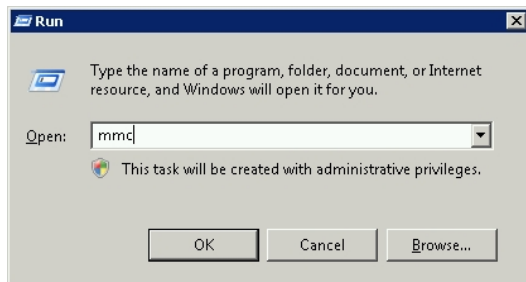
- Copy the certificate file somewhere onto your IIS server system.
- Ensure the certificate file can be read by the process that runs IIS.
Default IIS User is:
 - NETWORK SERVICE for Windows Servers 2003 and 2008.
 - ASPNET for XP and Windows Server 2000

The following screen shot shows the properties for the pfx file, with a NETWORK SERVICE user having been assigned permission to read the file.

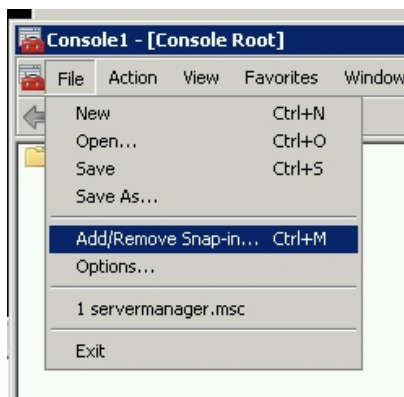


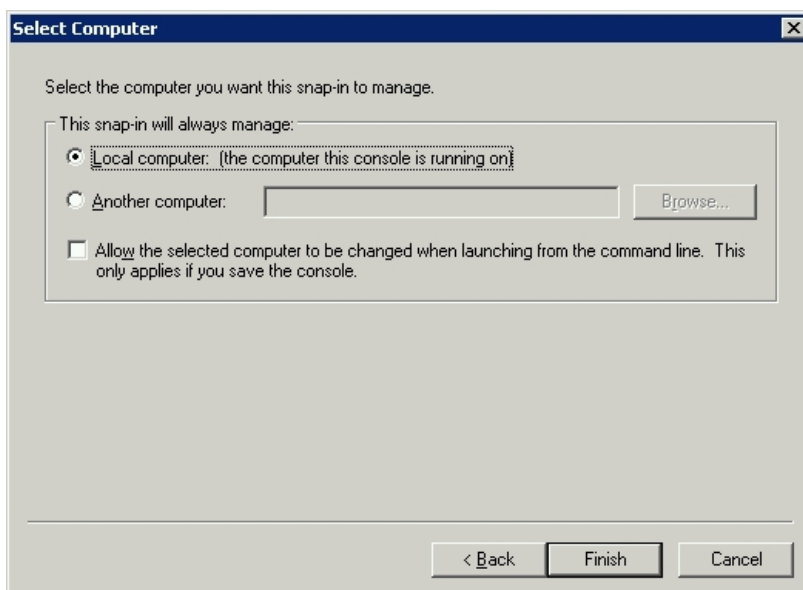
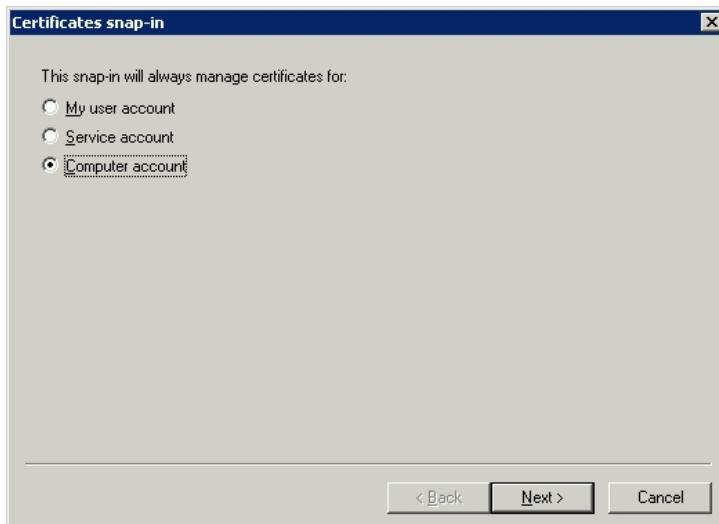
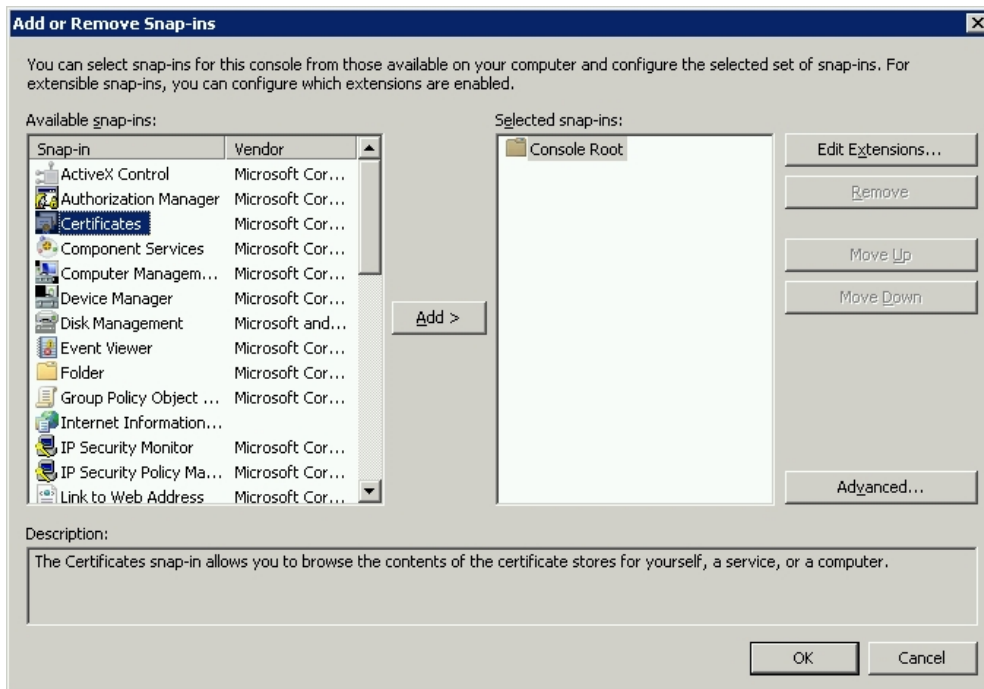
Step 4. Load the certificate into the servers Local Certificate Store

- Start MMC

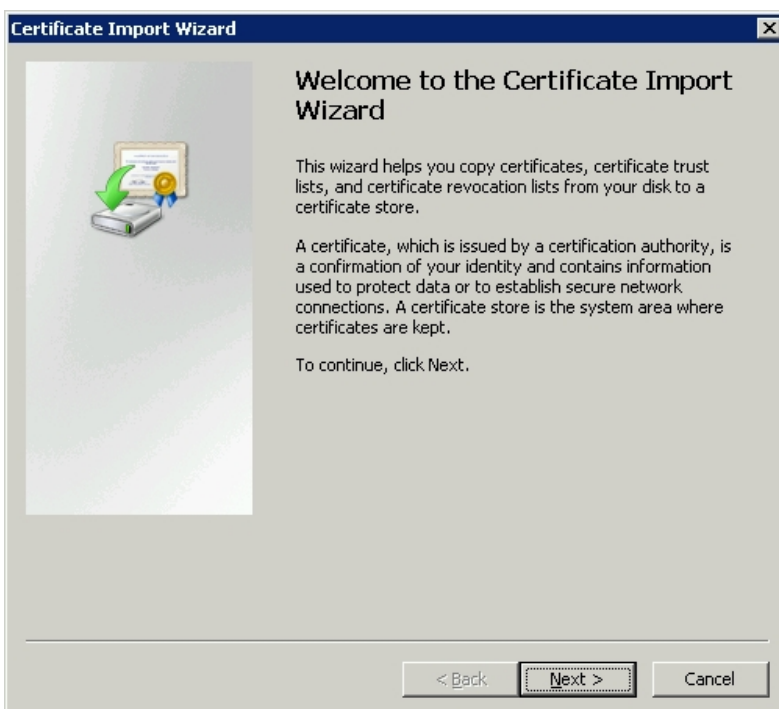
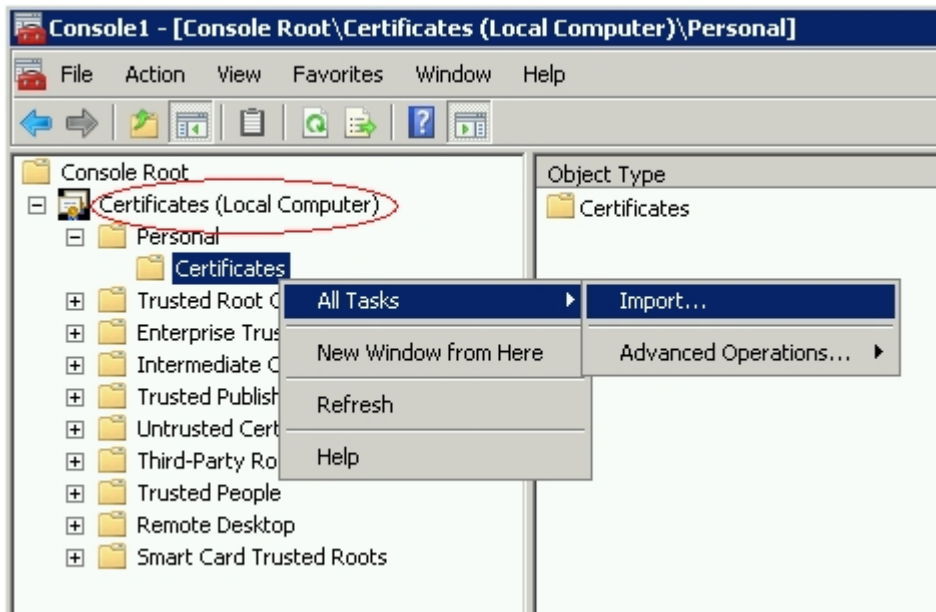


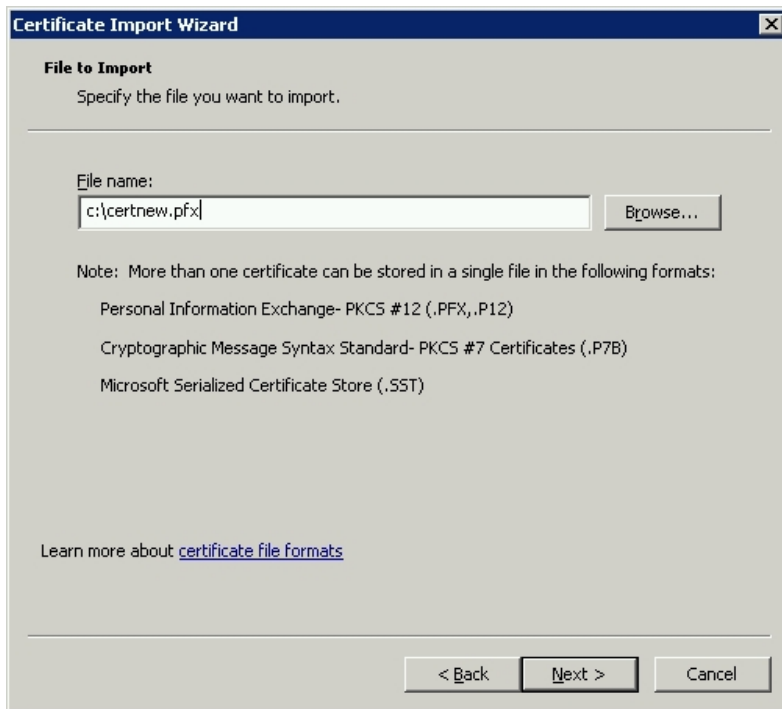
- Add the Certificates SnapIn. Select Computer Account for the local computer.





- Import the certificate to the Local Computer Store.
Right Click on the Personal->Certificates folder then select All Tasks->Import and follow the Certificate Import Wizard.





Certificate Import Wizard

File to Import
Specify the file you want to import.

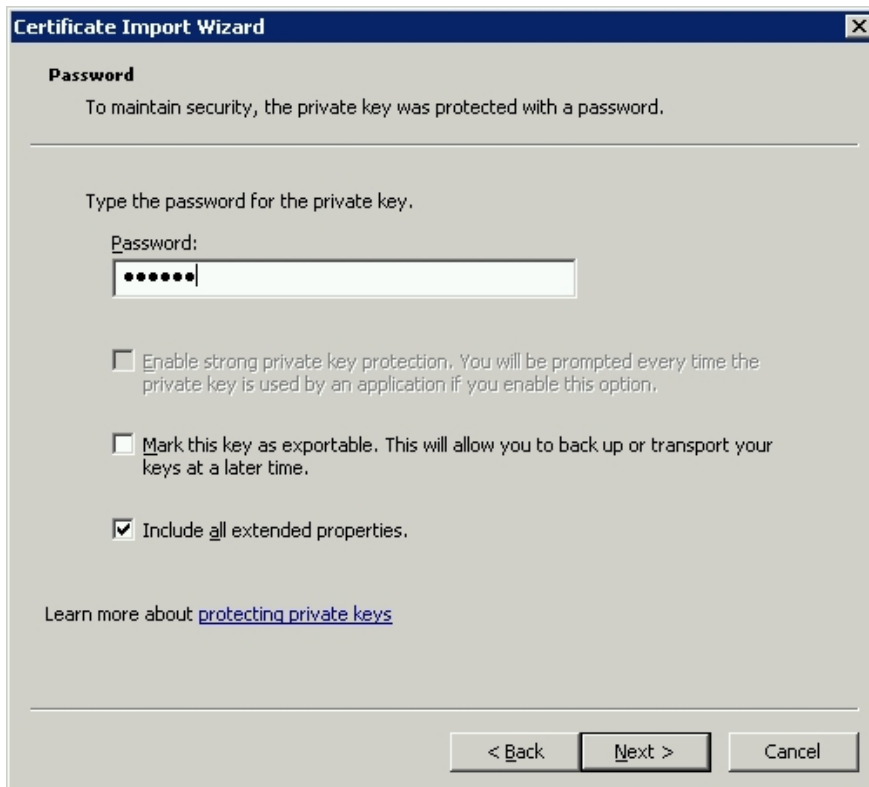
File name:

Note: More than one certificate can be stored in a single file in the following formats:

- Personal Information Exchange- PKCS #12 (.PFX,.P12)
- Cryptographic Message Syntax Standard- PKCS #7 Certificates (.P7B)
- Microsoft Serialized Certificate Store (.SST)

Learn more about [certificate file formats](#)

< Back Next > Cancel



Certificate Import Wizard

Password
To maintain security, the private key was protected with a password.

Type the password for the private key.

Password:

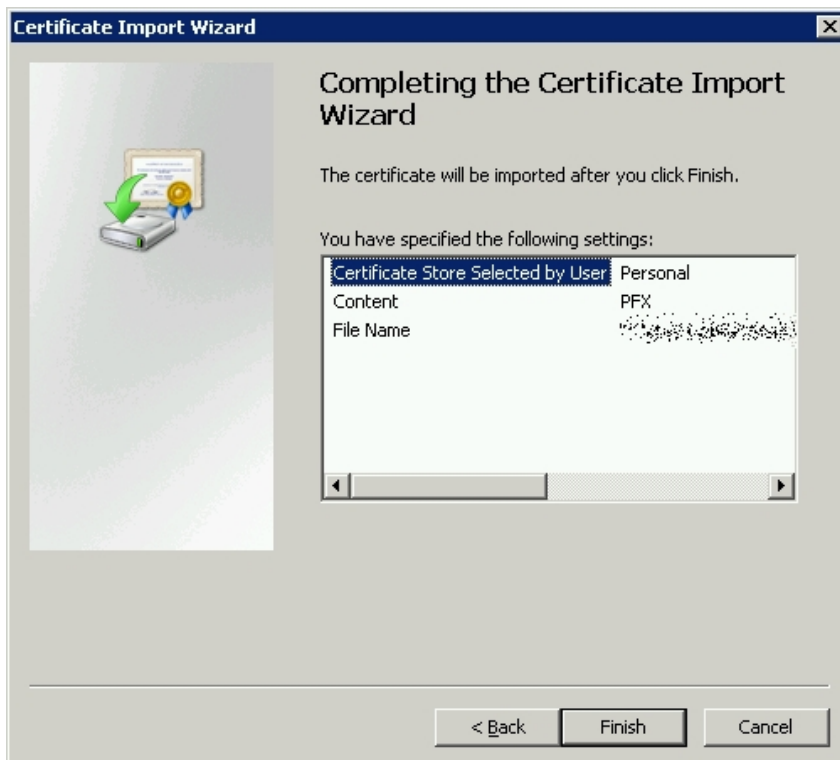
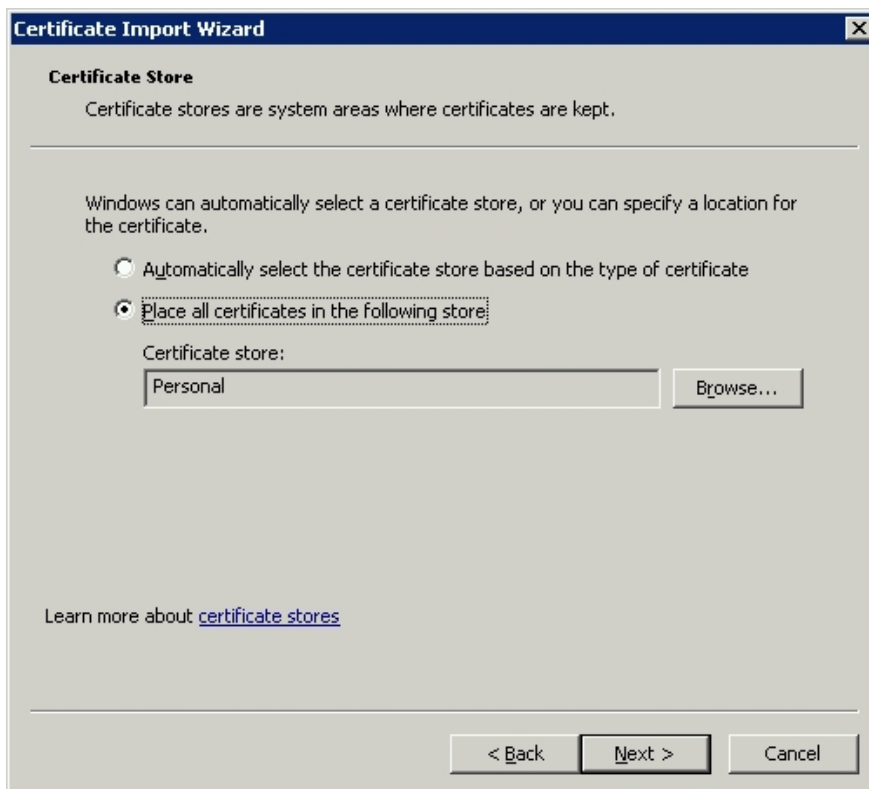
☐ Enable strong private key protection. You will be prompted every time the private key is used by an application if you enable this option.

☐ Mark this key as exportable. This will allow you to back up or transport your keys at a later time.

☒ Include all extended properties.

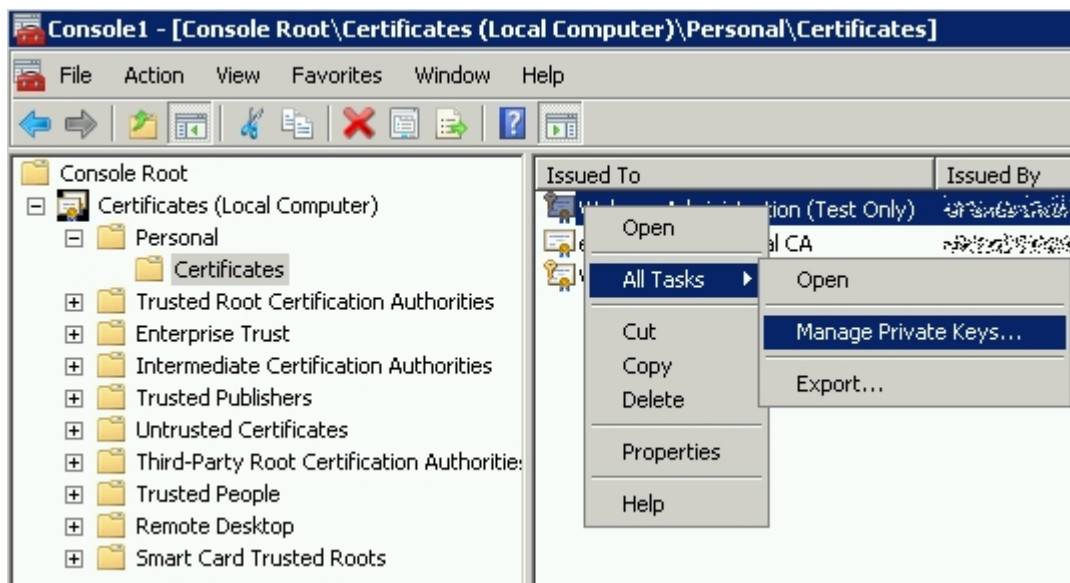
Learn more about [protecting private keys](#)

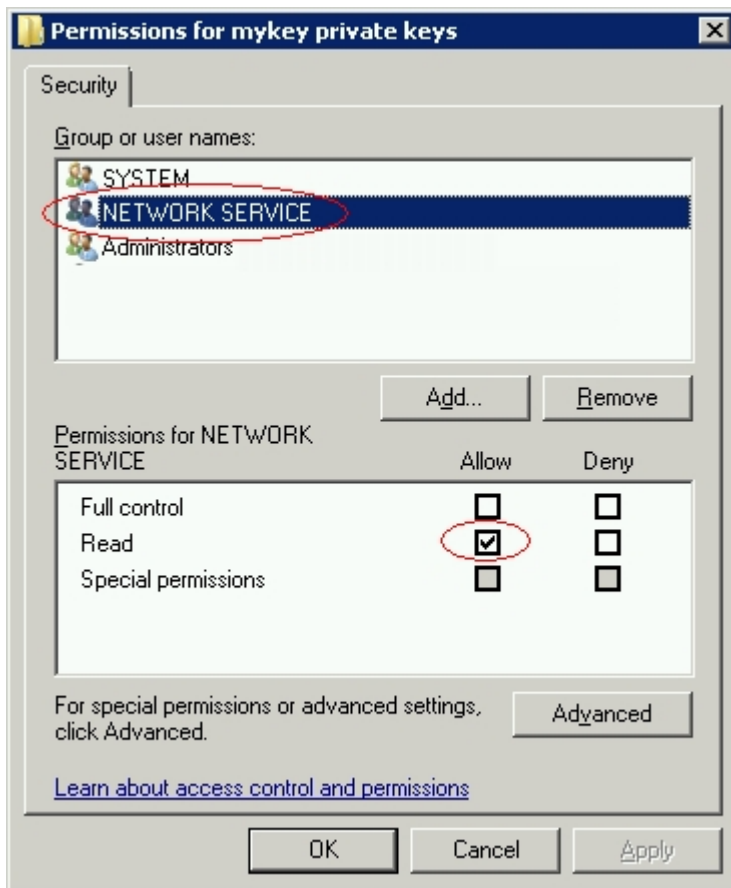
< Back Next > Cancel





- Assign Read permissions for the IIS user to access the Certificate details in the certificate store.
Right click on the imported certificate and select All Tasks->Manage Private Keys
The screen shots below show that the NETWORK SERVICE user has been given read permission to our freshly imported Key.
Please note that if you are using Windows Server 2003 you will need to use winhttpconfig.exe (which is part of the Windows 2003 resource kit) to perform this step. Additional details on this step are outlined in the Appendix.



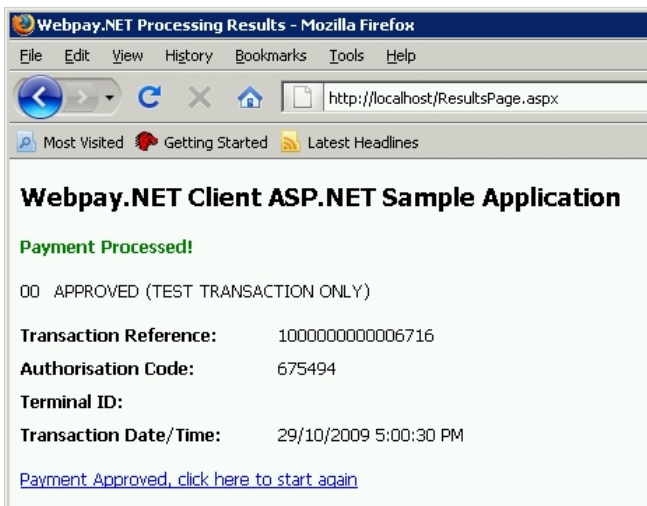
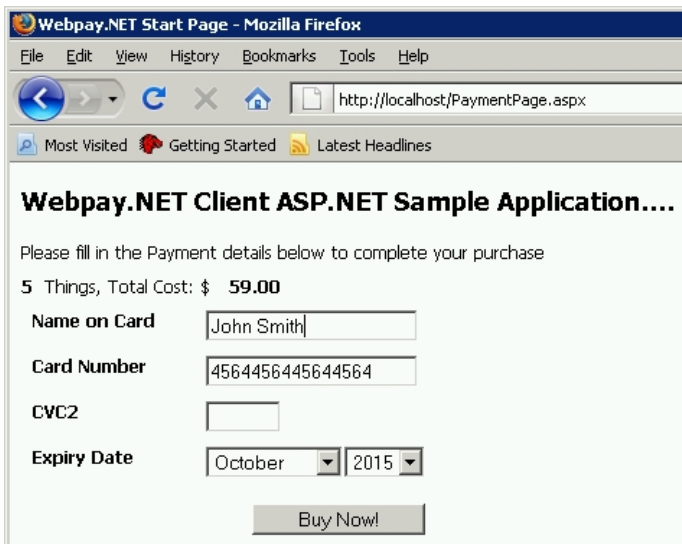
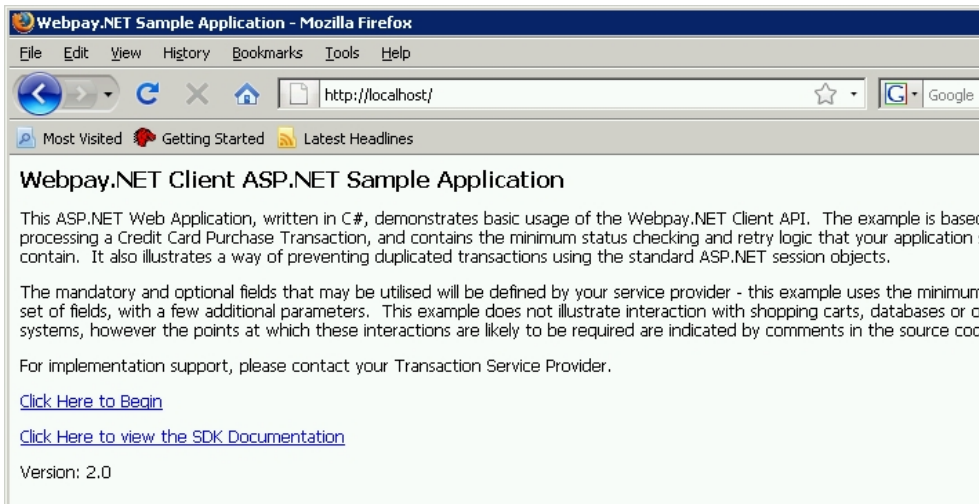


Once the appropriate permissions have been supplied you can test the application.

Please note: If you receive the error: **webpay.client.ConnectException: A call to SSPI failed**

Then this indicates that the process running the application does not have the correct permission to access the certificate.

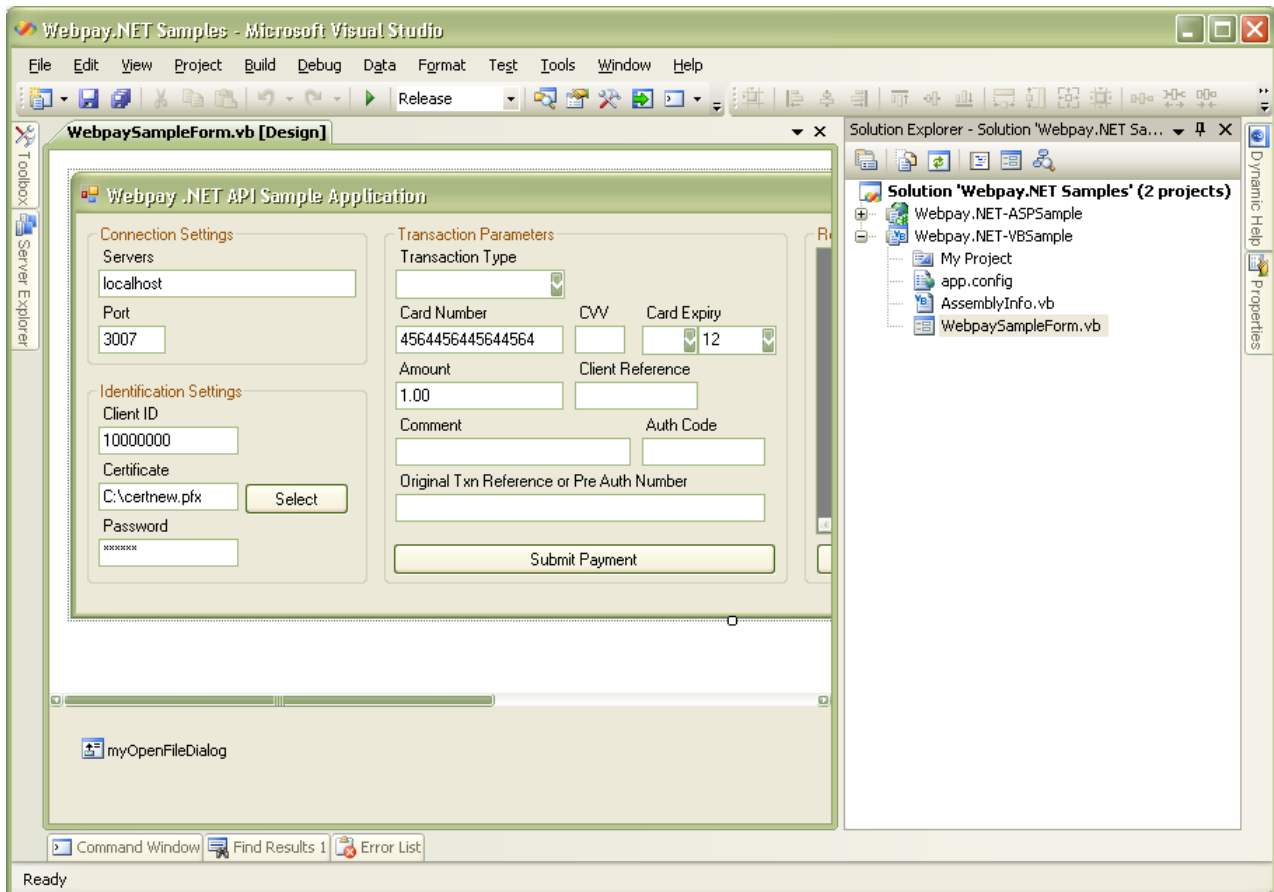
The following screen shots show the sample application running under IIS.



VB Sample Application

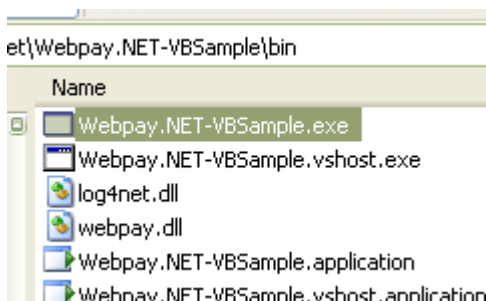
This application, written in Visual Basic .NET, demonstrates the implementation of the Webpay.NET Client API in a normal Windows GUI. Like the Web Application, it also contains the logic necessary for automated status checking in the event of a processing error. However, since it is a single threaded, single user application, it does not contain session management logic.

This project consists of only one Windows Form, as shown below. All parameters are driven by the user through the input boxes provided – in a normal end-user application, the connection and identification parameters would normally be loaded from a configuration file, possibly with a separate “Options” form loaded from a Menu. It would also be necessary to add logic for obtaining input from another part of the program, a database or external process and appropriately handling the transaction results.



Simply 'Build' the project to create the executable file: **Webpay.NET-VBSample.exe**

This can be located and run from the samples **bin** directory.



Troubleshooting

If your output fails any of the tests above, please review the steps you have taken so far. The following questions may assist in determining common connection problems:

- Is the **Webpay.dll** assembly file in the same directory as your executable?
- Is the right location of the Webpay client certificate file specified? You may need to specify the file's full path.
- Is the password correct for the Webpay client certificate?
- Is the correct address and test port number specified for the Webpay gateway? You may want to use "nslookup" or "dig (domain information groper)" to ensure that the address resolves correctly.
- Are you behind a firewall? You may need to contact your Security Administrator to ensure the appropriate ports have been defined through the firewall.

If you still can't identify the problem, call your Service Provider representative. Be prepared to e-mail the output of the test program.

Network Errors

Commercial and freeware tools can be found on the Internet to assist you in testing your network connectivity to the Webpay Secure Transaction Servers.

Unexpected Errors While Testing

In some cases, the system may return errors when you are expecting an approved or other code during testing. Please check the response code you receive against those contained in the *Creditcard Transactions Specifications* document.

Some response codes may be generated at any time and this behaviour is perfectly normal. For example, the System may return the code **A6** for **Server Busy** if several other merchants are conducting load tests at the same time.

Before reporting errors, ensure that you are able to replicate them, so that our support team can diagnose properly.

Common errors and solutions

The following section outlines some common errors and their solutions.

Error Message	Comment
Error Details: System.Exception: Invalid certificate Path: File Not Found at C:\net.pfx at webpay.client.Webpay..ctor(String clientID, String certificatePath, String certificatePassphrase) at Webpay.NET_ASPSample.PaymentPage.buyButton_Click(Object sender, EventArgs e) in C:\dev\Webpay.Net API v3.0\Webpay.NET-ASPSample\PaymentPage.aspx.cs:line 100	<p>Ensure Certificate has permission to be accessed by the process running IIS.</p> <p>Default User is:</p> <ul style="list-style-type: none"> • NETWORK SERVICE for Windows Servers 2003 and 2008. • ASPNET for XP and Windows Server 2000 <p>Simply right click on the file and provide read permission to the correct user.</p> <p>To check which user: View the Application Pools tab in IIS Manager and check the Identity column.</p>
Error Details: webpay.client.ConnectException: A call to SSPI failed, see inner exception. at webpay.client.Webpay.connect() at webpay.client.Webpay.disconnectedInitTransaction() at webpay.client.Webpay.initTransaction() at webpay.client.Webpay.execute() at	<p>This error indicates that the Webpay object can not access the specified Certificate. Please ensure that your certificate has been imported.</p> <p>To 'import' the certificate, double click on the supplied pfx certificate file and</p>

Webpay.NET_ASPSample.PaymentPage.processPayment(Webpay webpayObject)	follow the instructions of the Certificate Import Wizard
System.MissingMethodException: Method not found: 'Boolean System.Threading.WaitHandle.WaitOne(Int32)'. at webpay.client.Webpay.CallWithTimeout(Action action, Int32 timeoutMilliseconds) at webpay.client.Webpay.execute() at Webpay.NET_VBSample.WebpaySampleForm.submitPaymentButton_Click(Object sender, EventArgs e)	This error indicates that you may need to update your .Net Framework. Check section <i>0 Minimum System Requirements</i> for details of which .Net Framework version is required. This website will let you know which version of the .Net framework you currently have installed: http://www.hanselman.com/smallestdotnet/

Switching on debugging

The Core Webpay Library (webpay.dll) includes inbuilt debug logging which by default is switched off. The following instructions illustrate how to enable debug logging should you require it.

The Webpay.Net dll and samples applications use the Log4Net logging framework.

The sample code is already configured to use Log4Net. In order to view the logging output of the Core Webpay Library you just need to update the sample code to 'switch' the internal Logging on.

To do this is a simple matter of setting the DEBUG flag to ON.

VB Example:

'The Debug Flag can be used to switch on the Internal Logging from the Webpay Library (Uses Log4Net)

```
webpayObject.setValue("DEBUG", "ON")
```

C# Example

//The Debug Flag can be used to switch on the Internal Logging from the Webpay Library (Uses Log4Net)

```
webpayObject.setValue("DEBUG", "ON");
```

The following Log4Net configuration files are available.

Logging Source	Logging Configuration File
webpay.dll (Core Webpay Library)	webpay.dll.config (located in the same directory as the webpay.dll file)
C# Sample Application	Log4Net.config
VB Sample Application	app.config Webpay.NET-VBSample.exe.config

Should you need to modify the default configuration files please consult the online log4Net documentation.

<http://logging.apache.org/log4net/release/manual/configuration.html>

Additional Information

Security

The *Webpay Transaction Server* maintains secure transactions through the use of SSL (Secure Sockets Layer). The SSL Handshake Protocol was originally developed by Netscape Communications Corporation to provide security and privacy over the Internet. Using SSL, we can provide client and server authentication, and ensure 128-bit encryption of all transaction details.

It is recommended that each merchant is identified by a **unique** X.509 digital certificate. Merchants that have multiple MerchantID's should be issued with one certificate for each MerchantID and it is essential that the correct certificate is used when performing a transaction, otherwise authentication errors will occur, and the transaction request will fail.

Performance

The Webpay Transaction Servers can be clustered to provide high availability, fail-over, and fast processing under load. The average time taken to complete a transaction is 6-7 seconds, but merchants may experience faster or slower transaction completion times due to a number of factors, including: the "distance" (or number of hops) between the merchant's server and the Webpay environment, the type and speed of connection, general internet congestion, high server load, etc. The API's automatically manage transaction and connection timeouts.

Connectivity

Depending upon the WTS implementation, a number of gateways may be utilised for communication with the Client APIs. This implementation ensures maximum connection and system failover. The API may be directed to multiple gateways, and if the first is unavailable the next gateway in sequence will be contacted. This is defined by including each gateway address separated by a comma when invoking the `setServers` function.

Test/Live Merchant Status

St.George Bank provides a Test and a Live system for merchant use. The Test system is a mirror of the Live system, with the exception that it sends the requests to a *simulated* banking environment.

Initially, you will need to develop and test your code using the test system. When you believe you are ready to "Go Live", you must contact the St.George Bank Support Team to begin the accreditation process.

The Webpay Servers produce a wide range of transactional responses. Within the testing regime, these responses are controllable through the allocation of the cents component in the amount value. To fully test your system's ability to handle and control varying responses, you should test all cents values between 00 and 99.

A complete list of all creditcard response codes is contained in The *Transactions Specifications* documents that accompany this kit.

Contact the St.George Bank Support team for the current Server parameters required when using the Test and Live systems.

Session and Concurrency Issues

When implementing the Webpay.NET API, it is important to consider the effect of multiple users accessing your application concurrently. A new transaction context should be created for every transaction, to provide a mechanism for maintaining a user's session, which will allow each transaction instance to be processed only once.

The sample ASP.NET web application shows one possible way of managing user sessions to prevent lost or duplicate transactions.work

Technical Support

8.1 IPG Web Site

The complete suite of API documentation and software is available from <https://www.ipg.stgeorge.com.au/download.asp>. Please visit/check this site often as any notices, upgrades, new API versions and documentation changes will be made available there.

8.2 Contacting St. George Bank

Before reporting errors, **ensure that you are able to replicate them**, so that we are able to diagnose properly.

Be prepared to have available for the Helpdesk or e-mail the following information:

- Operating system + API type (i.e. – W2003 + .Net API)
- Basic Hardware description (i.e. – P3 800 + 512 meg RAM)
- Debug logs & Log file containing the values of all fields in the transaction
- Description of the error (when and how it happened)
- Error code and Error message

The St. George Bank Technical Support Team can be contacted in the following ways:

- Telephone via the St. George 24/7 EFTPOS Helpdesk 1300 650 977
- Email to: ipgsupport@stgeorge.com.au
- On-Line at <https://www.ipg.stgeorge.com.au>

Appendix

Appendix A – Files included in this Developer Kit

NOTE: The certificate file (eg. net.pfx) is not included in the kit. For security purposes, this file will be supplied separately.

Documentation

Webpay .NET API Developer Guide
WebpayDocumentation.xml
Webpay.Net API Help File.chm

This document.
Webpay.Net API Class Library documentation
Windows Help File version of the
WebpayDocumentation.xml file

Base libraries

webpay.dll
webpay.dll.config

The .NET API Dynamic Link Library.
Config file for the DLL. Used to manage internal logging.

3rd Party libraries

log4net.dll

Webpay.dll uses the log4net.dll. Include this in your project if you need to see the internal webpay.dll logging.

VB.NET Sample

Webpay.NET-VBSample.exe
Webpay.NET-VBSample.vbproj
Sample Source Code (.vb and .resx files)
app.config (in the project space)
Webpay.NET-VBSample.exe.config (Once the project is built)

Precompiled VB.NET Sample Program
Project file for VB.NET Test Program
VB.NET Source for Sample Program
Basic configuration file for the sample application.

C# ASP Sample

Webpay.NET-ASPSample.csproj
Sample Source Code (.aspx and .cs files)
Log4Net.config
Web.config

Project file for C# ASP Sample Program
C# ASP Source for Sample Program
Logging configuration file
Config file

Visual Studio .NET 2008

Webpay.NET Samples.sln

Solution file for sample programs

Appendix B – Glossary

WTS	Webpay Transaction Server
-----	---------------------------

Appendix C – Advanced Configuration Options

The webpay.dll can be passed some advanced configuration options through the applications configuration file.

The configuration file in the sample applications are:

Project	Filename in project	Filename after build
VB.NET Sample	App.config	Webpay.NET-VBSample.exe.config
C# ASP Sample	Web.config	Web.config

The following advanced configuration options are available:

Execute Method timeout setting

This value determines the maximum timelimit that the execute() function will run for. Once this timeout limit is reach a TimeoutException will be thrown. Should a TimeoutException be thrown a status request will need to be processed to determine the outcome of the transaction.

Property Name	webpay.dll.executeMethodTime out	For example: <add key="webpay.dll.executeMethodTimeout" value="5000" />
Minimum Value	30000	Milliseconds
Maximum Value	60000	Milliseconds
Default Value	45000	Milliseconds. This value is used if no config value is available or if it is invalid.

Confirmation of this configuration setting will be written to the debug log with the following message:
Timeout value webpay.dll.executeMethodTimeout set to [45000];

SSLWrite timeout setting

This value determines the maximum length of time the API will block waiting for data while writing to the SSL stream.

Property Name	webpay.dll.sslTimeoutWrite	For example: <add key="webpay.dll.sslTimeoutWrite" value="5000" />
Minimum Value	1000	Milliseconds
Maximum Value	30000	Milliseconds
Default Value	15000	Milliseconds. This value is used if no config value is available or if it is invalid.

Confirmation of this configuration setting will be written to the debug log with the following message:
Timeout value webpay.dll.sslTimeoutWrite set to [5000];

SSLRead timeout setting

This value determines the maximum length of time the API will block waiting for data while reading from the SSL stream.

Property Name	webpay.dll.sslTimeoutRead	For example: <add key="webpay.dll.sslTimeoutRead" value="5000" />
Minimum Value	1000	Milliseconds
Maximum Value	30000	Milliseconds
Default Value	15000	Milliseconds. This value is used if no config value is available or if it is invalid.

Confirmation of this configuration setting will be written to the debug log with the following message:
Timeout value webpay.dll.sslTimeoutRead set to [5000];

X509 Storage options

This value provides additional configuration options to the API to allow it to support alternate x509Storage options.

Property Name	webpay.dll.X509KeyStorageFlag	For example: <add key="webpay.dll.X509KeyStorageFlag" value="MachineKeySet" />
Available Options	DefaultKeySet, MachineKeySet, UserKeySet	For more details on these options: http://msdn.microsoft.com/en-us/library/system.security.cryptography.x509certificates.x509keystorageflags%28v=VS.100%29.aspx
Default Value	None	If no config value is available or if it is invalid, then a call to the X509Certificate constructor is made without the X509KeyStorageFlags directive.

Confirmation of this configuration setting will be written to the debug log with the following message:
Timeout value webpay.dll.sslTimeoutRead set to [5000];

Sample configuration

```

<appSettings>
  <add key="webpay.dll.sslTimeoutWrite" value="5000" />
  <add key="webpay.dll.sslTimeoutRead" value="5000" />
  <add key="webpay.dll.executeMethodTimeout" value="45000" />
  <add key="webpay.dll.X509KeyStorageFlag" value="DefaultKeySet" />
</appSettings>

```

Appendix D - Security Considerations for older systems

Older versions of Windows Server and IIS have different security requirements to those outlined in the main document. Please consider the following information should you be running this API on an older system.

Windows 2000 Server

ASP applications use the ASPNET local account. In order for the application to be able to read the certificate file, Read access must be granted to the certificate file for the ASPNET local account. For example: the default certificate location used in the sample applications is c:\net.pfx.

Windows Server 2003

If you are running IIS v5.0, then ASP applications use the ASPNET local account. In order for the application to be able to read the certificate file, Read access must be granted to the certificate file for the ASPNET local account.

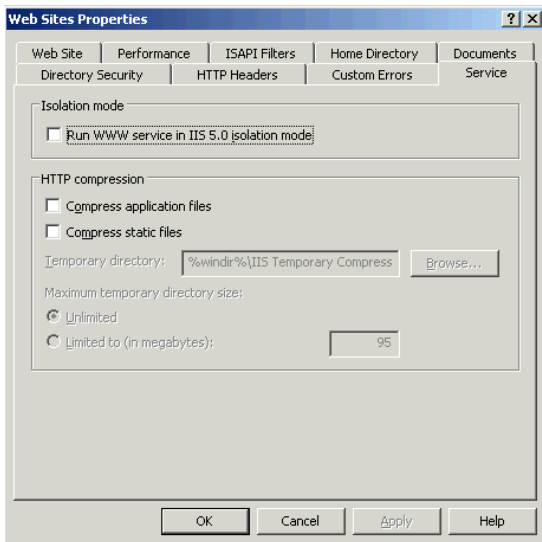
If you are running IIS v6.0, it can run in one of two modes:

- Worker Process Isolation Mode
- IIS 5.0 Isolation Mode

The default isolation mode upon installing IIS 6.0 depends on whether you perform a clean installation or an upgrade.

- After a clean install of IIS 6.0, IIS runs in worker process isolation mode.
- After an upgrade from an earlier version of IIS 6.0, the isolation mode is the same as configured on the previously-installed version of IIS 6.0.
- After an upgrade from IIS 5.0 or IIS 4.0, IIS 6.0 runs in IIS 5.0 isolation mode by default to maintain compatibility with your existing applications.

To determine which mode you are using, open the Web Sites properties page and look on the Service tab. If "Run WWW service in IIS 5.0 isolation mode" is ticked, then you are running in IIS 5.0 Isolation Mode. Otherwise, you are running in Worker Process Isolation Mode.

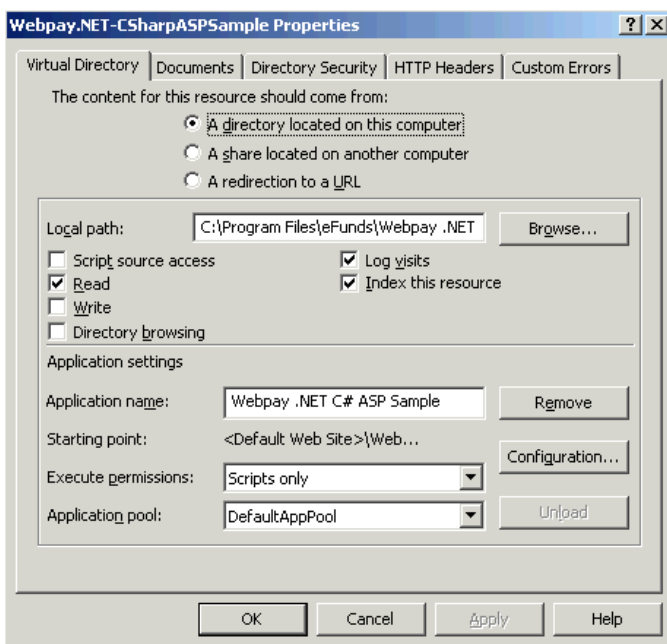


If you are running in IIS 5.0 Isolation Mode, the web application runs as the LocalSystem account. The LocalSystem account can read, execute, and change most of the resources on the computer. Thus, no special permissions need to be granted to the certificate file.

If you are running in Worker Process Isolation Mode, by default, processes run with the Network Service identity. The Network Service account has lower access rights than the default account for IIS 5.0 isolation mode. However, no special permissions need to be granted to the certificate file.

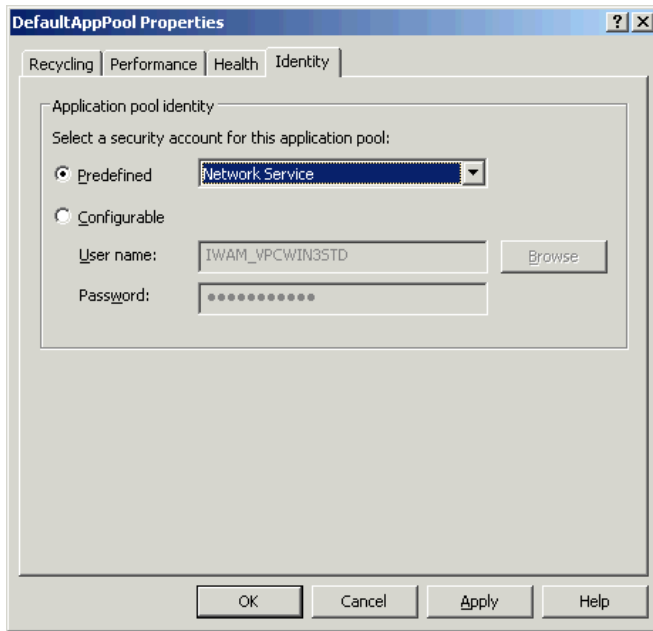
Under Worker Process Isolation Mode, it is possible to configure the identity of the application pool. The identity of an application pool is the name of the account under which the application pool's worker process runs. If you set the Identity to anything other than the predefined levels (Network Server, Local Service, Local System), then you must set the permission accordingly on the certificate file.

First, determine which application pool your website is running in. Open the website's Properties page.



The Application pool setting will list the web site's application pool.

Next, open the Properties page for the listed application pool. Select the Identity tab.



If the Configurable option is being used, then the User Name specified must be granted Read access to the certificate file.

Appendix E – Windows 2003 Certificate Store Permissions

To assign the IIS user permission to use the Imported certificate in a windows 2003 server you will need to use **winhttpCertCfg.exe**. It is part of the Win2003 Resource kit and can be downloaded from the following location:

<http://www.microsoft.com/downloads/details.aspx?familyid=c42e27ac-3409-40e9-8667-c748e422833f&displaylang=en>

The following command illustrates how to give the **NETWORK SERVICE** user access to a key with the subject **Webpay**.

winhttpcertcfg.exe -g -c LOCAL_MACHINE\MY -s "Webpay" -a "NETWORK SERVICE"

NB: Modify the above command to suit your certificate and user access requirements.

Troubleshooting:

The following command can be used to display which users currently have access to the certificate:

winhttpcertcfg.exe -l -c LOCAL_MACHINE\MY -s "Webpay"

NB: Modify the above command to suit your certificate details.