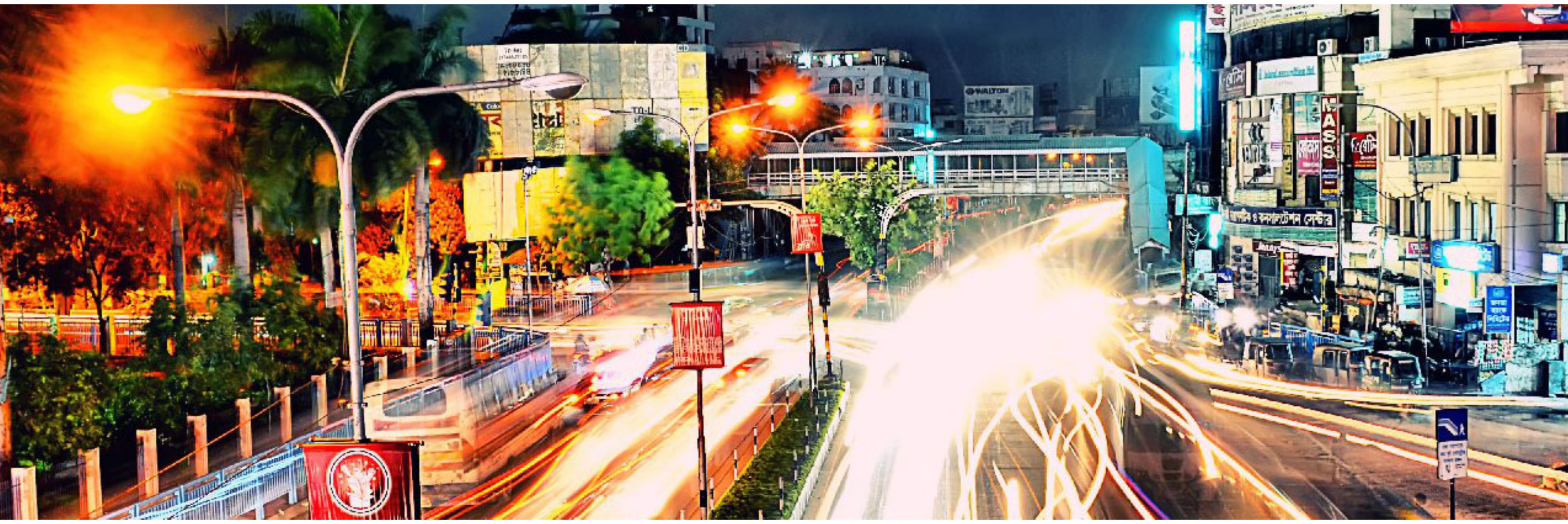
# SOAP Toolkits for Web Services Developer Guide



CyberSource®
A Visa Solution

## CyberSource Contact Information

For general information about our company, products, and services, go to http://www.cybersource.com.

For sales questions about any CyberSource Service, email sales@cybersource.com or call 650-432-7350 or 888-330-2300 (toll free in the United States).

For support information about any CyberSource Service, visit the Support Center: http://www.cybersource.com/support

## Copyright

## Restricted Rights Legends

## Trademarks

**Revision:** July 2020

# Contents

# Recent Revisions to This Document

| Release | Changes |
|---|---|
| July 2020 | Added endpoints and Business Center URLs for India. |
| February 2020 | • Removed Perl and ASP chapters and references.<br>• Updated links to sample code.<br>• Changed mentions of *.NET 3.0* to *.NET 3.0 and later*. |
| January 2016 | Fixed the URL for the perl sample code. |
| September 2015 | Updated the production server URL and the test server URL. |
| August 2015 | Changed mentions of .NET 3.0 to .NET 3.0 and later. |

# Configuring SOAP Toolkits for Web Services

SOAP toolkits are for merchants who use the SOAP protocol with a secure authentication method. With the SOAP toolkits, you do not need to download and configure a CyberSource client. To use any of the toolkits, your system must support these features:

- **HTTPS**: HTTP with TSL 1.2 encryption. The CyberSource servers do not support persistent HTTP connections.
- **SOAP 1.1:** Version 1.1 of the Simple Object Access Protocol.
- **Document/literal (unwrapped)**: Style of the WSDL used by the CyberSource Web Services. With this style, the entire content of the SOAP body is defined in a schema.
- **UsernameToken**: Authentication mechanism specified in WS-Security 1.0. in the header of the SOAP message.

## Supported Toolkits

CyberSource has tested and supports only the toolkits listed below. You can implement a toolkit on a platform that is not tested or supported, but CyberSource cannot guarantee that you can use such an implementation with the Web Services.

| Toolkits | Supported Platforms |
|---|---|
| PHP 5.2.1 | Windows, Linux, Solaris |
| .NET:<br>.NET 2.0 and WSE 3.0<br>.NET 3.0 (WCF) and later | Windows |
| C++ with gSOAP:<br>gSOAP 2.7.9c for Windows<br>gSOAP 2.7.9e for Linux<br>gSOAP 2.7.9d for Mac OS X | Windows, Linux, Mac OS |
| Java with Apache Axis and WSS4J | Windows, Linux, Solaris |

CyberSource recommends that you use logging only when troubleshooting problems. To comply with all Payment Card Industry (PCI) and Payment Application (PA) Data Security Standards regarding the storage of credit card and card verification number data, the logs that are generated contain only masked credit card and card verification number (CVV, CVC2, CVV2, CID, CVN) data. For more information about PCI and PA requirements, see www.visa.com/cisp.

Follow these guidelines when working with log files:

- Use debugging temporarily for diagnostic purposes only.
- If possible, use debugging only with test credit card numbers.
- Never store clear text card verification numbers.
- Delete the log files as soon as you no longer need them.
- Never email personal and account information, such as customers' names, addresses, card or check account numbers, and card verification numbers to CyberSource.

## Destination URLs for SOAP Messages

The latest version of the API is located at https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor or in India https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor.

When constructing your SOAP messages, use the following target URLs:

- Test environment: https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor
- Production environment: https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor
- Production environment in India: https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor

`1.x` is not a placeholder for the version number but an integral part of the URL.

## Transaction Key

### Context

Before you can send requests for Internet Commerce Suite (ICS) services, you must create a security key for your CyberSource merchant ID. Use this key to replace the placeholder value for `TRANSACTION_KEY` in the code samples.

***IMPORTANT:*** You must use separate transaction keys for the test and production environments.

1. Log in to the Business Center.
   - Live transactions: https://ebc2.cybersource.com/ebc2/
   - Live transactions in India: https://ebc2.in.cybersource.com/ebc2/
   - Test transactions: https://ebctest.cybersource.com/ebc2/
2. On the left navigation pane, click the **Payment Configuration** icon.

3 Click **Key Management**. The Key Management page appears.

4 In the Search toolbar, select the Merchant ID for which you want to create a key.

5 Click **Generate Key**. The Create Key page appears.

6 Select the type of key that you want to generate.

7 Click **Next Step**.

8 Select a key type, and generate a new key.

9 Click **Submit**. The Key Management page appears.

---

***IMPORTANT:*** Be sure to store the test and production environment transaction keys in different locations. Be careful not to overwrite a key in the wrong directory.

## SOAP Message Sample

Before using this sample, replace ***N.NN*** with the current API version and use your merchant ID and password.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-s
ecext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>yourMerchantID</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-pro
file-1.0#PasswordText">yourPassword</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-N.NN">
      <merchantID>yourMerchantID</merchantID>
      <merchantReferenceCode>MRC-123</merchantReferenceCode>
      <billTo>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
        <street1>1295 Charleston Road</street1>
        <city>Mountain View</city>
        <state>CA</state>
        <postalCode>94043</postalCode>
        <country>US</country>
        <email>null@cybersource.com</email>
      </billTo>
      <item id="0">
        <unitPrice>5.00</unitPrice>
        <quantity>1</quantity>
      </item>
      <item id="1">
        <unitPrice>10.00</unitPrice>
        <quantity>2</quantity>
      </item>
      <purchaseTotals>
        <currency>USD</currency>
      </purchaseTotals>
      <card>
        <accountNumber>4111111111111111</accountNumber>
        <expirationMonth>11</expirationMonth>
        <expirationYear>2020</expirationYear>
      </card>
      <ccAuthService run="true"/>
    </requestMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

# Constructing SOAP with PHP 5.2.1

This section describes how to construct SOAP messages to process transactions with CyberSource.

Before starting this process, download and install the third-party software. CyberSource tested these versions:

| Software Tested | Description |
|---|---|
| • Linux Kernel 2.4<br>• Windows XP Pro with SP2<br>• Solaris | Operating system versions tested. |
| PHP 5.2.1 | PHP software. The SOAP extension is provided only in versions 5.2.1 and later. |
| libxml2 2.6.23 | 2.6.11 is the minimum version required by the SOAP extension. |
| openssl 0.9.8d | SSL library; 0.9.6 is the minimum required version by the SOAP extension. |

Although the SOAP extension does not have built-in support for Web Services Security, you can add the required header elements for the UsernameToken information to the outgoing request. The code in the sample file shows how to extend the `SoapClient` class and how to override its `__doRequest()` method (lines 17 to 49) to insert the UsernameToken information.

Test the client application with the following Cybersource sample code: https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_php.

The sample PHP files contain many comments and a sample card authorization. Choose the file appropriate for you:

- `cli-sample.php` if you use the command-line interface
- `web-sample.php` if you use the Web interface

Be sure that you understand the content of the file and that you replace the generic values of the variables, such as your merchant ID and password, with your own values.

## Installing PHP on Windows

### Context

If you are running PHP on Windows, your PHP application requires these two extensions: `SOAP` and `OpenSSL`.

1. If an extensions directory is not already present, create an extensions directory in the `php.ini` file:

   ```
   extension_dir ="C:\PHP\extensions"
   ```

2. Download the ZIP package from http://www.php.net/downloads.php.

   The Windows installer package does not include extensions.

3. Copy `php_soap.dll` and `php_openssl.dll` from the package to the extensions directory.

4. In the extension section of `php.ini`, add a reference to the DLLs:

   ```
   extension=php_soap.dll

   extension=php_openssl.dll
   ```

## Installing PHP on Linux

### Context

If you are running PHP on Linux, your PHP application requires these three extensions: `SOAP`, `OpenSSL`, and `libxml`.

1. To find out if your existing PHP application already has these extensions, run this command:

   ```
   php -i | grep configure
   ```

   - If the output shows these three extensions, skip Steps 2 and 3, and proceed to Building and Running the Sample:

     ```
     --enable-soap
     --with-openssl
     --with-libxml-dir
     ```

- If the output does not show all three extensions, proceed to Step 2.

***IMPORTANT:*** CyberSource is not responsible for build errors that you might encounter during Steps 2 and 3.

**2** To build your PHP application with the `SOAP`, `OpenSSL`, and `libxml` extensions, navigate to the directory where the PHP source was installed. Run the `configure` command with the three required extensions and any other extension previously included in your PHP application. For example:

```
./configure '--prefix=your_target_dir' '--enable-soap'
'--with-libxmldir=your_libxml_dir' '--with-openssl=your_openssl_dir'
```

**3** In the same directory, build and install your application.

```
make

make install
```

## Building and Running the Sample

**Context** To test your client, modify the variables in the sample files, and launch the application.

**1** In your sample PHP file, replace the variables with your own values:

```
MERCHANT_ID

TRANSACTION_KEY
```

Note that the URL for the CyberSource API (`WSDL_URL`) is set to the test environment and for a specific version of the API. Always use the most current version of the API.

**2** Run the script `php <`*`sample PHP file`*`>`.

In the reply file, you can see the result of the request and all of the fields that are returned.

## Modifying Your Script

After you configure and test your application, you can modify it as needed. Be sure to use separate transaction keys for the test and production environments.

- To access the test and production environments, use these values for `WSDL_URL` (line 7):
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com

    - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, update the version number in the URL.
- To add or delete API fields, modify the source code.

# Constructing SOAP with .NET 2.0 and WSE 3.0

This section describes how to construct SOAP messages to process transactions with CyberSource.

Before starting this process, download and install the required third-party software:

| Software Tested | Description |
|---|---|
| Windows XP Pro with SP2 | Operating system version tested. |
| Visual Studio 2005 | Includes .NET 2.0. |
| WSE 3.0 | Web Services Enhancements for Microsoft .NET, which is used to authenticate the user with the UsernameToken class. |

Test the client application with the sample code files available from https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_net_wse.

The sample files, `sample_wse30.vb` (VB) and `sample_wse30.cs` (C#), provide the code to process your transactions. To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, be sure to replace the generic values of the variables with your own.

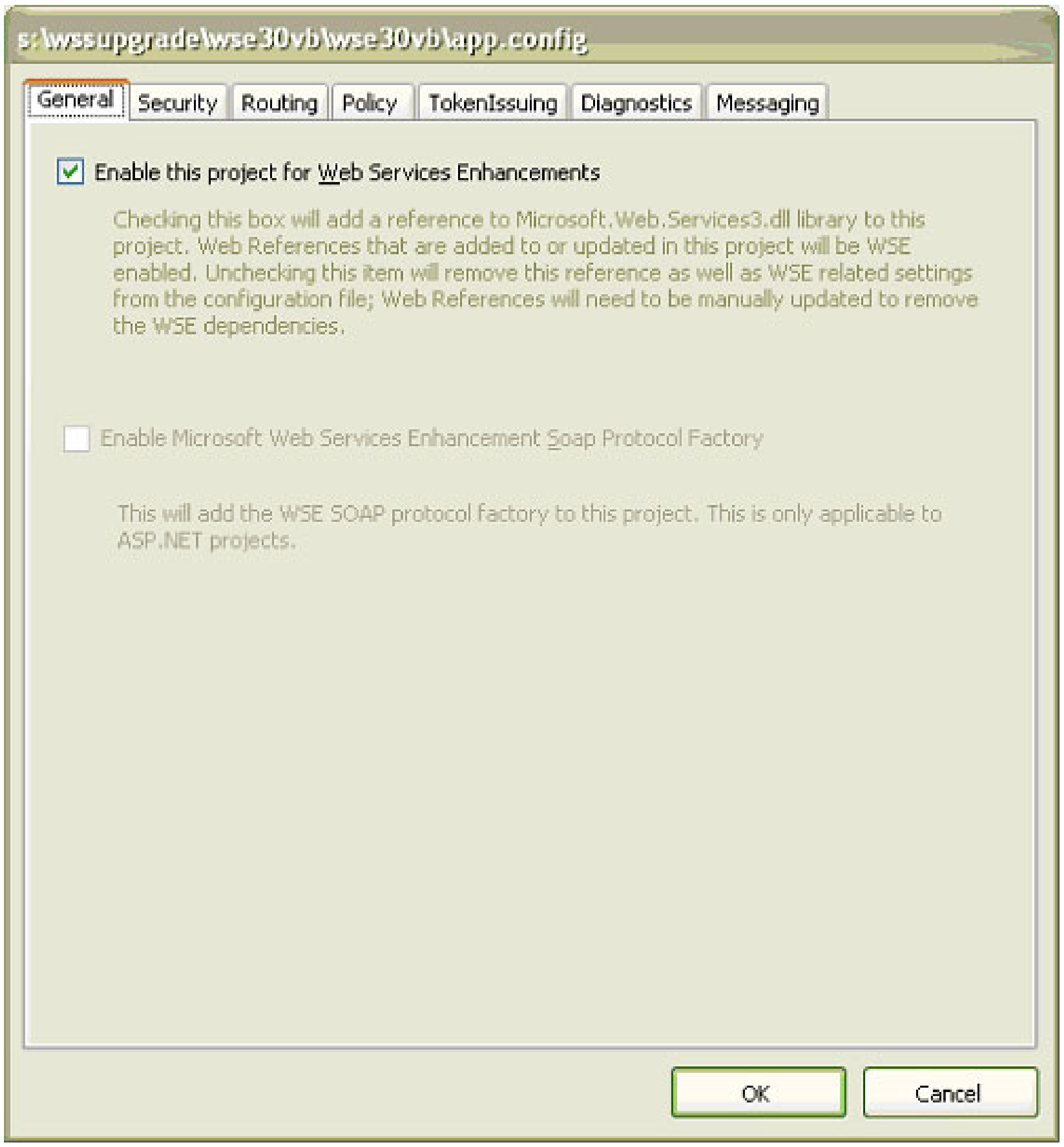
## Preparing Your Application

### Context

To test the sample code provided, you must have a console application.

1. Create a new application, or open your existing application in Visual Studio.
2. Right-click the project node and choose **WSE Settings 3.0**. The app config dialog box appears. If **WSE Settings 3.0** does *not* appear, proceed as follows:
    a. Reinstall WSE 3.0 by using the Add or Remove Programs menu.
    b. In the installer, select **Modify** and install the Visual Studio Tools option.
    c. Restart Visual Studio.
    d. Repeat Steps 1 and 2.

3 In the dialog box under the General tab, check **Enable this project for Web Services Enhancements**.



s:\wssupgrade\wse30vb\wse30vb\app.config

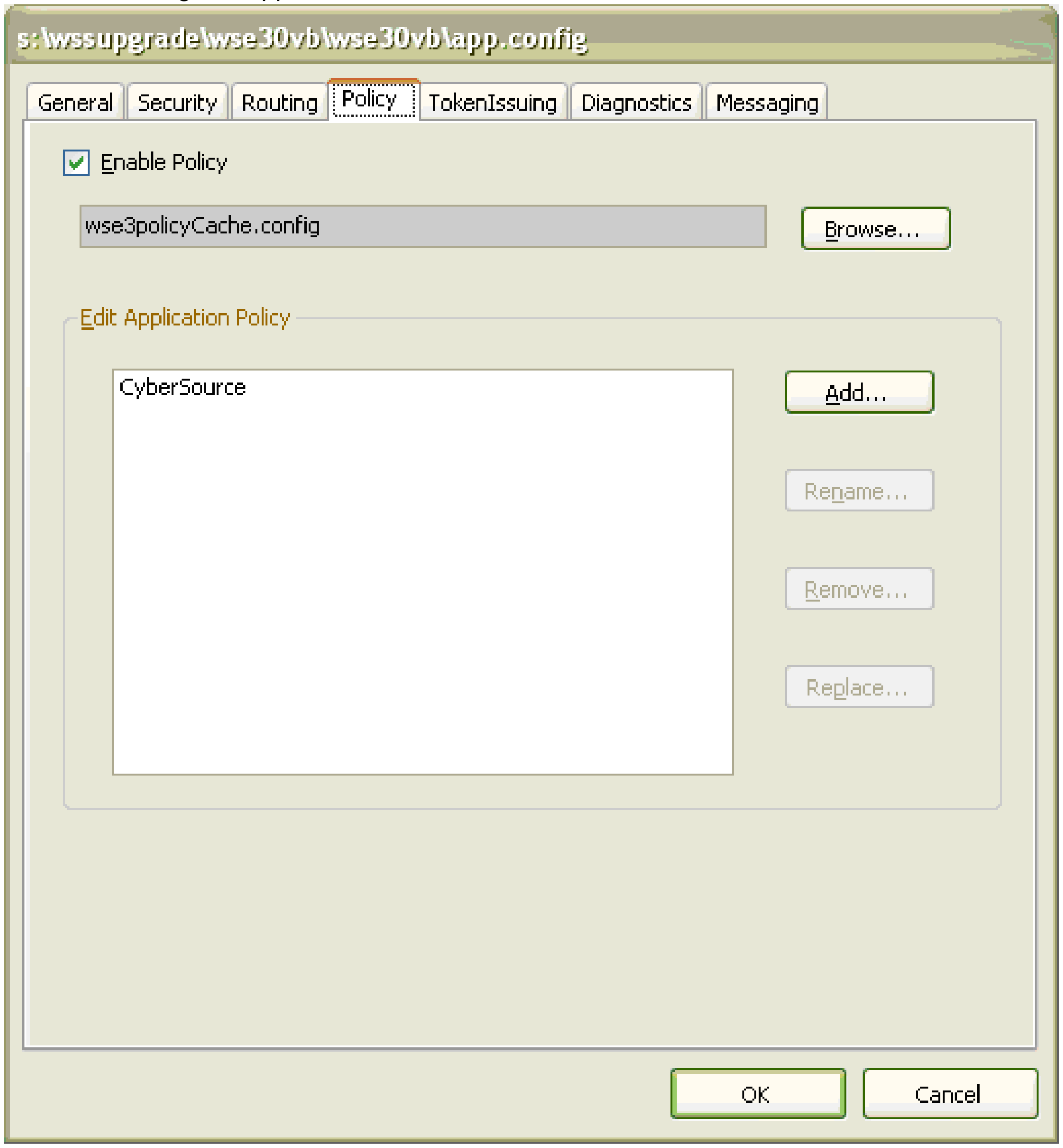
[✓] Enable this project for Web Services Enhancements

Checking this box will add a reference to Microsoft.Web.Services3.dll library to this project. Web References that are added to or updated in this project will be WSE enabled. Unchecking this item will remove this reference as well as WSE related settings from the configuration file; Web References will need to be manually updated to remove the WSE dependencies.

[ ] Enable Microsoft Web Services Enhancement Soap Protocol Factory

This will add the WSE SOAP protocol factory to this project. This is only applicable to ASP.NET projects.

4 Click the Policy tab and check **Enable Policy**. Click **Add**. The Add or Modify Policy Friendly Name dialog box appears.


s:\wssupgrade\wse30vb\wse30vb\app.config

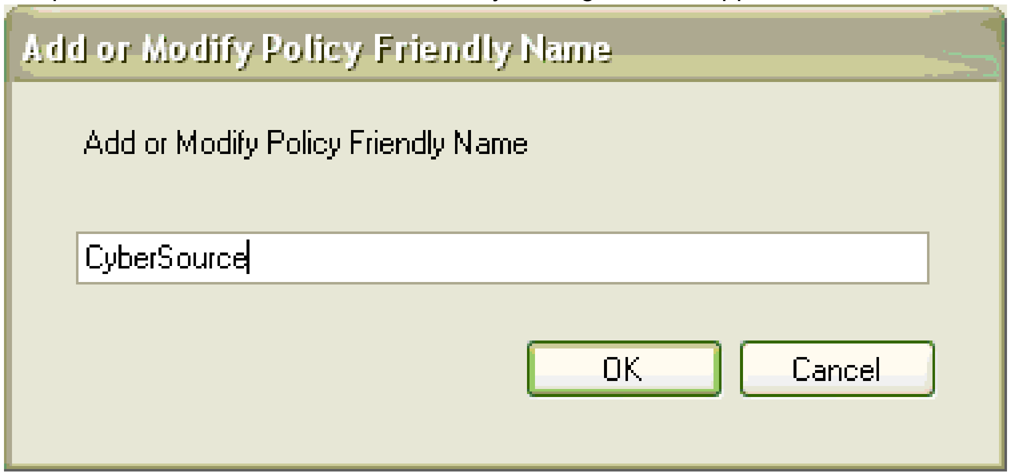
[✓] Enable Policy

wse3policyCache.config

Edit Application Policy

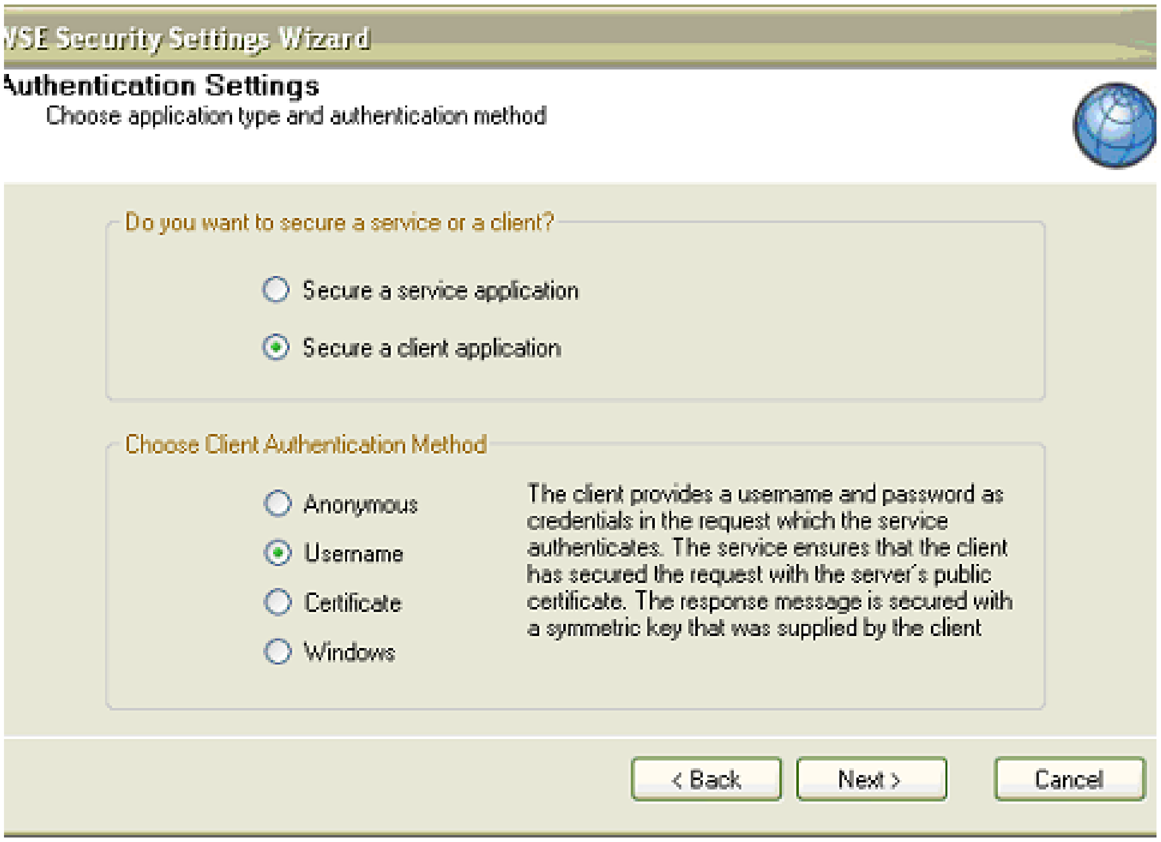
CyberSource

5 In the **Add or Modify Policy Friendly Name** field, enter **`CyberSource`**. If you use a name other than CyberSource, you must modify the value of the *`POLICY_NAME`* variable in the sample code. Click **OK**. The WSE Security Settings Wizard appears.


Add or Modify Policy Friendly Name
Add or Modify Policy Friendly Name
OK
Cancel

CyberSource

6 In the **Do you want to secure a service or a client** field, choose **Secure a client application**. In the **Choose Client Authentication Method** field, choose **Username**. Click **Next**.

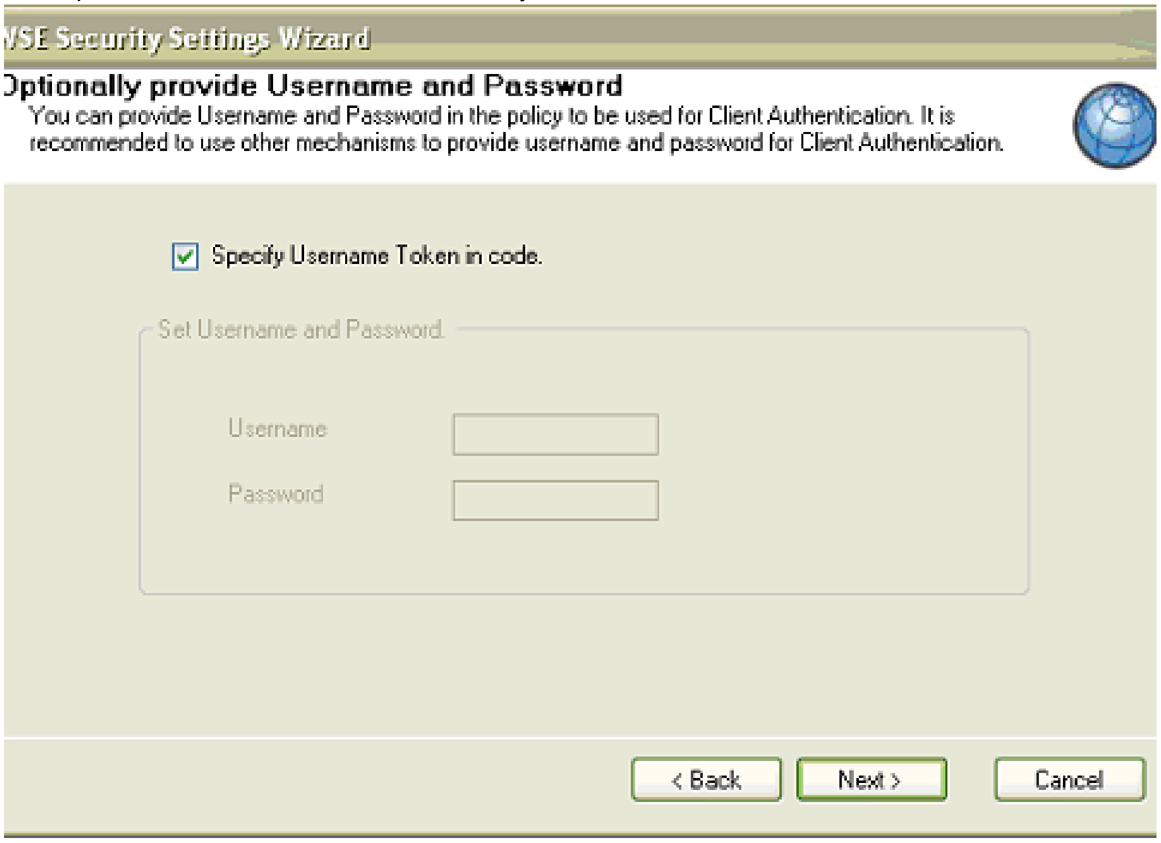

VSE Security Settings Wizard
Authentication Settings
Choose application type and authentication method
Do you want to secure a service or a client?
Choose Client Authentication Method
< Back
Next >
Cancel

Secure a service application [✓ Secure a client application]

Anonymous [✓ Username] Certificate Windows

The client provides a username and password as credentials in the request which the service authenticates. The service ensures that the client has secured the request with the server's public certificate. The response message is secured with a symmetric key that was supplied by the client

7 On the next page, **Specify Username Token in code** is checked by default. If you leave this default setting your password appears as plain text in the wse3policyCache.config policy. However, if you specify the username and password in the code, you can retrieve the password from a database or from any other source. Click **Next**.



VSE Security Settings Wizard

Optionally provide Username and Password

You can provide Username and Password in the policy to be used for Client Authentication. It is recommended to use other mechanisms to provide username and password for Client Authentication.

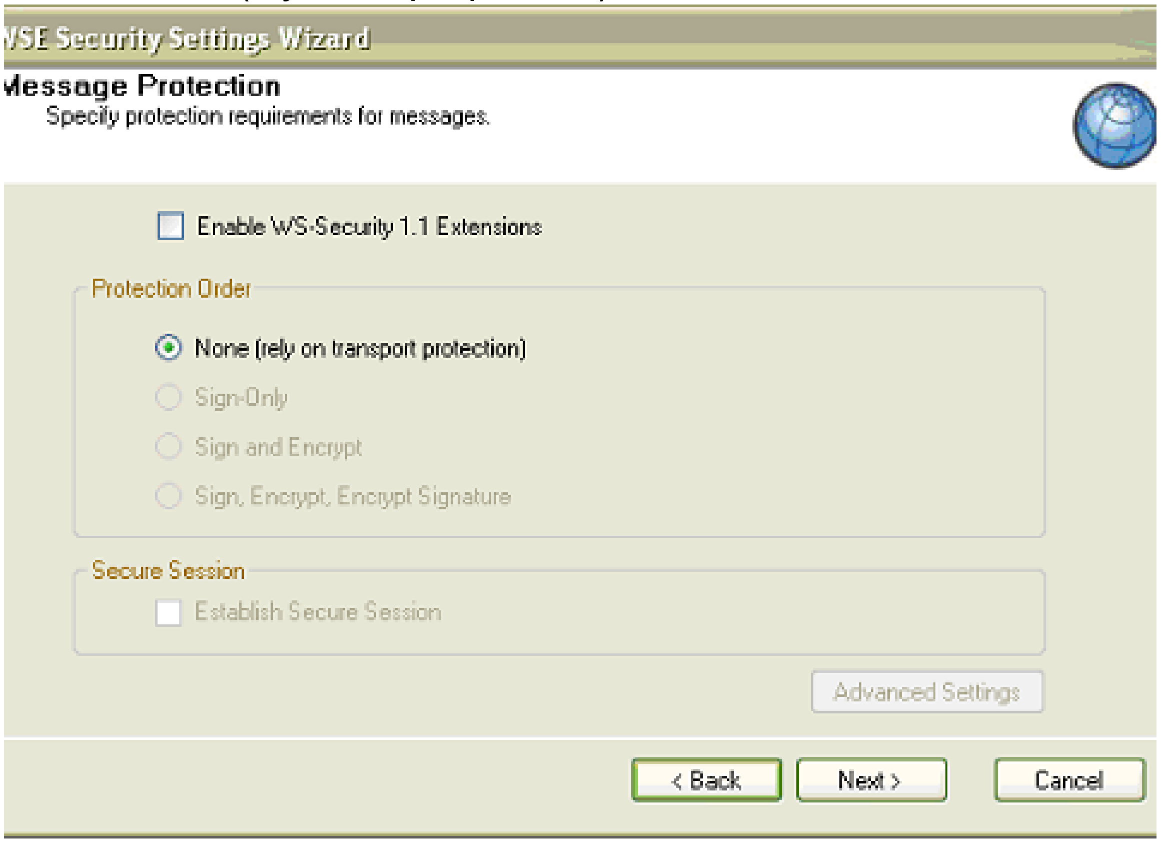
[✓] Specify Username Token in code.

Set Username and Password

Username

Password

8 Uncheck **Enabled WS-Security 1.1 Extensions**. Unchecking this option automatically selects **None (rely on transport protection)**. Click **Next**.


WSE Security Settings Wizard

Message Protection

Specify protection requirements for messages.

☐ Enable WS-Security 1.1 Extensions

Protection Order

● None (rely on transport protection)

○ Sign-Only

○ Sign and Encrypt

○ Sign, Encrypt, Encrypt Signature

Secure Session

☐ Establish Secure Session

9 To exit the wizard, click **Finish**.

10 To save your changes, click **OK**.

# Sending Requests to CyberSource

## Context

To add a Web reference to CyberSource, follow these steps:

1 In the Solution Explorer, right-click the project node and choose **Add Web Reference**.

2 In the Add Web Reference dialog box, in the **URL** field, enter the URL for CyberSource's Web Service:

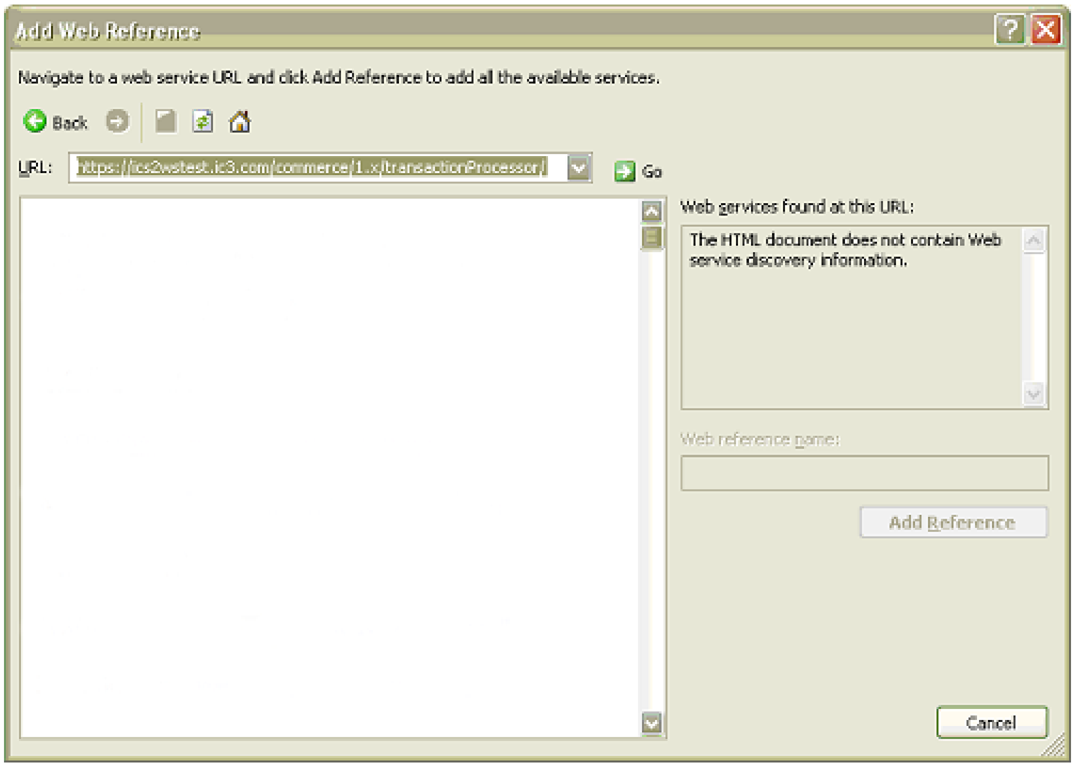
- Test environment: https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor
- Production environment: https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor
- Production environment in India: https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor

---

***IMPORTANT:*** You must use separate transaction keys for the production and test environments.

---

3 Click the **Go** button beside the **URL** field.



Navigate to a web service URL and click Add Reference to add all the available services.

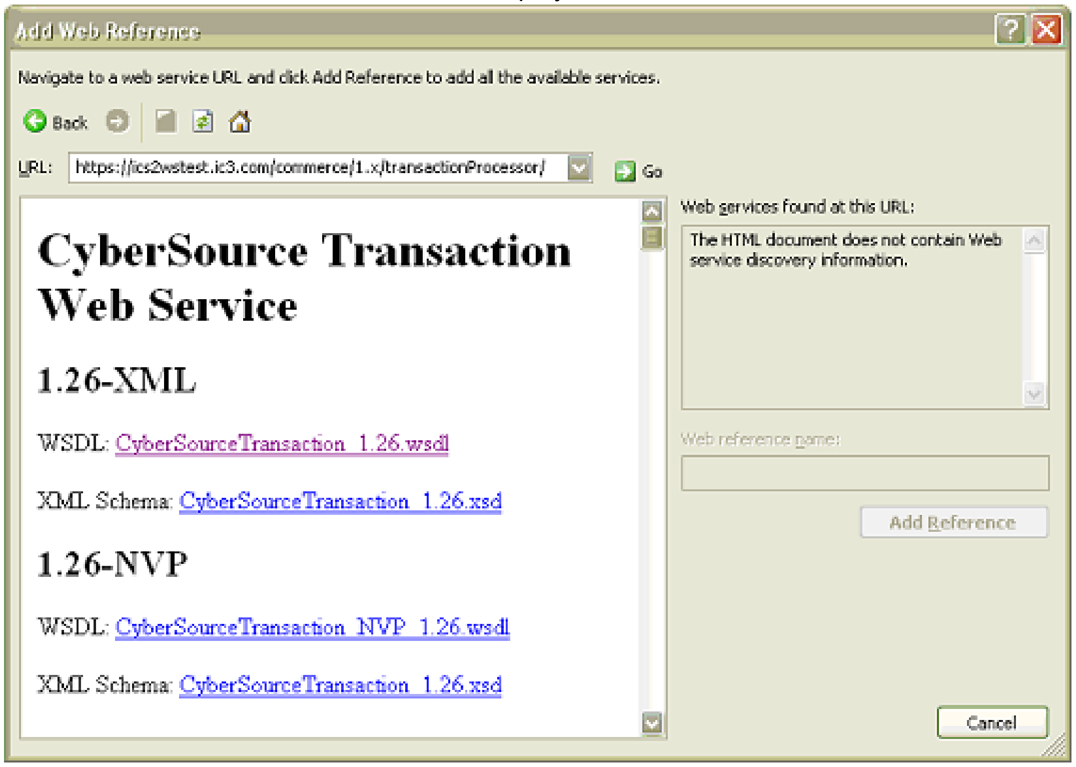
URL: https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor/

Web services found at this URL:

The HTML document does not contain Web service discovery information.

Web reference name:

The available server API versions are displayed.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor/

CyberSource Transaction Web Service

1.26-XML

WSDL: CyberSourceTransaction_1.26.wsdl

XML Schema: CyberSourceTransaction_1.26.xsd

1.26-NVP

WSDL: CyberSourceTransaction_NVP_1.26.wsdl

XML Schema: CyberSourceTransaction_1.26.xsd

Web services found at this URL:

The HTML document does not contain Web service discovery information.

Web reference name:

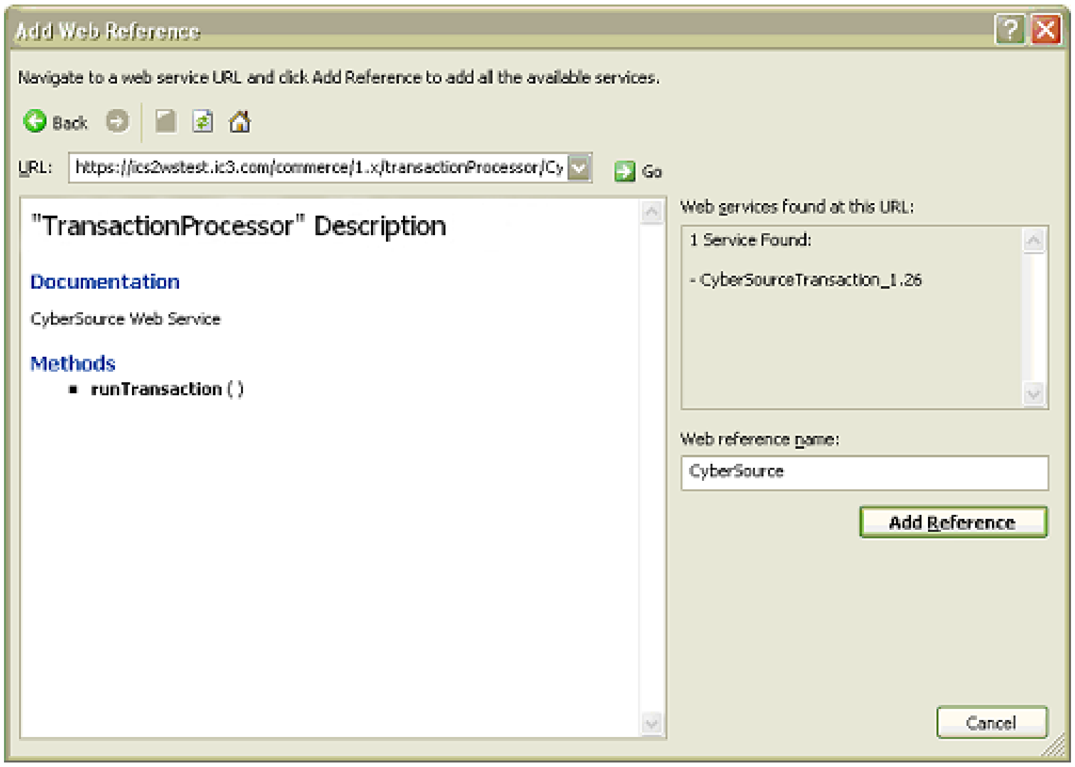
4 To display the content of the most current WSDL, click the top link.

The next step, Step 5, is not required to run the application. However, if you decide to use a name other than *CyberSource*, use a name that is not associated with a particular server so that you can easily change between the test and production servers. In addition, change the import statement in the sample code using the following format, where `myapp` is your project default name space:

```
import myapp.com.ic3.ics2wstesta;
```

5 Change the name that is displayed in the **Web reference name** field to **`CyberSource`**. The Web reference name is used in the name space that you need to import in your code. For example, if your project's default name space is `myapp`, and you set the Web reference name to CyberSource, you will import `myapp.CyberSource`. Depending on the URL that you

entered in Step 2, the default Web Reference Name in the field on the right side of the window is either `com.ic3.ics2wstesta` or `com.ic3.ics2wsa`.


Add Web Reference
Navigate to a web service URL and click Add Reference to add all the available services.
Back
URL:
Go
Web services found at this URL:
Web reference name:
Add Reference
Cancel

https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor/Cy

"TransactionProcessor" Description

Documentation

CyberSource Web Service

Methods

- runTransaction ( )

1 Service Found:

- CyberSourceTransaction_1.26

CyberSource

**6** Click **Add Reference**.

This generates the proxy classes that process the request and the reply.

## Building the Sample and Testing the Net Client

### Context

To test your client, follow these steps.

**1** In `sample_wse.cs` or `sample_wse.vb`, modify the values of the following variables:

- *`MERCHANT_ID`*
- *`TRANSACTION_KEY`*
- *`LIB_VERSION`*
- *`POLICY_NAME`*

**2** Add the sample file to your application.

**3** Launch the application.

The reply file contains the request result and all returned fields. When client testing is finished, write the code to use the client application.

## Modifying the .NET Client and Code

After you configure and test your application, you can modify it as needed:

***IMPORTANT:*** You must use separate transaction keys for the test and production environments

- To alternate between the test and production environments, change the host in the URL in your application or Web configuration file:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, do the following:
  - In the Solution Explorer, under the Web References node, click the CyberSource web reference.
  - Update the value of the Web Reference URL to the version that you want to use, such as 1.86 in this example: https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor/CyberSourceTransaction_1.86.wsdl
  - Rebuild your application.
- To add or delete API fields, modify your source code.

# Constructing SOAP with .NET 3.0 (WCF) and Later Versions

This section describes how to construct SOAP messages with .NET 3.0 and later to process transactions with CyberSource.

Before starting this process, download and install the required third-party software:

| Software Tested | Description |
|---|---|
| Windows XP Pro with SP2 | Operating system version tested. |
| Visual Studio 2005 | Includes .NET 2.0. |
| Microsoft Windows SDK | Software Development Kit that contains necessary tools, such as `svcutil.exe`. |
| .NET Framework 3.0 and later Redistributable Package | Includes the Windows Communication Foundation. ***IMPORTANT:*** After installing the software, you must reboot your computer to ensure that `svcutil` is recognized as a shell command. |

Test the client application with the sample code files available from https://github.com/CyberSource/cybersource-soap-toolkit/blob/master/sample_wcf.cs.

To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, be sure that you replace the generic values of the variables with your own.

## Creating and Testing the .NET 3.0 Client Using Sample Code

### Context

To reach the .NET 3.0 (and later) command shell and create the client, follow these steps:

1. Go to **Start > All Programs > Microsoft Windows SDK > CMD Shell**.
2. Change directory (`cd`) to find the sample code (`sample_wcf.cs`).
3. Generate the proxy classes as follows:

```
svcutil /config:sample_wcf.exe.config
https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor/
CyberSourceTransaction_N.NN.wsdl
```

where

***N.NN* is the latest server API version. For the latest version, navigate to https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor**

Two files are generated:

> The`CyberSourceTransactionWS.cs` file contains the proxy classes.

> The `sample_wcf.exe.config` file is the configuration file for your application.

4 In `sample_wcf.exe.config`, change the security mode from `Transport` to `TransportWithMessageCredential`.

The security element should now read: `<security mode="TransportWithMessageCredential">`

5 To write the code to process your transactions, use `sample_wcf.cs`:

a Add your own values to *`MERCHANT_ID`* and *`TRANSACTION_KEY`*.

b Build the executable file as follows:

```
csc /out:sample_wcf.exe /target:exe
/reference:"C:\WINDOWS\Microsoft.NET\Framework\v3.0\Windows
Communication Foundation\System.ServiceModel.dll"
CyberSourceTransactionWS .cs sample_wcf.cs
```

```
csc /out:sample_wcf.exe /target:exe
/reference:"C:\WINDOWS\Microsoft.NET\Framework\v3.0\Windows
Communication Foundation\System.ServiceModel.dll"
CyberSourceTransactionWS .cs sample_wcf.cs
```

The `sample_wcf.exe` file is created.

6 Run the `sample_wcf.exe file`.

7 The reply file contains the request result and all returned fields. When client testing is finished, write the code to use the client application.

## Modifying the .NET 3.0 Client and Code

After you configure and test your application, you can modify it as needed:

***IMPORTANT:*** You must use different transaction keys for the test and production environments.

- To alternate between the test and production environments, change the host in the *`endpoint`* address in the configuration file:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, follow Steps 1-4 in the Creating and Testing the Client Using Sample Code.
- To add or delete API fields, modify the source code.

# Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

This section describes how to construct SOAP messages to process transactions with CyberSource.

Before starting this process, download and install the required third-party software. CyberSource tested these versions:

| Software Tested | Description |
|---|---|
| Windows XP Pro with SP2 | Operating system version tested. |
| gSOAP 2.7.9c | Soap toolkit. Download and unzip the latest win32 ZIP file from http://sourceforge.net/projects/gsoap2/ |
| OpenSSL 0.9.8d | You may use the pre-built package available from http://www.slproweb.com/products/Win32OpenSSL.html.<br>If you do, before installing the software, make a copy of `libeay32.dll` and `ssleay32.dll`, which are located in `c:\windows\system32`. Otherwise, these files are overwritten during installation. |
| Microsoft Visual Studio 2005 | Development environment tested. |

Test the client application with the sample code files available from https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_gsoap.

To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, be sure that you replace the generic values of the variables with your own.

## Generating the Windows Client Code

**Context** The pre-built `wsdl2h` tool does not support SSL, so you cannot point the tool directly to *https://ics2wstesta.ic3.com* or *https://ics2wsa.ic3.com* or in India *https://ics2wsa.in.ic3.com*.

1. Download the latest WSDL and XSD files to the same directory from the following URL: https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor or in India https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor
2. Save the files under the following names:

```
CyberSourceTransaction_1.26.wsdl
```

```
CyberSourceTransaction_1.26.xsd
```

3 To generate the header file, run the following script in the directory to which you downloaded the WSDL and XSD files:

```
gsoap_directory\bin\wsdl2h -t gsoap_directory\WS\WS-typemap.dat -s
-o cybersource.h CyberSourceTransaction_N.NN.wsdl
```

The `gsoap_directory` is the directory from which you extracted gSoap. ***N.NN*** is the version number of the WSDL file that you downloaded. This script creates the `cybersource.h` header file. You may change the name of the file; however, the following steps refer to the file as `cybersource.h`.

While you can safely ignore any warning messages, be sure that no two line items in your requests have the same ID.

4 In the `cybersource.h` header file, add the following line to the Import section:

```
#import "WS-Header.h"
```

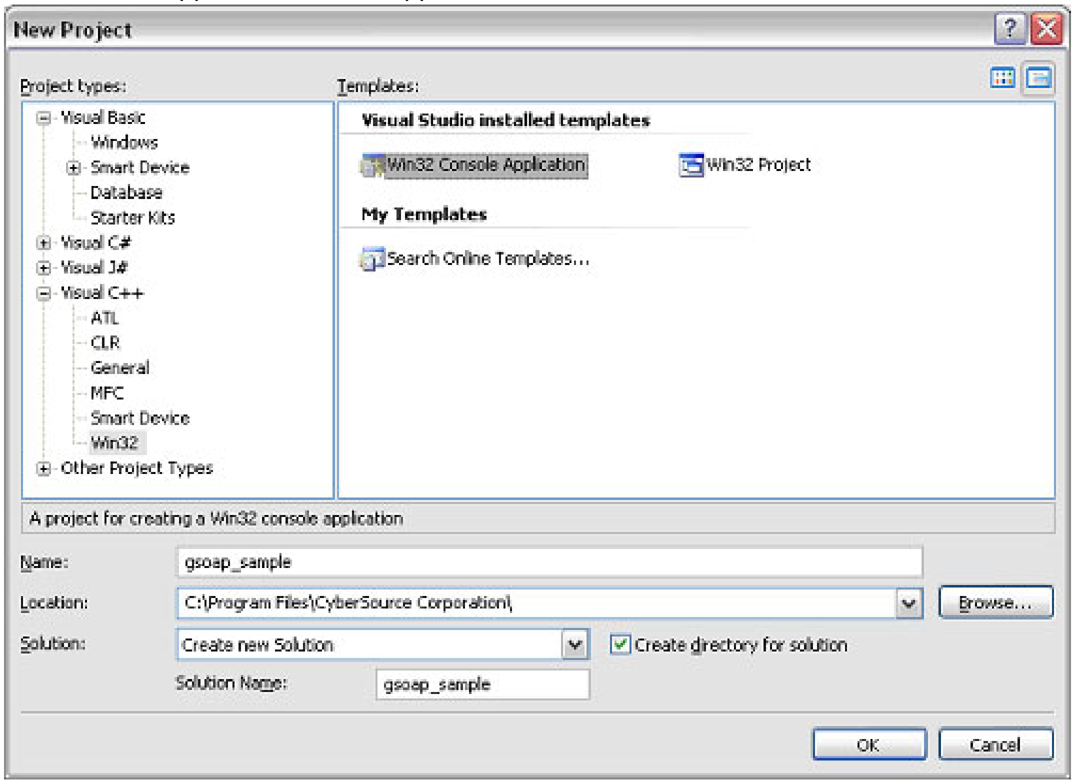
5 To generate the client source code, run this script:

```
gsoap_directory\bin\soapcpp2 -C -Igsoap_directory\import
cybersource.h
```

## Building the Windows Client

1. To create a non-CLR (Common Language Runtime) C++ project, open Visual Studio and choose **Project types > Visual C++ > Win32**.

2. Enter a name and location for the project. This sample uses a Win32 Console Application with `gsoap_sample` as the name of the solution. Click **OK**. When you start a new C++ project, the Win32 Application Wizard appears.



New Project

Project types: Visual Basic (Windows, Smart Device, Database, Starter Kits), Visual C#, Visual J#, Visual C++ (ATL, CLR, General, MFC, Smart Device, Win32), Other Project Types

Templates:

Visual Studio installed templates: Win32 Console Application, Win32 Project

My Templates: Search Online Templates...

A project for creating a Win32 console application

Name: gsoap_sample

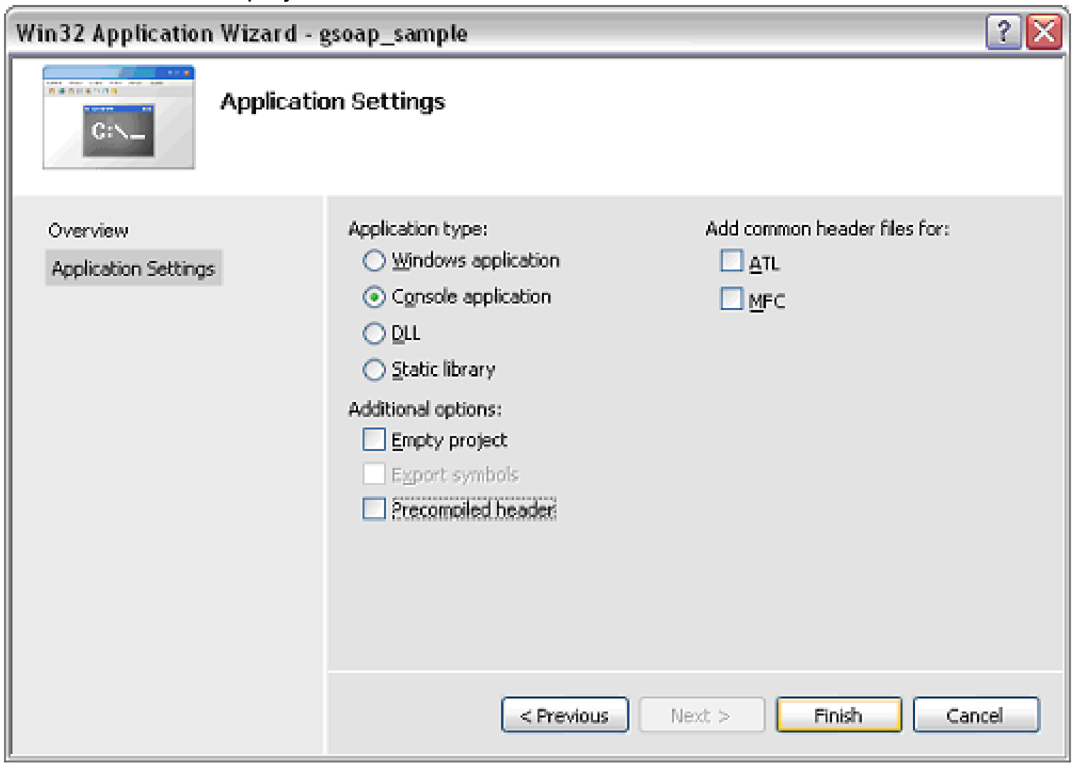
Location: C:\Program Files\CyberSource Corporation\

Solution: Create new Solution

Create directory for solution ✓

Solution Name: gsoap_sample

3 On the Application Settings page, uncheck the **Precompiled header** option and click **Finish**. Your new project is created.



Win32 Application Wizard - gsoap_sample

Application Settings

Overview

Application Settings

Application type:

- Windows application
- [✓ Console application]
- DLL
- Static library

Additional options:

- ☐ Empty project
- ☐ Export symbols
- ☐ Precompiled header

Add common header files for:

- ☐ ATL
- ☐ MFC

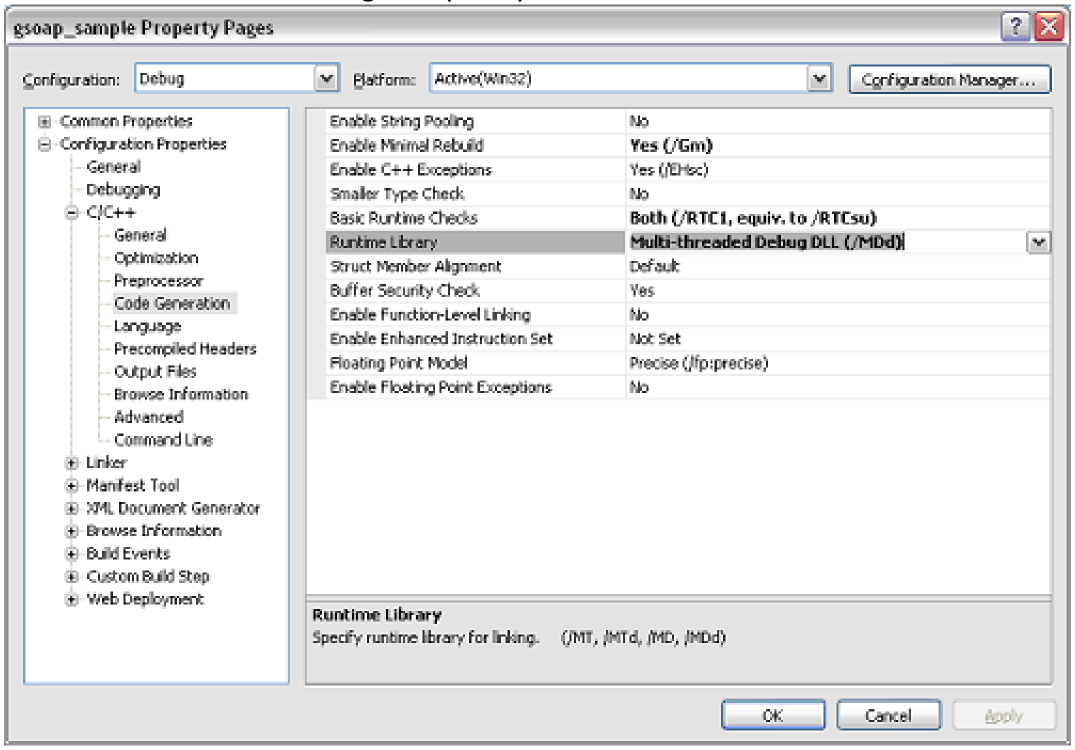
4 Choose **Project > gsoap_sample Properties**.

5 In the navigation pane, expand **Configuration Properties > C/C++** and click **Code Generation**.

6 Verify that the Debug and Release configurations are using the correct default runtime libraries:

**a** In the Configuration drop-down menu in the upper-left corner of the Property Pages, choose **Debug**.

**b** In the properties listed in the main panel, verify that Runtime Library reads **Multi-threaded debug DLL (/MDd)**.

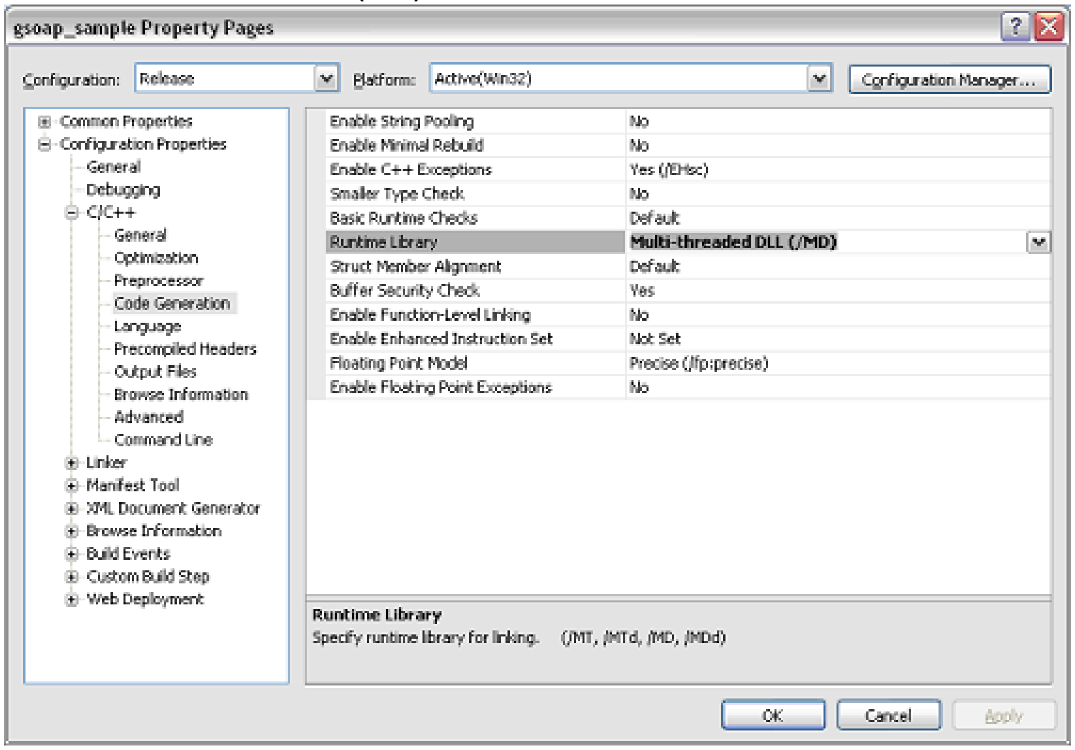


| | |
|---|---|
| Enable String Pooling | No |
| Enable Minimal Rebuild | **Yes (/Gm)** |
| Enable C++ Exceptions | Yes (/EHsc) |
| Smaller Type Check | No |
| Basic Runtime Checks | **Both (/RTC1, equiv. to /RTCsu)** |
| Runtime Library | **Multi-threaded Debug DLL (/MDd)** |
| Struct Member Alignment | Default |
| Buffer Security Check | Yes |
| Enable Function-Level Linking | No |
| Enable Enhanced Instruction Set | Not Set |
| Floating Point Model | Precise (/fp:precise) |
| Enable Floating Point Exceptions | No |

**Runtime Library**
Specify runtime library for linking. (/MT, /MTd, /MD, /MDd)

**c** Return to the Configuration drop-down menu in the upper-left corner and choose **Release**.

**d** In the properties displayed in the main panel, verify that Runtime Library reads **Multi-threaded DLL (/MD)**.


gsoap_sample Property Pages

Configuration: Release   Platform: Active(Win32)   Configuration Manager...

- Common Properties
- Configuration Properties
  - General
  - Debugging
  - C/C++
    - General
    - Optimization
    - Preprocessor
    - Code Generation
    - Language
    - Precompiled Headers
    - Output Files
    - Browse Information
    - Advanced
    - Command Line
  - Linker
  - Manifest Tool
  - XML Document Generator
  - Browse Information
  - Build Events
  - Custom Build Step
  - Web Deployment

| | |
|---|---|
| Enable String Pooling | No |
| Enable Minimal Rebuild | No |
| Enable C++ Exceptions | Yes (/EHsc) |
| Smaller Type Check | No |
| Basic Runtime Checks | Default |
| Runtime Library | Multi-threaded DLL (/MD) |
| Struct Member Alignment | Default |
| Buffer Security Check | Yes |
| Enable Function-Level Linking | No |
| Enable Enhanced Instruction Set | Not Set |
| Floating Point Model | Precise (/fp:precise) |
| Enable Floating Point Exceptions | No |

**Runtime Library**
Specify runtime library for linking. (/MT, /MTd, /MD, /MDd)

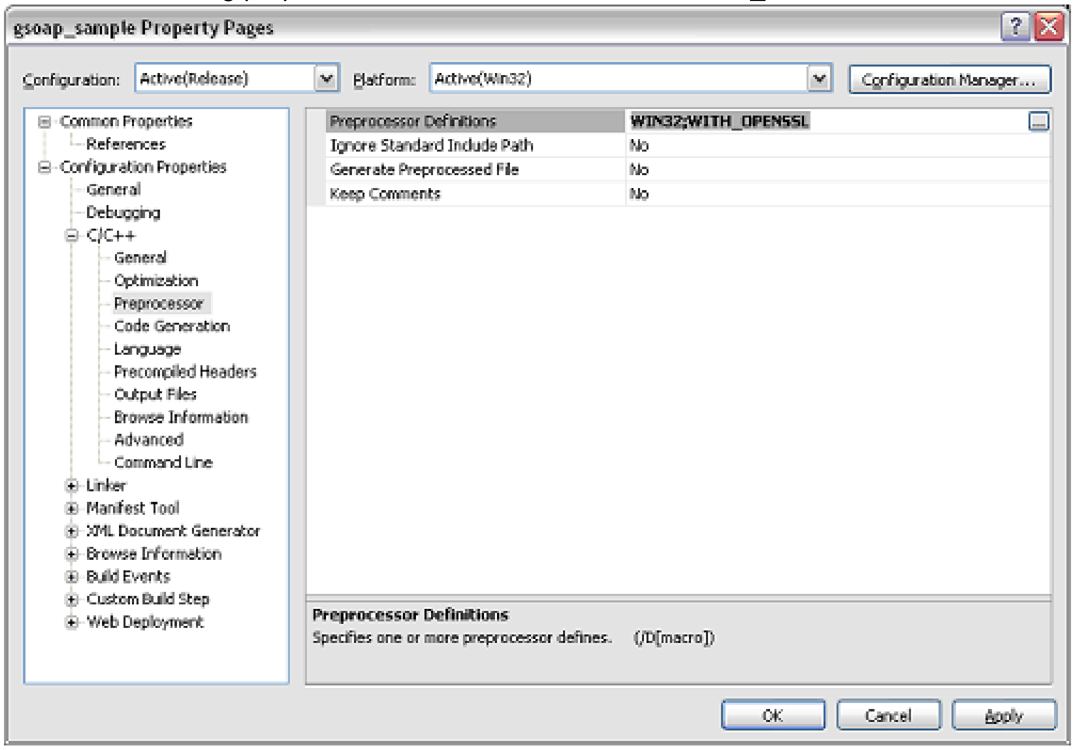
**e** Click **OK**. You are returned to the project.

**7** In the project, replace the `gsoap_sample.cpp` file, included in the project, with the files in the table below.

Before adding the source files to the plugin directory, you must change their file extensions from `.c` to `.cpp`.

| | |
|---|---|
| CyberSource sample | sample.cpp |
| Generated by gSOAP | soapC.cpp<br>soapClient.cpp |
| Included in gSOAP package | gsoap_directory\dom.cpp*<br>gsoap_directory\stdsoap2.cpp<br>gsoap_directory\mod_gsoap\gsoap_win\wininet\gsoapWinInet.cpp<br>gsoap_directory\plugin\smdevp.cpp gsoap_directory\plugin\wsseapi.cpp *<br>gsoap_directory is the directory in which you downloaded and extracted gSOAP |

8 Add the following preprocessor definitions: **WIN32** and **WITH_OPENSSL**



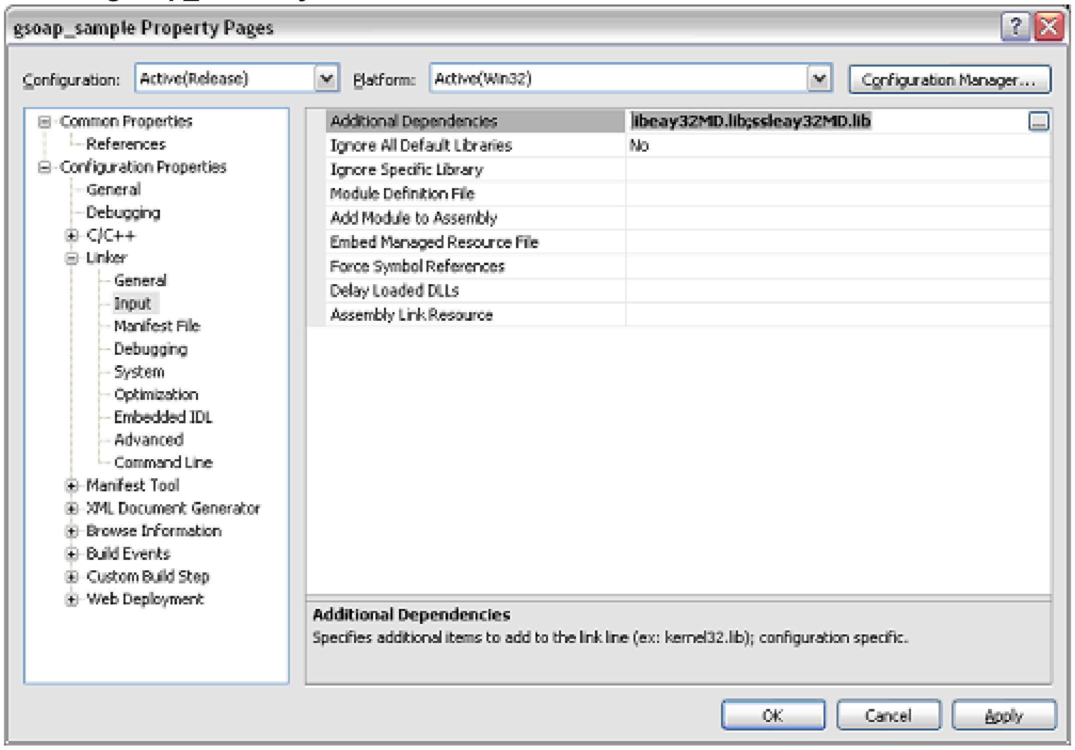
| | |
|---|---|
| Preprocessor Definitions | WIN32;WITH_OPENSSL |
| Ignore Standard Include Path | No |
| Generate Preprocessed File | No |
| Keep Comments | No |

**Preprocessor Definitions**
Specifies one or more preprocessor defines. (/D[macro])

9 In the sidebar, navigate to **Configuration Properties > C/C++ > General**, and add the following directories to your project's Additional Include Directories:

- **gsoap_directory**
- **gsoap_directory**\mod_gsoap\gsoap_win\wininet
- **gsoap_directory**\plugin

- **gsoap_directory**\include


gsoap_sample Property Pages

Configuration: Active(Release)   Platform: Active(Win32)

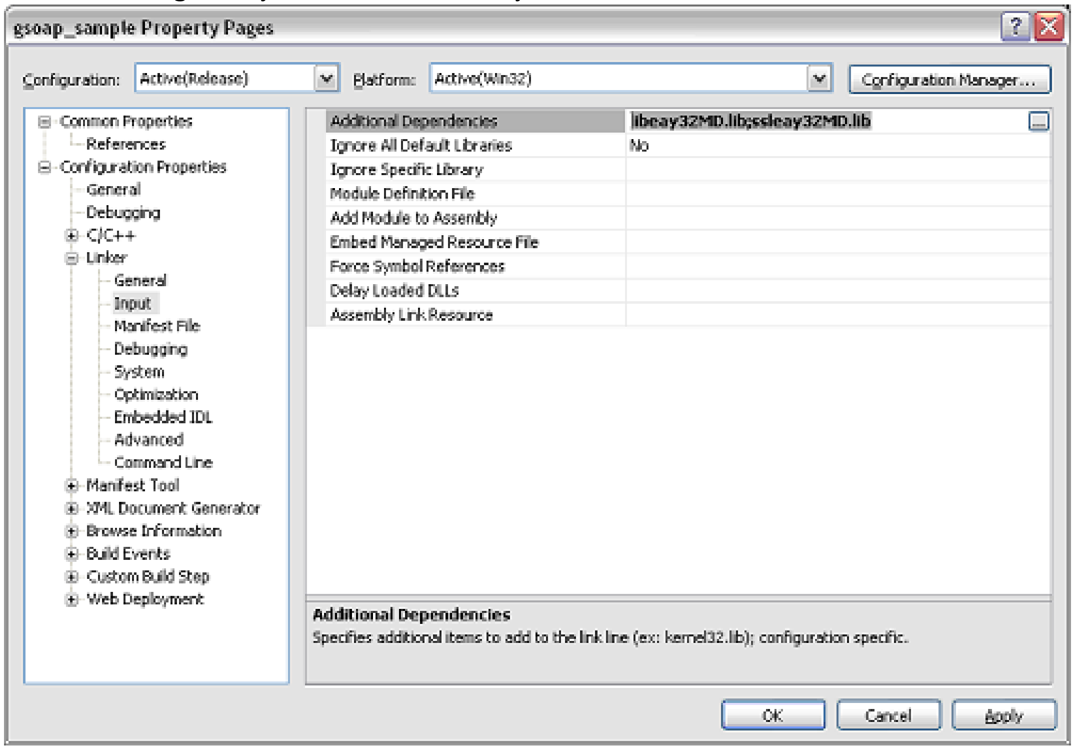
| | |
|---|---|
| Additional Dependencies | libeay32MD.lib;ssleay32MD.lib |
| Ignore All Default Libraries | No |
| Ignore Specific Library | |
| Module Definition File | |
| Add Module to Assembly | |
| Embed Managed Resource File | |
| Force Symbol References | |
| Delay Loaded DLLs | |
| Assembly Link Resource | |

**Additional Dependencies**
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

10 In the sidebar, navigate to **Configuration Properties > Linker > Input**, and add the following libraries; in the upper Configuration field's drop-down menu, choose each option in turn:

- **Release**: libeay32MD.lib and ssleay32MD.lib

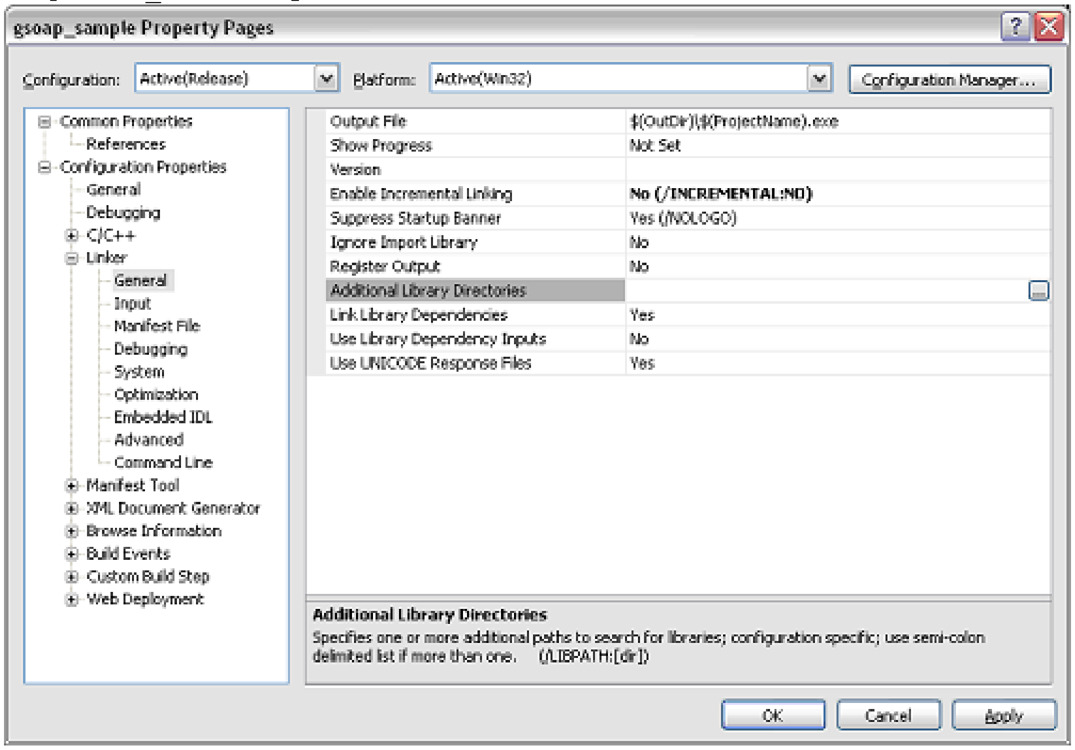
- **Debug**: libeay32MDd.lib and ssleay32MDd.lib



| | |
|---|---|
| Additional Dependencies | libeay32MD.lib;ssleay32MD.lib |
| Ignore All Default Libraries | No |
| Ignore Specific Library | |
| Module Definition File | |
| Add Module to Assembly | |
| Embed Managed Resource File | |
| Force Symbol References | |
| Delay Loaded DLLs | |
| Assembly Link Resource | |

**Additional Dependencies**
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.

11 Add the following directory to your project's Additional Library Directories: `openssl_directory\lib\VC`



| | |
|---|---|
| Output File | $(OutDir)\$(ProjectName).exe |
| Show Progress | Not Set |
| Version | |
| Enable Incremental Linking | No (/INCREMENTAL:NO) |
| Suppress Startup Banner | Yes (/NOLOGO) |
| Ignore Import Library | No |
| Register Output | No |
| Additional Library Directories | |
| Link Library Dependencies | Yes |
| Use Library Dependency Inputs | No |
| Use UNICODE Response Files | Yes |

**Additional Library Directories**
Specifies one or more additional paths to search for libraries; configuration specific; use semi-colon delimited list if more than one. (/LIBPATH:[dir])

12 In `gsoap_directory\stdsoap2.cpp`, find the following calls:

`ASN1_item_d2i`

`meth->d2i`

13 In each call, cast the second parameter `(&data)` as `const unsigned char **`. The calls now read:

ext_data = ASN1_item_d2i(NULL, (const unsigned char **) &data, ext->value->length, ASN1_ITEM_ptr(meth->it)); ext_data = meth->d2i(NULL, (const unsigned char **) &data, ext->value->length);

14 If you see the following compiler error in the `stdsoap2.cpp` file,

```
error C2440: '=' : cannot convert from 'const char *' to 'char *
```

cast the first parameter as `char *` as follows:

```
t = strchr((char *) s, ',');
```

15 Inside `soap_wsse_get_BinarySecurityTokenX509`, find `d2i_X509` in `gsoap_directory\plugin\wsseapi.cpp`.

16 To the existing cast, add `const` as follows: cert = d2i_X509(NULL, (const unsigned char**)&data, size); You can now test the client.

## Building the Sample and Testing the Windows Client

**Context** To test your client, follow these steps.

***IMPORTANT:*** You must use separate transaction keys for test and production environments.

1 In `sample.cpp`, modify the values of the following variables:

- *`MERCHANT_ID`*
- *`TRANSACTION_KEY`*
- *`SERVER_URL`*
- *`LIB_VERSION ENVIRONMENT`*

Use *`LIB_VERSION`* only if you are using a different version of gSOAP.

2 Run the client.

The code included in gSOAP causes the Visual C++ compiler to generate several warnings. You can safely ignore these warnings.

The reply file contains the request result and all returned fields. When client testing is finished, write the code to use the client application.

## Modifying the Windows Client and Code

After you configure and test your application, you can modify it as needed:

***IMPORTANT:*** You must use separate transaction keys for the test and production environments.

- To alternate between the test and production environments, change the URL assigned to `service.endpoint`. In the sample file, set the *`SERVER_URL`* variable to the appropriate value:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, rebuild your client by following the steps in Generating the Windows Client Code.
- To add or delete API fields, modify your source code.

# Constructing SOAP with C++ and gSOAP 2.7.9e for Linux

This section describes how to construct SOAP messages to process transactions using Linux.

Before starting this process, download and install the required third-party software. These versions were tested:

| Software Tested | Description |
|---|---|
| Linux Kernel 2.6 | Operating system version tested. |
| gSOAP 2.7.9e | SOAP toolkit. You can download it fromhttp://sourceforge.net/project/showfiles.php?group_id=52781 |
| OpenSSL 0.9.8 | Current version of the toolkit implementing SSL.<br>Most Linux installations already contain this package. If your package does not or if it has an old version, download the source from http://www.openssl.org. |
| gcc 4.1.2 | C/C++ compiler tested. |

The `sample.cpp` file provides the code to process your transactions.

Test the client application with the sample code files from https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_gsoap.

To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, be sure that you replace the generic values of the variables with your own.

`Makefile` provides targets to easily create and build the sample client.

## Preparing the Linux Development Environment

1. Download the latest gSOAP package for Linux.
2. Open the package by running this command:

```
tar xvfz gsoap_linux_2.7.9e.tar.gz
```

3 At the same level as the `gSOAP` directory created in the previous step, create these items with the appropriate command:

| Item | Description | Command: |
|---|---|---|
| `client` | Directory for the client-related files (`Makefile` and `sample.cpp`) | `mkdir client` |
| `gsoap` | Symbolic link to your gSOAP directory | `ln -s <gsoap_path> gsoap`where *<gsoap_path>* is the path to the `gSOAP` directory that you created in Step 2. |

4 In the `gsoap/stdsoap2.cpp` file, find the following calls:

`ASN1_item_d2i` (occurs once)

`meth-d2i` (occurs twice)

5 In each call, cast the second parameter (`&data`) as *const unsigned char* **.

The calls should now read:

```
ext_data = ASN1_item_d2i(NULL, (const unsigned char **) &data ...
```

```
ext_data = meth->d2i(NULL, (const unsigned char **) &data ...
```

6 In the `gsoap/plugin/wsseapi.c` file, find `cert = d2i_X509`.

7 Add `const` to the second argument's cast as follows:

```
cert = d2i_X509(NULL, (const unsigned char**)&data, size);
```

## Generating the Linux Client Code

### Context

The pre-built `wsdl2h` tool does not support SSL, so you cannot point the tool directly to https://ics2wstesta.ic3.com or https://ics2wsa.ic3.com, or https://ics2wsa.in.ic3.com in India.

1 Download the latest WSDL and XSD files to the `client` directory from either of these URLs:

https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor

https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor

In India: https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor

2 To ensure that `Makefile` finds the WSDL file that you downloaded in the previous step, rename this file as `CyberSourceTransaction.wsdl` by removing the version number.

---

***IMPORTANT:*** Do not rename the XSD file.

3 Run the `make header` script.

The following warning appears, which you can safely ignore. However, make sure that no two line items in your requests have the same ID.

```
Warning: element 'xsd:unique' at level 2 was not recognized and will be ignored.
```

4 In the newly generated `cybersource.h header` file, add the following line to the Import section:

```
#import "WS-Header.h"
```

5 Run the `make source` script.

## Building the Sample and Testing the Linux Client

1 In the `sample.cpp` file, modify the values of the following variables:

- *`MERCHANT_ID`*
- *`TRANSACTION_KEY`*
- *`SERVER_URL`*
- *`LIB_VERSION`* (if using a different gSOAP version)
- *`ENVIRONMENT`*

Do not use `LIB_VERSION` variable unless you are using a different version of gSOAP.

2 Run the `make` **`cybsdemo`** script. You can safely ignore the warning messages. `cybsdemo` is now ready to use. The reply file contains the request result and all returned fields. When client testing is finished, write the code to use the client application.

3 Run the sample by executing:

```
./cybsdemo
```

## Modifying Linux Client and Code

### Context

After your application is configured and tested, you can modify it as needed:

***IMPORTANT:*** You must use separate transaction keys for the test and production environments.

- To alternate between the test and production environments, change the URL assigned to the service.endpoint. In the sample file, set the *SERVER_URL* variable to the appropriate value:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, rebuild your client by following the steps in Generating the Linux Client Code and Building the Sample and Testing the Linux Client.
- To add or delete API fields, modify your source code.

# Constructing SOAP with C++ and gSOAP 2.7.9d for Mac OS X

This section describes how to construct SOAP messages to process transactions with CyberSource.

Before starting this process, download and install the required third-party software. CyberSource tested these versions:

| Software Tested | Description |
|---|---|
| Mac OS X | Operating system version tested. |
| gSOAP 2.7.9d | SOAP toolkit<br>Download and unzip the latest Mac OS X package from http://sourceforge.net/projects/gsoap2/ |
| openssl 0.9.7 | OpenSSL version that is part of Mac OS X. |
| gcc 4.0.1 | C/C++ compiler tested. |

The `sample.cpp` file provides the code to process your transactions.

Test the client application with the sample code files from https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_gsoap.

To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, be sure to replace the generic values of the variables with your own values.

## Preparing the Development Environment

1. Download and unzip the latest gSOAP package for Mac OS X.
2. Open the package by running this command:

   ```
   tar xvfz gsoap_macosx_S2.7.9d.tar.g
   ```

3. Under the same parent directory, create these items with the appropriate command:

| Item | Description | Command: |
|---|---|---|
| `client` | Directory for the client-related files (`Makefile` and `sample.cpp`) | `mkdir client` |
| `gsoap` | Symbolic link to your gSOAP directory | gsoap -> gsoap-macosx-2.7 |

## Generating the Mac Client Code

**Context** The pre-built `wsdl2h` tool does not support SSL, so you cannot point the tool directly to *https://ics2wstesta.ic3.com* or *https://ics2wsa.ic3.com*, or *https://ics2wsa.in.ic3.com* in India.

1 Download the latest WSDL and XSD files to the `client` directory from either of the following URLs:

https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor

https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor

In India: https://ics2wsa.in.ic3.com/commerce/1.x/transactionProcessor

2 To ensure that the `Makefile` file finds the WSDL file that you downloaded in the previous step, rename it as `CyberSourceTransaction.wsdl` by removing the version number.

***IMPORTANT:*** Do not rename the XSD file.

3 Run the `make header` script.

You will receive warning messages, which you can safely ignore. However, ensure that no two line items in your requests have the same ID.

4 In the newly generated the `cybersource.h` header file, add the following line to the Import section:

```
#import "WS-Header.h"
```

5 Run the `make source` script.

## Building the Sample and Testing the Mac Client

1. In the `sample.cpp` file, modify the values of the following variables:

   - *`MERCHANT_ID`*
   - *`TRANSACTION_KEY`*
   - *`SERVER_URL`*
   - *`LIB_VERSION`*
   - *`ENVIRONMENT`*

   Do not use the *`LIB_VERSION`* variable unless you are using a different version of gSOAP.

2. Run the `make cybsdemo` script. Ignore the warning messages. `cybsdemo` is now ready to use.

3. Run the sample by executing:

   ```
   ./cybsdemo
   ```

   The reply file contains the request result and all returned fields. When client testing is finished, write the code to use the client application.

## Modifying the Mac Client and Code

### Context

After your application is configured and tested, you can modify it as needed:

***IMPORTANT:*** You must use separate transaction keys for the test and production environments.

- To alternate between the test and production environments, change the URL assigned to the service.endpoint. In the sample file, set the *`SERVER_URL`* variable to the appropriate value:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, rebuild your client by following the steps in Generating the Mac Client Code and Building the Sample and Testing the Mac Client.
- To add or delete API fields, modify your source code.

# Constructing SOAP with Apache Axis and WSS4J

This section describes how to construct SOAP messages to process transactions with CyberSource using Apache Axis and WSS4J.

Before starting this process, download and install the required third-party software. CyberSource tested these versions:

| Software Tested | Description |
|---|---|
| • Windows XP Professional with SP<br>• Linux<br>• Solaris | Operating systems tested |
| JDK 1.5 | Java Development Kit |
| Apache Axis 1.4 | SOAP toolkit<br>Download and unzip the latest package from http://ws.apache.org/axis |
| Apache WSS4J 1.5.1 | WS-Security package<br>Download and unzip the latest package from http://ws.apache.org/wss4 |
| Apache XML Security 1.4.0 | XML security package<br>Download the latest package from http://santuario.apache.org/download.html and extract `xmlsec-N.N.N.jar` |
| activation.jar | JDK JavaBeans Activation Framework add-on that you can download from http://www.oracle.com/technetwork/java/jaf11-139815.html |
| mail.jar | JDK Java Mail add-on that you can download from http://java.sun.com/products/javamail/ |

The `Sample.java:` sample file provides the code to process your transactions.

Test the client application with the sample code files available from https://github.com/CyberSource/cybersource-soap-toolkit/tree/master/sample_axis_wss4j
.

To help understand and use the code, the files contain many comments and a sample card authorization. Before using the files, replace the generic values of the variables with your own.

The `SamplePWCallback.java:` sample file is a Password Callback Handler, which provides the password to WSS4J.

The `SampleDeploy.wsdd:` file is a sample deployment descriptor file used by WSS4J.

## Generating and Building the Stubs

1. From each of the following packages, add these items to your classpath:
   - The current directory (.)
   - These files:

| Package | Files |
|---|---|
| Apache Axis | • `axis.jar`<br>• `commons-discovery-0.2.jar`<br>• `commons-logging-1.0.4.jar`<br>• `jaxrpc.jar`<br>• `log4j-1.2.8.jar`<br>• `saaj.jar`<br>• `wsdl4j-1.5.1.jar` |
| Apache WSS4J | `wss4j-1.5.1.jar` |
| Apache XML Security | `xmlsec-1.4.0.jar` |
| JDK JavaBeans Activation Framework | `activation.jar` |
| JDK Java Mail | `mail.jar` |

2. From a command prompt, go to the directory in which you downloaded the CyberSource sample code `Sample.java`.

3. To generate the stubs, execute this command without line breaks:

   ```
   java org.apache.axis.wsdl.WSDL2Java -p com.cybersource.stub
   https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor/Cyber
   SourceTransaction_N.NN.wsdl
   ```

   where:

   `com.cybersource.stub` is the package name that will be used for the generated classes. You can choose a different package name if you wish. However, the rest of the steps and the sample code refer to this value.

   ***N.NN*** is the CyberSource API version. Find the latest version here: https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor

4. To compile the source code, execute this command:

   ```
   javac com/cybersource/stub/*.java
   ```

5. Create a jar file by using the compiled classes:

   ```
   jar cf cybersource.jar com/cybersource/stub/*.class
   ```

6. Add the newly created `cybersource.jar` file to your classpath.

## Building the Sample and Testing the Client

### Context

To build the sample and test your client, modify the variables in the sample files, and run the application.

1. In the `Sample.java` file, modify the values of *`MERCHANT_ID`*.
2. In the `SamplePWCallback.java` file, modify the value of *`TRANSACTION_KEY`*.
3. Compile the samples as follows:

   ```
   javac Sample.java SamplePWCallback.java
   ```

4. Run the sample as follows:

   ```
   java -Daxis.ClientConfigFile=SampleDeploy.wsdd Sample
   ```

   The reply file contains the request result and all returned fields. When testing the client is finished, write the code to use the client application.

## Modifying the Client and Code

After your application is configured and tested, you can modify it as needed:

***IMPORTANT:*** You must use separate transaction keys for the test and production environments.

- To alternate between the test and production environments, set *`SERVER_URL`* to the appropriate value:
  - Test environment: ics2wstesta.ic3.com
  - Production environment: ics2wsa.ic3.com
  - Production environment in India: ics2wsa.in.ic3.com
- To update the version of the CyberSource API, rebuild the client by following the steps in Generating and Building the Stubs.
- To add or delete API fields, modify your source code.