CyberSource SOAP Toolkits for Web Services

Developer Guide

January 2016

CyberSource®

the power of payment

CyberSource Corporation HQ | P.O. Box 8999 | San Francisco, CA 94128-8999 | Phone: 800-530-9095

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 at http://www.cybersource.com/support.

Copyright

Restricted Rights Legends

Trademarks

Contents

Recent Revisions to This Document

Release	Changes
January 2016	Fixed the URL for the perl sample code. See page 33.
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*.
October 2013	This revision contains only editorial changes and no technical updates.
August 2013	This revision contains only editorial changes and no technical updates.
January 2013	Added important note about the new requirement for separate transaction security keys for CyberSource production and test environments. For more information, see "Transaction Key," page 13.

About This Guide

Audience

This guide is written for application developers who want to configure SOAP toolkits that are used with CyberSource Web Services.

Using the SOAP toolkits requires programming skills in one of the following programming languages:

- ASP
- C, C++
- Java/Cold Fusion
- .NET
- Perl
- PHP

Purpose

This guide describes tasks you must complete to configure and use the CyberSource SOAP toolkits.

Scope

This guide describes how to use the SOAP toolkits to build and test an API client for all supported programming languages. It does not describe how to implement CyberSource services. For information about how to use CyberSource APIs to implement CyberSource services, see "Related Documents," page 9.

Conventions

Note and Important Statements

Note	A *Note* contains helpful suggestions or references to material not contained in the document.

Important	An *Important* statement contains information essential to successfully completing a task or learning a concept.

Text and Command Conventions

Convention	Usage
bold	▪ Field and service names; for example: Include the **ics_applications** field. ▪ Items that you are instructed to act upon; for example: Click **Save**.
italic	▪ Filenames and pathnames. For example: Add the filter definition and mapping to your *web.xml* file. ▪ Placeholder variables for which you supply particular values.
`monospace`	▪ XML elements. ▪ Code examples and samples. ▪ Text that you enter in an API environment; for example: Set the **davService_run** field to `true`.

Related Documents

CyberSource Services Documentation

This guide (*SOAP Toolkits for Web Services Developer Guide*) contains information about how to configure the SOAP toolkits.

In contrast, CyberSource services documentation contains information about how to:

- Determine what to put in requests sent to CyberSource.
- Interpret what is contained in the reply from CyberSource.

Each type of CyberSource service has associated documentation:

- *Getting Started with CyberSource Advanced for the Simple Order API* (PDF | HTML)
- *Credit Card Services Using the Simple Order API* (PDF | HTML)
- *Electronic Check Services for the Simple Order API* (PDF | HTML)

Refer to the Support Center for complete CyberSource technical documentation:

http://www.cybersource.com/support_center/support_documentation

Customer Support

For support information about any CyberSource service, visit the Support Center:

http://www.cybersource.com/support

CHAPTER 1

Configuring SOAP Toolkits for Web Services

SOAP toolkits are designed 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 protocol with SSL encryption
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 header of the SOAP message

Note

The CyberSource servers do not support persistent HTTP connections.

This guide provides instructions for developers who need to configure SOAP toolkits that will be used with CyberSource Web Services.

Supported Toolkits

The toolkits are available in many platform options, including open source.

Important CyberSource has tested and will support only the toolkits listed below. Although you can implement a toolkit on a platform that is not tested or supported, CyberSource cannot guarantee that you will be able to use such an implementation with the Web Services.

Toolkits	Supported Platforms
PHP 5.2.1	Windows, Linux, Solaris
.NET .NET 2.0 and WSE 3.0 .NET 3.0 (WCF) and later	Windows
Perl 5.8.8 and SOAP::Lite 0.69	Windows, Linux, Solaris
C++ with gSOAP gSOAP 2.7.9c for Windows gSOAP 2.7.9e for Linux gSOAP 2.7.9d for Mac OS X	Windows, Linux, Mac OS
Java with Apache Axis and WSS4J	Windows, Linux, Solaris

Important 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 PABP requirements, see www.visa.com/cisp. Follow these guidelines:

- 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 to CyberSource personal and account information, such as customers' names, addresses, card or check account numbers, and card verification numbers.

Destination URLs for SOAP Messages

The latest version of the API is located at:

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

When constructing your SOAP messages, use these target URLs:

- Test environment:

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

- Production environment:

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

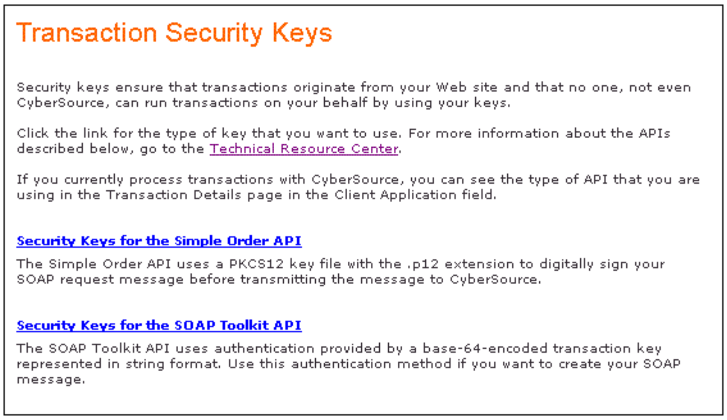
Note `1.x` is not a placeholder for the version number but an integral part of the URL.

Transaction Key

Before you can send requests for ICS (Internet Commerce Suite) 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.

Step 1 In the navigation pane of the Business Center, click **Account Management** > **Transaction Security Keys**.



Transaction Security Keys

Security keys ensure that transactions originate from your Web site and that no one, not even CyberSource, can run transactions on your behalf by using your keys.

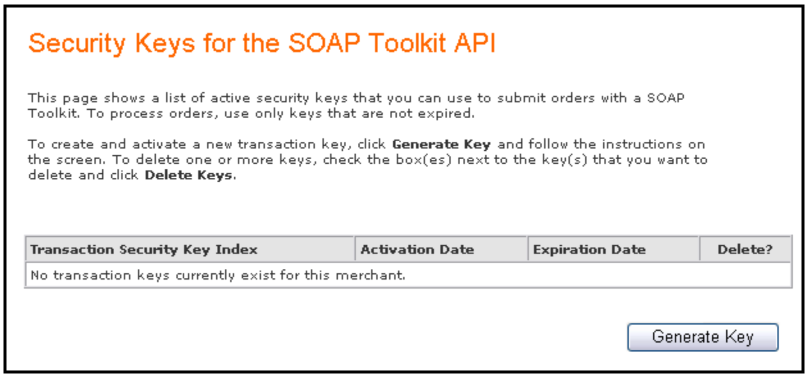
Click the link for the type of key that you want to use. For more information about the APIs described below, go to the Technical Resource Center.

If you currently process transactions with CyberSource, you can see the type of API that you are using in the Transaction Details page in the Client Application field.

Security Keys for the Simple Order API
The Simple Order API uses a PKCS12 key file with the .p12 extension to digitally sign your SOAP request message before transmitting the message to CyberSource.

Security Keys for the SOAP Toolkit API
The SOAP Toolkit API uses authentication provided by a base-64-encoded transaction key represented in string format. Use this authentication method if you want to create your SOAP message.

Step 2 Click **Security Keys for the SOAP Toolkit API**.



Security Keys for the SOAP Toolkit API

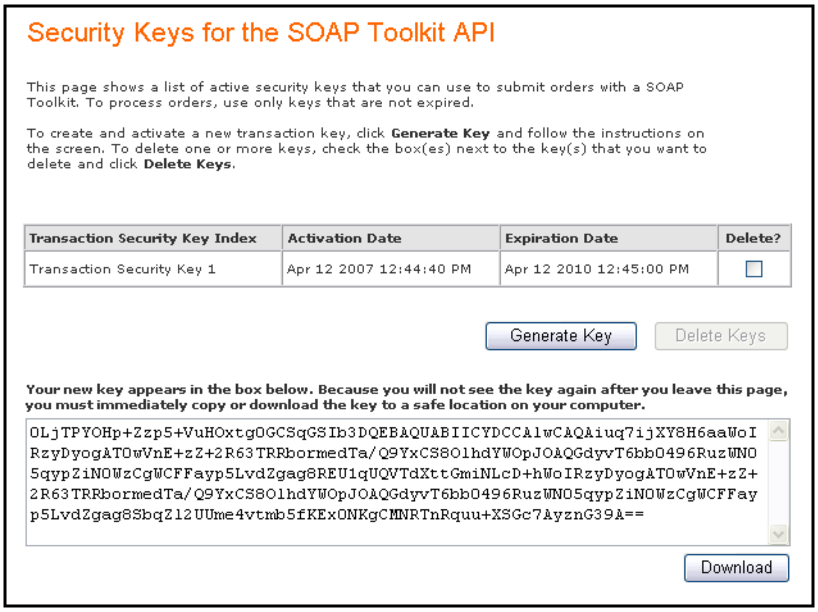
This page shows a list of active security keys that you can use to submit orders with a SOAP Toolkit. To process orders, use only keys that are not expired.

To create and activate a new transaction key, click **Generate Key** and follow the instructions on the screen. To delete one or more keys, check the box(es) next to the key(s) that you want to delete and click **Delete Keys**.

Transaction Security Key Index	Activation Date	Expiration Date	Delete?
No transaction keys currently exist for this merchant.			

Step 3 Click **Generate Key**.

Your new key appears immediately below the table. Because the key content will disappear as soon as you leave the Web page, save your key now. If you forget to do so, you will need to delete the old key before creating a new one.



Security Keys for the SOAP Toolkit API

This page shows a list of active security keys that you can use to submit orders with a SOAP Toolkit. To process orders, use only keys that are not expired.

To create and activate a new transaction key, click **Generate Key** and follow the instructions on the screen. To delete one or more keys, check the box(es) next to the key(s) that you want to delete and click **Delete Keys**.

Transaction Security Key Index	Activation Date	Expiration Date	Delete?
Transaction Security Key 1	Apr 12 2007 12:44:40 PM	Apr 12 2010 12:45:00 PM	☐

Your new key appears in the box below. Because you will not see the key again after you leave this page, you must immediately copy or download the key to a safe location on your computer.

```
OLjTPYOHp+Zzp5+VuHOxtgOGCSqGSIb3DQEBAQUABIICYDCCAlwCAQAiuq7ijXY8H6aaWoI
RzyDyogATOwVnE+zZ+2R63TRRbormedTa/Q9YxCS8OlhdYWOpJOAQGdyvT6bbO496RuzWNO
5qypZiNOWzCgWCFFayp5LvdZgag8REU1qUQVTdXttGmiNLcD+hWoIRzyDyogATOwVnE+zZ+
2R63TRRbormedTa/Q9YxCS8OlhdYWOpJOAQGdyvT6bbO496RuzWNO5qypZiNOWzCgWCFFay
p5LvdZgag8SbqZl2UUme4vtmb5fKExONKgCMNRTnRquu+XSGc7AyznG39A==
```

Step 4 Click **Download**.

Save the key in a secure location.

Important Be sure to use separate locations for the test and production environment transaction keys. Be careful not to overwrite a key in the wrong directory.

Sample SOAP Message

If you want to use this example, make sure to replace ***N.NN*** with the current API version.

```
<?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-secext-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-profile-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>
```

CHAPTER 2

Constructing SOAP with PHP 5.2.1

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

Before starting, you need to download and install the third-party software. CyberSource tested these versions:

Software Tested	Description
▪ Linux Kernel 2.4 ▪ Windows XP Pro with SP2 ▪ 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 WS Security, you can add the required header elements for UsernameToken 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:

Sample code	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 Make 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.

Preparing your PHP Installation

Follow the instructions in the section that applies to your operating system.

Windows Operating System

Your PHP application requires at least these extensions: `SOAP` and `OpenSSL`.

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

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

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

Note The extensions are not included in the Windows installer package.

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

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

```
extension=php_soap.dll
extension=php_openssl.dll
```

Linux Operating System

Your PHP application requires at least these extensions: `SOAP`, `OpenSSL`, and `libxml`.

Step 1 To find out if your existing PHP application meets these requirements, run this command:

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

- If the output shows these three extensions, skip steps 2 and 3, and proceed to the next section:

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

  or

- If the output does not show all three extensions, proceed to step 2 and 3.

Note CyberSource is not responsible for build errors that you may encounter during Steps 2 and 3.

Step 2 To build your PHP application with the `SOAP`, `OpenSSL`, and `libxml` extensions, navigate to the directory where the PHP source was installed and 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'
```

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

```
make
make install
```

Building and Running the Sample

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

Step 1 In your sample PHP file, replace the placeholder values with your own:

```
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.

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

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

Modifying your Script

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 as follows the value of `WSDL_URL` (line 7) that is passed to the constructor of `ExtendedClient`:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, update the version number in the URL.
- To add or delete API fields, modify your source code.

CHAPTER 3

Constructing SOAP with .NET 2.0 and WSE 3.0

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

Before starting, you need to 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 following CyberSource sample code:

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

Preparing Your Application

To test the sample code provided, you must have a Console Application.

Step 1 Create a new application or open your existing application in Visual Studio.

Step 2 Right-click the project node and select **WSE Settings 3.0**.

If WSE Settings 3.0 does not appear, proceed as follows:

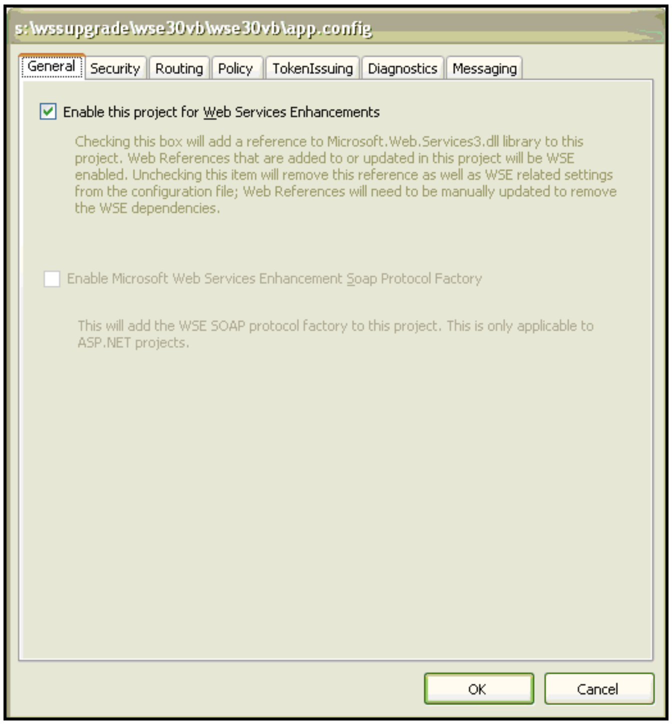
a Reinstall WSE 3.0 by using the Add or Remove Programs menu.

b In the installer, select Modify and install the Visual Studio Tools option.

c When done, restart Visual Studio.

d Repeat steps 1 and 2.

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



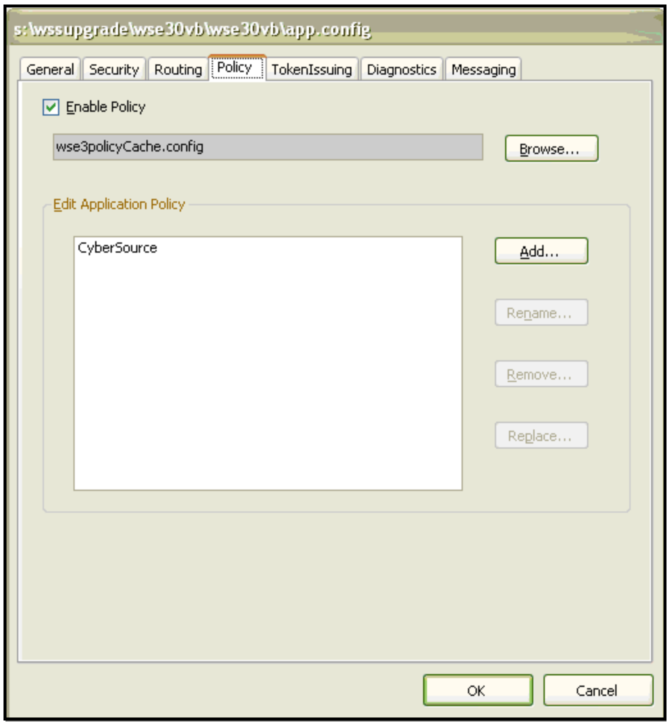
[✓] Enable this project for Web Services Enhancements

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.

Step 4 Click the Policy tab and check **Enable Policy**.



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

☑ Enable Policy

wse3policyCache.config

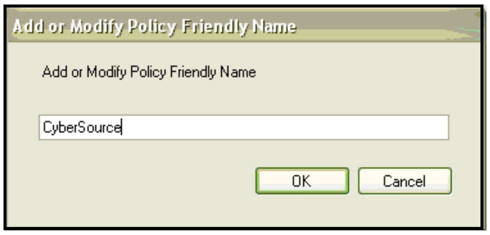
Edit Application Policy

CyberSource

Step 5 Click **Add**.

Step 6 In the field, enter CyberSource.

If you use a name other than CyberSource, you will need to modify the value of the variable `POLICY_NAME` in the sample code.



Add or Modify Policy Friendly Name

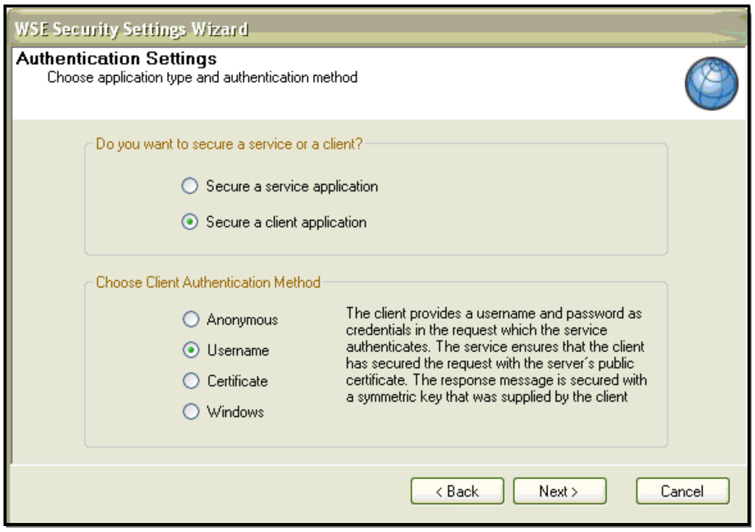
Add or Modify Policy Friendly Name

CyberSource

The WSE Security Settings Wizard appears.

Step 7 Click **Next**.

Step 8 Click **Secure a client application** and **Username**.



WSE Security Settings Wizard

Authentication Settings
Choose application type and authentication method

Do you want to secure a service or a client?

○ Secure a service application
● Secure a client application

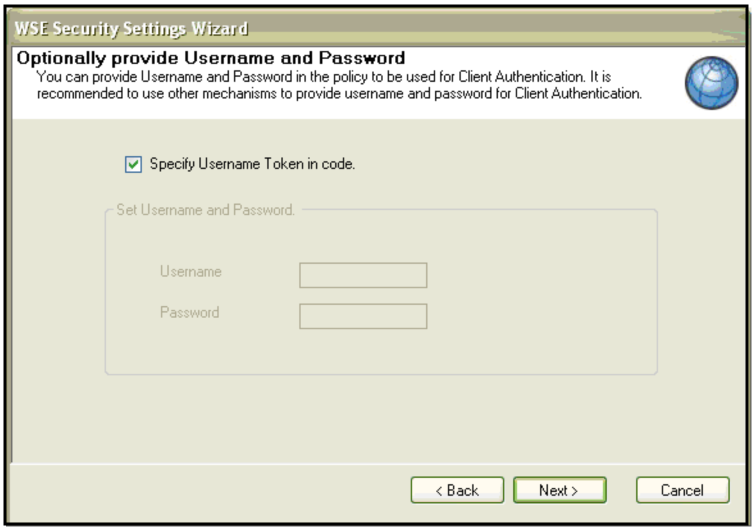
Choose Client Authentication Method

○ 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

Step 9 Click **Next**.

On the next page, **Specify Username Token in code** is checked by default. CyberSource recommends that you leave the setting as is. Otherwise, your password appears as plain text in `wse3policyCache.config`. On the other hand, if you specify the user name and password in the code, you can retrieve the password from a database or from any other source.



WSE 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.

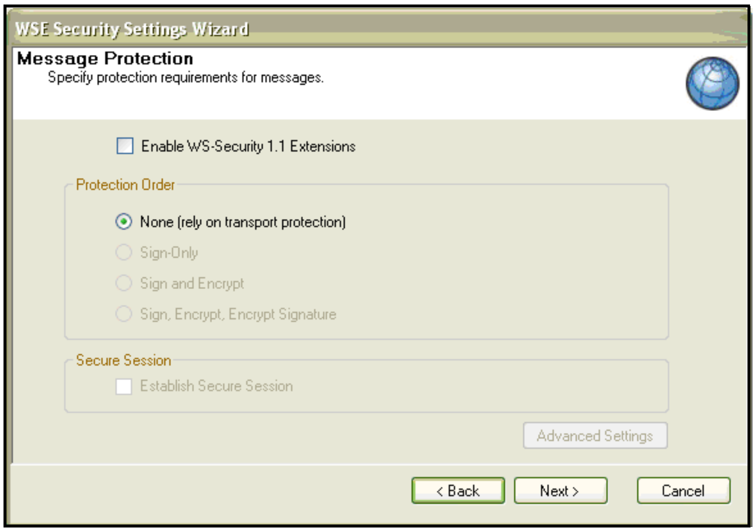
☑ Specify Username Token in code.

Set Username and Password.

Username	
Password	

Step 10 Click **Next**.

Step 11 Uncheck **Enabled WS-Security 1.1 Extensions**, which causes **None (rely on transport protection)** to be automatically selected.



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

Step 12 Click **Next**.

Step 13 To exit the wizard, click **Finish**.

Step 14 To save your changes, click **OK**.

Sending Requests to CyberSource

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

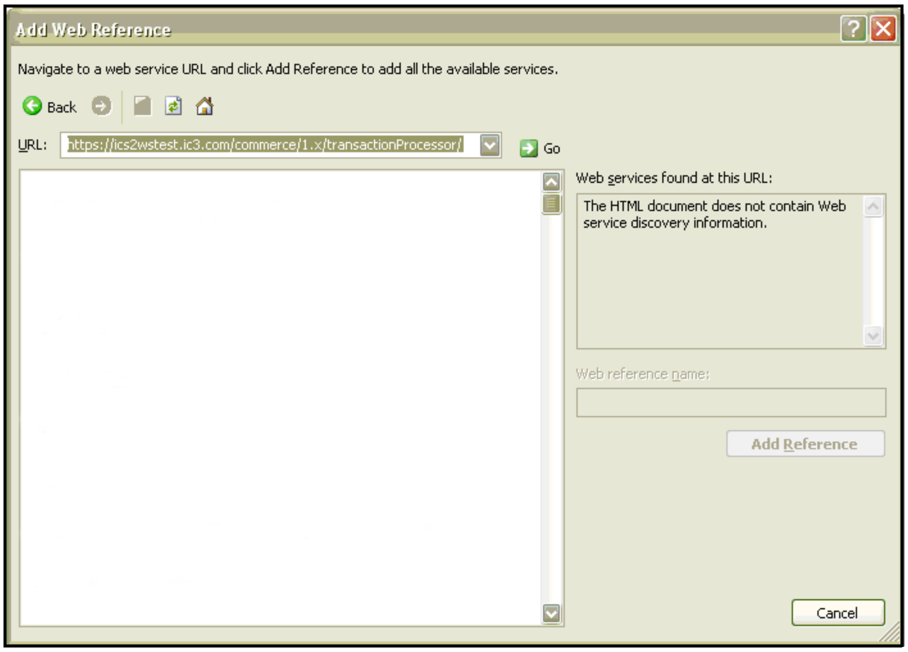
Step 1 In the Solution Explorer, right-click the project node and select **Add Web Reference**.

Step 2 In the dialog box, enter the URL for CyberSource's Web Service:

Important

You must use separate transactions keys for the production and test environments.

Test environment	`https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor`
Production environment	`https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor`



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/

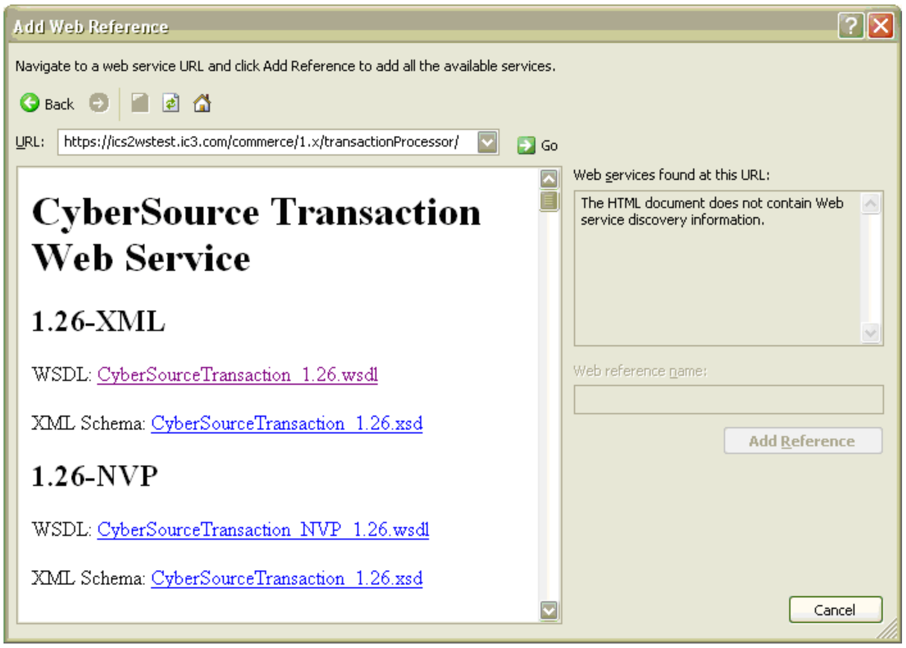
Web services found at this URL:

The HTML document does not contain Web service discovery information.

Web reference name:

Step 3 Click **Go**.

The available server API versions are displayed.



Add Web Reference

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:

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

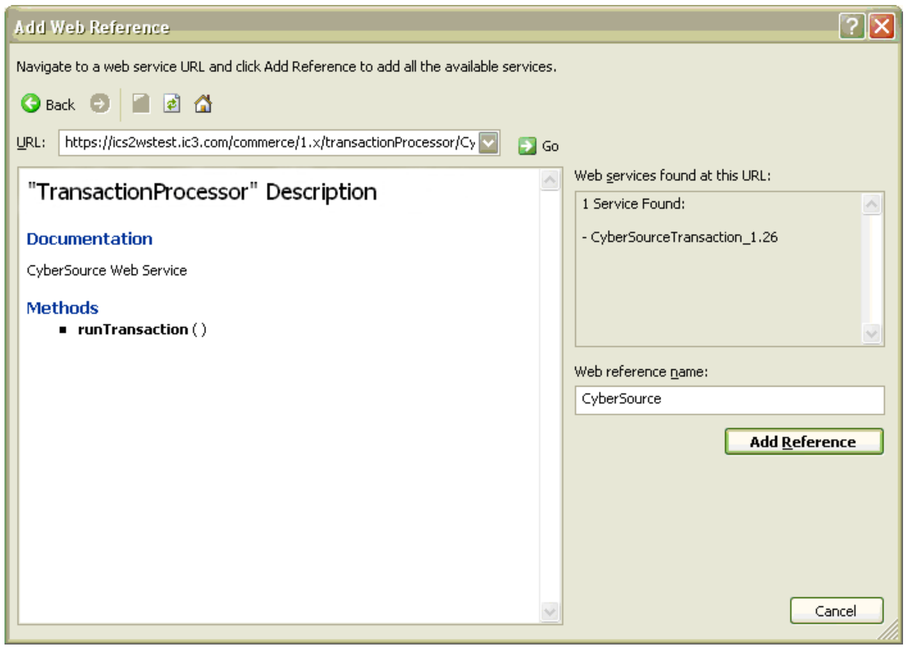
Step 5 Change the Web Reference Name to CyberSource.

The Web Reference Name will be 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`.

Note 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 as follows:

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



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/Cy

"TransactionProcessor" Description

Documentation

CyberSource Web Service

Methods

- runTransaction ()

Web services found at this URL:

1 Service Found:

- CyberSourceTransaction_1.26

Web reference name: CyberSource

Step 6 Click **Add Reference**.

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

Building the Sample and Testing the Client

To test your client, follow these steps.

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

```
MERCHANT_ID
TRANSACTION_KEY
LIB_VERSION
POLICY_NAME
```

Modify `POLICY_NAME` only if needed.

Step 2 Add the sample file to your application.

Step 3 Run the application.

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 your Client and your 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, change the host in the URL in your application or Web configuration file:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, follow these steps:

  a In the Solution Explorer, under the Web References node, click the CyberSource web reference.

  b 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
  ```

  c Rebuild your application.

- To add or delete API fields, modify your source code.

CHAPTER 4

Constructing SOAP with .NET 3.0 (WCF) and Later

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

Before starting, you need to 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 **Note** After installing the software, make sure that you reboot your computer to ensure that `svcutil` is recognized as a shell command.

Test the client application with the following CyberSource sample code:

`sample_wcf.cs` provides the code to process your transactions. To help you understand and use the code, the file contains many comments and a sample card authorization. Before using the file, make sure that you understand each section and that you replace the generic values of the variables with your own.

Creating and Testing the Client by Using the Sample Code

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

Step 1 Go to Start > All Programs > Microsoft Windows SDK > CMD Shell.

Step 2 Change directory (cd) to locate the sample code (`sample_wcf.cs`).

Step 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, point your browser to `https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor`.

Two files are generated:

- `CyberSourceTransactionWS.cs` contains the proxy classes.
- `sample_wcf.exe.config` is the configuration file for your application.

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

The security element now reads:

`<security mode="TransportWithMessageCredential">`

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

a Modify the values of `MERCHANT_ID` and `TRANSACTION_KEY` with your own values.

b Build the executable 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 Co
mmunication Foundation\System.ServiceModel.dll" CyberSourceTr
ansactionWS .cs sample_wcf.cs
```

The `sample_wcf.exe` is created.

Step 6 Run `sample_wcf.exe`.

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 your Client and your 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, change the host in the `endpoint` address in the configuration file:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, repeat the steps in the previous section except Step 5.
- To add or delete API fields, modify the source code.

CHAPTER 5

Constructing SOAP with Perl 5.8.8 and SOAP::Lite 0.69

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

Before starting, you need to download and install the required third-party software. CyberSource tested these versions:

Software Tested	**Description**
▪ Linux Kernel 2.4 ▪ Windows ▪ Solaris	Operating systems tested
Perl 5.8.8	Perl software
SOAP::Lite 0.69	Perl module for SOAP messages
OpenSSL 0.9.7	SSL library
Crypt::SSLeay 0.53_02	Download from http://search.cpan.org/dist/Crypt-SSLeay/SSLeay.pm
URI 1.35	Download from http://search.cpan.org/dist/URI/
XML::Parser 2.34	Download from http://search.cpan.org/dist/XML-Parser/
libwww-perl 5.8.0.5	Download from http://search.cpan.org/

Test the client application with the following Cybersource sample code:

sample_perl.pl provides the code to process your transactions.

To help you understand and use the code, the file contains many comments and a sample card authorization. Before using the file, make sure that you understand its content and that you replace the generic values of the variables with your own values.

Installing SOAP::Lite

You can install the SOAP::Lite module by using either of the following methods. Regardless of the method that you choose, make sure that `Client HTTPs support` is set to `yes`. If `Crypt::SSLeay` is properly installed, the default is `yes`.

Using the CPAN Module

Step 1 Run this command:

```
perl -MCPAN -e shell
```

Step 2 Inside the shell, run this command:

```
install SOAP::Lite
```

If the CPAN module is configured to follow prerequisites, the prerequisites will be installed automatically. For more information, see the CPAN module documentation.

Not Using the CPAN Module

Run the following scripts for each module that `SOAP::Lite` requires and one last time for `SOAP::Lite`:

```
cd package_directory
perl Makefile.PL
make
make test
make install
```

The ***`package_directory`*** is the directory where you unpacked the package.

Building the Sample and Testing the Client

To test your client, follow these steps.

Step 1 In `sample.pl`, modify `MERCHANT_ID` and `TRANSACTION_KEY` with your own values.

Step 2 Run the script `perl sample.pl`.

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 your Client and your 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, change the host in the URL that you pass to `proxy()`. In the sample, set `CYBS_HOST` to the appropriate host:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, update the version number in the URI that you pass to `uri()`. In the sample, set `CYBS_VERSION` to your new target version.
- To add or delete API fields, modify the source code.

CHAPTER 6

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

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

Before starting, you need to 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 at http://www.slproweb.com/products/Win32OpenSSL.html. 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 will be overwritten during installation.
Microsoft Visual Studio 2005	Development environment tested

Test the client application with the following Cybersource sample code:

Sample code	`sample.cpp` provides the code to process your transactions. To help you understand and use the code, the file contains many comments and a sample card authorization. Before using the file, make sure that you understand the many sections and that you replace the generic values of the variables with your own.

Generating the Client Code

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

Step 1 Download to the same directory the latest WSDL and XSD files from the following URL:

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

Step 2 Save the files as follows:

```
CyberSourceTransaction_1.26.wsdl
CyberSourceTransaction_1.26.xsd
```

Step 3 To generate the header file, run the following script in the directory where 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 where 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 `cybersource.h`.

Note

You will receive the following warnings, 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.
Warning: element 'xsd:selector' at level 3 was not recognized
and will be ignored.
Warning: element 'xsd:field' at level 3 was not recognized and
will be ignored.
Warning: element 'xsd:unique' at level 2 was not recognized and
will be ignored.
Warning: element 'xsd:selector' at level 3 was not recognized
and will be ignored.
Warning: element 'xsd:field' at level 3 was not recognized and
will be ignored.
```

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

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

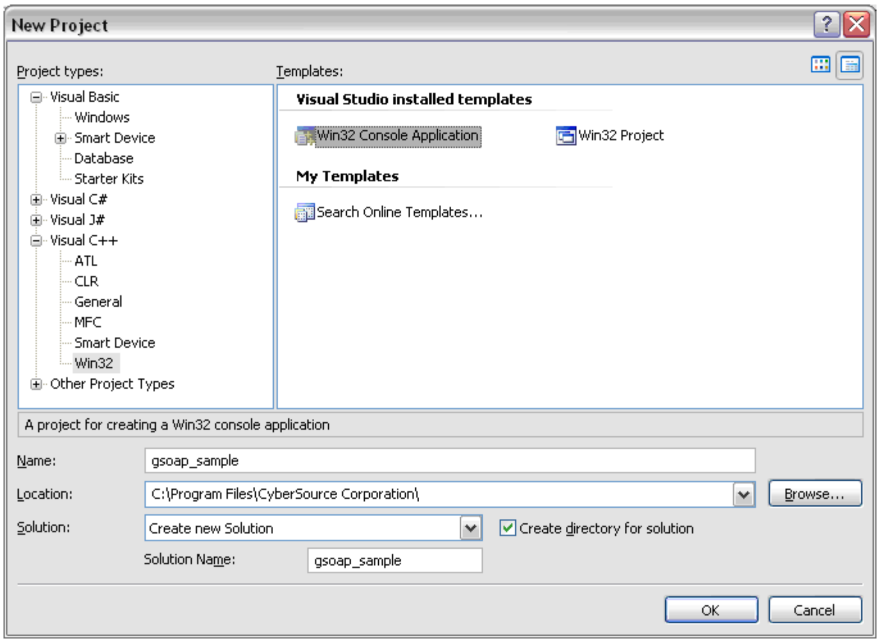
Step 5 To generate the client source code, run this script:

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

Building the Client

Step 1 In Visual Studio 2005, create a non-CLR (Common Language Runtime) C++ project.

The sample code provided is for a Win32 Console Application. The Win32 Application Wizard appears. This sample uses a Win32 Console Application with `gsoap_sample` as the name of the solution.



A project for creating a Win32 console application

Name: gsoap_sample

Location: C:\Program Files\CyberSource Corporation\

Solution: Create new Solution

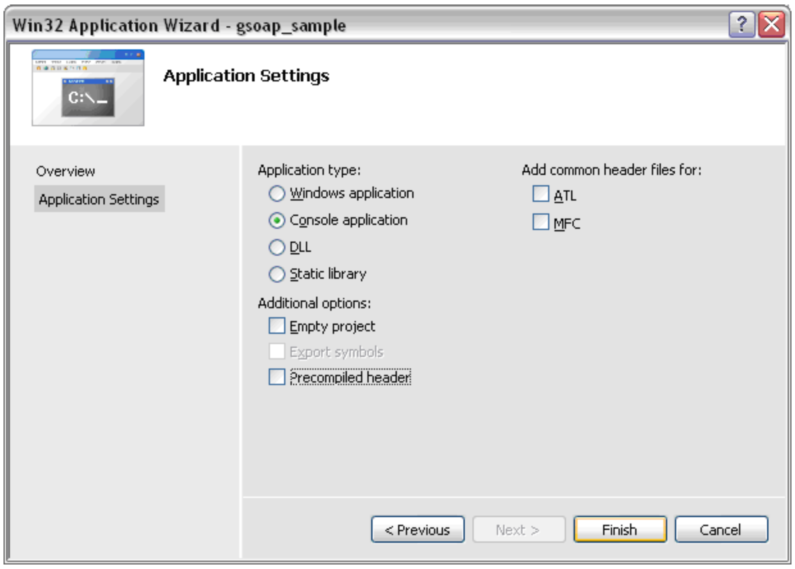
[✓] Create directory for solution

Solution Name: gsoap_sample

Step 2 Click **OK** and **Next**.

The Application Settings page appears.

Step 3 On the Application Settings page, uncheck **Precompiled header** and click **Finish**.



Win32 Application Wizard - gsoap_sample

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

Your new project is created.

Step 4 Select **Project -> gsoap_sample Properties**.

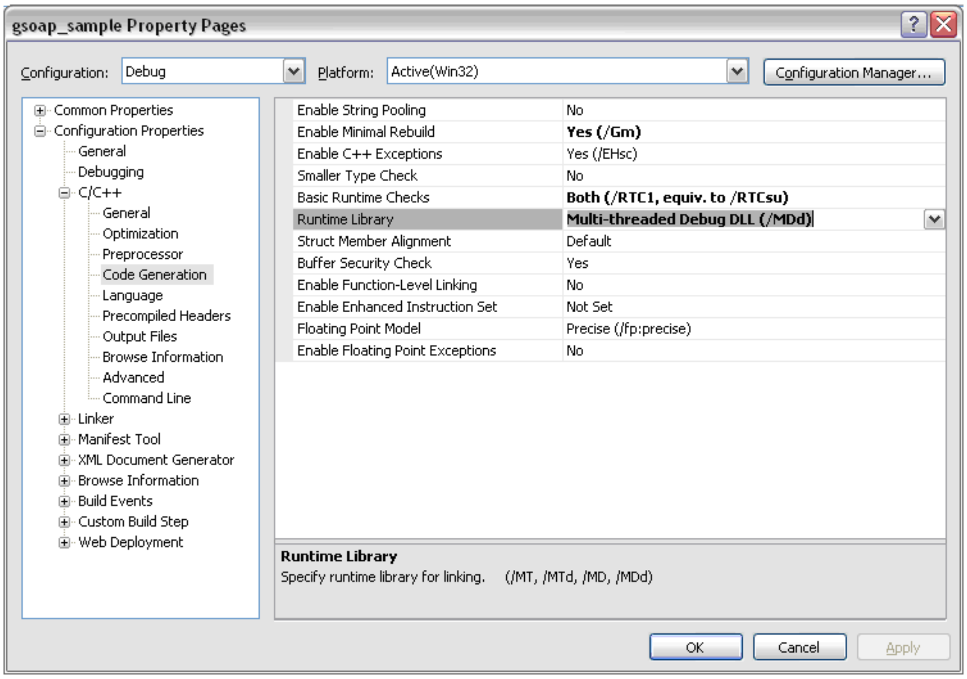
Step 5 In the navigation pane, expand **Configuration Properties -> C/C++** and click **Code Generation**.

Step 6 Make sure that the default configuration runtime libraries are selected as follows:

- Debug configuration:

  a In the Configuration drop-down menu in the upper-left corner, select **Debug**.

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



gsoap_sample Property Pages

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

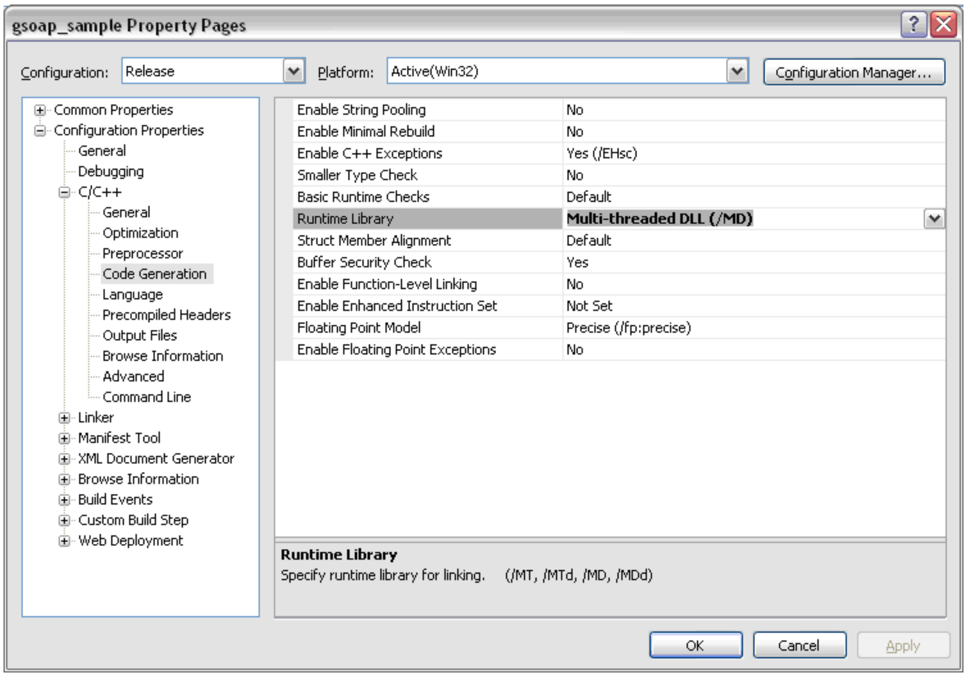
Enable String Pooling	No
Enable Minimal Rebuild	**Yes (/Gm)**
Enable C++ Exceptions	Yes (/EHsc)
Smaller Type Check	No
Basic Runtime Checks	**Both (/RTC1, equiv. to /RTCsu)**
Runtime Library	**Multi-threaded Debug DLL (/MDd)**
Struct Member Alignment	Default
Buffer Security Check	Yes
Enable Function-Level 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)

- Release configuration:

  a In the Configuration drop-down menu in the upper-left corner, select **Release**.

  b In the main panel, verify that Runtime Library reads `Multi-threaded DLL (/MD)`.



gsoap_sample Property Pages

Configuration: Release Platform: Active(Win32)

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)

Step 7 Click **OK**.

You are returned to the project.

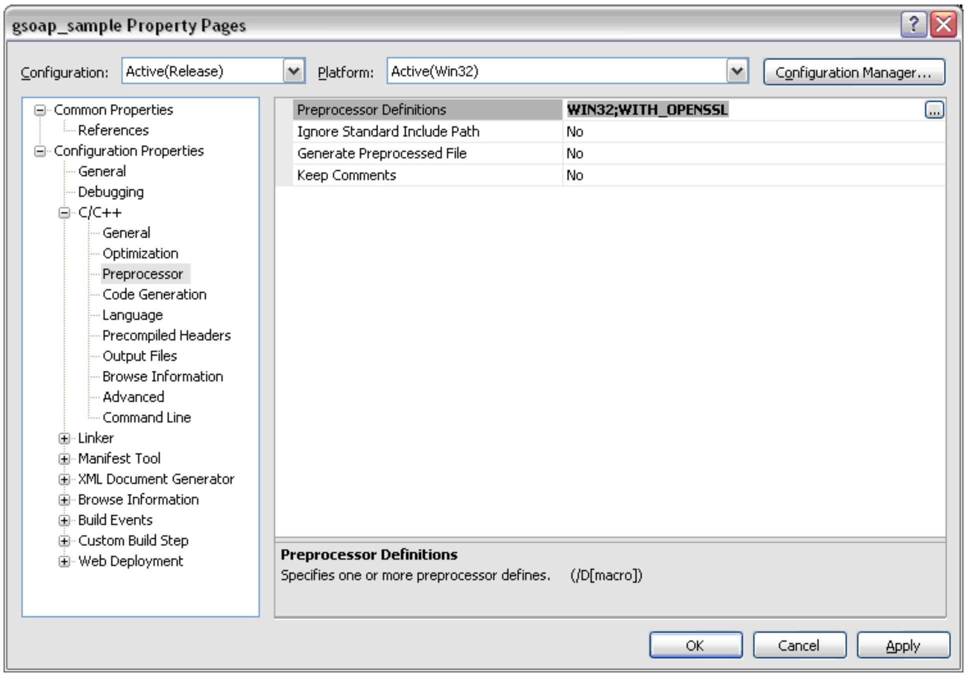
Step 8 Replace `gsoap_sample.cpp`, included in the project, with the following files:

Note Before adding the source files to the `plugin` directory, change their file extension from `.c` to `.cpp`.

CyberSource sample	`sample.cpp`
Generated by gSOAP	`soapC.cpp` `soapClient.cpp`
Included in gSOAP package	***gsoap_directory***`\dom.cpp *` ***gsoap_directory***`\stdsoap2.cpp` ***gsoap_directory***`\mod_gsoap\gsoap_win\wininet\gsoapWinInet.cpp` ***gsoap_directory***`\plugin\smdevp.cpp` ***gsoap_directory***`\plugin\wsseapi.cpp` **** gsoap_directory*** is the directory where you downloaded and extracted gsoap

Step 9 Add the following preprocessor definitions:

```
WIN32
WITH_OPENSSL
```


gsoap_sample Property Pages

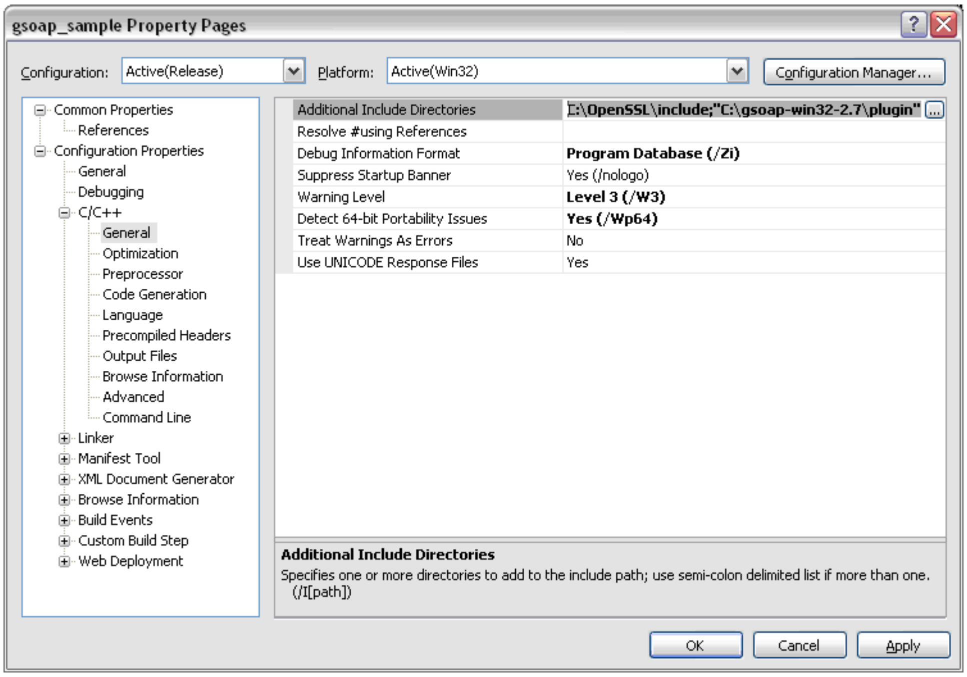
Configuration: Active(Release) Platform: Active(Win32)

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])

Step 10 In the Configuration Properties > C/C++ > General, 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)

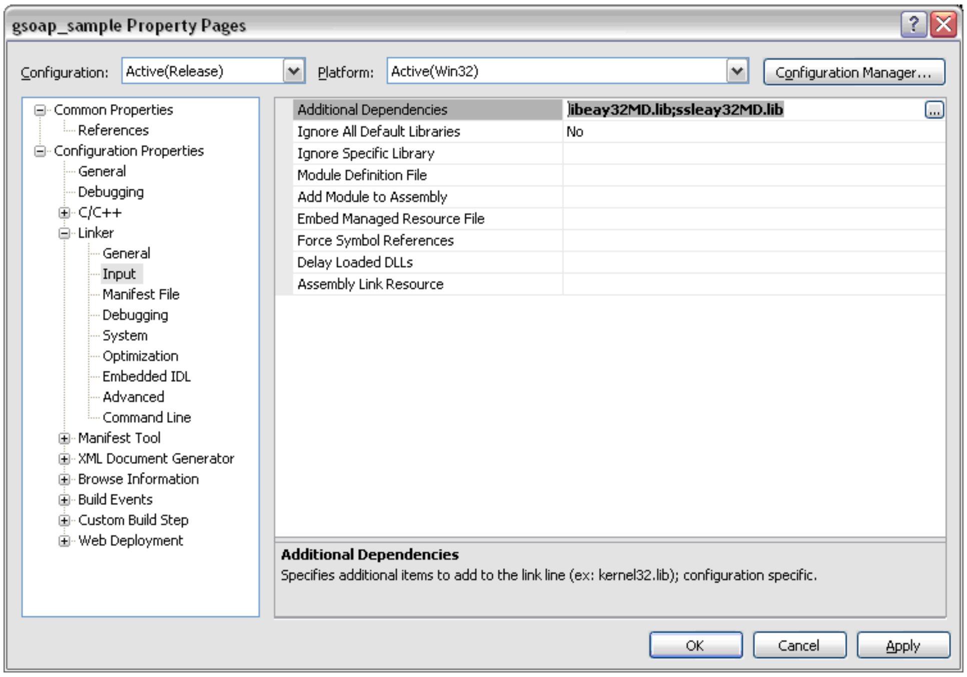
Additional Include Directories	C:\OpenSSL\include;"C:\gsoap-win32-2.7\plugin"
Resolve #using References	
Debug Information Format	Program Database (/Zi)
Suppress Startup Banner	Yes (/nologo)
Warning Level	Level 3 (/W3)
Detect 64-bit Portability Issues	Yes (/Wp64)
Treat Warnings As Errors	No
Use UNICODE Response Files	Yes

Additional Include Directories
Specifies one or more directories to add to the include path; use semi-colon delimited list if more than one. (/I[path])

Step 11 In the Configuration Properties > Linker > Input, add the following libraries.

In the Configuration drop-down menu, select each option in turn:

Release	`libeay32MD.lib` `ssleay32MD.lib`
Debug	`libeay32MDd.lib` `ssleay32MDd.lib`

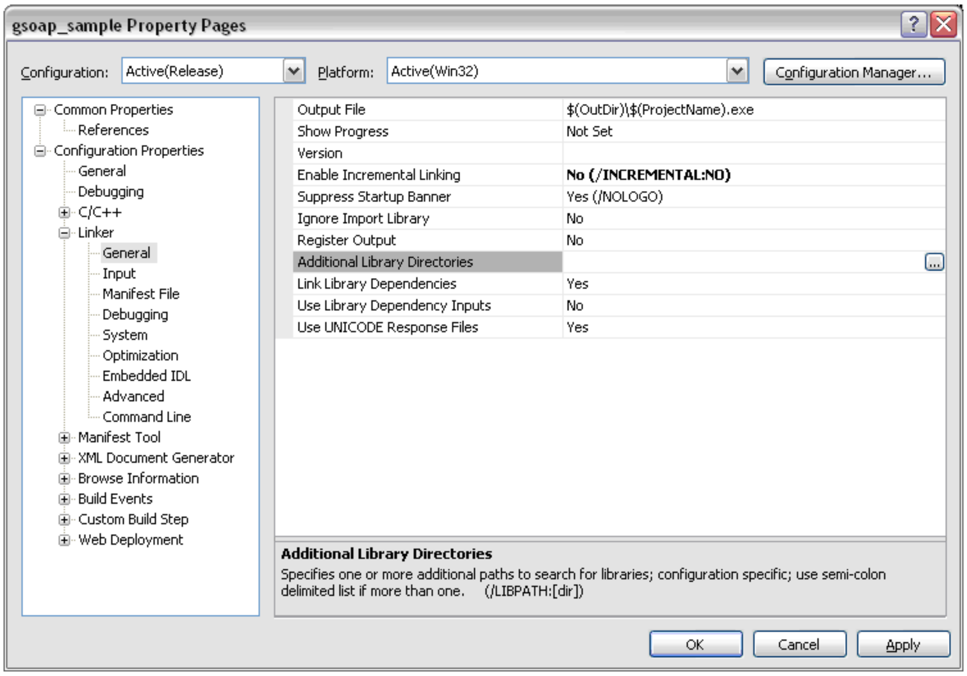


Additional Dependencies	libeay32MD.lib;ssleay32MD.lib
Ignore All Default Libraries	No
Ignore Specific Library	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded DLLs	
Assembly Link Resource	

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

Step 12 Add the following directory to your project's **Additional Library Directories**:

```
openssl_directory\lib\VC
```



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

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])

Step 13 In ***gsoap_directory***\stdsoap2.cpp, and find the following calls:

```
ASN1_item_d2i
meth->d2i
```

Step 14 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);
```

Step 15 If you see the following compiler error in stdsoap2.cpp

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

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

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

Step 16 Find d2i_X509 in gsoap_directory\plugin\wsseapi.cpp, inside soap_wsse_get_BinarySecurityTokenX509.

Step 17 To the existing cast, add ***const*** as follows:

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

You are ready to test the client.

Building the Sample and Testing the Client

To test your client, follow these steps.

Important You must use separate transaction keys for the test and production environments.

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

```
MERCHANT_ID
TRANSACTION_KEY
SERVER_URL
LIB_VERSION
ENVIRONMENT
```

Only use `LIB_VERSION` if you are using a different version of gSOAP.

Step 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 testing the client is finished, write the code to use the client application.

Modifying your Client and your 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, change the URL assigned to `service.endpoint`. In the sample file, set `SERVER_URL` to the appropriate value:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, rebuild your client by following the steps in "Generating the Client Code," page 37.
- To add or delete API fields, you only need to modify your source code.

CHAPTER 7

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

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

Before starting, you need to download and install the required third-party software. CyberSource tested these versions:

Software Tested	Description
Linux Kernel 2.6	Operating system version tested
gSOAP 2.7.9e	SOAP toolkit. You can download it from http://sourceforge.net/project/showfiles.php?group_id=52781
OpenSSL 0.9.8	Current version of the toolkit implementing SSL. **Note** 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

Test the client application with the following Cybersource sample code:

Sample code	`sample.cpp` provides the code to process your transactions. To help you understand and use the code, the file contains many comments and a sample card authorization. Before using the file, make sure that you understand each section and 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 Development Environment

Step 1 Download the latest gSOAP package for Linux.

Step 2 To open the package, run this command:

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

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

Create a…	Named	With this command:
Directory for the client-related files (`Makefile` and `sample.cpp`)	`client`	`mkdir client`
Symbolic link to your `gSOAP` directory	`gsoap`	`ln -s <gsoap_path> gsoap` where *<gsoap_path>* is the path to the `gSOAP` directory that you created in Step 2.

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

`ASN1_item_d2i`	(occurs once)
`meth-d2i`	(occurs twice)

Step 5 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_data = meth->d2i(NULL, (const unsigned char **) &data ...
```

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

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

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

Generating the Client Code

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

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

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

or

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

Step 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.

Note Do not rename the XSD file.

Step 3 Run the `make header` script.

Note You will receive the following warnings, 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.
```

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

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

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

Step 5 Run the `make source` script.

Building the Sample and Testing the Client

To test your client, follow these steps.

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

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

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

Step 2 Run the script `make cybsdemo`.

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 testing the client is finished, write the code to use the client application.

Step 3 Run the sample by executing `./cybsdemo`.

Modifying your Client and your 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, change the URL assigned to `service.endpoint`. In the sample file, set `SERVER_URL` to the appropriate value:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, rebuild your client by following the steps in "Generating the Client Code" and "Building the Sample and Testing the Client."
- To add or delete API fields, you only need to modify your source code.

CHAPTER

8

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

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

Before starting, you need to 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 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

Test the client application with the following Cybersource sample code:

Sample code	`sample.cpp` provides the code to process your transactions. To help you understand and use the code, the file contains many comments and a sample card authorization. Before using the file, make sure that you understand the many sections and 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 Development Environment

Step 1 Download and unzip the latest gSOAP package for Mac OS X.

Step 2 To open the package, run this command:

```
tar xvfz gsoap_macosx_S2.7.9d.tar.gz
```

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

Create a…	Named	With this command:
Directory for the client-related files (`Makefile` and `sample.cpp`)	`client`	`mkdir client`
Symbolic link to your `gSOAP` directory	`gsoap`	`gsoap -> gsoap-macosx-2.7`

Generating the Client Code

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

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

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

or

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

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

Note Do not rename the XSD file.

Step 3 Run the `make header` script.

Note You will receive the following warnings, 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.
Warning: element 'xsd:unique' at level 2 was not recognized and
will be ignored.
```

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

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

Step 5 Run the `make source` script.

Building the Sample and Testing the Client

To test your client, follow these steps.

Step 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.

Step 2 Run the `make cybsdemo` script.

Ignore the warning messages. `cybsdemo` is now ready to use.

Step 3 Run the sample by executing `./cybsdemo`.

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 your Client and your 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, change the URL assigned to `service.endpoint`. In the sample file, set `SERVER_URL` to the appropriate value:

Test environment	`ics2wstesta.ic3.com`
Production environment	`ics2wsa.ic3.com`

- To update the version of the CyberSource API, rebuild your client by following the steps in "Generating the Client Code," page 54, and "Building the Sample and Testing the Client."
- To add or delete API fields, you only need to modify your source code.

CHAPTER 9

Constructing SOAP with Apache Axis and WSS4J

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

Before starting, you need to download and install the required third-party software. CyberSource tested these versions:

Software Tested	Description
■ Windows XP Professional with SP2 ■ Linux ■ Solaris	Operating systems tested
JDK 1.5	Java Development Kit
Apache Axis 1.4	SOAP toolkit Download and unzip the latest package from http://ws.apache.org/axis
Apache WSS4J 1.5.1	WS-Security package Download and unzip the latest package from http://ws.apache.org/wss4j
Apache XML Security 1.4.0	XML security package 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/

Test the client application with the following Cybersource sample code:

Sample code	`Sample.java`: sample file, which provides the code to process your transactions. The file contains comments and a sample card authorization. Before using the file, make sure that you understand each section and that you replace the generic values of the variables with your own. `SamplePWCallback.java`: sample Password Callback Handler, which provides the password to WSS4J. `SampleDeploy.wsdd`: sample deployment descriptor file used by WSS4J.

Generating and Building the Stubs

Step 1 From each of these packages, add these items to your classpath:

- The current directory (`.`)
- These files:

Package	Files
Apache Axis	■ `axis.jar` ■ `commons-discovery-0.2.jar` ■ `commons-logging-1.0.4.jar` ■ `jaxrpc.jar` ■ `log4j-1.2.8.jar` ■ `saaj.jar` ■ `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`

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

Step 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/
CyberSourceTransaction_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. The latest version is located:

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

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

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

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

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

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

Building the Sample and Testing the Client

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

Step 1 In `Sample.java`, modify the values of `MERCHANT_ID`.

Step 2 In `SamplePWCallback.java`, modify the value of `TRANSACTION_KEY`.

Step 3 Compile the samples as follows:

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

Step 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 Your Client and Your 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`

- To update the version of the CyberSource API, rebuild the client by following the steps in "Generating and Building the Stubs," page 58.
- To add or delete API fields, modify your source code.