CyberSource SOAP Toolkits for Web Services

Developer Guide

January 2016
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

© 2016 CyberSource Corporation. All rights reserved. CyberSource Corporation (“CyberSource”) furnishes this document and the software described in this document under the applicable agreement between the reader of this document (“You”) and CyberSource (“Agreement”). You may use this document and/or software only in accordance with the terms of the Agreement. Except as expressly set forth in the Agreement, the information contained in this document is subject to change without notice and therefore should not be interpreted in any way as a guarantee or warranty by CyberSource. CyberSource assumes no responsibility or liability for any errors that may appear in this document. The copyrighted software that accompanies this document is licensed to You for use only in strict accordance with the Agreement. You should read the Agreement carefully before using the software. Except as permitted by the Agreement, You may not reproduce any part of this document, store this document in a retrieval system, or transmit this document, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written consent of CyberSource.

Restricted Rights Legends

For Government or defense agencies. Use, duplication, or disclosure by the Government or defense agencies is subject to restrictions as set forth the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

For civilian agencies. Use, reproduction, or disclosure is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial Computer Software Restricted Rights clause at 52.227-19 and the limitations set forth in CyberSource Corporation's standard commercial agreement for this software. Unpublished rights reserved under the copyright laws of the United States.

Trademarks

CyberSource, The Power of Payment, CyberSource Payment Manager, CyberSource Risk Manager, CyberSource Decision Manager, CyberSource Connect, Authorize.Net, and eCheck.net are trademarks and/or service marks of CyberSource Corporation. All other brands and product names are trademarks or registered trademarks of their respective owners.
**Contents**

**Recent Revisions to This Document**  6

**About This Guide**  7
- Audience  7
- Purpose  7
- Scope  7
- Conventions  8
  - Note and Important Statements  8
  - Text and Command Conventions  8
- Related Documents  9
  - CyberSource Services Documentation  9
- Customer Support  9

---

**Chapter 1**  Configuring SOAP Toolkits for Web Services  10
- Supported Toolkits  11
- Destination URLs for SOAP Messages  12
- Transaction Key  13
- Sample SOAP Message  15

**Chapter 2**  Constructing SOAP with PHP 5.2.1  16
- Preparing your PHP Installation  17
  - Windows Operating System  17
  - Linux Operating System  18
- Building and Running the Sample  19
- Modifying your Script  19

**Chapter 3**  Constructing SOAP with .NET 2.0 and WSE 3.0  20
- Preparing Your Application  20
- Sending Requests to CyberSource  25
Modifying Your Client and Your Code  60
## Recent Revisions to This Document

<table>
<thead>
<tr>
<th>Release</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2016</td>
<td>Fixed the URL for the perl sample code. See page 33.</td>
</tr>
<tr>
<td>September 2015</td>
<td>Updated the production server URL and the test server URL.</td>
</tr>
<tr>
<td>August 2015</td>
<td>Changed mentions of <code>.NET 3.0</code> to <code>.NET 3.0 and later</code>.</td>
</tr>
<tr>
<td>October 2013</td>
<td>This revision contains only editorial changes and no technical updates.</td>
</tr>
<tr>
<td>August 2013</td>
<td>This revision contains only editorial changes and no technical updates.</td>
</tr>
<tr>
<td>January 2013</td>
<td>Added important note about the new requirement for separate transaction security keys for CyberSource production and test environments. For more information, see &quot;Transaction Key,&quot; page 13.</td>
</tr>
</tbody>
</table>
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

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

![Note]

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

![Important]

Text and Command Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Usage</th>
</tr>
</thead>
</table>
| **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
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:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTPS</td>
<td>HTTP protocol with SSL encryption</td>
</tr>
<tr>
<td>SOAP 1.1</td>
<td>Version 1.1 of the Simple Object Access Protocol</td>
</tr>
<tr>
<td>Document/literal (unwrapped)</td>
<td>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.</td>
</tr>
<tr>
<td>UsernameToken</td>
<td>Authentication mechanism specified in WS-Security 1.0</td>
</tr>
<tr>
<td></td>
<td>In header of the SOAP message</td>
</tr>
</tbody>
</table>

The CyberSource servers do not support persistent HTTP connections.

Note

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.

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

> **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](http://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

---

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

---

**Note**
Chapter 1  Configuring SOAP Toolkits for Web Services

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.

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

Important

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 base64-encoded transaction key represented in string format. Use this authentication method if you want to create your SOAP message.
Chapter 1  Configuring SOAP Toolkits for Web Services

**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**.

<table>
<thead>
<tr>
<th>Transaction Security Key Index</th>
<th>Activation Date</th>
<th>Expiration Date</th>
<th>Delete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>no transaction keys currently exist for this merchant</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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**.

<table>
<thead>
<tr>
<th>Transaction Security Key Index</th>
<th>Activation Date</th>
<th>Expiration Date</th>
<th>Delete?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Security Key 1</td>
<td>Apr 12 2007 12:44:46 PM</td>
<td>Apr 12 2019 12:45:00 PM</td>
<td></td>
</tr>
</tbody>
</table>

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.  

`OL37VO9Pve-ZEp+7aH30xwq0GCGGq38D3D96AQUARIIYDCC15CAQxP1q70U8226HoaqkI8I k5DySopiXYnUe=52J6K3FQboTm7sW/G97cC5b5IhF5WqB13hYmY76b049169K4697bKIO dbxy1MQHr9XCFSpfp5=3261yq2p2E71+9qV7iKdC62ZC36qg07Jf0U0VfE+zE+ 32657T82om4sGetj_897sC883h9yYp0to2A0dly74b4k455b980Y3Qpy7fIKHWeCgYCCiyw p51UxY2egZ5bq122WE4n4Xb6S5EKo9H9GQCFRP79RQwu4C5Gc7Lza93594==`

**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
<?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>
    </requestMessage>
  </soapenv:Body>
</soapenv:Envelope>
```
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:

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

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:

<table>
<thead>
<tr>
<th>Sample code</th>
<th>The sample PHP files contain many comments and a sample card authorization. Choose the file appropriate for you:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&quot;cli-sample.php&quot; if you use the command-line interface</td>
</tr>
<tr>
<td></td>
<td>&quot;web-sample.php&quot; if you use the Web interface</td>
</tr>
</tbody>
</table>

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"
```


The extensions are not included in the Windows installer package.

---

**Note**

**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:

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

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

```bash
./configure '--prefix=your_target_dir' '--enable-soap' '--with-libxml-dir=your_libxml_dir' '--with-openssl=your_openssl_dir'
```

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

```bash
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`:

<table>
<thead>
<tr>
<th>Test environment</th>
<th>Production environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ics2wstesta.ic3.com</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.
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:

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP Pro with SP2</td>
<td>Operating system version tested</td>
</tr>
<tr>
<td>Visual Studio 2005</td>
<td>Includes .NET 2.0</td>
</tr>
<tr>
<td>WSE 3.0</td>
<td>Web Services Enhancements for Microsoft .NET, which is used to authenticate the user with the UsernameToken class.</td>
</tr>
</tbody>
</table>

Test the client application with the following CyberSource sample code:

Sample code

<table>
<thead>
<tr>
<th>sample_wse30.vb (VB) and sample_wse30.cs (C#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provide the code to process your transactions.</td>
</tr>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

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:

- Reinstall WSE 3.0 by using the Add or Remove Programs menu.
- In the installer, select Modify and install the Visual Studio Tools option.
- When done, restart Visual Studio.
- Repeat steps 1 and 2.
Step 3  In the dialog box, check Enable this project for Web Services Enhancements.
Step 4  Click the Policy tab and check Enable Policy.

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.

Step 7  Click Next.
Step 8  Click **Secure a client application** and **Username**.

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.

Step 10  Click **Next**.
Step 11  Uncheck **Enabled WS-Security 1.1 Extensions**, which causes **None (rely on transport protection)** to be automatically selected.

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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td><a href="https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor">https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor</a></td>
</tr>
<tr>
<td>Production environment</td>
<td><a href="https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor">https://ics2wsa.ic3.com/commerce/1.x/transactionProcessor</a></td>
</tr>
</tbody>
</table>

Important: You must use separate transactions keys for the production and test environments.
Step 3 Click Go.
The available server API versions are displayed.

Step 4 To display the content of the most current WSDL, click the top link.
Chapter 3  Constructing SOAP with .NET 2.0 and WSE 3.0

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`.

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;
```

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:

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

```
Important
```

- To alternate between the test and production environments, change the host in the URL in your application or Web configuration file:

<table>
<thead>
<tr>
<th>Environment</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production environment</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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/1.86/transactionProcessor/CyberSourceTransaction_1.86.wsdl
```

  c. Rebuild your application.

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

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP Pro with SP2</td>
<td>Operating system version tested</td>
</tr>
<tr>
<td>Visual Studio 2005</td>
<td>Includes .NET 2.0</td>
</tr>
<tr>
<td>Microsoft Windows SDK</td>
<td>Software Development Kit that contains necessary tools, such as svcutil.exe</td>
</tr>
<tr>
<td>.NET Framework 3.0 and later</td>
<td>Includes the Windows Communication Foundation</td>
</tr>
<tr>
<td>Redistributable Package</td>
<td>Note After installing the software, make sure that you reboot your computer to ensure that svcutil is recognized as a shell command.</td>
</tr>
</tbody>
</table>

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 \texttt{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:

- \texttt{CyberSourceTransactionWS.cs} contains the proxy classes.
- \texttt{sample_wcf.exe.config} is the configuration file for your application.

**Step 4** In \texttt{sample_wcf.exe.config}, change the security mode from \texttt{Transport} to \texttt{TransportWithMessageCredential}.

The security element now reads:

```
<security mode="TransportWithMessageCredential">
```

**Step 5** To write the code to process your transactions, use \texttt{sample_wcf.cs}:

a) Modify the values of \texttt{MERCHANT_ID} and \texttt{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 Communication Foundation\System.ServiceModel.dll"* CyberSourceTransactionWS .cs sample_wcf.cs
```

The \texttt{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:

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

  **Important**

- To alternate between the test and production environments, change the host in the **endpoint address** in the configuration file:

<table>
<thead>
<tr>
<th>Environment</th>
<th>Host URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.
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:

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Kernel 2.4</td>
<td>Operating systems tested</td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td></td>
</tr>
<tr>
<td>Perl 5.8.8</td>
<td>Perl software</td>
</tr>
<tr>
<td>SOAP::Lite 0.69</td>
<td>Perl module for SOAP messages</td>
</tr>
<tr>
<td>OpenSSL 0.9.7</td>
<td>SSL library</td>
</tr>
<tr>
<td>Crypt::SSLeay 0.53_02</td>
<td>Download from <a href="http://search.cpan.org/dist/Crypt-SSLeay/SSLeay.pm">http://search.cpan.org/dist/Crypt-SSLeay/SSLeay.pm</a></td>
</tr>
<tr>
<td>URI 1.35</td>
<td>Download from <a href="http://search.cpan.org/dist/URI/">http://search.cpan.org/dist/URI/</a></td>
</tr>
<tr>
<td>XML::Parser 2.34</td>
<td>Download from <a href="http://search.cpan.org/dist/XML-Parser/">http://search.cpan.org/dist/XML-Parser/</a></td>
</tr>
<tr>
<td>libwww-perl 5.8.0.5</td>
<td>Download from <a href="http://search.cpan.org/">http://search.cpan.org/</a></td>
</tr>
</tbody>
</table>

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.
Chapter 5  Constructing SOAP with Perl 5.8.8 and SOAP::Lite 0.69

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:

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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.
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:

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP Pro with SP2</td>
<td>Operating system version tested</td>
</tr>
<tr>
<td>gSOAP 2.7.9c</td>
<td>Soap toolkit</td>
</tr>
</tbody>
</table>


| OpenSSL 0.9.8d        | You may use the pre-built package available at [http://www.slproweb.com/products/Win32OpenSSL.html](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:

<table>
<thead>
<tr>
<th>Sample code</th>
<th>sample.cpp provides the code to process your transactions.</th>
</tr>
</thead>
</table>

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 \texttt{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:

```plaintext
gsoap_directory\bin\wsdl2h -t gsoap_directory\WS\WS-typemap.dat -s -o cybersource.h CyberSourceTransaction_\texttt{N.NN}.wsdl
```

The \texttt{gsoap_directory} is the directory where you extracted gSoap.

\texttt{N.NN} is the version number of the WSDL file that you downloaded.

This script creates the \texttt{cybersource.h} header file. You may change the name of the file. However, the following steps refer to \texttt{cybersource.h}.

\begin{tcolorbox}[colback=white]
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.
Warning: element 'xsd:field' at level 3 was not recognized and will be ignored.

\end{tcolorbox}

**Step 4** In \texttt{cybersource.h}, add the following line to the Import section:

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

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

```plaintext
gsoap_directory\bin\soapcpp2 -C -I gsoap_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.

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

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

![Application Settings](image)

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)**.
Chapter 6  Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

- 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)**.

  ![Image of soap_sample Property Pages](image)

  **Step 7**  Click **OK**.

  You are returned to the project.
Chapter 6  Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

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

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

Note

<table>
<thead>
<tr>
<th>CyberSource sample</th>
<th>sample.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated by gSOAP</td>
<td>soapC.cpp</td>
</tr>
<tr>
<td></td>
<td>soapClient.cpp</td>
</tr>
</tbody>
</table>
| Included in gSOAP      | 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
Chapter 6  Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

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`
Step 11  In the Configuration Properties > Linker > Input, add the following libraries.
In the Configuration drop-down menu, select each option in turn:

<table>
<thead>
<tr>
<th>Release</th>
<th>libeay32MD.lib</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ssleay32MD.lib</td>
</tr>
<tr>
<td>Debug</td>
<td>libeay32MDd.lib</td>
</tr>
<tr>
<td></td>
<td>ssleay32MDd.lib</td>
</tr>
</tbody>
</table>

![Screenshot of gsoap_sample Property Pages](image)
Chapter 6  Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

Step 12  Add the following directory to your project’s **Additional Library Directories**:  
openssl\openssl_directory\lib\VC

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.
Chapter 6  Constructing SOAP with C++ and gSOAP 2.7.9c for Windows

Building the Sample and Testing the Client

To test your client, follow these steps.

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:

- 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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production environment</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.

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

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

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.
Preparation 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:

<table>
<thead>
<tr>
<th>Create a...</th>
<th>Named</th>
<th>With this command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory for the client-related files (Makefile and sample.cpp)</td>
<td>client</td>
<td>mkdir client</td>
</tr>
<tr>
<td>Symbolic link to your gSOAP directory</td>
<td>gsoap</td>
<td>ln -s &lt;gsoap_path&gt; gsoap</td>
</tr>
</tbody>
</table>

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 ...
```

**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);
```
Chapter 7  Constructing SOAP with C++ and gSOAP 2.7.9e for Linux®

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.

Do not rename the XSD file.

Step 3  Run the make header script.

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:

- 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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.

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

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac OS X</td>
<td>Operating system version tested</td>
</tr>
<tr>
<td>gSOAP 2.7.9d</td>
<td>SOAP toolkit</td>
</tr>
<tr>
<td></td>
<td>Download and unzip the latest Mac OS X package from <a href="http://sourceforge.net/projects/gsoap2/">http://sourceforge.net/projects/gsoap2/</a></td>
</tr>
<tr>
<td>openssl 0.9.7</td>
<td>OpenSSL version that is part of Mac OS X</td>
</tr>
<tr>
<td>gcc 4.0.1</td>
<td>C/C++ compiler tested</td>
</tr>
</tbody>
</table>

Test the client application with the following Cybersource sample code:

<table>
<thead>
<tr>
<th>Sample code</th>
<th>sample.cpp provides the code to process your transactions.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Makefile provides targets to easily create and build the sample client.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Create a…</th>
<th>Named</th>
<th>With this command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory for the client-related files (Makefile and sample.cpp)</td>
<td>client</td>
<td>mkdir client</td>
</tr>
<tr>
<td>Symbolic link to your gSOAP directory</td>
<td>gsoap</td>
<td>gsoap -&gt; gsoap-macosx-2.7</td>
</tr>
</tbody>
</table>

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:

- 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:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.
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:

<table>
<thead>
<tr>
<th>Software Tested</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP Professional with SP2</td>
<td>Operating systems tested</td>
</tr>
<tr>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td>Solaris</td>
<td></td>
</tr>
<tr>
<td>JDK 1.5</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>Apache Axis 1.4</td>
<td>SOAP toolkit</td>
</tr>
<tr>
<td></td>
<td>Download and unzip the latest package from <a href="http://ws.apache.org/axis">http://ws.apache.org/axis</a></td>
</tr>
<tr>
<td>Apache WSS4J 1.5.1</td>
<td>WS-Security package</td>
</tr>
<tr>
<td></td>
<td>Download and unzip the latest package from <a href="http://ws.apache.org/wss4j">http://ws.apache.org/wss4j</a></td>
</tr>
<tr>
<td>Apache XML Security 1.4.0</td>
<td>XML security package</td>
</tr>
<tr>
<td></td>
<td>Download the latest package from <a href="http://santuario.apache.org/download.html">http://santuario.apache.org/download.html</a> and extract xmlsec-N.N.N.jar</td>
</tr>
<tr>
<td>mail.jar</td>
<td>JDK Java Mail add-on that you can download from <a href="http://java.sun.com/products/javamail/">http://java.sun.com/products/javamail/</a></td>
</tr>
</tbody>
</table>
Test the client application with the following Cybersource sample code:

<table>
<thead>
<tr>
<th>Sample code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample.java</td>
<td>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.</td>
</tr>
<tr>
<td>SamplePWCallback.java</td>
<td>sample Password Callback Handler, which provides the password to WSS4J.</td>
</tr>
<tr>
<td>SampleDeploy.wsdd</td>
<td>sample deployment descriptor file used by WSS4J.</td>
</tr>
</tbody>
</table>

### Generating and Building the Stubs

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

<table>
<thead>
<tr>
<th>Package</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Axis</td>
<td>axis.jar</td>
</tr>
<tr>
<td></td>
<td>commons-discovery-0.2.jar</td>
</tr>
<tr>
<td></td>
<td>commons-logging-1.0.4.jar</td>
</tr>
<tr>
<td></td>
<td>jaxrpc.jar</td>
</tr>
<tr>
<td></td>
<td>log4j-1.2.8.jar</td>
</tr>
<tr>
<td></td>
<td>saaj.jar</td>
</tr>
<tr>
<td></td>
<td>wsdl4j-1.5.1.jar</td>
</tr>
<tr>
<td>Apache WSS4J</td>
<td>wss4j-1.5.1.jar</td>
</tr>
<tr>
<td>Apache XML Security</td>
<td>xmlsec-1.4.0.jar</td>
</tr>
<tr>
<td>JDK JavaBeans Activation Framework</td>
<td>activation.jar</td>
</tr>
<tr>
<td>JDK Java Mail</td>
<td>mail.jar</td>
</tr>
</tbody>
</table>

**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
java org.apache.axis.wsdl.WSDL2Java -p com.cybersource.stub
https://ics2wstesta.ic3.com/commerce/1.x/transactionProcessor/
CyberSourceTransaction_1.9.wsd1
```

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.

- `1.9` 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:

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

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

```bash
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:

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

Step 4  
Run the sample as follows:

```bash
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:

- To alternate between the test and production environments, set `SERVER_URL` to the appropriate value:

<table>
<thead>
<tr>
<th>Environment</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test environment</td>
<td>ics2wstesta.ic3.com</td>
</tr>
<tr>
<td>Production</td>
<td>ics2wsa.ic3.com</td>
</tr>
</tbody>
</table>

- 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.

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