CyberSource Simple Order API Client for ASP

Developer's Guide

August 2010

CyberSource®

the power of payment

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 650-965-6000 or 888-330-2300 (toll-free in the United States).

Additional Contact Information for Enterprise Business Center Users

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

For Customer Support, contact the appropriate office:

- Within the United States, go to http://www.cybersource.com/esupport.
- Outside the United States, email eu_support@cybersource.com or call +44 0 1189 653 545.

Additional Contact Information for Business Center Users

For technical support questions, go to the Home page in the Business Center to see the contact information appropriate for your account.

Visit the Business Center, your central location for managing your online payment transactions, at https://businesscenter.cybersource.com.

Contents

Documentation Changes

The following table lists changes made in the last six releases of this document:

Release	Changes
August 2010	• Clarified how to use logs to comply with PCI and PABP requirements. For more information, see "EnableLog" on page 17 and "Logger" on page 23.
January 2008	• Added notes stating that the CyberSource servers do not support persistent HTTP connections. For name-value pairs, see "Requesting ICS Services" on page 30. For XML, see "Requesting ICS Services" on page 42.
October 2006	• Added information regarding PCI and PABP compliance. See the description of EnableLog in Table 1 on page 16 and see "Logger" on page 23.
July 2006	• Added important information about configuring your client API host to a unique, public IP address. See "System Requirements" on page 9. • Modified the path to the sample files. For more information, see "Creating and Sending Requests" on page 30 (NVP) or "Sample Code" on page 42 (XML). • Modified Appendix A, "Generating Security Keys," on page 55. The process for generating a key is now identical in both Business Centers.
March 2006	• Updated the information in "Other Necessary Documentation" on page 2 • Updated the information in "Going Live" on page 12.
August 2005	• Added information about the type of character encoding that the client supports. See "System Requirements" on page 9.

Chapter 1

Introduction

This chapter provides an introduction to the CyberSource® Simple Order API Client for ASP. The chapter includes these sections:

Who Should Read This Guide

You should use this guide if you are a merchant or a reseller of CyberSource services who wants to use ASP COM objects to access CyberSource's Internet Commerce Suite^{SM} (ICS) services.

You can be:

- A merchant or reseller using the Business Center at https://businesscenter.cybersource.com.
- An enterprise merchant using the Enterprise Business Center at https://ebc.cybersource.com.

Most of this guide's content is applicable to both types of users. The guide indicates if particular information is applicable only to one type of user.

You will need to know which type of user you are when you install the client and generate security keys. If you are not sure whether you should use the Business Center or the Enterprise Business Center, look at the registration email that you received when you or someone at your company opened your account with CyberSource. The email includes your login ID and password and indicates whether to use the login with the Business Center or the Enterprise Business Center.

If you do not have access to the registration email or have lost it, contact your account manager.

Other Necessary Documentation

This section lists other documents that you need to integrate with CyberSource.

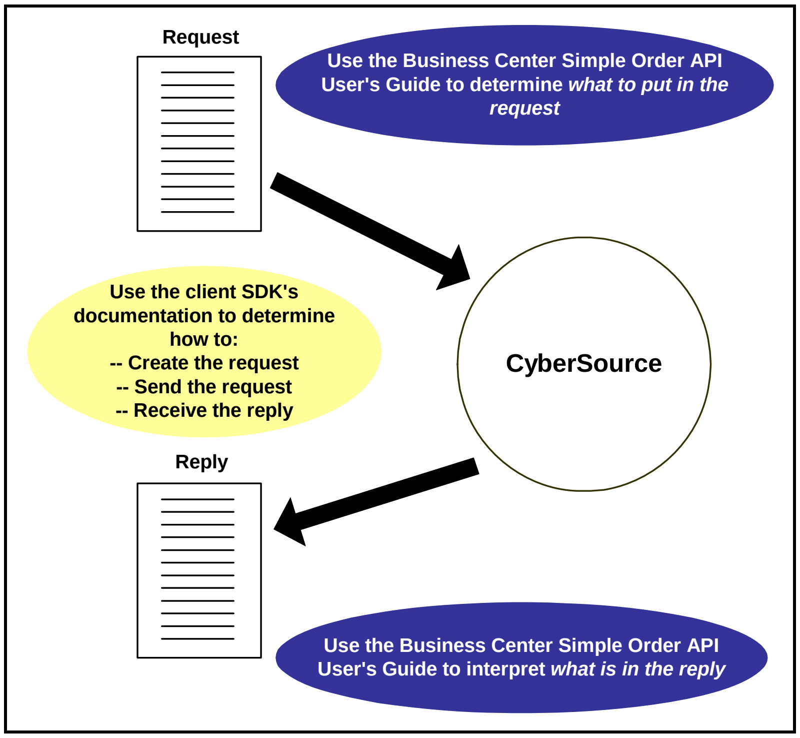
General Documentation

This documentation is available in the client package download:

- `README` file
- `CHANGES` file
- Sample code files (see "Sample Scripts" on page 6)

Business Center Users

For merchants and resellers using the Business Center, the Business Center Simple Order API User's Guide describes the API for using the CyberSource ICS services for payment processing. This and other related documentation is available in the Business Center.

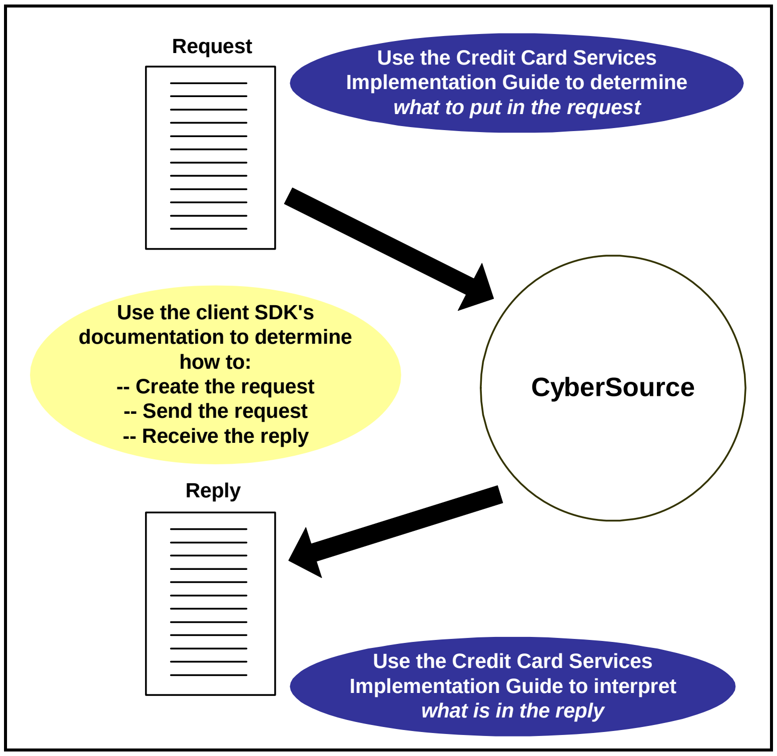

Request
Use the Business Center Simple Order API User's Guide to determine *what to put in the request*
Use the client SDK's documentation to determine how to:
-- Create the request
-- Send the request
-- Receive the reply
CyberSource
Reply
Use the Business Center Simple Order API User's Guide to interpret *what is in the reply*

Enterprise Business Center Users

If you use the Enterprise Business Center, you need these guides:

- Credit Card Services Implementation Guide: Describes the API for using the CyberSource ICS Credit Card Services. Make sure when you read this guide that you read the Simple Order API chapter and **NOT** the SCMP API chapter.
- Reporting Developer's Guide: Describes how to download the CyberSource reports you will be using to manage your orders.
- Implementation guides for any other ICS services you are planning to use. These are available in the documentation area of the Support Center.

Important The Simple Order API was originally referred to as the Web Services API in the CyberSource documentation. You may still see old references to the Web Services API in some locations.


Request
Use the Credit Card Services Implementation Guide to determine *what to put in the request*
Use the client SDK's documentation to determine how to:
-- Create the request
-- Send the request
-- Receive the reply
CyberSource
Reply
Use the Credit Card Services Implementation Guide to interpret *what is in the reply*

Using ASP in a Hosted Environment

If you are operating in a hosted environment (with an Internet Service Provider hosting your Web store), then read this section.

To use the CyberSource Simple Order API client for ASP, you must register several dll's (Microsoft Dynamic Linked Libraries). These dll's ensure that your transactions are secure while being sent to CyberSource. If you use a hosted environment, you must check with your hosting provider (ISP) to make sure that they support the registration of custom dll's. If you are unable to find any documentation related to your hosting provider's support of dll's, then contact them with the following statement:

> **CyberSource requires the registration of three dll's for use by my e-commerce software. These dll's need to be registered using regsvr32. CyberSource ensures the safety and functionality of these dll's. Please let me know your policy for supporting this implementation.**

Note that it is also possible that other merchants who use your hosting provider may also use CyberSource, and so the hosting provider may have already installed the CyberSource ASP client and registered the dll's. In that case, CyberSource recommends you verify with your hosting provider which version of the client they have installed and registered. If the client you want to use is newer, ask them to register the newer dll's.

If you have any questions regarding the above information or installation of the client, please contact Customer Support.

Choosing Your API and Client Version

You need to choose an API and a client.

API Variation

With this client package, you can use either of these variations of the Simple Order API:

- Name-value pairs, which are simpler to use
- XML, which requires you to create and parse XML documents

The test that you run immediately after installing the client uses name-value pairs.

Client Versions

CyberSource updates the Simple Order API on a regular basis to introduce new API fields and functionality. To determine the latest version of the server-side API for the ICS services, go to https://ics2ws.ic3.com/commerce/1.x/transactionProcessor/.

The Simple Order API Client for ASP also has a version, but it is not the same as the API version. The client version represents the version of the client-side code that you use to access the ICS services.

When configuring the client, you indicate which version of the API you want to use. When setting this parameter, do not use the current version of the client; use the current version of the API.

Sample Code Available

The client contains sample scripts and sample ASP pages.

Basic ASP Page Example

Creating an ASP page that uses VBScript to access ICS services is easy. The example below shows the primary code required to send a Simple Order API request for credit card authorization and process the reply. The example uses name-value pairs. For a more complete example, see the sample program and sample ASP pages included in the package (see "Sample Scripts" on page 6). Chapter 4, "Using Name-Value Pairs," on page 29 shows you how to create the code.

```
' Set oMerchantConfig properties
Dim oMerchantConfig
set oMerchantConfig =
    Server.CreateObject( "CyberSourceWS.MerchantConfig" )
oMerchantConfig.MerchantID = "infodev"
oMerchantConfig.KeysDirectory = "\keys"
oMerchantConfig.SendToProduction = "0"
oMerchantConfig.TargetAPIVersion = "1.18"

' Create request hashtable
Dim oRequest
set oRequest = Server.CreateObject( "CyberSourceWS.Hashtable" )

' We want to do credit card authorization in this example
oRequest.Value( "ccAuthService_run" ) = "true"

' Add required fields
oRequest.Value( "billTo_firstName" ) = "Jane"
oRequest.Value( "billTo_lastName" ) = "Smith"
oRequest.Value( "billTo_street1" ) = "1295 Charleston Road"
oRequest.Value( "billTo_city" ) = "Mountain View"
oRequest.Value( "billTo_state" ) = "CA"
oRequest.Value( "billTo_postalCode" ) = "94043"
oRequest.Value( "billTo_country" ) = "US"
oRequest.Value( "billTo_email" ) = "jsmith@example.com"
oRequest.Value( "card_accountNumber" ) = "4111111111111111"
oRequest.Value( "card_expirationMonth" ) = "12"
oRequest.Value( "card_expirationYear" ) = "2010"
oRequest.Value( "purchaseTotals_currency" ) = "USD"

' There are two items in this example
oRequest.Value( "item_0_unitPrice" ) = "12.34"
oRequest.Value( "item_1_unitPrice" ) = "56.78"

' Add optional fields here according to your business needs

' create Client object and send request
dim oClient
set oClient = WScript.CreateObject( "CyberSourceWS.Client" )
dim varReply, nStatus, strErrorInfo
nStatus = oClient.RunTransaction( _
          oMerchantConfig, Nothing, Nothing, oRequest, _
          varReply, strErrorInfo )

' Handle the reply. See “Handling the Return Status” on page 32.
ProcessReply nStatus, varReply, strErrorInfo
```

Sample Scripts

The client contains two sample scripts, one for using name-value pairs and one for using XML. See “Testing the Client” on page 11 or see the README file for more information about using the `AuthCaptureSample.wsf` script to test the client.

- For name-value pairs: See `AuthCaptureSample.wsf` in `<`*`installation directory`*`>\samples\nvp`.
- For XML: CyberSource recommends that you examine the name-value pair sample code listed above before implementing your code to process XML requests.

  For the XML sample code, see `AuthSample.wsf` in `<`*`installation directory`*`>\samples\xml`. Also see the `auth.xml` XML document that the script uses.

Sample ASP Pages

The client download package also includes sample ASP pages in the <*installation directory*>\samples\store directory. You also have shortcuts to several of the files from the Start menu at Start > Programs > CyberSource Simple Order API for ASP > Sample Store.

Table 1 Files in aspSample Directory

File	Description
`global.asa`	Sets up the MerchantConfig object and stores it in a Session variable.
`EditFile.bat`	Batch file used with the Start menu program shortcuts. Not used directly with the ASP sample pages.
`checkout.asp`	Displays the contents of the shopping basket, prompts for address and payment information.
`checkout2.asp`	Authorizes the order and displays the result.
`store_footer.asp`	Footer used in the checkout pages.
`store_header.asp`	Header used in the checkout pages.

To use the sample ASP pages:

1. In IIS, create a virtual directory that points to the <*installation directory*>\samples\store directory.
2. Go to Start > Programs > CyberSource Simple Order API for ASP > Sample Store > Edit configuration script (global.asa) and set the properties of the MerchantConfig object to the correct values. For more information about the MerchantConfig settings, see "MerchantConfig Properties" on page 16.
3. If you have enabled logging, make sure that the Internet guest user account (default is IUSR_<machine name>) has Write permission on the `logs` directory.
4. Open a Web browser and type the following URL:

   ```
   http://localhost/<virtual directory>/checkout.asp
   ```

Chapter 2

Installing and Testing the Client

This chapter explains how to install, configure, and test the Simple Order API Client for ASP and includes these sections:

System Requirements

Your system must meet the following minimum requirements:

- Windows Installer 2.0 or later

  If the msi file does not run, you may not have a suitable Windows Installer. You can download version 2.0 or later from the Microsoft Windows Installer Downloads page.

- Windows 2000 or XP

- If using Windows XP: MSXML 4.0 Service Pack 2 (Microsoft® XML Core Services)

  For the download, go to the Microsoft site, scroll to the bottom of the page and download the `msxml.msi` file.

- If using an operating system earlier than Windows 2000 SP3: WinHTTP 5.0

Note CyberSource recommends that your system use at least Windows 2000 SP3 because it comes with WinHTTP 5.1, which fixes bugs in the 5.0 version.

The SDK supports UTF-8 encoding.

Important Failure to configure your client API host to a unique, public IP address will cause inconsistent transaction results.

CyberSource strongly recommends configuring your client API host to a unique, public IP address. Failure to do so will cause inconsistent transaction results. The client API request ID algorithm uses a combination of IP address and system time, along with other values. In some architectures it is possible that this combination will not yield unique identifiers.

Creating a Security Key

The first thing you need to do is create your security key. The client uses the security key to add a digital signature to every request that you send. This signature helps ensure that no one else can use your CyberSource account to process orders. You will specify the location of your key when you configure the client.

Important If you do not protect your security key, the security of your CyberSource account may be compromised.

To generate your key, you log in to either the Business Center or the Enterprise Business Center, depending on which one you normally will use. If you are not sure which one to use, see "Who Should Read This Guide" on page 1.

Business Center Users

Resellers

If you are a reseller using the Business Center, you can use the CyberSource Security Libraries for the Simple Order API to create keys on behalf of your merchants.

Merchants

If you are a merchant using the Business Center, you create your security key by using a Java applet that you access through the Business Center. See Appendix A, "Merchants Using the Business Center," on page 55 for instructions.

Enterprise Business Center Users

If you are an enterprise merchant using the Enterprise Business Center, you create your key by using a Java applet that you access through the Enterprise Business Center. See Appendix A, "Merchants Using the Enterprise Business Center," on page 57 for instructions.

Installing the Client

To install the client:

1. Go to the client downloads page on the Support Center.
2. Download the latest client package. You can save the file in any directory.
3. Double-click the downloaded msi file.

   The installation wizard opens.
4. Follow the instructions on the screen.

   The client is now installed on your system. The default installation directory for the package contents is the `c:\simapi-asp-n.n.n` directory. The Start menu now includes a new item for the CyberSource Simple Order API for ASP.
5. Test the client. See "Testing the Client" on page 11.

You have now installed and tested the client. You are ready to create your own code for requesting ICS services. Finish reading this chapter, and then move on to either Chapter 4, "Using Name-Value Pairs," on page 29 if you plan to use name-value pairs, or Chapter 5, "Using XML," on page 41 if you plan to use XML.

Testing the Client

After you have installed the client, test it immediately to ensure the installation was successful. To test the client:

1. Modify the test script to include your merchant ID and directory to your security key:
   a. Select Start > Programs > CyberSource Simple Order API for ASP > NVP Sample > Edit configuration script (config.vbs).
   b. In the `config.vbs` script, replace the default values for your merchant ID and the directory where your security key is stored. The lines of code look like this before you edit them:

      ```
      oMerchantConfig.MerchantID = "your_case_sensitive_merchant_id"
      oMerchantConfig.KeysDirectory = "your_keys_dir(e.g.
      baseDir\simapi-net-n.n.n\keys)"
      ```
   c. Save the script.
2. Run the test by selecting Start > Programs > CyberSource Simple Order API for ASP > NVP Sample > Run test script (AuthCaptureSample.wsf). This test requests an authorization and then a follow-on capture if the authorization is successful.

   A command shell opens and displays the results of the test.

   - If the test is successful, you see a decision of ACCEPT for both the credit card authorization and the follow-on capture.
   - If the test is not successful, you see a different decision value or an error message.

If the test fails:

1. Check to see that the `config.vbs` changes that you made in Step b above are correct.
2. Run the test again.
3. If the test still fails, look at the error message and determine the return status value (a number from 1 to 8).
4. See the descriptions of the status values in Chapter 3, "Possible Return Status Values," on page 26, and follow any instructions given there for the error you received.
5. Run the test again.
6. If the test still fails, contact Customer Support.

Going Live

When you have completed all of your system testing and are ready to accept real transactions from your customers, you are ready to go live.

Business Center Users

If you are a Business Center user, you can use the Business Center site to go live (see the Business Center User's Guide for more information).

Important You must also configure your client so that it sends transactions to the production server and not the test server. See the description of the SendToProduction property in Table 1 on page 16.

After you go live, use real card numbers and other data to test every card type you support. Because these are real transactions in which you are buying from yourself, use small amounts, such as one dollar, to do the tests. Process an authorization, then capture the authorization, and later refund the money. Use your bank statements to verify that money is deposited into and withdrawn from your merchant bank account as expected. If you have more than one CyberSource merchant ID, test each one separately.

Enterprise Business Center Users

If you are an Enterprise Business Center user, use the procedure below to go live.

When you go live, your CyberSource account is updated so that you can send transactions to CyberSource's production server. If you have not already done so, you will need to provide your banking information to CyberSource so that your processor can deposit funds to your merchant bank account.

To go live:

1. Go to the CyberSource Knowledgebase at http://www.cybersource.com/esupport.
2. Submit a question (click the **Submit a Question** link).

3 Log in with your CyberSource merchant ID and password (the same ones you use to log in to the Enterprise Business Center).

4 On the **Submit a Question** page, fill in the contact information.

5 For the **General Topic**, select **Getting Set Up on CyberSource**.

6 For the **Specific Topic**, select **Go Live**.

7 For the **Subject**, enter “Go Live”.

8 In the question field, indicate that you would like to go live.

9 If you have not already submitted your banking information to CyberSource, include the information in the question field and select the check box for **This question contains sensitive banking information**.

10 Click **Submit question**.

You will receive an email with your support ticket number, and a CyberSource representative will contact you to complete the process.

11 Once CyberSource has confirmed that you are live, make sure to update your system so that you send requests to the production server and not the test server. See the description of the SendToProduction property in Table 1 on page 16.

After you go live, use real card numbers and other data to test every card type, currency, and CyberSource application that your integration supports. Because these are real transactions in which you are buying from yourself, use small amounts, such as one dollar, to do the tests. Use your bank statements to verify that money is deposited into and withdrawn from your merchant bank account as expected. If you have more than one CyberSource merchant ID, test each one separately.

Deploying the Client to Another Computer

You can deploy the client to another computer without running the installer that CyberSource provided. To do this:

1 Copy to the destination computer all the binaries (`*.dll` and `*.pdb`) in the `<`*`installation directory`*`>/lib` directory, or include them in your own installer.

2 If the destination computer does not have MSXML 4.0 Service Pack 2 (Microsoft XML Core Services), copy the following files from `%WINDIR%\system32` into the same directory on the destination computer:

```
msxml4.dll
msxml4r.dll
```

3 Register `msxml4.dll`:

- a Open a command prompt.
- b Go to the `%WINDIR%\system32` directory.
- c At the prompt, type `regsvr32 msxml4.dll`

4 Register the CyberSource `*.dll` files:

- a Open a command prompt.
- b Go to the directory where the binaries are installed on the destination computer.

c At the prompt, type

```
regsvr32 CybsWSSecurity.dll
regsvr32 CyberSourceWS.dll
```

You have deployed the client to the destination computer.

Updating the Client to Use a Newer API Version

CyberSource will periodically update the Simple Order API (previously called the Web Services API). You can update your existing client to work with the new API version. Go to https://ics2ws.ic3.com/commerce/1.x/transactionProcessor/ for a list of the available API versions.

To update the client to use a newer API version, update the value for the TargetAPIVersion in the MerchantConfig object (see Table 1, "MerchantConfig Properties," on page 16). For example, to use the 1.18 version of the API, set the TargetAPIVersion to `"1.18"`.

Chapter 3

Client Objects

This chapter describes the client's objects:

MerchantConfig

The MerchantConfig object controls these types of settings for the client:

- Security
- Server
- Timeout
- Logging

MerchantConfig Properties

The data type for all of the properties is string.

Table 1 MerchantConfig Properties

Property	Description
LogString	Returns a string containing a comma-separated list of the MerchantConfig object's properties and their values.
MerchantID	Merchant ID. If you specify a merchant ID in the request (either using the Hashtable object if using name-value pairs, or the XML document if using XML), the one in the request takes precedence over the one in the MerchantConfig object.
KeysDirectory	Location of the merchant's security key. UNC paths are allowed. The client includes a `keys` directory that you can use. **Note** CyberSource recommends that you store your key locally for faster request processing.
SendToProduction	Flag that indicates whether the transactions for this merchant should be sent to the production server. Use one of these values: • `"0"`: Do not send to the production server; send to the test server. • `"1"`: Send to the production server.
TargetAPIVersion	Version of the Simple Order API to use, such as `1.18`. Do not set this property to the current version of the client. Instead set it to an available API version. See "Client Versions" on page 4 for more information. **Note** Go to https://ics2ws.ic3.com/commerce/1.x/transactionProcessor/ to see an up-to-date list of the available versions. See the *Simple Order API Release Notes*, for information about what has changed in each version.
KeyFilename	Name of the security key filename for the merchant, if you have changed it from the default value of ***<merchantID>***`.p12`.
ServerURL	Alternative server URL to use. See Using Alternate Server Properties below for more information. Give the complete URL, as it will be used exactly as you specify here.
NamespaceURI	Alternative namespace URI to use. See Using Alternate Server Properties below for more information. Give the complete namespace URI, as it will be used exactly as you specify here.
EffectiveKeyFilename	If AlternateKeyFilename is not empty, this property simply returns the value of AlternateKeyFilename. If AlternateKeyFilename is empty, the property looks for a bstrMerchantID parameter that you provide. If you do not provide one, it derives the key filename from the MerchantID property.

Table 1 MerchantConfig Properties (Continued)

Property	Description
EffectiveServerURL	If AlternateServerURL is not empty, this property simply returns the value of AlternateServerURL. Otherwise, it derives the server URL from the TargetAPIVersion and SendToProduction properties.
EffectiveNamespaceURI	If AlternateNamespaceURI is not empty, this property simply returns the value of AlternateNamespaceURI. Otherwise, it derives the namespace URI from the property TargetAPIVersion. This is particularly useful if you are passing in the request as XML and do not want to hardcode the namespace URI in the XML string. You can call this property to dynamically set the namespace URI of your `<requestMessage>` element.
Timeout	Length of timeout in seconds. The default is 110.
EnableLog	Flag directing the client to log transactions and errors. Possible values: • `0`: Do not enable logging. • `1`: Enable logging. This property is ignored if you use a Logger object with the request. **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. 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. For more information about PCI and PABP requirements, see www.visa.com/cisp.
LogDirectory	Directory to write the log file to. UNC paths are allowed. Note that the client will not create this directory for you; you must specify an existing directory.The client includes a `logs` directory that you can use. This property is ignored if you use a Logger object with the request.

Table 1 MerchantConfig Properties (Continued)

Property	Description
LogFilename	Log filename. The client uses `cybs.log` by default. This property is ignored if you use a Logger object with the request.
LogMaximumSize	Maximum size in megabytes for the log file. When the log file reaches this size, it is archived into `cybs.log.<`***`yyyymmdd`***`T`***`hhmmssxxx`***`>` and a new log file is started. The ***`xxx`*** indicates milliseconds. The default value is `"10"`. This property is ignored if you use a Logger object with the request.

Example Setting Merchant Configuration Object Properties

This example sets the MerchantID property of a MerchantConfig object.

```
Dim oMerchantConfig
set oMerchantConfig =
    Server.CreateObject( "CyberSourceWS.MerchantConfig" )
oMerchantConfig.MerchantID = "merchant123"
```

Using Alternate Server Properties

Use the ServerURL and NamespaceURI properties if CyberSource changes the convention used to specify the server URL and namespace URI, but has not updated the client yet.

For example, these are the server URLs and namespace URI for accessing the ICS services using the Simple Order API version 1.18:

- Test server URL:

  `https://ics2wstest.ic3.com/commerce/1.x/transactionProcessor/`
- Production server URL:

  `https://ics2ws.ic3.com/commerce/1.x/transactionProcessor/`
- Namespace URI:

  `urn:schemas-cybersource-com:transaction-data-1.18`.

If in the future CyberSource changes these conventions, but has not yet provided a new version of the client, you will be able to configure your existing client to use the new server and namespace conventions required by the ICS server.

MerchantConfig Method

MerchantConfig has one method:

Copy

Table 2 Copy Method

Syntax	`Copy( oMerchantConfig, fNonEmptyOnly )`
Description	Copies the properties of `oMerchantConfig`.
Returns	Nothing
Parameters	• `oMerchantConfig`: The specific MerchantConfig object. • `fNonEmptyOnly`: Flag indicating which properties to copy. Set to `true` to copy only the properties that are not null or empty. Set to `false` to copy all properties, whether or not null or empty.

ProxyConfig

The ProxyConfig object controls proxy settings.

ProxyConfig Properties

The data type for all of the properties is string.

Table 3 ProxyConfig Properties

Property	Description
LogString	Returns a string containing a comma-separated list of the ProxyConfig object's properties and their values.
ProxySetting	Type of proxy server to use. Use one of these values: • `"0"`: Use WinHTTP proxy configuration, which is set using `proxycfg.exe`, a WinHTTP tool available in Windows 2000 SP3 and later, or as part of the Resource Kit in earlier Windows versions. • `"1"`: Do not use a proxy server. • `"2"`: Use the proxy server specified in the ProxyServer property.
ProxyServer	Proxy server to use. Only used if ProxySetting=`"2"`. Allowable formats include: • *`<http://>server<:port>`* • *`<http://>IP address<:port>`* The `http://` and `port` are optional.
Username	User name used to authenticate against the proxy server if required. If the proxy server requires the domain name during authentication, add the domain name and a backslash: ***`<domain>`***`\`***`<username>`***
Password	Password used to authenticate against the proxy server, if required.

Example Setting ProxyConfig Properties

This example sets a ProxyConfig object to use a proxy server.

```
Dim oProxyConfig
set oProxyConfig =
    Server.CreateObject( "CyberSourceWS.ProxyConfig" )
oProxyConfig.ProxySetting = "2"
oProxyConfig.ProxyServer = varServer ‘ variable that contains server
oProxyConfig.Username = varUsername  ‘ variable that contains
                                     ‘ username
oProxyConfig.Password = varPassword  ‘ variable that contains
                                     ‘ password
```

ProxyConfig Method

ProxyConfig has one method:

Copy

Table 4 Copy Method

Syntax	`Copy( oProxyConfig, fNonEmptyOnly )`
Description	Copies the properties of `oProxyConfig`.
Returns	Nothing
Parameters	• `oProxyConfig`: The specific ProxyConfig object. • `fNonEmptyOnly`: Flag indicating which properties to copy. Set to `true` to copy only the properties that are not null or empty. Set to `false` to copy all properties, whether or not null or empty.

Hashtable

You use the Hashtable object to store the request and reply information. You use the Hashtable object only if you are using name-value pairs, and not XML.

Hashtable Properties

The data type for all of the properties is string.

Table 5 Hashtable Properties

Property	Description
LogString	Returns a newline-character–separated list of the Hashtable object's properties and their values.
Value	Reads or writes a value for the specified name. Accepts the name as a parameter. See the example below.
Content	Reads the content of the Hashtable as a string. Accepts a delimiter as a parameter, which is used to concatenate the name-value pairs. See the example below.
Names	Returns the array of name-value pairs. See the example below.
Overwrite	Gets or sets the overwrite mode. Allowable values: • `True`: If the name that is being set already exists, its value is replaced with the new one. This is the default value upon instantiation. • `False`: The current value for the name is kept. The overwrite mode that you set is used for all field assignments until you reset it to the opposite value. See the example below.

The following examples show how to use the Hashtable object's properties.

Example Using the Value Property of the Hashtable Object

The following code shows how to set name-value pairs in a Hashtable object.

```
Dim oRequest
set oRequest = Server.CreateObject( "CyberSourceWS.Hashtable" )
oRequest.Value( "ccAuthService_run" ) = "true"
oRequest.Value( "card_accountNumber" ) = "4111111111111111"
```

Note For the Hashtable object, the Value property is the default property. Thus you can leave out the ".Value" when setting and retrieving values. For example, the following syntax is valid:

```
oRequest( "ccAuthService_run" ) = "true"
```

Example Using the Content Property of the Hashtable Object

The following code shows how to get the content of the Hashtable object as a string:

```
content = oRequest.Content (vbCRLf)
```

Example Using the Names Property of the Hashtable Object

The following code shows how to list the name-value pairs in the Hashtable object

```
names = oRequest.Names
For Each name in names
    Response.Write name & "=" & oRequest.Value(name)
Next
```

Example Using the Overwrite Property of the Hashtable Object

```
oRequest.Overwrite = true
oRequest.Value("name") = "valueA"
oRequest.Value("name") = "valueB"    ' overwrites valueA
oRequest.Overwrite = false
oRequest.Value("name") = "valueC"    ' does not overwrite anymore
                                     ' name=valueB
```

Hashtable Method

The Hashtable object has one method:

Delete

Table 6 Delete Method

Syntax	`Delete ( bstrName )`
Description	Deletes the entry for `bstrName` in the Hashtable, if it exists.
Returns	Nothing
Parameters	• `bstrName`: The name for the name-value pair to delete.

Logger

Use the Logger object if you want to log more information than the client already does.

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.

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.

For more information about PCI and PABP requirements, see www.visa.com/cisp.

Logger Properties

The data type for all of the properties is string.

Table 7 Logger Properties

Property	Description
Filename	Name of the log file. Field can include an absolute or relative path. For example, `c:\logs\file.log` or `..\file.log`. UNC paths are allowed. This property is read-only and can only be set by calling the PrepareFile method.
MaximumSize	Maximum size in megabytes for the log file. When the log file reaches this size, it is archived into <***`logfilename`***>.<***`yyyymmdd`***`T`***`hhmmssxxx`***> and a new log file is started. The ***`xxx`*** indicates milliseconds. The default value is `"10"`. This property is read-only and can only be set by calling the PrepareFile method.

Logger Methods

The Logger object has the following methods:

- PrepareFile
- Log
- LogTransactionStart

PrepareFile

Table 8 PrepareFile Method

Syntax	`PrepareFile( bstrFilename, bstrMaxSizeInMB )`
Description	Initializes the properties Filename and MaximumSize with `bstrFilename` and `bstrMaxSizeInMB`, respectively. The method then prepares the file by checking if the file has exceeded the maximum size. If it has, the method archives the file and creates a new empty file. You must call this method before calling the other Logger methods.
Returns	Nothing
Parameters	• `bstrFilename`: The file name to use. • `bstrMaxSizeInMB`: Maximum size for the file.

Log

Table 9 Log Method

Syntax	`Log( bstrType, bstrText )`
Description	Writes an entry to the log file by using this format *`<timestamp> <thread id> <logType> > <logText>`* Example: 2003-11-07 22:47:12.484 1512 INFO > Missing field
Returns	Nothing
Parameters	• `bstrType`: A short name describing what the log entry is for. For example, the client uses values such as REQUEST, REPLY, FAULT, INFO, TRANSTART, ERROR. To make the log file easy to view, it is recommended that you use only a maximum of 9 characters for this parameter. • `bstrText`: The text to be logged in the file.

LogTransactionStart

Table 10 LogTransactionStart Method

Syntax	`LogTransactionStart( oMerchantConfig, oProxyConfig )`
Description	Writes an entry that marks the start of a transaction and also logs the content of the supplied `oMerchantConfig` and `oProxyConfig` objects. Note that `oProxyConfig` can be Nothing, but `oMerchantConfig` must point to a MerchantConfig object. This method is optional but recommended. If you do not supply a Logger object in the request, the client creates its own Logger object and runs the LogTransactionStart method itself, creating a TRANSTART entry in the log file. If you supply a Logger object in the request, the client does not call this method on your Logger object, so you must call the method yourself before logging anything else in the log file. Alternately, you can choose some other way to indicate the start of the transaction in the log file.
Returns	Nothing
Parameters	• `oMerchantConfig`: The MerchantConfig object used for the transaction being logged. • `oProxyConfig`: The ProxyConfig object used for the transaction being logged. Can be Nothing.

Fault

The Fault object contains information if a fault occurs when the client sends the request. The following table lists the Fault object's properties. The data type for all of the properties is string.

Table 11 Fault Properties

Property	**Description**
FaultDocument	The entire, unparsed fault document.
FaultCode	The fault code, which indicates where the fault originated.
FaultString	The fault string, which describes the fault.
RequestID	The requestID for the request.

Client

The Client object has one method.

RunTransaction

Table 12 RunTransaction Method

Syntax	`nStatus = RunTransaction ( oMerchantConfig, oProxyConfig, oLogger, varRequest, varReply, bstrErrorInfo)`
Description	Sends the request to the ICS server and receives the reply
Returns	A number from 0 to 8 that indicates the status of the request. See "Possible Return Status Values" on page 26 for descriptions of the values.
Parameters	• `oMerchantConfig`: A MerchantConfig object. Required. • `oProxyConfig`: A ProxyConfig object. May be Nothing. • `oLogger`: A Logger object. May be Nothing. Supply one only if you want to log more information than what the client logs. If you supply a Logger object, the client does not call its LogTransactionStart method. See "LogTransactionStart" on page 25 for more information. • `varRequest`: A variant that can be one of the items listed below in Possible varRequest Values. • `varReply`: A variant that can be one of the items listed below in Possible varReply Values. • `bstrErrorInfo`: If status is not 0, this contains a BSTR that has more information about the status. If status is 0, this parameter is empty.

Possible Return Status Values

The runTransaction method returns a status indicating the result of the request. The following table describes the possible status values.

Table 13 Possible Status Values

Value	Description
0	**Result:** The client successfully received a reply. **Manual action to take:** None
1	**Result:** An error occurred before the request could be sent. This usually indicates a configuration problem with the client. **Manual action to take:** Fix the problem described in `bstrErrorInfo.`

Table 13 Possible Status Values (Continued)

Value	Description
2	**Result:** An error occurred while sending the request. **Manual action to take:** None.
3	**Result:** An error occurred while waiting for or retrieving the reply. **Manual action to take:** Check the Transaction Search screens on the Business Center (for merchants using the Business Center) or Enterprise Business Center (for enterprise merchants) to see if the request was processed, and if so, if it succeeded. Update your transaction database appropriately.
4	**Result:** The client received a reply or a fault, but an error occurred while processing it. **Value of varReply:** Contains the server’s raw reply. **Manual action to take:** Examine the contents of `varReply`. If you cannot determine the status of the request, then check the Transaction Search screens on the Business Center (for merchants using the Business Center) or Enterprise Business Center (for enterprise merchants) to see if the request was processed, and if so, if it succeeded. Update your transaction database appropriately.
5	**Result:** The server returned a fault with FaultCode set to CriticalServerError. **Manual action to take:** Check the Transaction Search screens in Business Center to see if the request succeeded. When searching for the request, use the request ID provided in the RequestID property of the Fault object.
6	**Result:** The server returned a fault with FaultCode set to ServerError, indicating a problem with the ICS server. **Manual action to take:** None
7	**Result:** The server returned a fault with FaultCode set to a value other than ServerError or CriticalServerError. Indicates a possible problem with merchant status or the security key. Could also indicate that the message was tampered with after it was signed and before it reached the ICS server. **Manual action to take:** Examine the FaultString and fix the problem. You may need to generate a new security key, or you may need to contact Customer Support if there are problems with your merchant status.
8	**Result:** The server returned an HTTP status code other than 200 (OK) or 504 (gateway timeout). Note that if a 504 gateway timeout occurs, then the status=3. **Value of varReply:** Contains the HTTP response body, or if none was returned, the literal `"(response body unavailable)"`. **Manual action to take:** None.

Possible varRequest Values

The `varRequest` parameter can be one of the following:

- Hashtable object: Makes the client use the name-value pairs
- DOMDocument40 object (part of MSXML 4.0): Makes the client use XML
- BSTR containing XML data: Makes the client use XML

Possible varReply Values

The `varReply` parameter can be one of these:

- Hashtable object: If `varRequest` was a Hashtable object and the return status is `0`
- DOMDocument40 object: If `varRequest` was a DOMDocument40 object and the return status is `0`
- BSTR containing the XML reply: If `varRequest` was a BSTR and the return status is 0
- Fault: If the return status is `5`, `6`, or `7`. See “Fault” on page 25 for information about the Fault object.
- BSTR containing the raw reply (the entire response body): If the return status is `4` or `8`
- Empty if the return status is `1`, `2`, or `3`

The following table summarizes the value of `varReply` that you receive for each status value. The `bstrErrorInfo` parameter in the reply always contains information about the status.

Table 14 Summary of varReply Values by Status

Status	0	1	2	3	4	5	6	7	8
DOMDocument40 or BSTR with XML or Hashtable (name-value pairs)	x								
BSTR with raw reply					x				x
Fault						x	x	x	

Chapter 4

Using Name-Value Pairs

This chapter explains how to use the client to request ICS services by using name-value pairs and includes these sections:

Other Necessary Documentation

Make sure that you have the guide that explains the API fields you need to use in your requests:

For merchants and resellers using the Business Center:

- Business Center Simple Order API User's Guide: Describes the API for using the CyberSource ICS Credit Card Services and other payment services.

For merchants using the Enterprise Business Center:

- Credit Card Services Implementation Guide: Describes the API for using the CyberSource ICS Credit Card Services. Make sure when you read this guide that you read the Simple Order API chapter, *not* the SCMP API chapter.
- Implementation guides for any other ICS services you are planning to use. These are available in the documentation area of the Support Center.

Requesting ICS Services

To request CyberSource ICS services, write code that:

- Collects information for the ICS services that you will use
- Assembles the order information into requests
- Sends the requests to the CyberSource server

Important The CyberSource servers do not support persistent HTTP connections.

- Processes the reply information

The instructions in this chapter explain how to write VBScript that requests ICS services. You will need another guide, as described in "Other Necessary Documentation" on page 29, to get a list of the API fields to use in your requests.

Creating and Sending Requests

Note The code in this chapter's example is incomplete. For a complete sample program, see the `AuthCaptureSample.wsf` file in the `<`*`installation directory`*`>\samples\nvp` directory, or see the sample ASP pages.

To use any ICS service, you must create and send a request that includes the required information for that service.

The following example shows basic code for requesting ICS services. In this example, Jane Smith is buying an item for $29.95.

Creating the MerchantConfig Object

First create a MerchantConfig object and set your merchant ID and other basic transaction settings:

```
Dim oMerchantConfig
set oMerchantConfig =
    Server.CreateObject( "CyberSourceWS.MerchantConfig" )
oMerchantConfig.MerchantID = "infodev"
oMerchantConfig.KeysDirectory = "\keys"
oMerchantConfig.SendToProduction = "0"
oMerchantConfig.TargetAPIVersion = "1.18"
```

Creating an Empty Request Hashtable

You next create a Hashtable to hold the request fields:

```
Dim oRequest
set oRequest = Server.CreateObject( "CyberSourceWS.Hashtable" )
```

Adding the Merchant ID

`You next add the CyberSource merchant ID to the request.`

Note If you specify a merchant ID in the Hashtable object, it overrides the merchant ID you specify in the MerchantConfig object.

```
oRequest.Value( "merchantID" ) = "infodev"
```

Adding Services to the Request Hashtable

You next indicate the service you want to use by adding the field to the request. For example, to request a credit card authorization:

```
oRequest.Value( "ccAuthService_run" ) = "true"
```

Requesting a Sale

You can request multiple services by adding additional fields to the request. For example, if you fulfill the order immediately, you can request credit card authorization and capture together (referred to as a “sale”):

```
oRequest.Value( "ccAuthService_run" ) = "true"
oRequest.Value( "ccCaptureService_run" ) = "true"
```

Adding Service-Specific Fields to the Request Hashtable

You next add the fields that are used by the services that you are requesting. If you request multiple services and they share common fields, you only need to add the field once.

```
oRequest.Value( "billTo_firstName" ) = "Jane"
oRequest.Value( "billTo_lastName" ) = "Smith"
oRequest.Value( "card_accountNumber" ) = "4111111111111111"
oRequest.Value( "item_0_unitPrice" ) = "29.95"
```

The example above shows only a partial list of the fields you need to send. Refer to "Other Necessary Documentation" on page 29 for information about the guide or guides that list all of the fields for the service(s) you are requesting.

Sending the Request

You next create a Client object and send the request:

```
dim oClient
set oClient = WScript.CreateObject( "CyberSourceWS.Client" )
dim varReply, nStatus, strErrorInfo
nStatus = oClient.RunTransaction( _
          oMerchantConfig, Nothing, Nothing, oRequest, varReply, _
          strErrorInfo )
```

Interpreting Replies

Handling the Return Status

You next handle the `nStatus` returned by the RunTransaction method. The `nStatus` indicates whether the ICS server received the request, if the client received the reply, and if there were any errors or faults during transmission. See "Possible Return Status Values" on page 26 for descriptions of each status value. For a different example, see the `AuthCaptureSample.wsf` file in the `<`*`installation directory`*`>\samples\nvp` directory.

```
if nStatus = 0 then
  dim strContent
  ' Read the value of the "decision" in the varReply hashtable.
  ' If decision=ACCEPT, indicate to the customer that the request was successful.
  ' If decision=REJECT, indicate to the customer that the order was not approved.
  ' If decision=ERROR, indicate to the customer an error occurred and to try again
  ' later.
  strContent = GetReplyContent( varReply ) ' get reply contents
  ' See “Processing the Reason Codes” on page 35 for how to process the reasonCode
  ' from the reply.
  ' Note that GetReplyContent() is included in this document to help you understand
  ' how to process reason codes, but it is not included as part of the sample scripts
  ' or sample ASP pages.

else HandleError nStatus, strErrorInfo, oRequest, varReply
end if
'---------------------
sub HandleError( nStatus, strErrorInfo, oRequest, varReply )
'---------------------
```

```
' HandleError shows how to handle the different errors that can occur.
select case nStatus

  ' An error occurred before the request could be sent.
  case 1
    ' Non-critical error.
    ' Tell customer the order could not be completed and to try again later.
    ' Notify appropriate internal resources of the error.

  ' An error occurred while sending the request.
  case 2
    ' Non-critical error.
    ' Tell customer the order could not be completed and to try again later.

  ' An error occurred while waiting for or retrieving the reply.
  case 3
    ' Critial error.
    ' Tell customer the order could not be completed and to try again later.
    ' Notify appropriate internal resources of the error.

  ' An error occurred after receiving and during processing of the reply.
  case 4
    ' Critical error.
    ' Tell customer the order could not be completed and to try again later.
    ' Look at the BSTR in varReply for the raw reply.
    ' Notify appropriate internal resources of the error.

  ' CriticalServerError fault
  case 5
    ' Critial error.
    ' Tell customer the order could not be completed and to try again later.
    ' Read the contents of the Fault object in varReply.
    ' Notify appropriate internal resources of the fault.

  ' ServerError fault
  case 6
    ' Non-critical error.
    ' Tell customer the order could not be completed and to try again later.
    ' Read the contents of the Fault object in varReply.

  ' Other fault
```

```
    case 7
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Read the contents of the Fault object in varReply.
      ' Notify appropriate internal resources of the fault.

    ' HTTP error
    Case 8
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Look at the BSTR in varReply for the raw reply.

  end select
```

Processing the Reason Codes

After the CyberSource server processes your request, it sends a reply message that contains information about the services you requested. You receive different fields depending on the services you request and the outcome of each service.

To use the reply information, you must integrate it into your system and any other system that uses that data. For example, you can store the reply information in a database and send it to other back office applications.

You must write an error handler to process the reply information that you receive from CyberSource. Do not show the reply information directly to customers. Instead, present an appropriate response that tells customers the result.

Important Because CyberSource may add reply fields and reason codes at any time, you should parse the reply data according to the names of the fields instead of their order in the reply.

The most important reply fields to evaluate are the following:

- **decision**: A one-word description of the results of your request. The decision is one of the following:
  - `ACCEPT` if the request succeeded
  - `REJECT` if one or more of the services in the request was declined
  - `REVIEW` if you are an enterprise merchant using CyberSource Decision Manager and it flags the order for review. See "Handling Reviews" on page 38 for more information.
  - `ERROR` if there was a system error. See "Retrying When System Errors Occur" on page 39 for more information.

- **reasonCode**: A numeric code that provides more specific information about the results of your request.

You also receive a reason code for each service in your request. You can use these reason codes to determine whether a specific service succeeded or failed. If a service fails, other services in your request may not run. For example, if you request a credit card authorization and capture, and the authorization fails, the capture will not run. The reason codes for each service are described in the Business Center Simple Order API User’s Guide (for Business Center users) or in the service’s implementation guide (for Enterprise Business Center users).

Important CyberSource reserves the right to add new reason codes at any time. If your error handler receives a reason code that it does not recognize, it should use the decision to interpret the reply.

```
' Note that GetReplyContent() is included in this document to help you
' understand how to process reason codes, but it is not included as part of
' the sample scripts or sample ASP pages.
'----------------
function GetReplyContent( varReply )
'----------------

  dim strReasonCode
  strReasonCode = varReply.Value( "reasonCode" )

  select case strReasonCode

    ' Success
    case "100"
      GetReplyContent _
      = "Request ID: " & varReply.Value( "requestID" ) & vbCrLf & _
      "Authorized Amount: " & _
      varReply.Value( "ccAuthReply_amount" ) & vbCrLf & _
      "Authorization Code: " & _
      varReply.Value( "ccAuthReply_authorizationCode" )

    ' Insufficient funds
    case "204"
      GetReplyContent = "Insufficient funds in account. Please use a different" & _
      "card or select another form of payment."

    ' add other reason codes here that you need to handle specifically

    ' For all other reason codes, return an empty string, in which case, you should
    ' display a generic message appropriate to the decision value you received.
    case else
      GetReplyContent = ""

  end select

end function
```

Handling Reviews

If you use CyberSource Decision Manager, you may also receive the `REVIEW` value in the **decision** field. `REVIEW` means that Decision Manager has marked the order for review based on how you configured the Decision Manager rules.

If you will be using Decision Manager, you have to determine how to handle the new `REVIEW` value. Ideally, you will update your order management system to recognize the `REVIEW` response and handle it according to your business rules. If you cannot update your system to handle the `REVIEW` response, CyberSource recommends that you choose one of these options:

- If you authorize and capture the credit card payment at the same time, treat the `REVIEW` response like a `REJECT` response. Rejecting any orders that are marked for review may be appropriate if your product is a software download or access to a Web site. If supported by your processor, you may also want to reverse the authorization.
- If you approve the order after reviewing it, convert the order status to `ACCEPT` in your order management system. You can request the credit card capture without requesting a new authorization.
- If you approve the order after reviewing it but cannot convert the order status to `ACCEPT` in your system, request a new authorization for the order. When processing this new authorization, you must disable Decision Manager. Otherwise the order will be marked for review again. For details about the API field that disables Decision Manager, see the Decision Manager Developer's Guide.

  Alternately, you can specify a custom business rule in Decision Manager so that authorizations originating from a particular internal IP address at your company are automatically accepted.

  If supported by your processor, you may want to reverse the original authorization.

Requesting Multiple Services

When you request multiple services in one request, CyberSource processes the services in a specific order. If a service fails, CyberSource does not process the subsequent services in the request.

For example, in the case of a sale (a credit card authorization and a capture requested together), if the authorization service fails, CyberSource will not process the capture service. The reply you receive only includes reply fields for the authorization.

This following additional example applies **to enterprise merchants only.**

Many ICS services include "ignore" fields that tell CyberSource to ignore the result from the first service when deciding whether to run the subsequent services. In the case of the sale, even though the issuing bank gives you an authorization code, CyberSource might decline the authorization based on the AVS or card verification results. Depending on your business needs, you might choose to capture these types of declined authorizations anyway. You can set the **businessRules_ignoreAVSResult** field to “true” in your combined authorization and capture request:

```
oRequest.Value( "businessRules_ignoreAVSResult") = "true"
```

This tells CyberSource to continue processing the capture even if the AVS result causes CyberSource to decline the authorization. In this case you would then get reply fields for both the authorization and the capture in your reply.

Note You are charged only for the services that CyberSource performs.

Retrying When System Errors Occur

You must design your transaction management system to include a way to correctly handle CyberSource system errors (when you successfully receive a reply and the reply’s **decision**=ERROR; see “Processing the Reason Codes” on page 35 for more information about the **decision**). Depending on which payment processor is handling the transaction, the error may indicate a valid CyberSource system error, or it may indicate a processor rejection because of some type of invalid data. In either case, CyberSource recommends that you do not design your system to endlessly retry sending a transaction in the case of a system error.

Instead, CyberSource recommends that you retry sending the request only two or three times, waiting a longer period of time between each successive retry. For example, after the first system error response, wait a short period of time (perhaps 30 seconds) and try sending the request again. If you receive the same error, wait a longer period of time (perhaps 1 minute) and try sending the request again. Depending on the situation, you may decide you can wait and try again after a longer period of time (perhaps 2 minutes). You should determine what is most appropriate for your business situation.

If after several retry attempts you are still receiving a system error, it is possible that the error is actually being caused by a processor rejection and not a CyberSource system error. In that case, CyberSource recommends that you either:

- Search for the transaction in the Enterprise Business Center or the Business Center (depending on which one you normally use), look at the description of the error on the Transaction Detail page, and call your processor to determine if and why they are rejecting the transaction.
- Contact CyberSource Customer Support to confirm whether your error is truly caused by a CyberSource system issue.

If Vital is your processor, you may want to follow the first suggestion as there are several common Vital processor responses that are returned to you as system errors and that only Vital can address.

Chapter 5

Using XML

This chapter describes how to request ICS services using XML and includes these sections:

Other Necessary Documentation

Make sure that you have the guide that explains the API fields (XML elements) you need to use in your requests:

For merchants and resellers using the Business Center:

- Business Center Simple Order API User's Guide: Describes the API for using the CyberSource ICS Credit Card Services and other payment services.

For merchants using the Enterprise Business Center:

- Credit Card Services Implementation Guide: Describes the API for using the CyberSource ICS Credit Card Services. Make sure when you read this guide that you read the Simple Order API chapter and **NOT** the SCMP API chapter.
- Implementation guides for any other ICS services you are planning to use. These are available in the documentation area of the Support Center.

Requesting ICS Services

To request CyberSource ICS services, write code that:

- Collects information for the ICS services that you will use
- Assembles the order information into requests
- Sends the requests to the CyberSource server

Important The CyberSource servers do not support persistent HTTP connections.

- Processes the reply information

The instructions in this chapter explain how to write VBScript that requests ICS services. You will need another guide, as described in "Other Necessary Documentation" on page 41, to get a list of the API fields (XML elements) to use in your requests.

Sample Code

CyberSource recommends that you examine the name-value pair sample code provided in `AuthCaptureSample.wsf` before implementing your code to process XML requests. The sample will give you a basic understanding of how to request CyberSource's ICS services. The sample code file is located in the `<`*`installation directory`*`>\samples\nvp` directory.

After examining that sample code, read this chapter to understand how to create code to process XML requests. Note that the code in this chapter's example is incomplete. For a complete sample program, see the `AuthSample.wsf` file in the `<`*`installation directory`*`>\samples\xml` directory.

Creating a Request Document

With the client, you can create an XML request document by using any application and send a request to CyberSource. For example, if you have a customer relationship management (CRM) system that uses XML to communicate with other systems, you can use the CRM system to generate request documents.

The XML document that you provide must be either a DOMDocument40 object (part of MSXML 4.0) or a string containing XML data.

The request document must be validated against the XML schema for CyberSource transactions. To view the `xsd` file for the version of the Simple Order API that you are using, go to https://ics2ws.ic3.com/commerce/1.x/transactionProcessor.

Important Make sure that the elements in your document appear in the correct order. Otherwise, your document will not be valid, and your request will fail.

The following example shows a basic XML document for requesting ICS services. In this example, Jane Smith is buying an item for $29.95.

The XML document in this example is incomplete. For a complete example, see auth.xml in `<installation directory>\samples\xml`.

Creating an Empty Request

Add the XML declaration and the document's root element:

```
<?xml version="1.0" encoding="utf-8"?>
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-
1.18">
</requestMessage>
```

When you construct a request, you need to indicate the correct namespace for the elements, and the namespace must use the same API version that you specify in the MerchantConfig object. For example, if you set `TargetAPIVersion=1.18`, the namespace must be `urn:schemas-cybersource-com:transaction-data-1.18`.

Note The XML document that you receive in the reply always uses a prefix of `c:` (for example, `xmlns:c="urn:schemas-cybersource-com:transaction-data-1.18"`). Make sure you use an XML parser that supports namespaces.

Adding the Merchant ID

You next add the CyberSource merchant ID to the request.

Note If you specify a merchant ID in the XML document, it overrides the merchant ID you specify in the MerchantConfig object.

```
<?xml version="1.0" encoding="utf-8"?>
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-
1.18">
  <merchantID>infodev</merchantID>
</requestMessage>
```

Adding Services to the Request

You next indicate the service that you want to use by creating an element for that service in the request, then setting the element's `run` attribute to `true`. For example, to request a credit card authorization:

```
<?xml version="1.0" encoding="utf-8"?>
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-
1.18">
  <merchantID>infodev</merchantID>
  <ccAuthService run="true"/>
</requestMessage>
```

Requesting a Sale

You can request multiple services by adding additional elements. For example, if you fulfill the order immediately, you can request a credit card authorization and capture together (referred to as a "sale"):

```
<?xml version="1.0" encoding="utf-8"?>
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-
1.18">
  <merchantID>infodev</merchantID>
  <ccAuthService run="true"/>
  <ccCaptureService run="true"/>
</requestMessage>
```

Adding Service-Specific Fields to the Request

You next add the fields that are used by the services you are requesting. Most fields are child elements of container elements; for example, a `<card>` element contains the customer's credit card information.

```
<?xml version="1.0" encoding="utf-8"?>
<requestMessage xmlns="urn:schemas-cybersource-com:transaction-data-
1.18">
  <merchantID>infodev</merchantID>
  <billTo>
    <firstName>Jane</firstName>
    <lastName>Smith</lastName>
  </billTo>
  <item id="0">
    <unitPrice>29.95</unitPrice>
  </item>
  <card>
    <accountNumber>4111111111111111</accountNumber>
  </card>
  <ccAuthService run="true"/>
</requestMessage>
```

The example above shows only a partial list of the fields you need to send. Refer to "Other Necessary Documentation" on page 41 for information about the guide or guides that list all of the fields for the service(s) you are requesting.

Sending Requests

Once you have created an XML document, you use VBscript in your ASP pages to send the request to CyberSource.

Creating the MerchantConfig Object

First create a MerchantConfig object and set your merchant ID and other basic transaction settings:

```
Dim oMerchantConfig
set oMerchantConfig =
    Server.CreateObject( "CyberSourceWS.MerchantConfig" )
oMerchantConfig.MerchantID = "infodev"
oMerchantConfig.KeysDirectory = "\keys"
oMerchantConfig.SendToProduction = "0"
oMerchantConfig.TargetAPIVersion = "1.18"
```

Note The namespace that you specify in the XML document must use the same API version that you specify in the MerchantConfig object. For example, if you set `TargetAPIVersion=1.18`, the namespace must be `urn:schemas-cybersource-com:transaction-data-1.18`.

Reading the XML Document

```
' read XML document
dim strXMLRequest
strXMLRequest = ReadTextFile( "MyXMLDocument.xml" )

' If you did not hardcode the namespace in the XML document, make sure
' to insert the effective namespace URI from the MerchantConfig object
' into the XML document.
```

Sending the Request

You next create a Client object and send the request:

```
dim oClient
set oClient = WScript.CreateObject( "CyberSourceWS.Client" )

' This example uses a DOMDocument40 object and not a string.
' See the AuthSample.wsf example in the samples\xml directory for an
example
```

```
' of how to handle a string.
' Create the DOMDocument40 object
dim varRequest
set varRequest = WScript.CreateObject( "Msxml2.DOMDocument.4.0" )

' Load the XML string into the DOMDocument40 object
varRequest.loadXML( strXMLRequest )

' send request
dim varReply, nStatus, strErrorInfo
nStatus = oClient.RunTransaction( _
          oMerchantConfig, Nothing, Nothing, varRequest, _
          varReply, strErrorInfo )
```

Interpreting Replies

Handling the Return Status

You next handle the `nStatus` returned by the RunTransaction method. The `nStatus` indicates whether the ICS server received the request, if the client received the reply, and if there were any errors or faults during transmission. See "Possible Return Status Values" on page 26 for descriptions of each status value. For a different example, see the `AuthSample.wsf` file in the client's `<`*`installation directory`*`>\samples\xml` directory.

```
' set the SelectionNamespace for later XPath queries
strNamespaceURI = oMerchantConfig.EffectiveNamespaceURI
varReply.setProperty "SelectionNamespaces", _
                     "xmlns:c='" & strNamespaceURI & "'"
if nStatus = 0 then
  dim strContent
  ' Get the contents of the <replyMessage> element
  dim oReplyMessage
  set oReplyMessage =
    varReply.SelectSingleNode( "c:replyMessage")
  ' Read the value of the "decision" in the oReplyMessage.
  ' If decision=ACCEPT, indicate to the customer that the request was successful.
  ' If decision=REJECT, indicate to the customer that the order was not approved.
  ' If decision=ERROR, indicate to the customer an error occurred and to try again
  ' later.
  ' Now get reason code results:
```

```
  strContent = GetReplyContent( oReplyMessage )
  ' See “Processing the Reason Codes” on page 49 for how to process the reasonCode
  ' from the reply.
  ' Note that GetReplyContent() is included in this document to help you understand
  ' how to process reason codes, but it is not included as part of the sample scripts
  ' or sample ASP pages.
else HandleError nStatus, strErrorInfo, oRequest, varReply
end if
'---------------------
sub HandleError( nStatus, strErrorInfo, oRequest, varReply )
'---------------------
  ' HandleError shows how to handle the different errors that can occur.
  select case nStatus

    ' An error occurred before the request could be sent.
    case 1
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Notify appropriate internal resources of the error.

    ' An error occurred while sending the request.
    case 2
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.

    ' An error occurred while waiting for or retrieving the reply.
    case 3
      ' Critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Notify appropriate internal resources of the error.

    ' An error occurred after receiving and during processing
    ' of the reply.
    case 4
      ' Critial error.
      ' Tell customer the order could not be completed and to try again later.
      ' Look at the BSTR in varReply for the raw reply.
      ' Notify appropriate internal resources of the error.
    ' CriticalServerError fault
    case 5
```

```
      ' Critial error.
      ' Tell customer the order could not be completed and to try again later.
      ' Read the contents of the Fault object in varReply.
      ' Notify appropriate internal resources of the fault.

    ' ServerError fault
    case 6
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Read the contents of the Fault object in varReply.

    ' Other fault
    case 7
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Read the contents of the Fault object in varReply.
      ' Notify appropriate internal resources of the fault.

    ' HTTP error
    Case 8
      ' Non-critical error.
      ' Tell customer the order could not be completed and to try again later.
      ' Look at the BSTR in varReply for the raw reply.

end select
```

Processing the Reason Codes

After the CyberSource server processes your request, it sends a reply message that contains information about the services you requested. You receive different fields depending on the services you request and the outcome of each service.

To use the reply information, you must integrate it into your system and any other system that uses that data. For example, you can store the reply information in a database and send it to other back office applications.

You must write an error handler to process the reply information that you receive from CyberSource. Do not show the reply information directly to customers. Instead, present an appropriate response that tells customers the result.

Important Because CyberSource may add reply fields and reason codes at any time, you should parse the reply data according to the names of the fields instead of their order in the reply.

The most important reply fields to evaluate are the following:

- **decision**: A one-word description of the results of your request. The decision is one of the following:
    - `ACCEPT` if the request succeeded
    - `REJECT` if one or more of the services in the request was declined
    - `REVIEW` if you use CyberSource Decision Manager and it flags the order for review. See “Handling Reviews” on page 52 for more information.
    - `ERROR` if there was a system error. See “Retrying When System Errors Occur” on page 53 for more information.
- **reasonCode**: A numeric code that provides more specific information about the results of your request.

You also receive a reason code for each service in your request. You can use these reason codes to determine whether a specific service succeeded or failed. If a service fails, other services in your request may not run. For example, if you request a credit card authorization and capture, and the authorization fails, the capture will not run. The reason codes for each service are described in the Business Center Simple Order API User’s Guide (for Business Center users) or in the service’s implementation guide (for Enterprise Business Center users).

Important CyberSource reserves the right to add new reason codes at any time. If your error handler receives a reason code that it does not recognize, it should use the decision to interpret the reply.

```
' Note that GetReplyContent() is included in this document to help you
' understand how to process reason codes, but it is not included as
' part of the sample scripts or sample ASP pages.
'----------------
function GetReplyContent( oReplyMessage )
'----------------
  dim strReasonCode
  strReasonCode = GetValue( oReplyMessage, "c:reasonCode" )

  select case strReasonCode
    ' Success
    case "100"
      GetReplyContent = _
        "Request ID: " & GetValue( oReplyMessage, "c:requestID" ) & _
            vbCrLf & _
        "Authorized Amount: " & _
        GetValue( oReplyMessage, "c:ccAuthReply/c:amount" ) & _
            vbCrLf & _
        "Authorization Code: " & _
        GetValue( oReplyMessage, _
            "c:ccAuthReply/c:authorizationCode" )

    ' Insufficient funds
    case "204"
      GetReplyContent = "Insufficient funds in account. Please " & _
      "use a different card or select another form of payment." & _

    ' add other reason codes here that you need to handle specifically

    ' For all other reason codes, return an empty string, in which case, you should
    ' display a generic message appropriate to the decision value you received.
    case else
      GetReplyContent = ""
  end select
end function
```

Handling Reviews

If you use CyberSource Decision Manager, you may also receive the `REVIEW` value in the **decision** field. `REVIEW` means that Decision Manager has marked the order for review based on how you configured the Decision Manager rules.

If you will be using Decision Manager, you have to determine how to handle the new `REVIEW` value. Ideally, you will update your order management system to recognize the `REVIEW` response and handle it according to your business rules. If you cannot update your system to handle the `REVIEW` response, CyberSource recommends that you choose one of these options:

- If you authorize and capture the credit card payment at the same time, treat the `REVIEW` response like a `REJECT` response. Rejecting any orders that are marked for review may be appropriate if your product is a software download or access to a Web site. If supported by your processor, you may also want to reverse the authorization.
- If you approve the order after reviewing it, convert the order status to `ACCEPT` in your order management system. You can request the credit card capture without requesting a new authorization.
- If you approve the order after reviewing it but cannot convert the order status to `ACCEPT` in your system, request a new authorization for the order. When processing this new authorization, you must disable Decision Manager. Otherwise the order will be marked for review again. For details about the API field that disables Decision Manager, see the Decision Manager Developer's Guide.

  Alternately, you can specify a custom business rule in Decision Manager so that authorizations originating from a particular internal IP address at your company are automatically accepted.

  If supported by your processor, you may want to reverse the original authorization.

Requesting Multiple Services

When you request multiple services in one request, CyberSource processes the services in a specific order. If a service fails, CyberSource does not process the subsequent services in the request.

For example, in the case of a sale (a credit card authorization and a capture requested together), if the authorization service fails, CyberSource will not process the capture service. The reply you receive only includes reply fields for the authorization.

This following additional example applies **to enterprise merchants only.**

Many ICS services include "ignore" fields that tell CyberSource to ignore the result from the first service when deciding whether to run the subsequent services. In the case of the sale, even though the issuing bank gives you an authorization code, CyberSource might decline the authorization based on the AVS or card verification results. Depending on your business needs, you might choose to capture these types of declined authorizations anyway. You can set the **businessRules_ignoreAVSResult** field to “true” in your combined authorization and capture request:

```
<businessRules>
    <ignoreAVSResult>true</ignoreAVSResult>
</businessRules>
```

This tells CyberSource to continue processing the capture even if the AVS result causes CyberSource to decline the authorization. In this case you would then get reply fields for both the authorization and the capture in your reply.

Note You are charged only for the services that CyberSource performs.

Retrying When System Errors Occur

You must design your transaction management system to include a way to correctly handle CyberSource system errors (when you successfully receive a reply and the reply’s **decision**=ERROR; see “Processing the Reason Codes” on page 49 for more information about the **decision**). Depending on which payment processor is handling the transaction, the error may indicate a valid CyberSource system error, or it may indicate a processor rejection because of some type of invalid data. In either case, CyberSource recommends that you do not design your system to endlessly retry sending a transaction in the case of a system error.

Instead, CyberSource recommends that you retry sending the request only two or three times, waiting a longer period of time between each successive retry. For example, after the first system error response, wait a short period of time (perhaps 30 seconds) and try sending the request again. If you receive the same error, wait a longer period of time (perhaps 1 minute) and try sending the request again. Depending on the situation, you may decide you can wait and try again after a longer period of time (perhaps 2 minutes). You should determine what is most appropriate for your business situation.

If after several retry attempts you are still receiving a system error, it is possible that the error is actually being caused by a processor rejection and not a CyberSource system error. In that case, CyberSource recommends that you either:

- Search for the transaction in the Enterprise Business Center or the Business Center (depending on which one you normally use), look at the description of the error on the Transaction Detail page, and call your processor to determine if and why they are rejecting the transaction.
- Contact CyberSource Customer Support to confirm whether your error is truly caused by a CyberSource system issue.

If Vital is your processor, you may want to follow the first suggestion as there are several common Vital processor responses that are returned to you as system errors and that only Vital can address.

Appendix A

Generating Security Keys

This appendix describes how to generate security keys for merchants using the Business Center and those using the Enterprise Business Center. If you are not sure which type of merchant you are, see "Who Should Read This Guide" on page 1.

Preparing to Generate a Key

Before generating one or more keys, you need to follow the steps required according to the URL that you use to log in to the Business Center.

Merchants Using the Business Center

The Simple Order API uses a public and private key pair key to prevent others from using your account. This security measure ensures that transactions originate from your web site and that no one, not even CyberSource, can run transactions on your behalf by using your private key.

The Security Keys page in the Business Center allows you to generate a certificate request, which is stored on your computer. The certificate request creates a public and private key pair that is used to encrypt transaction data sent to CyberSource. Your private key is required to generate encrypted data but is never transmitted over the Internet.

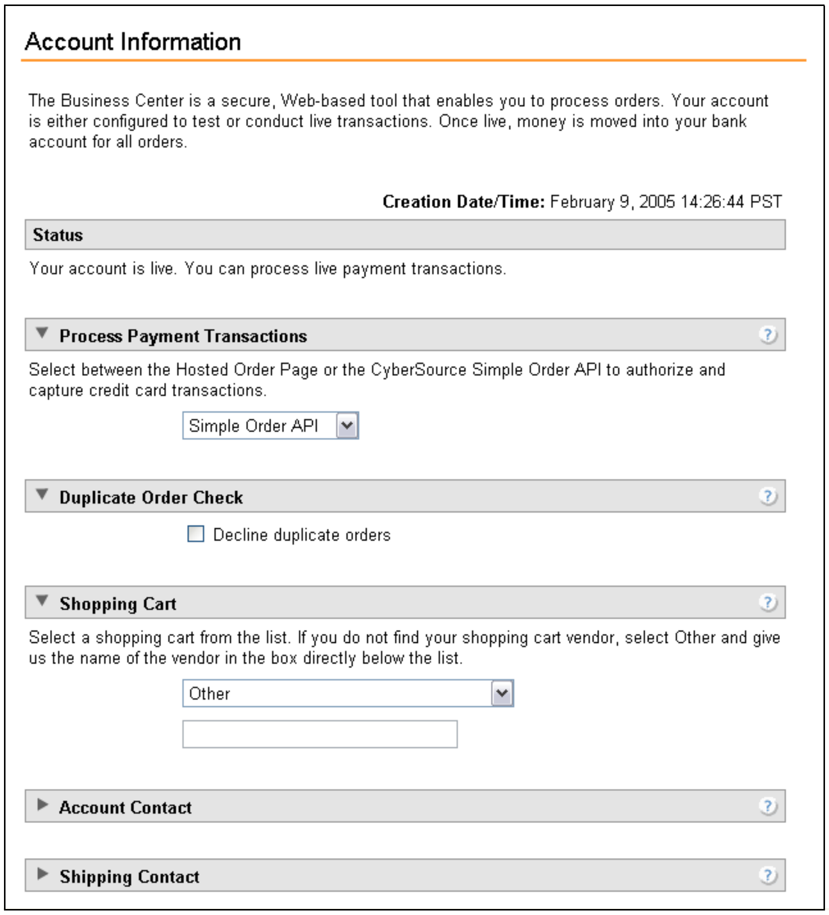
A Java applet is used to create your new key (***`<username>.p12)`***. The applet downloads 1.5 MB of executable code if your browser supports the plug-in.

Note If the applet fails to load properly, CyberSource recommends that you download the latest revision of Internet Explorer or Netscape Navigator and try again.

Before communicating with the CyberSource server, the applet encrypts your password by using an SSL certificate. Your password is sent with the key generation request to validate your identity.

To create a security key pair:

1. Log in to the Business Center at https://businesscenter.cybersource.com.
2. Click **Settings** > **Account Info** in the navigation pane.
   The Account Information page appears.
3. In the Process Payment Transactions section, select Simple Order API.



Account Information

The Business Center is a secure, Web-based tool that enables you to process orders. Your account is either configured to test or conduct live transactions. Once live, money is moved into your bank account for all orders.

Creation Date/Time: February 9, 2005 14:26:44 PST

Status

Your account is live. You can process live payment transactions.

Process Payment Transactions

Select between the Hosted Order Page or the CyberSource Simple Order API to authorize and capture credit card transactions.

Simple Order API

Duplicate Order Check

☐ Decline duplicate orders

Shopping Cart

Select a shopping cart from the list. If you do not find your shopping cart vendor, select Other and give us the name of the vendor in the box directly below the list.

Other

Account Contact

Shipping Contact

4. In the Shopping Cart section, select Other.
5. Scroll to the bottom of the page and click **Update**.
6. In the navigation pane, click **Transaction Security Keys**.
   The Transaction Security Keys page appears.
7. Proceed with "Generating a Key" on page 57.

Merchants Using the Enterprise Business Center

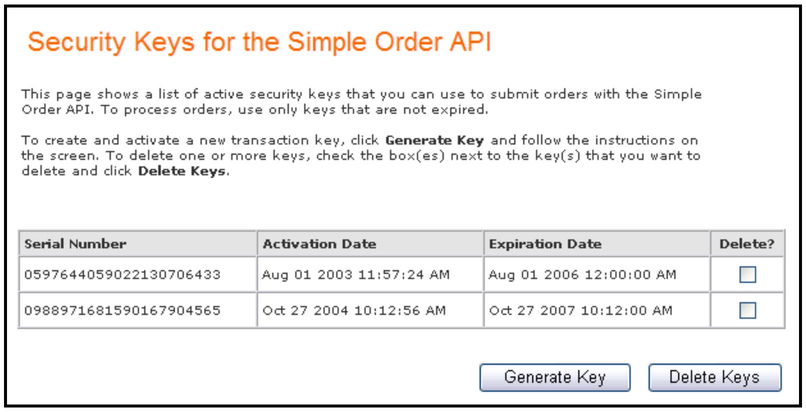
Before you can send requests for ICS services, you must use a Java applet to create a security key for your CyberSource merchant ID. The Java applet is on the Business Center's Web site. If you have problems creating a key, make sure your Web browser allows you to run Java applets.

Note The Java applet works only if you use version 1.4.1 of Sun's Java plug-in. If the applet fails to load properly, CyberSource recommends that you download the latest revision of your browser and try again.

1 To create a security key, log into the Business Center.
2 In the navigation pane. click the **Account Management** > **Transaction Security Keys**.
  The Transaction Security Keys page appears.
3 Proceed with the next section.

Generating a Key

This figure shows the Transaction Security Keys page. Follow the instructions below to create a key.

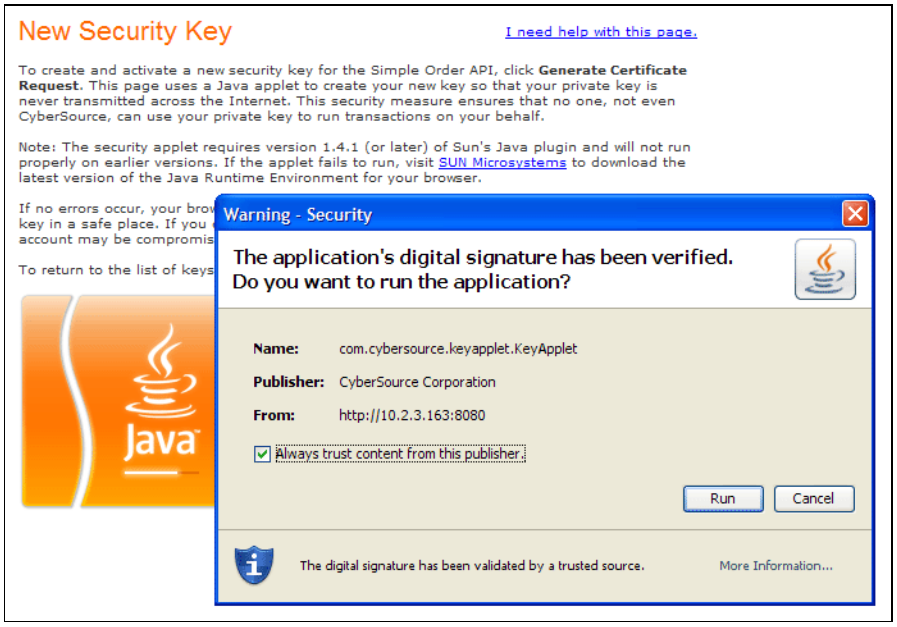

Security Keys for the Simple Order API

This page shows a list of active security keys that you can use to submit orders with the Simple Order API. 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**.

Serial Number	Activation Date	Expiration Date	Delete?
0597644059022130706433	Aug 01 2003 11:57:24 AM	Aug 01 2006 12:00:00 AM	☐
0988971681590167904565	Oct 27 2004 10:12:56 AM	Oct 27 2007 10:12:00 AM	☐

4 Click **Generate Key**.
  The download may take several minutes during which the applet may appear as a large gray box. A warning message appears.


New Security Key

I need help with this page.

To create and activate a new security key for the Simple Order API, click **Generate Certificate Request**. This page uses a Java applet to create your new key so that your private key is never transmitted across the Internet. This security measure ensures that no one, not even CyberSource, can use your private key to run transactions on your behalf.

Note: The security applet requires version 1.4.1 (or later) of Sun's Java plugin and will not run properly on earlier versions. If the applet fails to run, visit SUN Microsystems to download the latest version of the Java Runtime Environment for your browser.

If no errors occur, your brow key in a safe place. If you account may be compromis

To return to the list of keys

Warning - Security

The application's digital signature has been verified.
Do you want to run the application?

Name: com.cybersource.keyapplet.KeyApplet

Publisher: CyberSource Corporation

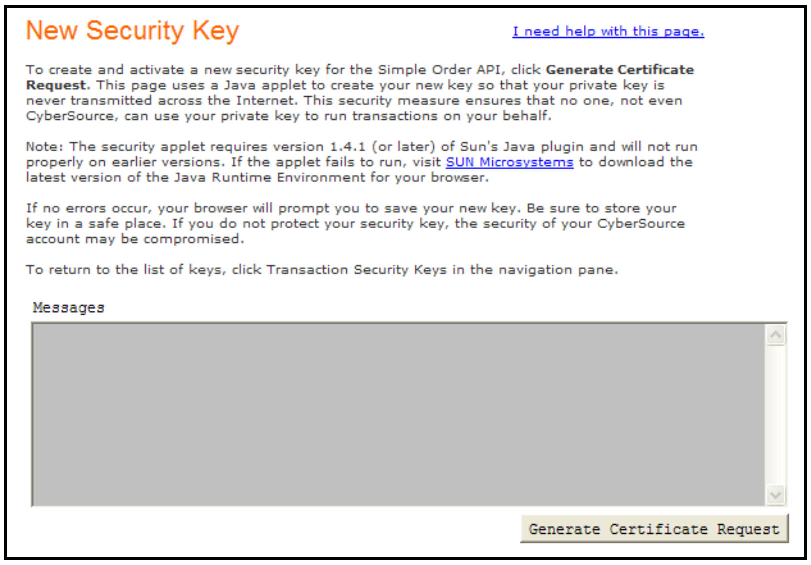
From: http://10.2.3.163:8080

[✓] Always trust content from this publisher.

The digital signature has been validated by a trusted source.

5 Verify that the certificate is signed by CyberSource Corporation, and click **Run**.

When the **Generate Certificate Request** button appears, the download is complete.


New Security Key

I need help with this page.

To create and activate a new security key for the Simple Order API, click **Generate Certificate Request**. This page uses a Java applet to create your new key so that your private key is never transmitted across the Internet. This security measure ensures that no one, not even CyberSource, can use your private key to run transactions on your behalf.

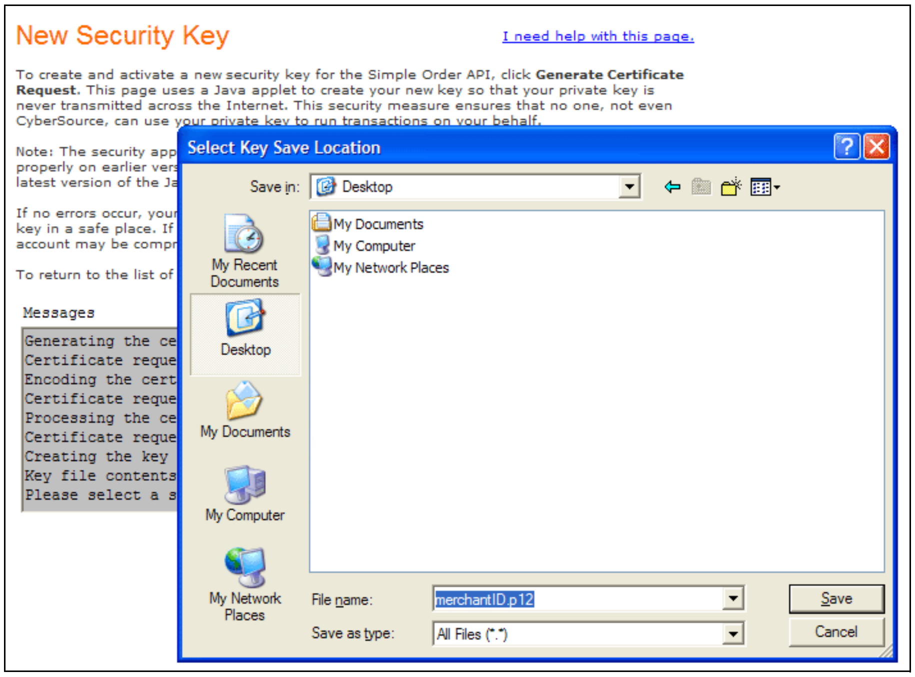
Note: The security applet requires version 1.4.1 (or later) of Sun's Java plugin and will not run properly on earlier versions. If the applet fails to run, visit SUN Microsystems to download the latest version of the Java Runtime Environment for your browser.

If no errors occur, your browser will prompt you to save your new key. Be sure to store your key in a safe place. If you do not protect your security key, the security of your CyberSource account may be compromised.

To return to the list of keys, click Transaction Security Keys in the navigation pane.

6 Click **Generate Certificate Request**.

While a new key is generated, messages appear in the box. Your browser opens the Save As dialog box.



New Security Key

I need help with this page.

To create and activate a new security key for the Simple Order API, click **Generate Certificate Request**. This page uses a Java applet to create your new key so that your private key is never transmitted across the Internet. This security measure ensures that no one, not even CyberSource, can use your private key to run transactions on your behalf.

Note: The security app
properly on earlier vers
latest version of the Ja

If no errors occur, your
key in a safe place. If
account may be compr

To return to the list of

Messages

```
Generating the ce
Certificate reque
Encoding the cert
Certificate reque
Processing the ce
Certificate reque
Creating the key
Key file contents
Please select a s
```

7 Choose a safe location for your key (`<username>.p12`).

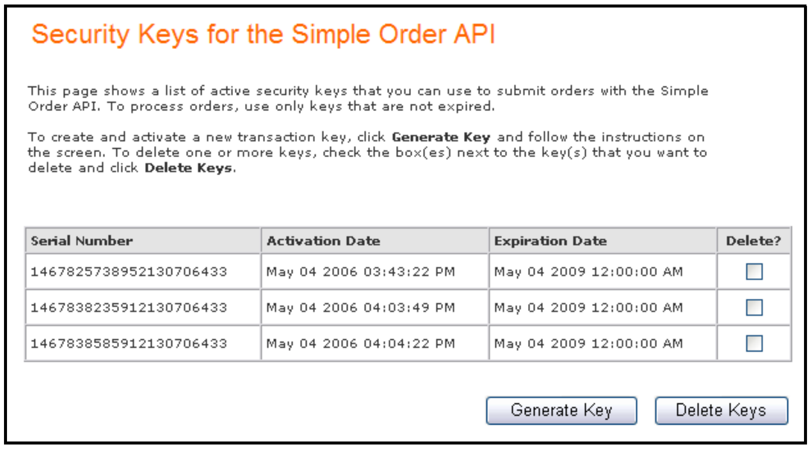
If you do not protect your security keys, the security of your CyberSource account may be compromised. The last line in the example below informs you that the process is finished and that a new key is stored in the location that you indicated:

```
Generating the certificate request. This may take several seconds.
Certificate request generated successfully.
Encoding the certificate request.
Certificate request encoded successfully.
Processing the certificate request. This may take several seconds.
Certificate request processed successfully.
Creating the key file contents.
Key file contents created successfully.
Please select a save location for your key file using the popup dialog.
Writing the key file to the file system.
Writing the key file to C:\Full_Path_that_you_chose.
Key file written to the file system successfully.
The password for the key file is your merchant id: <Merchant_ID>.

The Certificate Manager has successfully completed all operations.
```

8 To verify that your key is active, click **Transaction Security Keys** in the navigation pane.

Your new key is listed at the bottom of the table.



Security Keys for the Simple Order API

This page shows a list of active security keys that you can use to submit orders with the Simple Order API. 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**.

Serial Number	Activation Date	Expiration Date	Delete?
1467825738952130706433	May 04 2006 03:43:22 PM	May 04 2009 12:00:00 AM	☐
1467838235912130706433	May 04 2006 04:03:49 PM	May 04 2009 12:00:00 AM	☐
1467838585912130706433	May 04 2006 04:04:22 PM	May 04 2009 12:00:00 AM	☐

Appendix B

Viewing a Security Key's Serial Number

The client uses a p12 security key to add a digital signature to every request that you send. During installation, you used a Java applet to generate that p12 key file (see "Creating a Security Key" on page 10).

In the Business Center (for Business Center merchants) or the Enterprise Business Center (for enterprise merchants), you can view a list of the keys that you have generated. The keys, however, are listed there by their serial numbers, and not by their file names. If you are not sure which of your keys is the active one recognized by CyberSource, you may need to look at the serial numbers for the key files that you have stored locally to match them with the information shown in the Business Center or Enterprise Business Center.

This section explains how to import a key file and view its serial number in a Web browser.

Importing the Key File

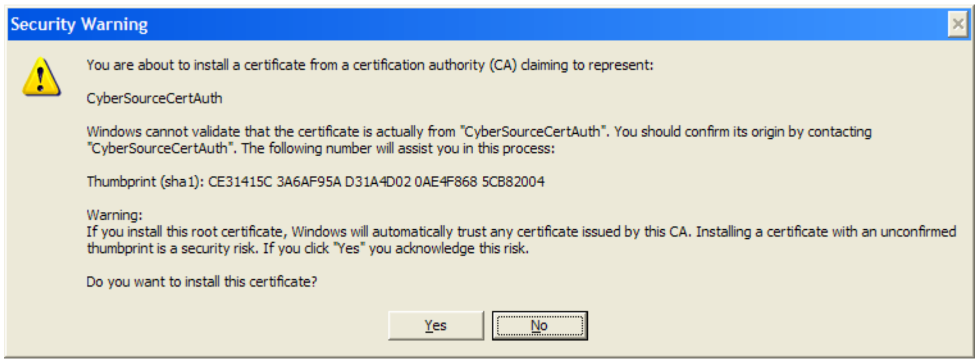
1. Locate and double-click the key file.

   The Certificate Import Wizard opens.
2. Click **Next**.

   The Wizard shows the path to the key file.
3. Click **Next**.
4. Type the password for the key file.

   The password is the merchant ID that you used to log into the Business Center to generate the key.
5. Do not select either check box.
6. Click **Next**.
7. Ensure that the option *Automatically select the certificate store based on the type of certificate* is selected.
8. Click **Next**.
9. Click **Finish**.

   A warning appears



Security Warning

You are about to install a certificate from a certification authority (CA) claiming to represent:

CyberSourceCertAuth

Windows cannot validate that the certificate is actually from "CyberSourceCertAuth". You should confirm its origin by contacting "CyberSourceCertAuth". The following number will assist you in this process:

Thumbprint (sha1): CE31415C 3A6AF95A D31A4D02 0AE4F868 5CB82004

Warning:
If you install this root certificate, Windows will automatically trust any certificate issued by this CA. Installing a certificate with an unconfirmed thumbprint is a security risk. If you click "Yes" you acknowledge this risk.

Do you want to install this certificate?

10 Click **Yes**.

A success message appears.

Viewing the Serial Number

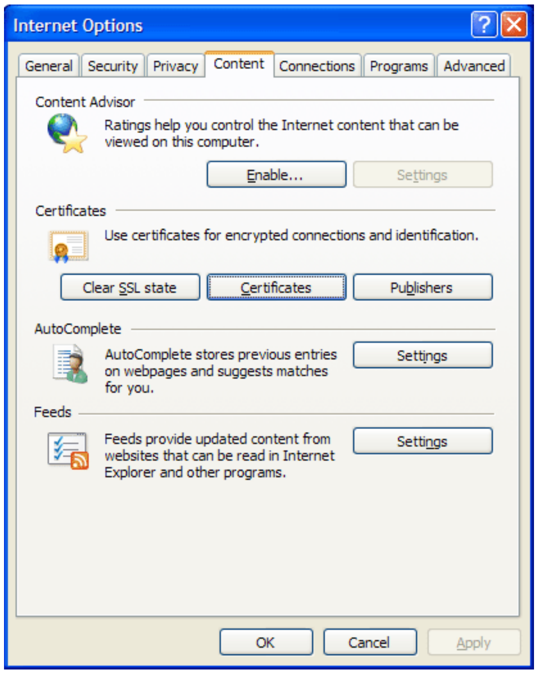
These instructions are for Internet Explorer 7. Modify them as needed for your browser.

1 Open Internet Explorer.

2 Click **Tools** > **Internet Options**….

The Internet Options window opens.

3 Click the Content tab.



Internet Options

Content Advisor

Ratings help you control the Internet content that can be viewed on this computer.

Certificates

Use certificates for encrypted connections and identification.

AutoComplete

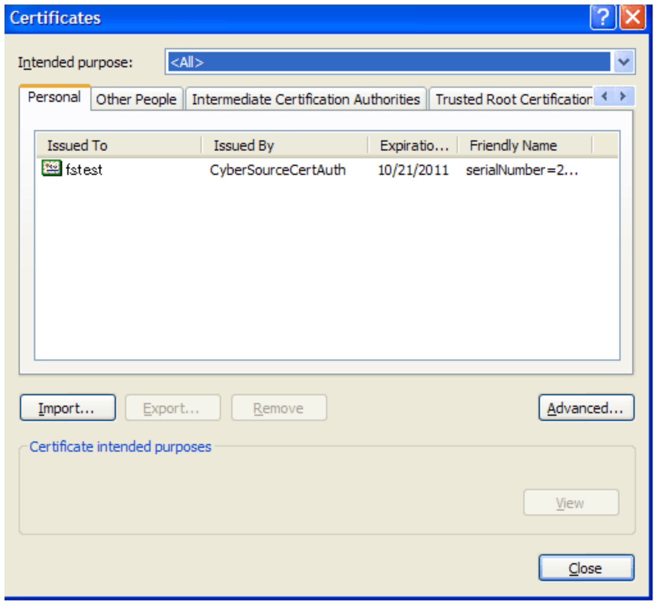
AutoComplete stores previous entries on webpages and suggests matches for you.

Feeds

Feeds provide updated content from websites that can be read in Internet Explorer and other programs.

4 Click **Certificates**.

The Certificates window shows a list of the certificates that have been imported. In this example, your key file is `fstest`.



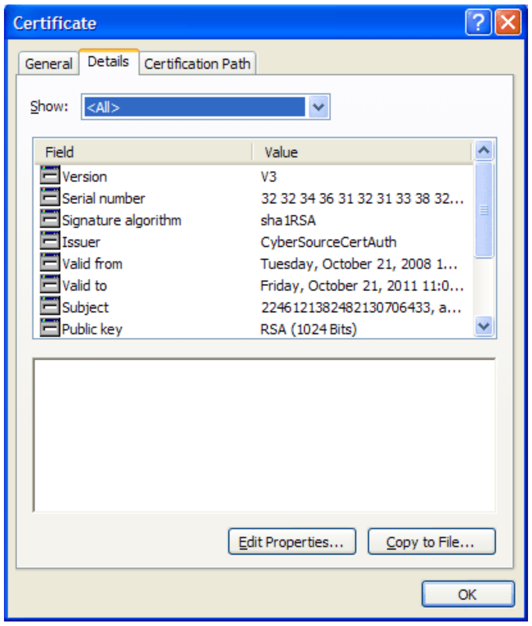
Issued To	Issued By	Expiratio...	Friendly Name
fstest	CyberSourceCertAuth	10/21/2011	serialNumber=2...

5 Double-click the key file that you imported in the previous section.

The Certificate window for that file opens.

6 Click the Details tab.

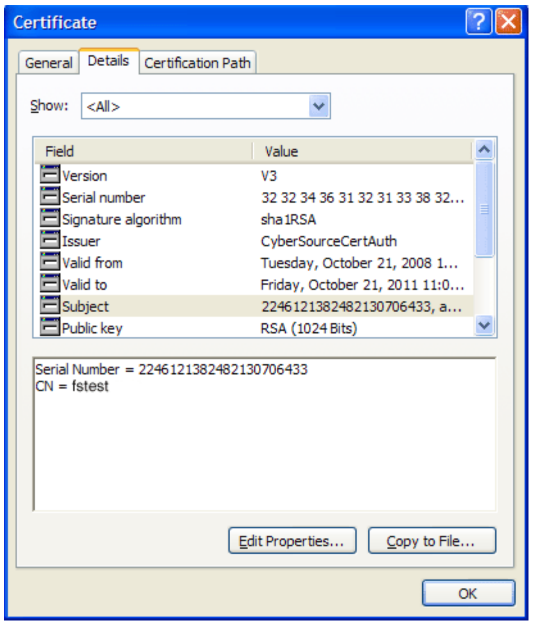
The window shows a list of fields and values, but the Serial Number field does not contain the serial number information that you want to see. Instead, the Subject field contains the correct information.



Field	Value
Version	V3
Serial number	32 32 34 36 31 32 31 33 38 32...
Signature algorithm	sha1RSA
Issuer	CyberSourceCertAuth
Valid from	Tuesday, October 21, 2008 1...
Valid to	Friday, October 21, 2011 11:0...
Subject	2246121382482130706433, a...
Public key	RSA (1024 Bits)

7 Click the Subject field.

The lower window displays the serial number for the key file.



Field	Value
Version	V3
Serial number	32 32 34 36 31 32 31 33 38 32...
Signature algorithm	sha1RSA
Issuer	CyberSourceCertAuth
Valid from	Tuesday, October 21, 2008 1...
Valid to	Friday, October 21, 2011 11:0...
Subject	2246121382482130706433, a...
Public key	RSA (1024 Bits)

```
Serial Number = 2246121382482130706433
CN = fstest
```

Appendix C

Using the Client Application Fields

This appendix explains the fields that you can use to describe your client application. Use these fields if you are building an application to sell to others, such as a shopping cart, that incorporates the client.

Important Do not use the fields in this appendix if you are only integrating the client with your own store.

Table 15 lists the client application fields that you can include in your request. You are not required to use any of these fields.

Table 15 Client Application Fields

Field Name	Description	Data Type and Length
clientApplication	Application or integration that uses the client (for example, `ShoppingCart Pro` or `Web Commerce Server`). Do not include a version number.	String (50)
clientApplicationVersion	The version of the application or integration (for example, `5.0` or `1.7.3`).	String (50)
clientApplicationUser	User of the application or integration (for example, `jdoe`).	String (30)

Index