



Direct Payment Component Windows NT SDK

Version 1.1
April 2004

This manual and accompanying electronic media are proprietary products of Optimal Payments Inc. They are to be used only by licensed users of the product.

© 1999–2004 Optimal Payments Inc. All rights reserved.

The information within this document is subject to change without notice. The software described in this document is provided under a license agreement, and may be used or copied only in accordance with this agreement. No part of this manual may be reproduced or transferred in any form or by any means without the express written consent of Optimal Payments Inc.

FirePay and FireCash are registered trademarks of Optimal Payments Inc. All other names, trademarks and registered trademarks are the property of their respective owners.

Optimal Payments Inc. makes no warranty, either express or implied, with respect to this product, its merchantability or fitness for a particular purpose, other than as expressly provided in the license agreement of this product. For further information, please contact Optimal Payments Inc.

International Head Office

Optimal Payments Inc.
2 Place Alexis Nihon, Suite 700
Westmount, Quebec H3Z 3C1
Canada

Tel.: (514) 380-2700

Fax: (514) 380-2760

Email: info@optimalpayments.com

Technical support: support@optimalpayments.com

Web: www.optimalpayments.com

U.K. Office

Optimal Payments Ltd.
Compass House
Vision Park
Histon, Cambridge CB4 9AD
England

Email: info@optimalpayments.co.uk

Web: www.optimalpayments.co.uk

Hull Office

Optimal Payments Inc.
75 Promenade du Portage
Gatineau, Quebec J8X 2J9
Canada

U.S. Offices

Optimal Payments Corp.
1800 West Loop South, #770
Houston, TX 77027

Optimal Payments Corp.
225 Franklin, 26th Floor
Boston, MA 02110

1 Installing Direct Payment

Software requirements	1-1
Merchant registration	1-1
Installing Direct Payment	1-2
For IIS	1-2
For VB and VC++	1-2
Testing Direct Payment.	1-3
IIS	1-3
VB/VC++	1-4

2 Using Direct Payment

What is Direct Payment?	2-1
Merchant transactions.	2-2
Submitting transaction requests	2-2
Failed transactions	2-4
Communication failure	2-5
Direct Payment component level	2-5
Payment service level	2-7
No response	2-7
Additional features	2-8
Secure communications	2-8
Proxy server support	2-8
Configuring for Direct Payment.	2-9
Direct Payment configuration file	2-9

A Error Return Codes

Error messages	A-1
----------------------	-----

- Direct Payment component error messages A-1
- Payment service error messages A-5
 - Action codes A-5
 - Error codes and strings A-6
 - Suberror codes and strings A-14
 - Unmapped suberror codes and strings A-18

B Sample ASP Script

- Sample script B-1
 - Dictionary Initialization B-2
 - Configuration file path setup B-2
 - Sample ASP script B-2

Index

Installing Direct Payment

Software requirements

The merchant application uses an ASP (Active Server Pages) script or a VB/VC++ application to implement the Direct Payment component. The Direct Payment component communicates with the Optimal Payments payment service over the Internet using SSL (Secure Sockets Layer). The following software is required:

- Microsoft Windows NT Server 4 or later, with Service Pack 6a, or Windows 2000.



*Your Windows NT Server must have **atl.dll** version 3.x before you can install the Direct Payment component.*

- Microsoft Internet Explorer 4.01 or later; **or**
- Netscape 4.x or later
- Microsoft Internet Information Server (IIS) 4.0 (if using ASP)

Merchant registration

Before using the Direct Payment component, the prospective merchant must register on the Optimal Payments Web site, by filling out an application form. Our sales department will then contact the merchant to finalize all necessary details. Once registration is complete, the merchant will receive an account number, an account transaction processing ID, and an account transaction processing password. These values are important, as they are used for merchant transaction requests with the Optimal Payments transaction processing service via the Direct Payment component. For more information, see the *Direct Payment Protocol Specification* manual.

Installing Direct Payment



If for any reason you have to reinstall the Direct Payment component, you must first remove the original installation.

For IIS

The Direct Payment component software is delivered in a compressed ZIP file for IIS.

To install Direct Payment component:

1. Unzip the contents of the ZIP file in its own folder.
2. Open a DOS window and change to the directory containing the files you just unzipped.
3. Run the *install.bat* file.
4. Follow the onscreen instructions to install the Direct Payment component.

For VB and VC++

The Direct Payment component software is packaged in an installation wizard for VB/C++. You need only to execute this wizard to install the product on your machine.

To install Direct Payment component:

1. Double-click *Setup.exe*. The Payment Component Client Setup window appears.
2. Follow the on-screen instructions.
3. Click OK. The installation is complete.

Testing Direct Payment



*In order for merchant transaction requests to be processed quickly and efficiently, ensure that all mandatory fields are validated **before** the request is sent to the transaction processor. That is, the merchant's request should not be sent if, for example, a credit card number is missing or has the wrong number of digits, or if the expiry date has already passed.*

IIS

When you install the Direct Payment component, a sample form is also installed in the following location:

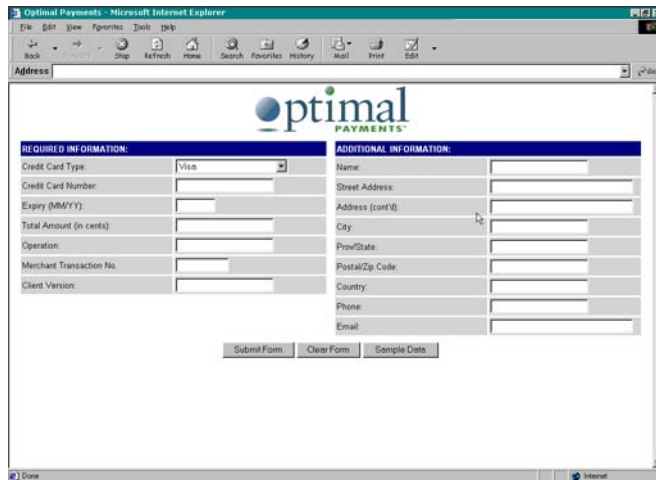
```
c:\InetPub\wwwroot\PaymentSample\sample_form.asp
```

The sample form is included to provide a means to test whether the Direct Payment component has been installed successfully.

To test your Direct Payment component installation:

1. Open the *sample.cfg* file and modify the following parameters:
 - merchantId (“Merchant ID” in your activation email)
 - merchantPwd (“Merchant Password” in your activation email)
 - account (“Account Number” in your activation email)
2. Modify the value of the *ConfigFile* variable found in the *sample_process.asp* file to the full path name and file name of your sample configuration file (e.g., c:\temp\sample.cfg).

- Open your Web browser and enter the following URL in the address field: `http://localhost/PaymentSample/sample_form.asp`. (Make sure IIS is running.) The following window opens.



- Click Sample Data.
- Click Submit Form.

If the Direct Payment component has been successfully installed, you will see the transaction status in your browser, and receive the following notification – “Credit card was processed successfully”.



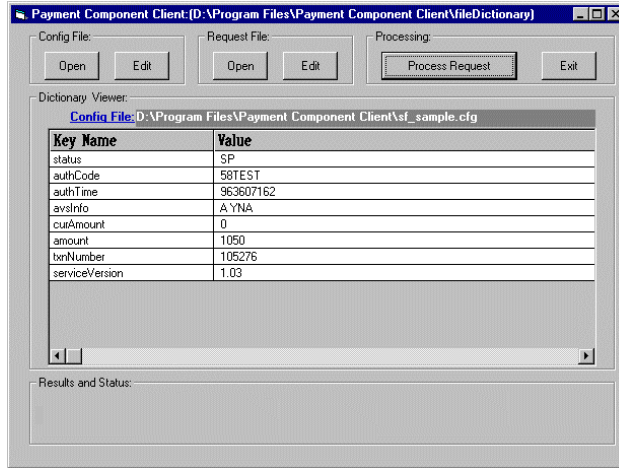
In addition to testing the Direct Payment component, the sample form functions as a demonstration of what a merchant’s purchase form can look like. It would, of course, be adapted to the needs of the merchant.

VB/VC++

When you install the Direct Payment component, a VB/VC++ client is also installed.

To test the client:

1. Click Start>Programs>Payment Component Client>Payment Component Client. If the Direct Payment component has been successfully installed, the following window appears:



2. In the Request File section, click Open.
3. In the Processing section, click Process Request.

If the Direct Payment component has functioned properly, you will see the transaction status at the bottom of the client in the Results and Status section, with the following notification – “Credit card was processed successfully. See the results above.”



In addition to testing the Direct Payment component, the sample form functions as a demonstration of what a merchant’s purchase form can look like. It would, of course, be adapted to the needs of the merchant.

Changing your configuration file

By default when you install the Direct Payment component, your configuration file is *sample.cfg*.

- To change which file is read by the Direct Payment component as your configuration file, in the Config File section click Open, then browse to the file desired.

- To edit the configuration file selected to be your default for transactions, in the Config File section click Edit. Your configuration file opens, permitting you to edit the information it contains.

Changing your request file

When you install the Direct Payment component, by default your request file is *fileDictionary*. The request file contains the name:value pairs from which transaction requests are built by the dictionary object.

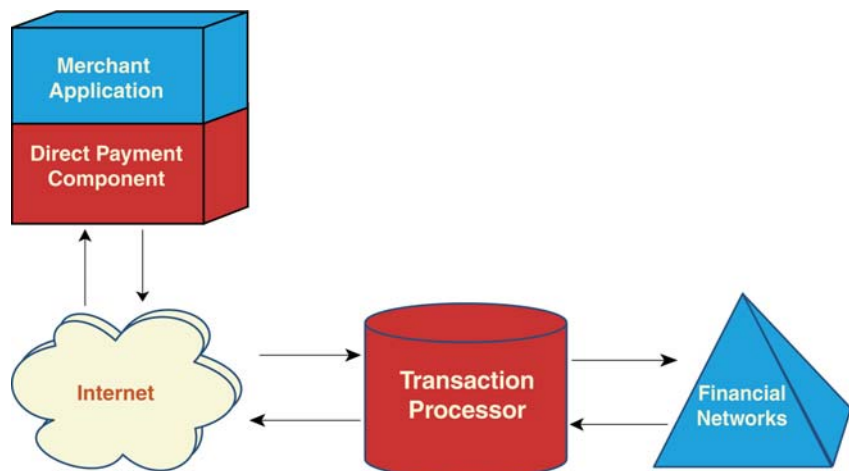
- To change which file is read by the Direct Payment component as your request file, in the Request File section click Open, then browse to the file desired.
- To edit the request file selected, in the Request File section click Edit. Your request file opens, permitting you to edit the information it contains.

Using Direct Payment

What is Direct Payment?

A merchant typically needs to send transaction requests in support of business transactions to a payment service. The merchant sends transaction requests (e.g., Purchase) to the Optimal Payments processor via the Direct Payment component, which acts as a transport medium.

The merchant application (ASP, VB, or VC++) uses a dictionary object to exchange relevant data with the Direct Payment component (see diagram below). The dictionary object is a data structure that stores information in pairs of keys, with a one-to-one key value (e.g., the credit card type might be *Visa*). This transaction request information is coded in an ASP script or in a VB or VC++ application. By executing this script or application, the information is communicated to the Direct Payment component, via the dictionary object, and the request is transmitted to the payment service. Results are also communicated back to the merchant application via the dictionary object.



The merchant uses Microsoft Internet Information Server (IIS) on an ASP (Active Server Pages) application or a VB or VC++ application, which in turn uses the Direct Payment component to communicate securely over the Internet, using SSL, with the payment service. The Direct Payment component uses sockets through SSL to communicate with the payment service.

Transactions are validated at both the Direct Payment component and the payment service levels. For example, the processor sends a response for both successful and failed transactions (e.g., if an invalid card type is supplied). The Direct Payment component sends an error message if information it requires is missing or incorrect (e.g., if your MerchantID value is missing).



In order to guarantee the security of customer information over the Internet, and in order to use the Direct Payment component and our processor, the merchant's Web site must be secure. Optimal Payments ensures the security of information transmitted between the merchant application and the payment service, but the merchant is responsible for securing information transmitted between the customer and the merchant's Web site (by, for example, having a certificate from a recognized certificate authority such as VeriSign).

Merchant transactions

The Direct Payment component supports merchant transaction functions such as Purchase, Transaction Lookup, and Query. For information on specific transaction requests, see the *Direct Payment Protocol Specification* manual.

Submitting transaction requests

The merchant application sends a variety of information to the payment service via the Direct Payment component. The application performs the following four tasks:

- Uses the *readConfigFile* method to send the path name of the merchant's configuration file to the payment service (see *Configuring for Direct Payment* on page 2-9).
- Uses the *processRequest* method to send the merchant's transaction requests (see *Merchant transactions* on page 2-2).

- Uses the *getStatus* method to request status information for a particular transaction request.
- Uses the *getErrorDescription* method to request error information for unsuccessful transactions.

The following is a brief overview of the process of submitting a transaction request via the Direct Payment component.



The purpose of the following examples is to demonstrate how to integrate an ASP script or a VB/VC++ application with the Direct Payment component. In order to scale up to meet the requirements of your enterprise, you will be able to adapt the working script or application provided by Optimal Payments. For information on testing your installation, see Testing Direct Payment on page 1-3.

ASP

An ASP script is used to transmit the information for these four tasks to Optimal Payments. A sample ASP script is supplied to the client with the software package (see Appendix B: *Sample ASP Script*). You may model your own ASP script after this, or you may have one already developed.

1. Register with the payment service. See *Merchant registration* on page 1-1 for more information.
2. Complete any fields in the configuration file that are either empty or that have values you want changed. See *Configuring for Direct Payment* on page 2-9 for more information.
3. Open the sample form with your browser, click Sample Data, and then click Submit Form. (The sample form is called *sample_form.asp*, and is placed by default in *c:\inetpub\wwwroot\payment sample* during installation.) This communicates the information in the fields to the Direct Payment component via the dictionary object. See Appendix B: *Sample ASP Script* for more information.
4. The Direct Payment component reads the parameters set in the merchant's configuration file and builds the request from the dictionary object.
5. The Direct Payment component then establishes a secure SSL connection with us over the Internet.
6. When the secure connection is established, the Direct Payment component sends the request to us, and waits for the result.

7. We process the transaction request and send the results back to the Direct Payment component, which in turn relays this information to the merchant application via the dictionary object. For more information on responses to each transaction type, see the *Direct Payment Protocol Specification* manual.

VB and VC++

1. Register with the payment service. See *Merchant registration* on page 1-1 for more information.
2. Complete any fields in the configuration file that are either empty or that have values you want changed. See *Configuring for Direct Payment* on page 2-9 for more information.
3. Click Start>Programs>Payment Component Client>Payment Component Client to open the sample application screen.
4. Click Process Request to process a test transaction. This communicates the information contained in the configuration file (*sample.cfg*) and the request file (*fileDictionary.txt*) to the Direct Payment component via the dictionary object.
5. The Direct Payment component reads the parameters set in the merchant's configuration file and builds the request from the dictionary object.
6. The Direct Payment component then establishes a secure connection with us over the Internet.
7. When the secure connection is established, the Direct Payment component sends the request to us, and waits for the result.
8. We process the transaction request and send the results back to the Direct Payment component, which in turn relays this information to the merchant application via the dictionary object. For more information on responses to each transaction type, see the *Direct Payment Protocol Specification* manual.

Failed transactions

Transaction requests are made to the payment service via the Internet. Normally, the merchant receives a response indicating that the transaction was successful. If the transaction request fails, an error response is

issued to the merchant. This response includes an error code and an error string.

There are four possible failure scenarios. The action to take varies with the type of error encountered.

- Communication failure
- Failure at the Direct Payment component level
- Failure at the payment service level
- No response is returned

Communication failure

In some instances, a transaction request is made, but due to the nature of the Internet, the transaction gets interrupted as a result of a communication failure. If this should happen, the Direct Payment component supports an active, automatic method for recovery. In the case of a communication failure, the following occurs:

- If the communication error occurs before the transaction request has been sent, the Direct Payment component automatically reinitiates communication.
- If the communication error occurs after a request has been sent but before a response has been received, the Direct Payment component sends a Transaction Lookup request to verify that the transaction request has been sent, and to resend the request if the transaction has not been completed.

Direct Payment component level

If the Direct Payment component returns an error, it indicates one of two things:

- Information required to execute the transaction request is missing (e.g., the *merchantId* field has not been filled out). In this case, the transaction request is not sent to the payment service.
- The transaction request is sent to the payment service, but some further problem prevents it from being processed. In this case, both a Direct Payment component and a payment service error message are returned.

Two important pieces of information are returned via the ASP script to the merchant application:

1. A status level number, indicating the nature of the error:

Status Level	Description
0	Transaction was processed successfully
-1	The configuration file is missing
-2	Some parameters are incorrect or missing
-3	A problem occurred calling a method
-4	A problem occurred using the dictionary object
-5	Communication problem
-6	The transaction status is unknown
-7	The server failed to process the request
-8	Manual intervention is required to process the request
-9	The transaction was aborted/rejected by the payment service.
-10	An error occurred when creating/using sockets
-11	An error occurred when initializing Winsock subsystem
-12	Proxy or host server unknown
-13	Client handshake error
-14	Cannot establish an SSL connection
-15	URL invalid
-16	Pseudo-random number generator (PRNG) not seeded correctly
-17	Server encountered internal error
-18	HTTP version not supported

Status Level	Description
-19	Bad HTTP request due to invalid syntax
-20	Authorization failed
-21	Web server refuses to fulfill the request
-22	Proxy authentication required
-23	Server is down
-24	Web server cannot fulfill the request

2. An error code and matching error string, describing the error more exactly. See *Direct Payment component error messages* on page A-1 for more information on the appropriate action to take.

Payment service level

An error at the payment service level indicates that the communication between the Direct Payment component and the payment service was successful, and that the transaction request was posted. However, an error of some sort has occurred with the payment service. Should this occur, the merchant receives a response from the payment service that includes an error code and an error string.

See *Payment service error messages* on page A-5 for more information on the appropriate action to take.

No response

Occasionally, the merchant receives no response at all. In this case, the user does not know whether or not the request was processed and does not have the payment service transaction number associated with the request (which would have been assigned if the request had in fact been processed). In such a case, the merchant must make a request for more information about the failed transaction, using the failure recovery transactions.

Additional features

In addition to transmitting merchant transaction requests, the Direct Payment component offers the following features:

- Secure communications (see *Secure communications* on page 2-8)
- Proxy server support (see *Proxy server support* on page 2-8)
- Active recovery in case of communication failure (see *Communication failure* on page 2-5)

Secure communications

The SSL protocol provides secure data communications through data encryption and decryption. The communication between the merchant application and our transaction processor is done over the Internet, via TCP/IP, using the Microsoft sockets (winsock.dll) with SSL.

Proxy server support

Merchants often employ proxy servers, in order to provide faster access to frequently accessed Web pages by reducing traffic. The proxy server maintains a cache of Web pages on a local server. Users who request frequently visited Web pages view them locally on the proxy server instead of at the original source on the Internet, reducing access time as well as bandwidth consumption.

Proxy servers are often used in conjunction with firewalls. Workstations on a network that is separated from the Internet by a firewall must use a proxy server outside the firewall to retrieve Web documents and other files and pass them through to the workstation.

The proxy server properties, read from the configuration file, are used to set the IP address or fully qualify the name on the proxy server used to connect to Internet. If you use a proxy server, you must provide the following two values in the configuration file:

- ProxyServer
- ProxyPort

The *ProxyServer* value provides the proxy server name and the *ProxyPort* value provides the connection port number of the proxy server you use to connect to the Internet. See *Configuring for Direct Payment* on page 2-9 for information on modifying your configuration file.

Configuring for Direct Payment

If you are using IIS, your ASP script provides the path name for your configuration file to the Direct Payment component. If you change this path from its default, where the configuration file is placed when you install the Direct Payment component, you must provide the new path in your ASP script (see *Configuration file path setup* on page B-2). If you are using VB/VC++, see *Changing your configuration file* on page 1-5 if you have changed the location of your configuration file.

Direct Payment configuration file

When you install the Direct Payment component, the configuration file is placed in following directory:

```
c:\program files\SureFire Commerce Direct Payment Component
for IIS\sample.cfg
```

Edit the configuration file with a text editor to ensure that the following fields are completed:

Field	Description
merchantId	Account transaction processing ID. See <i>Merchant registration</i> on page 1-1.
merchantPwd	Account transaction processing password. See <i>Merchant registration</i> on page 1-1.
account	Merchant account number. See <i>Merchant registration</i> on page 1-1.
PaymentServerURL	The payment service URL. This is given in the configuration file at installation.
PaymentServerPort	The payment service TCP port number
ProxyServer	Your proxy server name (only if you use a proxy server).
ProxyPort	Your proxy server port number (only if you use a proxy server).
HTTPVersion	The HTTP version used.

Field	Description
Cipher	Cipher suite used to implement the SSL communication.
Timeout	<p>Time, in seconds, the Direct Payment component waits for a response from the payment service before terminating communication. Default=120. Range=60–180.</p> <p>NOTE: You will not receive an error message if you set the TimeOut value outside of this range. Optimal Payments recommends staying within this range, however.</p>
Retries	<p>Number of times the Direct Payment component tries to establish communication before returning an error message. Default=3. Range=2–5.</p> <p>NOTE: You will not receive an error message if you set the Retries value outside of this range. Optimal Payments recommends staying within this range, however.</p>
LogLevel	<p>The level of information that is logged by the Direct Payment component, which is viewed by Event Viewer. Two levels are possible:</p> <ul style="list-style-type: none"> • Information – logs successful transactions. • Error – logs failed transactions. <p>If you want to log all transactions, use the following syntax: LogLevel=Error, Information</p>



Your configuration file contains sensitive information – your account transaction processing ID and password, and your merchant account number. Be sure to keep your configuration file protected from illegitimate access.

Error Return Codes

Error messages

There are two classes of error messages associated with the use of the Direct Payment component:

- Direct Payment component errors
- Payment service errors, related to processing the request

The status, error code, and error string are returned to the merchant application and appear in the ASP script.

Direct Payment component error messages

The following table lists all errors that can be returned by the Direct Payment component.

Error Code	Error String	Action
ERROR(50)	PaymentServerURL is missing.	Add our payment processor's URL to the configuration file.
ERROR(51)	MerchantID is missing.	Add your merchant ID to the configuration file.
ERROR(52)	ProxyServer is missing.	Add the proxy server name to the configuration file.
ERROR(53)	ProxyPort is missing.	Add the proxy server port number to the configuration file.
ERROR(54)	Configuration file missing.	Verify that the path to the configuration file is in your ASP script, and that the path is correct.

Error Code	Error String	Action
ERROR(55)	Reading error occurred in the configuration file.	Retry. If error persists, contact technical support.
ERROR(56)	Error opening the configuration file.	Retry. If error persists, please contact technical support.
ERROR(57)	Configuration file not supplied.	Verify that the path to the configuration file is in your ASP script, and that the path is correct.
ERROR(58)	MerchantPwd is missing.	Add the merchant password to the configuration file.
ERROR(59)	AccountID is missing.	Add the merchant account ID to the configuration file.
ERROR(60)	The Cipher is missing.	Add the Cipher to the configuration file.
ERROR(61)	HTTPVersion is missing.	Add the HTTPVersion parameter to the configuration file.
ERROR(62)	PaymentServerPort is missing.	Add the PaymentServerPort to the configuration file.
ERROR(63)	PaymentServerPort is invalid.	Please verify the PaymentServerPort.
ERROR(64)	ProxyPort is invalid.	Please verify the ProxyPort.
ERROR(65)	HTTPVersion is invalid.	Please verify the HTTPVersion.
ERROR(66)	The dictionary object is empty.	Fill your dictionary object before sending request.
ERROR(67)	Cannot read the dictionary object.	Retry. If error persists, please contact technical support.
ERROR(68)	Cannot write request response in the dictionary object.	Please do a Transaction Lookup or Query request to ascertain the status.
ERROR(69)	Call readCfgFile method first.	Your script must call the readCfgFile method before calling the processRequest method.

Error Code	Error String	Action
ERROR(70)	The server failed to process your transaction.	Verify the parameters in your request. Retry.
ERROR(71)	Request sent but no response received.	Please do a Transaction Lookup or Query request to ascertain the status.
ERROR(72)	Manual intervention required to process your request.	Please contact technical support.
ERROR(73)	Transaction aborted/rejected by the server. Request not correct.	Verify your request parameters.
ERROR(74)	Communication error occurred.	Retry. If error persists, please contact technical support.
ERROR(75)	Cannot initialize the Winsock subsystem	Retry. If error persists, check your Winsock dll.
ERROR(76)	The URL is invalid.	Verify that you have the correct URL in your configuration file.
ERROR(77)	Cannot create SSL context.	Retry.
ERROR(78)	Cannot create SSL structure.	Retry.
ERROR(79)	Cannot set the cipher specified.	Verify the Cipher parameter specified in the configuration file.
ERROR(80)	PRNG not seeded correctly.	Retry.
ERROR(81)	Cannot set the socket descriptor as I/O for TLS/SSL.	Retry.
ERROR(82)	Cannot establish SSL connection with the remote server.	Retry. If error persists, please contact technical support.
ERROR(83)	Communication timed out.	Please increase the timeout value in your configuration file.
ERROR(84)	Cannot connect to the server.	Retry. If error persists, please contact technical support.
ERROR(85)	Cannot send the request.	Retry. If error persists, please contact technical support.

Error Code	Error String	Action
ERROR(86)	Cannot read the data.	Retry. If error persists, please contact technical support.
ERROR(87)	Socket creation failed.	Retry. If error persists, please contact technical support.
ERROR(88)	Proxy server specified is unknown.	Verify the ProxyServer parameter set in the configuration file.
ERROR(89)	Server specified is unknown.	Verify the PaymentServerURL parameter in the configuration file.
ERROR(90)	Cannot build the HTTP header.	Please verify your URL.
ERROR(91)	Network is down.	Retry. If error persists, please contact technical support.
ERROR(92)	Connection aborted.	Retry. If error persists, please contact technical support.
ERROR(93)	Connection forcibly closed by the server.	Retry. If error persists, please contact technical support.
ERROR(94)	Request could not be understood by the server.	Please verify your URL and/or your HTTPHeader parameters.
ERROR(95)	Request requires user authentication.	Verify your proxy server parameters.
ERROR(96)	Server refuses to fulfill your request.	Retry. If error persists, please contact technical support.
ERROR(97)	Proxy authentication required.	Verify your proxy server parameters.
ERROR(98)	Server encountered internal error.	Retry. If error persists, please contact technical support.
ERROR(99)	HTTP protocol version not supported by the server.	Verify your HTTPVersion parameter in the configuration file.
ERROR(100)	Service unavailable.	Retry. If error persists, please contact technical support.

Error Code	Error String	Action
ERROR(101)	Server encountered problem fulfilling your request.	Please verify your URL and/or your HTTPHeader parameters.

Payment service error messages

If, after sending a transaction request, you receive an error message from the payment service, some or all of the following parameters are returned (in addition to some request-specific parameters):

Parameter	Description
status	E indicates that an error occurred.
errCode	An integer value associated with the error that occurred.
errString	String that describes the error that occurred.
subError	Lower level error that occurred. This value is only used when trying to resolve issues in co-operation with technical support.
subErrorString	String that describes the lower level error that occurred. This value is only used when trying to resolve issues in co-operation with technical support.
clientVersion	The version of the protocol that the payment service is running.

Action codes

The payment service returns an error code and an error string for any error encountered. There is also an action code associated with each error (not returned by the payment service). In the table in Error codes and strings below, find the error code returned to you in order to find the action code associated with it.

The meanings for the action code abbreviations are as follows:

- **AR** = Authorization Refused. The card cannot be authorized. Ask the user to verify credit card information or to use a different credit card.

- **CP** = Customer Parameter. The customer has provided incorrect information. Ask the customer to correct the information.
- **IE** = Internal Error. There is a problem on the system that you should report to technical support. You should also determine the status of the transaction using a Transaction Lookup request.
- **MP** = Merchant Parameter. Your application has provided incorrect information. Verify your information.
- **SR** = Service is Restarting. Please retry later.

Error codes and strings

The table immediately below contains all the error codes and error strings that might be returned while sending transaction requests, in addition to action codes (which are not returned by the payment service). The right-most column lists the action codes associated with each error.

Error Code	Error String	Description	Action Code
1	Error in HTTP environment.	HTTP level used not supported by server side. Should not occur.	IE
2	No response from process within timeout settings. Please do a Transaction Lookup to determine the transaction status.	After the transaction was sent, no response was received, because the transaction was never processed. The clearing network could be down.	IE
3	Payment service is currently restarting. Retry later. If the problem persists, please contact technical support.	Our gateway process connecting to a clearing house is temporarily down – most likely due to a restart because of connectivity problems with the clearing house.	SR
4	Could not read configuration file. Please contact technical support.	Server side error. Should not occur.	IE
5	Request method Get not allowed.	Only POST method is supported.	MP

Error Code	Error String	Description	Action Code
20	Remote validation error. Please verify request parameters.	One of the required parameters is not valid.	MP
21	Request validation failed. Please verify request parameters.	One of the required parameters is not valid.	MP
30	Request processing failure. Please contact technical support.	Server side error. Unlikely to occur, but could happen as a result of a configuration error.	IE
31	Request processing failure. Please contact technical support.	Server side error. Should not occur.	IE
32	Request not accepted. Please verify request parameters.	This error occurs if the request comes from an IP not configured for the merchant.	MP
33	Request processing failure. Please contact technical support.	Server side error. Should not occur.	IE
34	Authorization refused.	This error usually results from a hard decline from the clearing house, or from declines due to fraud prevention measures. A sub-error code occurs in the latter case.	AR
56	Invalid amount. Please verify request parameters.	An amount greater than the range supported was entered.	IE (MP)
57	Invalid CVD indicator. Please verify request parameters.	An incorrect value was used for the <i>cvdIndicator</i> parameter.	IE (MP)
58	Invalid CVD value. Please verify request parameters.	An incorrect value was used for the <i>cvdValue</i> parameter.	IE (MP)
63	Invalid account ID. Please verify request parameters.	Some account ID values sent with the transaction do not correspond with the values stored in the our database (e.g., incorrect <i>merchantPwd</i> entered).	MP

Error Code	Error String	Description	Action Code
91	Invalid payment information. Please verify request parameters	The card number, the brand, expiry date, or a combination thereof is incorrect. The suberror text describes the problem in more detail.	CP
92	Invalid payment method. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the payment method (<i>payMethod</i>) parameter.	MP
93	Invalid card type. Please verify request parameters.	This error results when a transaction is attempted with a card type that is not supported.	CP
101	Internal error. Please contact technical support.	Server side error. Should not occur.	IE
111	Could not assign name. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the name parameter.	CP
113	Could not assign address. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the address parameter.	CP
116	Could not assign province. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the province parameter.	CP
117	Could not assign zip. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the zip parameter.	CP
118	Could not assign country. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the country parameter.	CP

Error Code	Error String	Description	Action Code
119	Could not assign email. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the email parameter.	CP
120	Could not assign phone number. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the phone number parameter.	CP
121	Could not assign merchantTxn. Please verify request parameters.	A transaction was attempted with the wrong data type entered for the <i>merchantTxn</i> parameter.	MP
130	Invalid expiry date value. Please verify request parameters.	The expiry date is incorrect.	MP
131	Operation not supported. Please verify request parameters.	The transaction attempted is unknown (Purchase or Credit are examples of known transaction types), or the account is not configured for the transaction attempted.	MP
132	Missing mandatory parameters for operation. Please verify request parameters.	A field that is mandatory for the transaction (e.g., <i>cardType</i>) was not sent with the transaction.	MP
133	Invalid amount format. Should be integer. Please verify request parameters.	The amount of a transaction must be given with no decimal (e.g., \$4.95 = 495).	MP
134	Invalid client version. Please verify request parameters.	The <i>clientVersion</i> parameter must be set to 1.1 in order to use current functionality.	MP
137	Invalid zip code length. Please verify request parameters.	The <i>zip</i> parameter must be a maximum of 10 alphanumeric characters.	MP
138	Invalid zip code length. Please verify request parameters.	The <i>zip</i> parameter must be a maximum of 10 alphanumeric characters.	MP

Error Code	Error String	Description	Action Code
139	Invalid expiry date format. Please verify request parameters.	The format for the <i>cardExp</i> parameter must be "MM/YY". E.g., September 2003 = 09/03	MP
161	Not authorized to make request. Please verify request parameters.	The user name and/or password included with the Settlement transaction request are not correct. These are the <i>merchantId</i> and <i>merchantPwd</i> parameters, respectively.	MP
163	Invalid txnNumber. Please verify request parameters.	The authorization number included with the Settlement transaction request is not correct or cannot be found.	MP
174	Request failed. Please contact technical support.	Server side error. Should not occur.	IE
175	Requested Settlement exceeds remaining Authorization.	A Settlement transaction request must be equal to or less than the amount remaining to settle on an Authorization.	MP
176	Invalid settlement amount.	A Settlement transaction request must be equal to or less than the amount remaining to settle on an Authorization.	MP
178	Transaction already fully settled.	There is no amount remaining to settle on the original Authorization.	MP
209	Payment brand not in store list.	The transaction request was sent with a credit card type for which the account is not configured.	MP
210	Payment instrument error. Please verify request parameters.	Server side error. Should not occur.	CP

Error Code	Error String	Description	Action Code
212	Authorization refused – AVS did not match.	This error occurs when AVS fails on a transaction that otherwise would have been successful.	AR
213	The Authorization was aborted.	Server side error. Should not occur.	AR
221	Authorization failed.	The transaction was not authorized. The suberror text describes the problem in more detail.	AR
222	Currency mismatch with store.	Server side error. Should not occur.	MP
234	Settlement refused because credit card did not pass negative database check.	A Settlement was attempted on a credit card that was entered into the negative database after the authorization that you are trying to settle was approved.	MP
281	Not authorized to make request. Please verify request parameters.	The user name and/or password included with the Query transaction request are not correct. These are the <i>merchantId</i> and <i>merchantPwd</i> parameters, respectively.	MP
284	Invalid transaction number. Please verify request parameters.	The transaction number included with the Query transaction request cannot be found.	MP
311	Not authorized to make request. Please verify request parameters.	The user name and/or password included with the Transaction Lookup transaction request are not correct. These are the <i>merchantId</i> and <i>merchantPwd</i> parameters, respectively.	MP
321	Not authorized to make request. Please verify request parameters.	The user name and/or password included with the Authorization transaction request are not correct. These are the <i>merchantId</i> and <i>merchantPwd</i> parameters, respectively.	MP

Error Code	Error String	Description	Action Code
331	Not authorized to make request. Please verify request parameters.	The user name and/or password included with the Credit transaction request are not correct. These are the <i>merchantId</i> and <i>merchant-Pwd</i> parameters, respectively.	MP
333	Invalid txnNumber. Please verify request parameters.	The authorization number included with the Credit transaction request is not correct or cannot be found.	MP
334	Credit refused because credit card did not pass negative database check.	A Credit was attempted to a credit card that was entered into the negative database after the Settlement that you are trying to credit was completed.	MP
345	Requested Credit exceeds remaining funds settled.	A Credit transaction request must be equal to or less than the amount of funds available to credit (i.e., the amount settled for that credit card).	MP
346	Invalid credit amount.	A Credit transaction request must be equal to or less than the amount of funds available to credit (i.e., the amount settled for that credit card).	MP
347	Internal error. Please contact technical support.	Server side error. Should not occur.	IE
348	No settled funds available for credit.	A Credit transaction request can only be made on a credit card that has settled amounts remaining on it.	MP
353	Unknown txnNumber. Please verify request parameters.	The transaction number included with the Settlement transaction request is incorrect.	MP

Error Code	Error String	Description	Action Code
356	Unknown merchant transaction, already fully credited, or no amount available for credit.	A Credit transaction request was attempted where there were no funds remaining to be settled.	MP
600	Invalid shipment method. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the shipment method (<i>shipMethod</i>) parameter.	MP
601	Invalid carrier. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the <i>carrier</i> parameter.	MP
651	Invalid previous customer. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the <i>previousCustomer</i> parameter, which indicates whether the customer has previously shopped online with this merchant.	MP
652	Invalid customer ID. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the <i>customerId</i> parameter.	MP
653	Invalid customer IP. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the customer's IP address (<i>customerIP</i>) parameter.	MP
701	Invalid product type. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the product type (<i>productType</i>) parameter.	MP
702	Invalid product code. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the product code (<i>productCode</i>) parameter.	MP

Error Code	Error String	Description	Action Code
731	Invalid transaction category. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the transaction category (<i>txnCategory</i>) parameter.	MP
751	Invalid merchant SIC code. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the ISO Standard Industry Code (<i>merchantSIC</i>) parameter.	MP
752	Invalid customer account open date. Please verify request parameters.	A transaction was attempted with an incorrect value entered for the parameter indicating the date the customer account was opened (<i>custAcctOpenDate</i>).	MP
771	Invalid user data. Please verify request parameters.	A transaction was attempted with an incorrect value entered for a user data (e.g., <i>userData04</i>) parameter.	MP



The merchant application should not encounter internal errors during normal operation of the payment service. If they are encountered, contact technical support.

Suberror codes and strings

Suberror Code	Suberror String	Action Code
1000	Approval	AR
1001	Unknown response from clearing network	AR
1002	Clearing network response is Reenter	AR
1003	Clearing network response is Referral	AR

Suberror Code	Suberror String	Action Code
1004	Clearing network response is Pickup	AR
1005	Clearing network response is Decline	AR
1006	Clearing network response is Timeout	AR
1007	Card in negative database	AR
1008	Invalid merchant number	AR
1010	CVV2 check failed	AR
1011	Approved with ID	AR
1012	Invalid request	AR
1013	Invalid amount	AR
1014	Invalid account	AR
1015	Retry	AR
1016	Invalid expiry date	AR
1017	PIN invalid	AR
1018	Unauthorized transaction	AR
1019	Max PIN retries	AR
1020	Duplicate transaction	AR
1021	Invalid account match	AR
1022	Invalid amount match	AR
1023	Invalid item number	AR
1024	Item voided	AR
1025	Must balance now	AR
1026	Use duplicate	AR
1027	No duplicate found	AR
1028	Invalid data	AR

Suberror Code	Suberror String	Action Code
1029	No transaction found	AR
1030	Approved but not captured	AR
1031	Approved auth only	AR
1032	Invalid bank ID	AR
1034	Transaction type invalid	AR
1035	Approved debit	AR
1036	DB unavailable2	AR
1037	DB unavailable3	AR
1038	DB unavailable4	AR
1039	Unauthorized user	AR
1040	Invalid card	AR
1041	DB issuer unavailable	AR
1042	Invalid pos card	AR
1043	Account type invalid	AR
1044	Invalid prefix	AR
1045	Invalid FIID	AR
1046	Verify	AR
1047	Invalid LIC	AR
1048	Invalid state	AR
1049	EDC unavailable	AR
1050	DB unavailable1	AR
1051	Scan unavailable	AR
1052	Exceeds max amount	AR

Suberror Code	Suberror String	Action Code
1053	Exceeds max uses	AR
1054	Unable to process	AR
1055	Invalid request for terminal	AR
1056	Invalid date	AR
1057	Invalid format	AR
1058	No pickup	AR
1059	No funds available	AR
1060	Exceed limit	AR
1061	Restricted card	AR
1062	Mac key incorrect	AR
1063	Exceed frequency limit	AR
1064	Retain card	AR
1065	Late response	AR
1067	No share arrangement	AR
1068	Function unavailable	AR
1069	Invalid key	AR
1070	Invalid lifecycle trans	AR
1071	Pin key error	AR
1072	Mac sync error	AR
1073	Security violation	AR
1074	IST unavailable	AR
1075	Invalid issuer	AR
1076	Invalid acquirer	AR
1077	Invalid originator	AR

Suberror Code	Suberror String	Action Code
1078	System error	AR
1079	Duplicate reversal	AR
1081	Credit card is blocked	AR
1082	Credit card is stolen	AR
1083	Credit card is forged	AR
4000	Declined by Risk Management	AR

Unmapped suberror codes and strings

An unmapped suberror code and suberror string are returned in the event that a suberror response is not mapped to a standard 4-digit suberror code and string. All unmapped suberror codes are between 0 and 999, making them easy to differentiate from our suberror codes, which are all greater than 1000.

Sample ASP Script

Sample script

During installation, a sample ASP script is placed in the following directory:

```
c:\InetPub\wwwroot\PaymentSample\sample_process.asp
```

The ASP script comprises the following seven sections:

Section	Function
Variable declaration	Declares the variables for the script.
Variable assignment	Assigns the variable values, retrieving them from the Request form object.
Dictionary Initialization	Creates the dictionary object used to communicate merchant information to the Direct Payment component.
Configuration File Path Setup	Provides the path for your configuration file to the Direct Payment component. If you change this path from the one given by default, you must make that change here.
ASP SSL payment component initialization	Reads your configuration file for transaction details, and returns an error if unable to do so.
Credit Card Processing	Sends the transaction request to the payment service.
DisplayDict() Function	Displays the results of the transaction request.

Dictionary Initialization

This section of the ASP script contains parameters for the transaction requests supported by the Direct Payment component. The values for these parameters must be placed in double quotation marks (" "), and separated by commas.



The parameter names are case sensitive. If you use your own ASP script, ensure that the names and spellings of your parameters are identical to the parameters in the sample ASP script, and in the parameter fields in the Direct Payment Protocol Specification manual.

In order to see the mandatory and optional fields for the transaction request you desire, see the corresponding transaction in the *Direct Payment Protocol Specification* manual.

Configuration file path setup

During installation, the path for the configuration file is set by default to:

```
c:\program files\SureFire Commerce Direct Payment Component for  
IIS\sample.cfg
```

If you change the location of the configuration file, you must also change the path given in this section of the ASP script, in order that the Direct Payment component can locate it.

Sample ASP script

```
<%@LANGUAGE=VBSCRIPT%>  
<%response.buffer=true%>  
<%  
  
'Declare variables  
dim cardType  
dim cardNumber  
dim cardExp  
dim amount  
dim operation
```

```
dim merchantTxn
dim clientVersion
dim custName1
dim streetAddr
dim streetAddr2
dim phone
dim email
dim city
dim province
dim zip
dim country

'Get info from Request form object
cardType = Request.Form("cardType")
cardNumber = Request.Form("cardNumber")
cardExp = Request.Form("cardExp")
amount = Request.Form("amount")
operation = Request.Form("operation")
merchantTxn = Request.Form("merchantTxn")
clientVersion = Request.Form("clientVersion")
custName1 = Request.Form("custName1")
streetAddr = Request.Form("streetAddr")
streetAddr2 = Request.Form("streetAddr2")
phone = Request.Form("phone")
email = Request.Form("email")
city = Request.Form("city")
province = Request.Form("province")
zip = Request.Form("zip")
country = Request.Form("country")

'=====
'===== Dictionary Initialization =====
```

```
'=====
Set Dict = Server.CreateObject ("Scripting.Dictionary")
'=====

Dict.Add "cardType", cardType
Dict.Add "cardNumber", cardNumber
Dict.Add "cardExp", cardExp
Dict.Add "amount", amount
Dict.Add "operation", operation
Dict.Add "merchantTxn", merchantTxn
Dict.Add "clientVersion", clientVersion
Dict.Add "custName1", custName1
Dict.Add "streetAddr", streetAddr
Dict.Add "streetAddr2", streetAddr2
Dict.Add "phone", phone
Dict.Add "email", email
Dict.Add "city", city
Dict.Add "province", province
Dict.Add "zip", zip
Dict.Add "country", country

'=====
'==== Configuration file path setup =====
'=====
ConfigFile = "c:\program files\SureFire Commerce Direct Payment
Component for IIS\sample.cfg"

'=====
'==== ASPSSLPayment component initialization =====
'=====
Set Payment = Server.CreateObject ("sfPaymentComponent.SFFalcon.1")
init = Payment.readConfigFile (ConfigFile)
```

```
if not init then
%>
  <html>
    <head>
      <Title>SureFire Commerce</title>
    </head>
    <body bgcolor=white>
      <table border=0 cellpadding=0 cellspacing=0 bgcolor=white
width=350>
        <tr>
          <td align=center>
            <br>
          </td>
        </tr>
      </table>
      <br>
      <table border=0 cellpadding=0 cellspacing=0 bgcolor=lightgrey
width=350>
        <tr>
          <td align=center>
            <font face="arial" size=3 color="Red">
              <b><i>** Credit card processing failed **</i></b>
              <br>
              <i>Status:<%=Payment.getStatus%> <br>Descrip-
tion:<%=Payment.getErrorDescription%></B></i>
            </font>
            <br>
          </td>
        </tr>
      </table>

<%
```

```
end if
'=====
'===== CREDIT CARD PROCESSING =====
'=====
if init then
%>
    <html>
        <head>
            <Title>SureFire Commerce</title>
        </head>
        <body bgcolor=white>
            <table border=0 cellpadding=0 cellspacing=0 bgcolor=white
width=350>
                <tr>
                    <td align=center>
                        <br>
                    </td>
                </tr>
            </table>
            <br>
            <table border=0 cellpadding=0 cellspacing=0 bgcolor=lightgrey
width=350>
                <tr>
                    <td align=center>
                        <font face="arial" size=3 color="Navy"><b>TRANSAC-
TION STATUS:</b></font><br>
                        <font face="arial" size=3
color="Navy"><b><i>Processing credit card...</i></b></font>
                        <br>
                    </td>
                </tr>
            </table>
```

```
        <BR><font face="arial" size=3 color="Navy"><b><i>Infor-
information Submitted:</I></b></font>
    <%
DisplayDict(1)
Response.Flush()
result = Payment.processRequest(Dict)
if result then
    %>
    <br>
    <table border=0 cellpadding=0 cellspacing=0 bgcolor=light-
grey width=350>
        <tr>
            <td align=center>
                <font face="arial" size=3 color="Navy"><b>TRANS-
ACTION STATUS:</b></font><br>
                <font face="arial" size=3 color="Green"><b><i>**
Credit card was processed successfully **</i></b></font><br>
            </td>
        </tr>
    </table>

    <BR><font face="arial" size=3 color="Navy"><b><i>Information
Returned:</I></b></font>
    <%
DisplayDict(1) 'After Processing
else
    %>
    <table border=0 cellpadding=0 cellspacing=0 bgcolor=lightgrey
width=350>
        <tr>
            <td align=center>
```

```

        <font face="arial" size=3 color="Red">
            <b><i>** Credit card processing failed **</i></b>
            <br>
            <i>Status:<%=Payment.getStatus%><br>Description:<%=Pay-
ment.getErrorDescription%></font></B></i>
        </font>
        <br>
    </td>
</tr>
</table>

```

```

<%
    DisplayDict(1) 'After Processing
end if

```

```

end if
%>
</body>
</html>
<%

```

```

Set Payment=nothing
Set Dict=nothing

```

```

'=====
'===== Request() Function =====
'=====

```

```

function Request_func ()
On error resume next
error = Payment.Process (Dict)
if err.number = 0 then

```

```
Request = 0
else
    Request = -1
end if
end function

'=====
'===== DisplayDict() Function =====
'=====

Sub DisplayDict(init_len)
Dim a, i, name
a = Dict.Keys 'Get the keys

if init_len <> 0 then
    'display dictionary first time
    Response.Write("<table border=0 cellpadding=2 cellspacing=0>")
    For i = 0 To Dict.Count-1
        if (i mod 2) = 0 then
            Response.Write "<tr>"
            end if
            name = a(i)
            Response.Write "<td><b> " & name & " &nbsp;<b></td><td>" &
Dict.Item(name) & "&nbsp;</td>"
            if (i mod 2) <> 0 then
                Response.Write "</tr>"
            end if
        next
        Response.Write("</table>")
    end if

End Sub

%>
```


A

- action codes A-5
- ASP script 2-1
 - configuration file path setup B-2
 - dictionary initialization B-2
 - sample B-2
 - submitting transaction requests 2-3

C

- certificate authority, using 2-2
- Cipher 2-10
- codes
 - action A-5
 - error A-6
- communication failure 2-5
- configuration file 2-9
 - Cipher 2-10
 - HTTPVersion 2-9
 - LogLevel 2-10
 - merchant account number
 - merchant account number 2-9
 - merchantId 2-9
 - merchantPwd 2-9
 - path to B-2
 - PaymentServerPort 2-9
 - PaymentServerURL 2-9
 - ProxyPort 2-9
 - ProxyServer 2-9
 - Retries 2-10
 - Timeout 2-10

D

- dictionary object 2-1
- Direct Payment component
 - client 1-5

- failure 2-5
- installing 1-2
- overview 2-1
- testing for IIS 1-3
- testing for VB/VC++ 1-4
- Direct Payment configuration file 2-9
 - changing 1-5

E

- error messages A-1
 - action codes A-5
 - Direct Payment component A-1
 - error codes and strings A-6
 - payment service A-5
 - status level 2-6
- Event Viewer
 - viewing logging information 2-10

F

- failure recovery 2-4

H

- HTTPVersion 2-9

I

- IIS 1-2
 - installing Direct Payment component 1-1, 1-2
- interrupted transaction requests 2-5

L

- logging information

viewing with Event Viewer 2-10
LogLevel 2-10

M

merchant
 application 2-2
 registration 1-1
 transactions 2-2
merchant interface
 validating fields 1-3
merchantId 2-9
merchantPwd 2-9

P

parameters
 error messages, from payment
 services A-5
payment service failure 2-7
PaymentServerPort 2-9
PaymentServerURL 2-9
proxy server support 2-8
ProxyPort 2-9
ProxyServer 2-9

R

reducing traffic
 proxy servers 2-8
registering for Direct Payment
 component 1-1
request file, for VB/VC++ 1-6
requirements
 software 1-1
response failure 2-7
Retries 2-10

S

security 2-2, 2-8
software requirements 1-1
status levels, failure 2-6

submitting transaction requests 2-2

T

testing
 Direct Payment component, for IIS 1-3
 Direct Payment component, for VB/
 VC++ 1-4
Timeout 2-10
transaction failure
 communication 2-5
 Direct Payment component 2-5
 no response 2-7
 payment service 2-7
 status levels 2-6
Transaction Lookup request 2-5
transaction requests 2-1, 2-2
 failed 2-4
 no response 2-7
 submitting 2-2
transaction validation 2-2

V

validating fields 1-3
VB 1-2
 application 2-1
 submitting transaction requests 2-4
 testing the component 1-4
VC++ 1-2
 application 2-1
 submitting transaction requests 2-4
 testing the component 1-4