



Java Direct Payment for Windows NT and UNIX

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 Java Direct Payment

Software requirements	1-1
Merchant registration	1-1
Installing Java Direct Payment on NT	1-2
Installing Java Direct Payment on UNIX	1-2
Adding our server to your trusted sites	1-3
Testing Java Direct Payment on NT and UNIX	1-3
Using the command line.	1-4
Using the Web interface.	1-5

2 Introduction to Java Direct Payment

What is Java Direct Payment?	2-1
Merchant transactions.	2-2
Submitting transaction requests.	2-2
Functions and features	2-3
Failed transactions	2-4
Communication failure.	2-4
Java Direct Payment level.	2-5
Payment service level	2-6
No response.	2-7
Additional features	2-7
Secure communications	2-7
Proxy server support.	2-7
Configuring for Direct Payment.	2-8

A Error Return Codes

Error messages A-1

Java Direct Payment error messages A-1

Payment service error messages A-6

 Action codes. A-6

 Error codes and strings A-7

 Suberror codes and strings A-15

 Unmapped suberror codes and strings A-19

Index

Installing Java Direct Payment

Java Direct Payment was developed for UNIX and NT platforms. It uses the Sun JSSE package to implement SSL communications with the Terra Payments processor over the Internet.

Software requirements

Java Direct Payment was developed for NT and UNIX platforms. It requires:

- Sun JRE *or* JDK 1.2.2 or higher
- Sun JSSE 1.0.2

Merchant registration

Before using Java Direct Payment, the prospective merchant must register on the Terra 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 must call technical support to receive an account ID, an account transaction processing ID, and a password. These values are important, as they are used when you submit a transaction via Java Direct Payment. For more information, see the *Direct Payment Protocol Specification* manual.

Installing Java Direct Payment on NT



In order for Direct Payment to be able to communicate securely with the our processor, you must register the Sun JSSE provider correctly. This registration process is found in the installation instructions for JSSE, downloaded from java.sun.com.

To install Java Direct Payment:

1. Unzip the *jPayment.zip* archive.

The package will be installed in the directory *jPayment*, which is created in your current directory.

2. Add *jPayment.jar* to the classpath environment variable or install them in the `<java_home>/jre/lib/ext` directory.
3. Complete the parameters in the *sample.properties* file with the information obtained during registration. Save the file.



In some cases, the user might have to manually include our server among their "trusted" sites (in other cases, this will have been done automatically).

Installing Java Direct Payment on UNIX



In order for Direct Payment to be able to communicate securely with our processor, you must register the Sun JSSE provider correctly. (This registration process is found in the installation instructions for JSSE, downloaded from java.sun.com).



You may need to log on as root in order to have the authorization to install the package.

To install Java Direct Payment:

1. Enter the following command to untar the archive *jPayment.tar*:

```
tar -xvf jPayment.tar
```

The package will be installed in the directory *jPayment*, which is created in your current directory.

2. Add *payment.jar* to the classpath environment variable or install them in the `<java_home>/jre/lib/ext` directory.
3. Complete the parameters in the *sample.properties* file with the information obtained during registration. Save the file.

Adding our server to your trusted sites

In some cases, you might have to manually include our server among your “trusted” sites (in other cases, this will have been done automatically). To do this, you must import our security certificate file, *firepay.cer*, using the `keytool` command.

To import our certificate file:

1. Rename the `<java_home>/jre/lib/security/cacerts` file to `<java_home>/jre/lib/security/cacerts.old`.
2. Enter the following `keytool` command to import the certificate to your *cacerts* file:

```
keytool -import -trustcacerts -alias firepay -file  
<path/to/certificatefile/firepay.cer> -keystore  
<java_home>/jre/lib/security/cacerts> -storepass  
[your password]
```

Where `[your password]` is the password you use to protect your list of trusted sites. You will be prompted for confirmation to trust the certificate.

3. Type Yes.

Testing Java Direct Payment on NT and UNIX

Java Direct Payment can be tested using one of these methods:

- Using the command line

- Using the Web interface

Using the command line

To test Java Direct Payment:

Run the client class provided in the package, with the following arguments:

```
java Client c:/dev/jPayment/properties/sample.properties Your_Built_Request
```

Where *Your_Built_Request* is:

```
cardType=VI&cardNumber=4387751111011&amount=2500&cardExp=04/04&operation=P&merchantTxn=100000&clientVersion=1.1&custName1=Joe Smith&streetAddr=123 Main Street&city=Toronto&province=ON&zip=H4B2Y1  
&country=CA&phone=514-555-5555&email=joe.smith@domain.com  
&cvdIndicator=1&cvdValue=123
```



When you log in as root, some Unix platforms will not include the current directory in the path for security reasons.

You should get the results of an authorized transaction:

```
status=SP&authCode=105779&authTime=1056049131  
&avsInfo=Y&cvdInfo=M&curAmount=0&amount=2500  
&txnNumber=1000006&clientVersion=1.1
```



If you get a transaction unrecoverable error, it is probably because our server is not in your list of "trusted" sites. See Installing Java Direct Payment on UNIX on page 1-2 for more information.

Using the Web interface

To test Java Direct Payment:

1. Copy the three jsp files (*sample_form.jsp*, *sample_process.jsp*, *sample_error.jsp*) into the Web site root folder.



Make sure your Web server's class path configuration contains *jpayout.jar*.

2. Within the *sample_process.jsp* file, modify the *configFile* variable to point to the correct location of your *sample.properties* file.
3. Enter the following URL in your browser address field: *localhost/sample_form.jsp*.

4. Click the Sample Data button.
5. Click the Submit Form button to send the request. You should get the results of an authorized transaction.

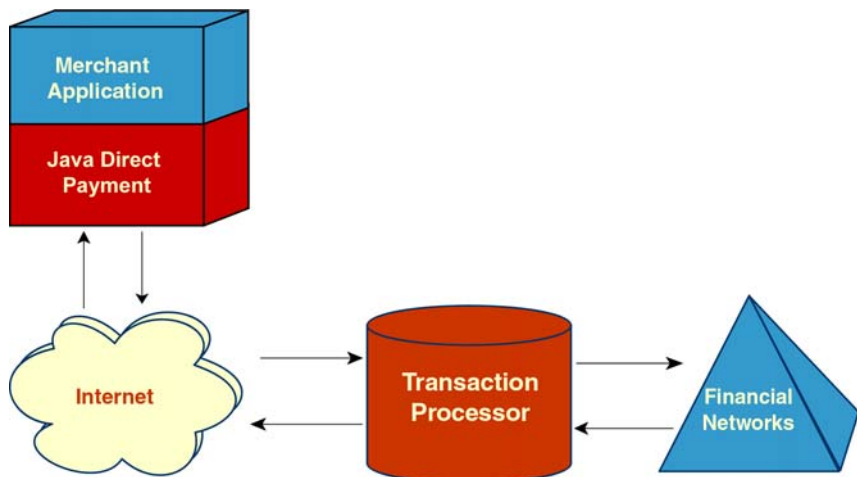


If you get a transaction unrecoverable error, it is probably because our server is not in your list of "trusted" sites. See *Installing Java Direct Payment on UNIX* on page 1-2 for more information.

Introduction to Java Direct Payment

What is Java 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 transaction processor via Java Direct Payment, which acts as a secure transport medium.



The merchant uses sockets through SSL (Secure Sockets Layer) to communicate securely with the payment server over the Internet.

Transactions are validated at both the Java Direct Payment and the payment service levels. For example, the transaction processor sends a response for both successful and failed transactions (e.g., if an invalid card type is supplied). Java Direct Payment 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 Java Direct Payment and our transaction 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 merchant application sends a variety of information to the payment service via Java Direct Payment.

The merchant application uses the process method to pass the required parameters (transaction parameters and the information from the configuration file – see *Configuring for Direct Payment* on page 2-8) to Java Direct Payment.

Java Direct Payment performs the following tasks:

- Opens an SSL connection, connects, and sends the request to the payment service using secure sockets.
- Waits for the response from the payment service and communicates it to the merchant application.
- Communicates the request status from the payment service to the merchant application.
- Communicates the error description, if applicable, to the merchant application.

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

Submitting transaction requests

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

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 2-8 for more information.
3. Initiate a request by following the steps in *Testing Java Direct Payment on NT and UNIX* on page 1-3.
4. Java Direct Payment reads the parameters set in the merchant's configuration file and builds the request from the merchant application.
5. Java Direct Payment then establishes a secure connection with us over the Internet.
6. When a secure connection is established, Java Direct Payment sends the request to us, and waits for the result.
7. We process the transaction request and send the results back to Java Direct Payment, which in turn relays this information to the merchant application. For more information on the responses to each transaction type, see the *Direct Payment Protocol Specification* manual.

Functions and features

Java Direct Payment implements five public methods in its visible interface, `RequestProcessing` class, which are used from the client application.

- The *boolean* `Init (String pathname)` method is used by the JSP application to send the pathname of the merchant's configuration file to Java Direct Payment. See *Configuring for Direct Payment* on page 2-8 for more information. This method returns *false* if an error occurs, and *true* otherwise.
- The *boolean* `Process (String Request)` method is used by the JSP application to send the request to us via Java Direct Payment. This method returns *false* if an error occurs, and *true* otherwise.
- The *String* `getErrorDescription()` method is used by the JSP application to request more information about the last error that occurred.
- The *int* `getStatus()` method is used by the JSP application to get the status of the processed request.

- The *String getResponse()* method is used by the JSP application to get the response returned. This method is used in the following cases:
 - *process()* method returned true and *getStatus()* method returned 0 (TRANSACTION_OK) – this means the transaction was successfully processed and the response will contain the server response.
 - *process()* method returned false and *getStatus()* method returned -8 (TRANSACTION_ERROR) – this means the server failed to process the transaction and the response will contain more information about the error.

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 Java Direct Payment 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, Java Direct Payment 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, Java Direct Payment automatically reinitiates communication.

- If the communication error occurs after a request has been sent but before a response has been received, Java Direct Payment 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.

Java Direct Payment level

If Java Direct Payment 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 completed). 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 Java Direct Payment and a payment service error message are returned.

Two important pieces of information are returned 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	Error with method call
-4	Error reading the configuration file
-5	Error occurred when using sockets
-6	Connection error
-7	Transaction not recoverable
-8	Server failed to process transaction
-9	Manual intervention required
-10	Transaction aborted
-11	Invalid URL
-12	Handshake error

2. An error code and matching error string, describing the error more exactly. See *Java Direct Payment 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 Java Direct Payment 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-6 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. For more information, see the *Direct Payment Protocol Specification* manual.

Additional features

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

- Secure communications (see *Secure communications* on page 2-7)
- Proxy server support (see *Proxy server support* on page 2-7)
- Active recovery in case of communication failure (see *Communication failure* on page 2-4)

Secure communications

SSL protocols provide secure data communications through data encryption and decryption. The communication between Java Direct Payment and our transaction processor is made with HTTP requests over the Internet, via TCP/IP, through a secure sockets by using SSL (Secure Sockets Layer).

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 2-8 for information on modifying your configuration file.

Configuring for Direct Payment

When you install Java Direct Payment, the configuration file is also installed. At this point you determine the pathname for your Java Direct Payment configuration file. In order to build and send transaction requests, Java Direct Payment must read the merchant's configuration file. Ensure 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 ID. See <i>Merchant registration</i> on page 1-1.
Cipher	Specified cipher. Default=SSL_RSA_WITH_RC4_128_MD5.
PaymentServerURL	The payment gateway URL. This is given in the configuration file at installation.

Field	Description
PaymentServerPort	Your payment server port number. Default=443.
Timeout	Time, in seconds, Java Direct Payment waits for a response from the payment service before terminating communication. Default=120. Range=60–180. 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.
Retries	Number of times Java Direct Payment tries to establish communication before returning an error message. Default=3. Range=2–5. 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.
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.
Keystore	Path to cacerts file, if server authentication is required.

Field	Description
LogLevel	The level of information that is logged by the Java Direct Payment. Three levels are possible: Information – Logs successful transactions. Error – Logs failed transactions. Trace – Logs all transactions. Default=Trace.
LogBasePath	Path to the log file.
LogFilename	Log file name.
LogMaxSize	Maximum size of the log file.

Error Return Codes

Error messages

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

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

Java Direct Payment error messages

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

Error Code	Error String	Action
ERROR(50)	PaymentServerURL is missing.	Add our payment processor's URL for to the configuration file.
ERROR(51)	PaymentServerPort is missing.	Add the payment server port to the configuration file.
ERROR(52)	merchantId is missing.	Add your merchant ID to the configuration file.
ERROR(53)	ProxyServer is missing.	Add the proxy server name to the configuration file.
ERROR(54)	ProxyPort is missing.	Add the proxy server port number to the configuration file.
ERROR(57)	Cipher is missing.	Add the cipher parameter to the configuration file.
ERROR(58)	merchantPwd is missing.	Add the merchant password to the configuration file.

Error Code	Error String	Action
ERROR(59)	account is missing.	Add the account number to the configuration file.
ERROR(60)	HTTPVersion is missing.	Add the HTTP version to the configuration file.
ERROR(61)	PaymentServerPort is invalid.	Add the server port number to the configuration file.
ERROR(62)	ProxyPort is invalid.	Check the ProxyPort value.
ERROR(63)	HTTPVersion is invalid.	Check the HTTPVersion value.
ERROR(64)	Configuration file not found.	Supply the exact path to your configuration file.
ERROR(65)	Error reading the configuration file.	Error occurred when reading the configuration file. Check the configuration file.
ERROR(66)	Access denied to the configuration file.	Check access rights to the configuration file.
ERROR(67)	Configuration file not readable.	Check the configuration file. It might be corrupted.
ERROR(68)	Call init() method first.	Method init() must be called first.
ERROR(69)	Port number should be a positive integer.	Verify that the port number supplied has the correct value.
ERROR(70)	Hostname should be non-null string.	Check the Hostname. It may be incorrect.
ERROR(71)	URL should be non-null string.	Check the URL. It may be invalid.
ERROR(72)	Request should be non-null string.	Check the request you submitted. It is empty.
ERROR(73)	Hostname you specified is unknown.	Check the hostname you specified.
ERROR(74)	Socket creation failed.	Retry later. If error persists, please contact technical support.

Error Code	Error String	Action
ERROR(75)	Cannot connect to server within Timeout value specified.	Increase the Timeout value parameter.
ERROR(76)	Socket error occurred.	Retry later. If error persists, please contact technical support.
ERROR(77)	Communication error occurred. Cannot send the request.	Retry later. If error persists, please contact technical support.
ERROR(78)	Communication error occurred. Cannot read the response for the Timeout value specified in your configuration file.	Perform a Transaction Lookup/Query to find out the transaction status.
ERROR(79)	Communication error occurred.	Check the URL. If error persists, please contact technical support.
ERROR(80)	Communication error occurred. MerchantTxn is required for Transaction Lookup recovery. Manual intervention is required to recover.	Please contact technical support.
ERROR(81)	Transaction aborted/rejected by server. Request could not be performed.	Retry later. If error persists, please contact technical support.
ERROR(82)	Server encountered internal error. Transaction unrecoverable.	Check the URL. If the URL is correct, perform a Transaction Lookup/Query to find out the transaction status.
ERROR(83)	Manual intervention required to process your request.	Contact technical support.

Error Code	Error String	Action
ERROR(84)	The requested resource must be accessed through the proxy server.	Use the proxy server. If error persists, please contact technical support.
ERROR(85)	The request could not be processed by the server due to invalid syntax in HTTP header.	Retry. If error persists, please contact technical support.
ERROR(87)	The server refuses to fulfill the request.	Retry later. If error persists, please contact technical support.
ERROR(88)	Invalid URL.	Check the URL.
ERROR(89)	Method in HTTP header not allowed.	Retry. If error persists, please contact technical support.
ERROR(90)	HTTP header invalid.	Please verify your URL and/or your HTTP header parameters.
ERROR(92)	The server timed out waiting for the request.	Retry. If error persists, please contact technical support.
ERROR(93)	The request could not be completed.	Retry. If error persists, please contact technical support.
ERROR(94)	Requested resource not available.	Retry. If error persists, please contact technical support.
ERROR(95)	The server refuses to accept the request without a defined content length.	Retry. If error persists, please contact technical support.
ERROR(96)	HTTP header invalid.	Please verify your URL and/or your HTTP header parameters.
ERROR(97)	Request too large.	Check the request.
ERROR(98)	URL too long.	Check the URL.
ERROR(99)	Request format unsupported.	Retry. If error persists, please contact technical support.

Error Code	Error String	Action
ERROR(100)	Server encountered internal error that prevented it from fulfilling the request.	Retry. If error persists, please contact technical support.
ERROR(101)	Server does not support functionality to fulfill the request.	Retry. If error persists, please contact technical support.
ERROR(102)	Invalid HTTP response from server.	Retry. If error persists, please contact technical support.
ERROR(103)	Service temporarily overloaded.	Retry later. If error persists, please contact technical support.
ERROR(104)	Gateway timed out.	Retry. If error persists, please contact technical support.
ERROR(105)	HTTP version not supported by server.	Check the HTTPVersion value.
ERROR(106)	HTTP header is invalid.	Retry. If error persists, please contact technical support.
ERROR(107)	The Cipher you specified is not supported.	Try other ciphers. If error persists, please contact technical support.
ERROR(108)	The server failed to process your transaction.	Check the server response for more information.
ERROR(109)	Timeout parameter is invalid.	Check the Timeout value.
ERROR(110)	Retries parameter is invalid.	Check the Retries value.
ERROR(111)	merchantId is invalid.	Check the merchantId value.
ERROR(112)	merchantPwd is invalid.	Check the merchantPwd value.
ERROR(113)	account is invalid.	Check the account value.
ERROR(114)	The Cipher you specified is refused by server.	Try other ciphers. If error persists, please contact technical support.

Error Code	Error String	Action
ERROR(115)	Connection refused by server. Check your PaymentServerPort.	Check the PaymentServerPort parameter. If error persists, please contact technical support.

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

Suberror Code	Suberror String	Action Code
1028	Invalid data	AR
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

Suberror Code	Suberror String	Action Code
1077	Invalid originator	AR
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.

A

account 2-8
action codes A-6

B

boolean Init 2-3
boolean Process 2-3

C

certificate authority, using 2-2
Cipher 2-8
codes
 action A-6
 error A-7
communication failure 2-4
configuration file
 account 2-8
 Cipher 2-8
 HTTPVersion 2-9
 LogBasePath 2-10
 LogFilename 2-10
 LogLevel 2-10
 LogMaxSize 2-10
 merchantId 2-8
 merchantPwd 2-8
 PaymentServerPort 2-9
 PaymentServerURL 2-8
 ProxyPort 2-9
 ProxyServer 2-9
 Retries 2-9
 Timeout 2-9
configuring
 Java Direct Payment 2-8

E

error messages A-1
 action codes A-6
 error codes and strings A-7
 from payment services A-6
 Java Direct Payment A-1
errors
 Java Direct Payment 2-5
 no response 2-7
 payment service level 2-6
 status level numbers 2-6

F

failed transactions 2-4
failure, communication 2-4
features
 secure communications 2-7
functions and features 2-3

H

HTTPVersion 2-9

I

installing
 Java Direct Payment on UNIX 1-2
 Java Direct Payment on Windows
 NT 1-2
int getStatus() 2-3

J

Java Direct Payment
 configuring 2-8
 error messages A-1
 errors 2-5
 installing on UNIX 1-2
 installing on Windows NT 1-2

- overview 2-1
- public methods 2-3
- testing 1-3

L

- LogBasePath 2-10
- LogFilename 2-10
- LogLevel 2-10
- LogMaxSize 2-10

M

- merchant registration 1-1
- merchant transactions 2-2
- merchantId 2-8
- merchantPwd 2-8

N

- no response 2-7

P

- parameters
 - error messages, from payment services A-6
- payment service error messages A-6
- payment service level errors 2-6
- PaymentServerPort 2-9
- PaymentServerURL 2-8
- proxy server support 2-7
- ProxyPort 2-9
- ProxyServer 2-9

R

- reducing traffic
 - proxy servers 2-7
- registering for Java Direct Payment 1-1
- RequestProcessing class 2-3
- requirements
 - software 1-1

- Retries 2-9

S

- secure communications 2-7
- security 2-2
- software requirements 1-1
- status level numbers 2-6
- String getErrorDescription() 2-3
- String getResponse() 2-4
- submitting transaction requests 2-2

T

- testing Java Direct Payment 1-3
- Timeout 2-9
- transaction requests, submitting 2-2
- transactions, failed 2-4
- transactions, merchant 2-2

U

- UNIX
 - installing Java Direct Payment 1-2

W

- Windows NT
 - installing Java Direct Payment 1-2