:: LinkPoint® Select API for PHP Version 3.0 INTEGRATION GUIDE

LinkPoint INTERNATIONAL®

Introduction to the API

What is the API?

The API is an Application Programming Interface: a collection of functions for processing payment transactions over the Internet in a highly secure manner.

Advantages of the API

Conducting credit card authorizations over the Internet in a secure environment is a very complex process. The API offers merchants a simpler payment solution.

Using the API is your key to connecting an online store electronically to the Payment Gateway with secure sockets layer (SSL) protection. All of your interactions with credit card processors and financial institutions are resolved and managed through the Payment Gateway. The payment processing communication highway is fast-paced with today’s equipment and technology becoming obsolete within days or months. Fortunately, the API handles all of the details for merchants in one package.

Smaller merchants can quickly and easily begin payment processing through the API. By contacting a participating sales agent, a merchant is guided through the online or paper application process. Upon approval, the same merchant can be running transactions within 24 hours.

Internet service providers (ISPs), commerce service providers (CSPs) and self-hosting merchants using the API do not have to invest in the logistical strategies, personnel, hardware, software or telecommunication expenses required for hosting secure data. Another advantage for ISP and CSP customers is obtaining access to the automated system for setting up and maintaining merchant accounts.

SECURE SOCKETS LAYER (SSL) ENCRYPTION

The API provides data encryption, server authentication, message integrity and optional client authentication for a TCP/IP connection by using version 3.0, Secure Socket Layers (SSL) protocol. SSL is a protocol developed by Netscape® to provide secure transmission of private information that is sent over the Internet. This protocol uses public and private key pairs to encrypt data. The public and private keys are dissimilar and each pair is unique; therefore, the keys you possess can verify your identity. The public key is distributed to the merchant, ISP or CSP in the form of a digital certificate, which contains information that can verify the key holder identity and the key validity. The private key is kept confidential and remains on the Gateway server. If data is encrypted with the private key, only the public key can decrypt it. If data is encrypted with the public key, only the private key can decrypt it. This process prevents the data from being compromised while in transit.

DEVELOP IN YOUR OWN LANGUAGE

You have your choice of several different web development languages for employing the API. The languages supported are:

- C++

- C#
- COM Object technology (allows you to develop in Visual Basic, VBScript, or ASP)
- PHP
- PERL®
- Java™
- .NET

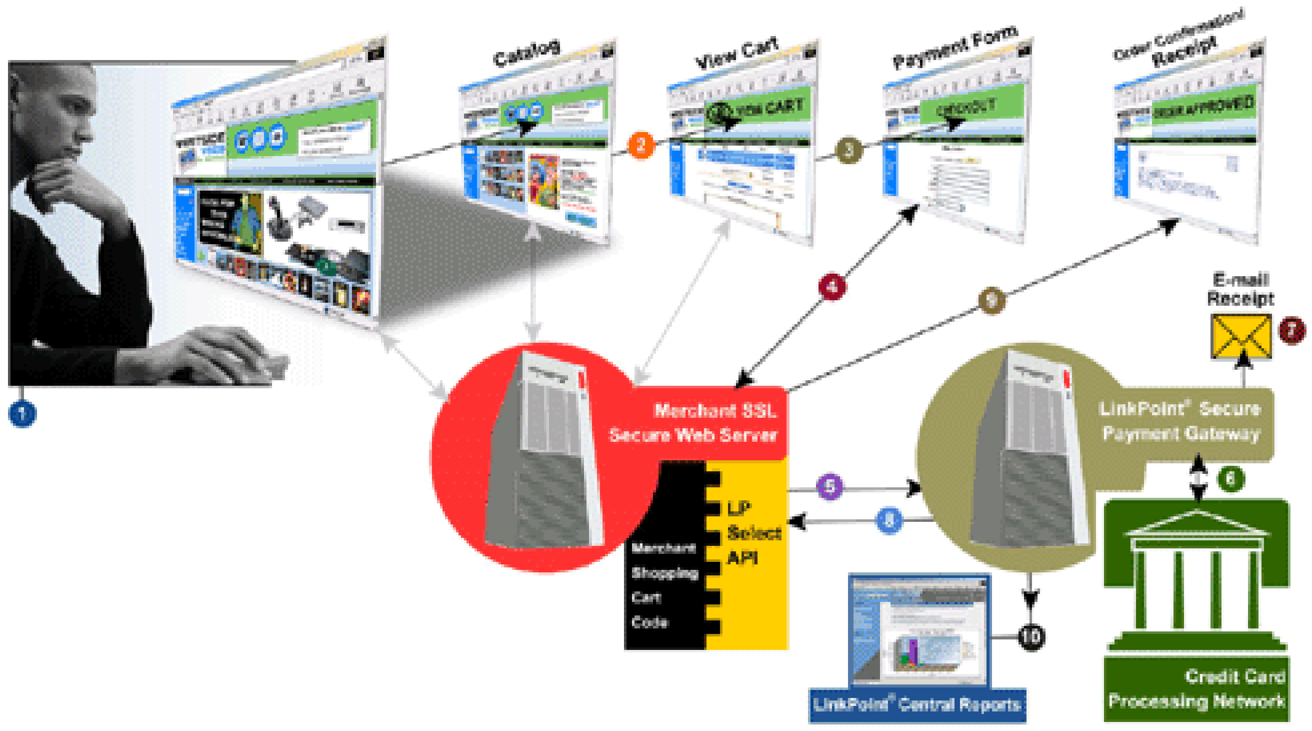
How the API Works

A merchant uses the API software modules to build the payment solution that fits the merchant's unique needs. Modules offered are:

- The payment module
- The shipping calculator module
- The tax calculator module
- The ESD module
- The VirtualCheck module
- The Recurring payments module

Merchants can pick and choose modules as needed. For credit card payments, all of the interactions with credit card processors and financial institutions are resolved and managed through the API payment module. For electronic check payments, all of the interactions with financial institutions are resolved and managed through the API VirtualCheck module.

Once a merchant has integrated the API payment module into a Web site, a customer can purchase items on the merchant's Web site and all payment details are processed automatically. Merchants can then review transaction activity by visiting LinkPoint Central. Let's follow a simple transaction as it travels from the customer's order to approval.


Catalog
View Cart
Payment Form
Order Confirmation/ Receipt
VIEW CART
CHECKOUT
ORDER APPROVED
E-mail Receipt
Merchant SSL Secure Web Server
LinkPoint® Secure Payment Gateway
LP Select API
Merchant Shopping Cart Code
LinkPoint® Central Reports
Credit Card Processing Network
1
2
3
4
5
6
7
8
9
10

1. A customer selects items for purchase from a merchant's online store over the Internet.
2. The customer clicks on the **View Cart** button and finalizes the order.
3. The customer clicks on the **Check Out** button and proceeds to the merchant's payment page.
4. The ordering information is received by the merchant's CSP or by the merchant's own server (self-hosting).

5. The merchant's customized software receives the information, uses the tax and shipping calculators as needed, calculates the order total and passes relevant order and payment information to the API payment module. The API payment module funnels the order data through the secure sockets layer (SSL) pipeline to the LinkPoint Secure Payment Gateway.
6. The LinkPoint Secure Payment Gateway calls and transmits the data through a dedicated, secure frame relay system to the banking network. (Once the transaction is approved and settled, funds are withdrawn from the credit card-issuing bank and deposited into the merchant bank.)
7. A confirmation e-mail is sent to the merchant and the purchaser when the transaction is completed.
8. The merchant's Web server receives the transaction response information. This generally happens within six seconds.
9. The merchant's Web server displays the transaction results to the customer.
10. The merchant uses LinkPoint Central reports to review transactions and mark items as shipped (e.g., ready for settlement).

What is Required to Integrate the API?

What you need to integrate the API depends on which language you choose to develop in.

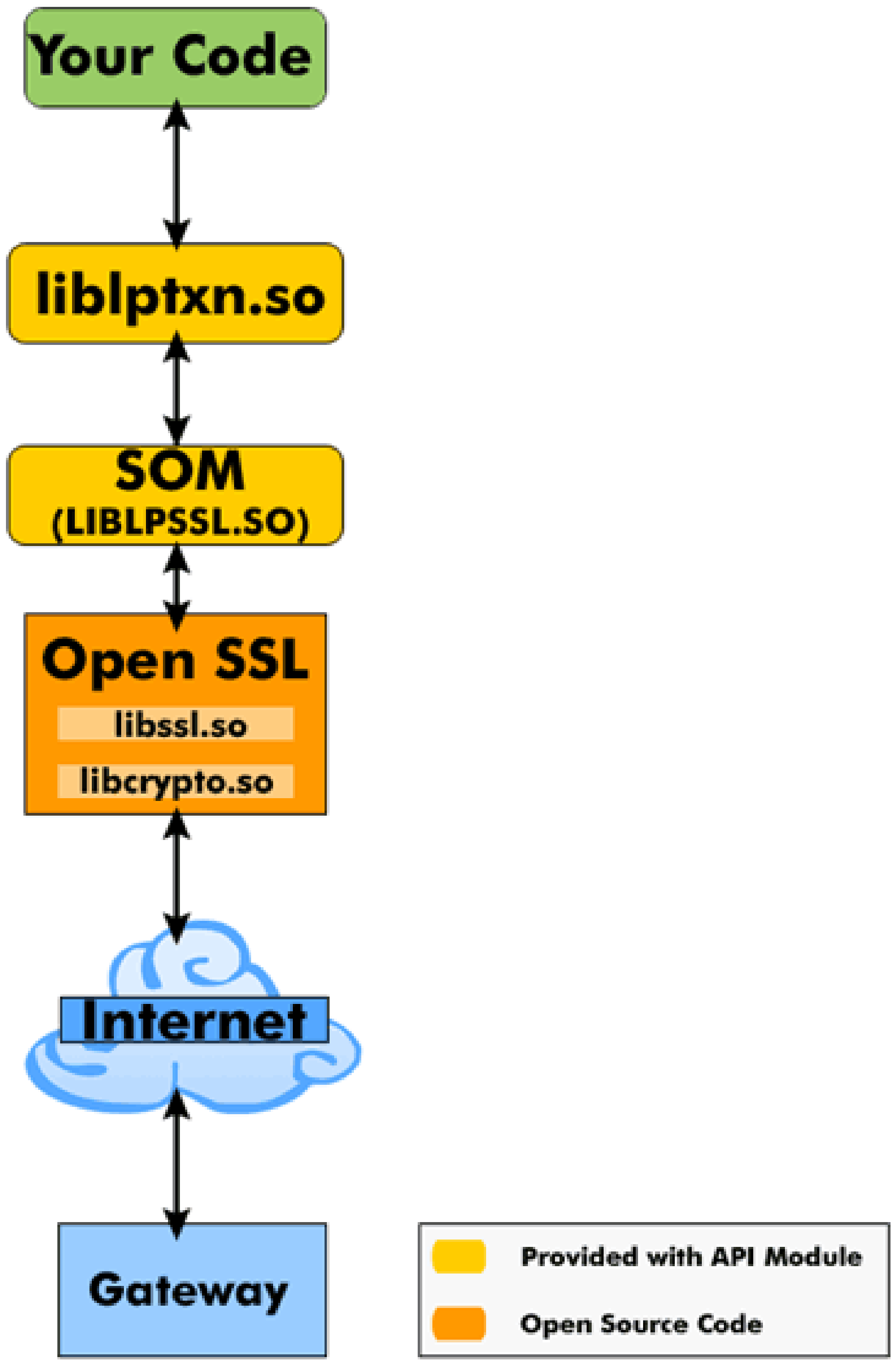
If you use PERL or PHP, you have some options. See *Prerequisites for PERL* and/or *Prerequisites for PHP* for further information.

If you are using any of the languages listed below, you will need to ensure that the OpenSSL libraries *libssl.so* and *libcrypto.so* (if using HP-UX, the file extensions would be .sl) are installed as shared objects on your Web server.

- C or C^{++}
- C$^{\#}$
- COM Object technology
- .NET technology
- Java^{TM}

See *What is OpenSSL?* for further information on OpenSSL and requirements for installing it.

The diagram below shows the flow between the different software modules.


Your Code
liblptxn.so
SOM
(LIBLPSSL.SO)
Open SSL
libssl.so
libcrypto.so
Internet
Gateway
Provided with API Module
Open Source Code

Prerequisites for Java

If you choose Java as your development language, you will first need to ensure that the OpenSSL libraries *libssl.so* and *libcrypto.so* (if using HP-UX, the file extensions would be .sl) are installed as shared objects on your Web server. See *What is OpenSSL?* for further information on OpenSSL and requirements for installing it.

You will also need to convert your PEM file to pkcs12 format. To do that run the following from a command prompt:

```
openssl pkcs12 -export -in YOURPEM.pem -inkey YOURPEM.pem -out YOURPEM.p12 -passout pass:YOURPASS -name "YOURNAME"
```

Where:

- YOURPEM - the name of your PEM file
- YOURPASS - any password
- YOURNAME - any arbitrary name

For example:

```
openssl pkcs12 -export -in 1234567.pem -inkey 1234567.pem -out 1234567.p12 -passout pass:987654321 -name "LinkPoint"
```

The output *.p12 file and the password are used to pass as parameters for JLinkPointTransaction object
If you have a problem converting your PEM file, please contact support.

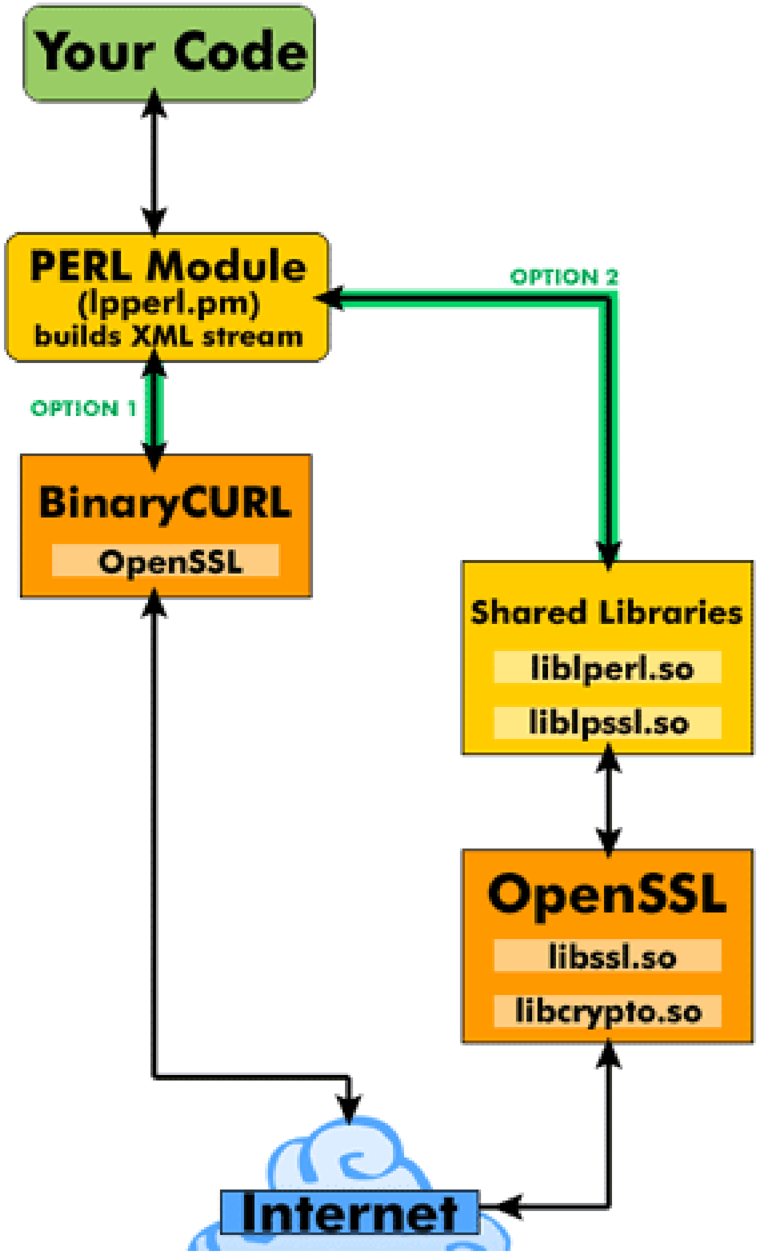
Prerequisites for PERL

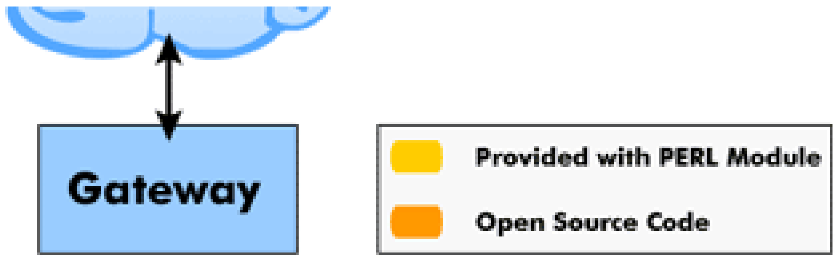
If you choose PERL as your development language, you will first need to know:

- Whether or not you have cURL and OpenSSL available on your Web server
- Whether you have root access and install privileges on your Web server

You have some choices for connecting to and sending transactions securely to the Gateway:

1. Use cURL (with OpenSSL) and DO NOT use the Shared Libraries provided with the PERL module. cURL must be built with SSL support enabled. (This is the default installation option for most cURL installations.)
2. Use the provided Shared Libraries. This option requires the two shared libraries lpbspssl.so and liblphp.so to be installed correctly (typically in /usr/lib) and also that you have OpenSSL built with 'shared object' support on your Web server. This option is recommended for those users that have admin privileges on their Web server for correct installation of the shared libraries.





Related Topics

- What is cURL? How do I get cURL?
- What is OpenSSL? How do I get OpenSSL?

Prerequisites for PHP

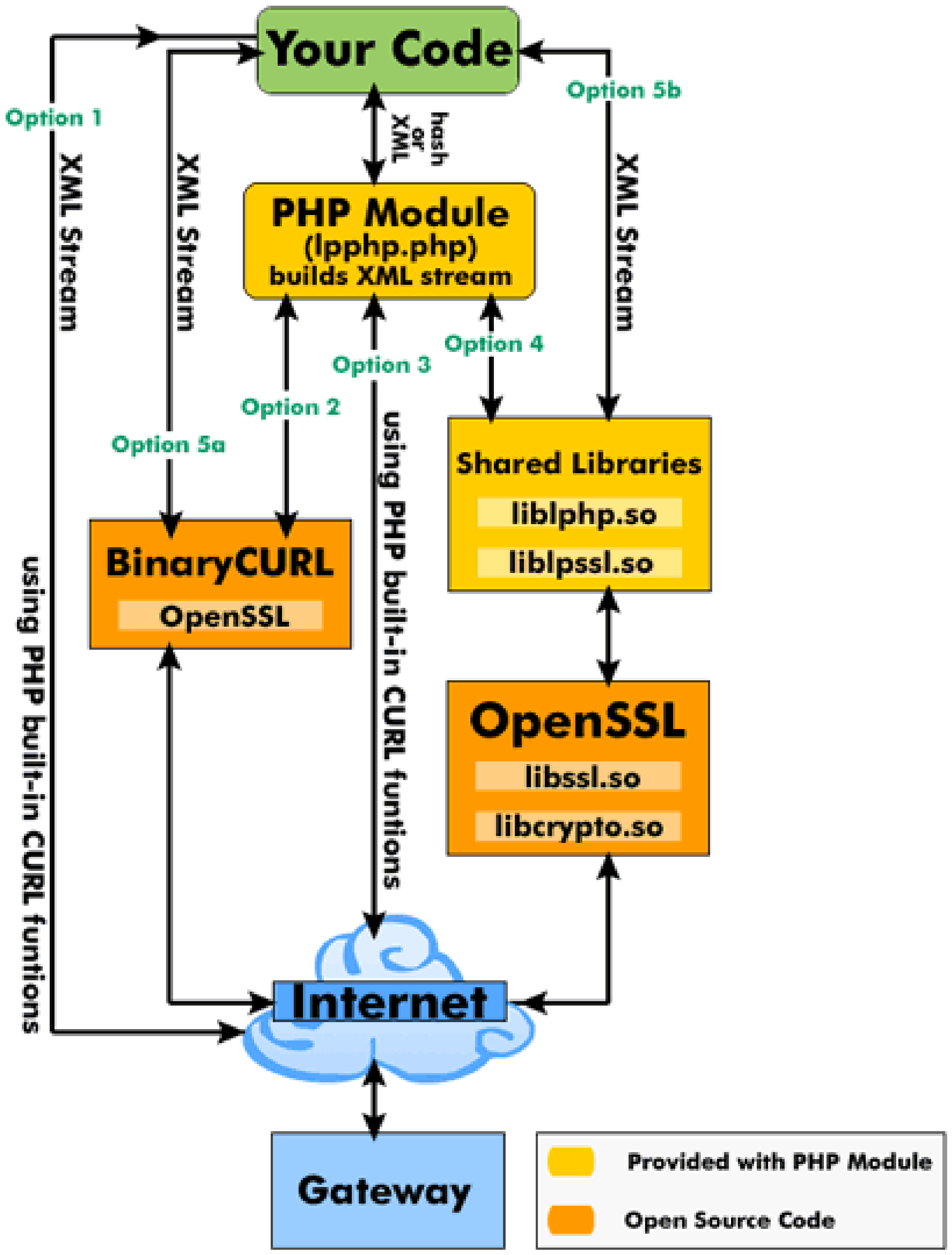
If you choose PHP as your development language, what you need to get started depends on:

- Whether or not you have cURL and OpenSSL available on your Web server
- Whether you have root access and install privileges on your Web server
- Whether you have a version of PHP with cURL functionality built-in

You have some choices for connecting to and sending transactions securely to the Gateway:

1. If you are using PHP with built-in cURL functionality, you have the option to build the XML string yourself and send it directly to the Gateway. This option requires a PHP version greater than 4.0.2 and that cURL support has been compiled into PHP. (This is the default setup for Red Hat® Linux® server installations.) This options does not use the PHP module to build the XML string, so users must manage their own XML output. This option is demonstrated in the PHP sample program PASS_XML_DIRECT.php.
2. Use cURL (with OpenSSL) along with the **lphp.php** module. Use this option with older versions of PHP or where PHP has not been compiled to support cURL. Accepts either hash-style input or XML.
3. Use the PHP built-in cURL methods* along with the **lphp.php** module. This is the easiest option to use. It requires PHP version greater than 4.0.2 and that cURL support is compiled into PHP. (This is the default setup for Red Hat Linux server installations.) Accepts either hash-style input or XML.
4. Use the lphp.php module along with the provided Shared Libraries and OpenSSL. This option requires the provided Shared Libraries **libspssl.so** and **liblphp.so** to be correctly installed (typically in /usr/lib) and that you have OpenSSL libraries built with 'shared object' support on your Web server. This option uses the **lphp.php** module to accept either hash-style input or XML. Recommended for users with admin privileges on their server to correctly install the shared libraries.
5. You may also build the XML stream yourself and send it:
   . via cURL, or
   b. via the Shared Libraries provided with the PHP module. This option requires the provided Shared Libraries **libspssl.so** and **liblphp.so** to be correctly installed (typically in /usr/lib) and that you have OpenSSL libraries built with 'shared object' support on your Web server. This option is demonstrated in the PHP sample program PASS_XML_LIB.php.

* To determine whether PHP cURL support is installed, enter the command: ***php -m***. PHP will list all compiled modules.



Making PHP and cURL work in Windows®

If you choose to use cURL on a Windows® platform, follow these steps to complete your PHP installation.

1. Open the file ***php.ini*** located in the ***winnt*** folder in a text editor such as Notepad. Remove the semicolon from this line:

   extension=php_cURL.dll

2. Copy ***php_cURL.dll*** into ***c:\php***\.
3. Verify that cURL works in PHP by doing the following:

. Open a text editor such as notepad.

b. Copy and paste this script into the file:

```
// script
phpinfo()
?>
```

c. Save the file in your local web directory as ***phpinfo.php***.

d. Using your web browser, type in the following URL: http://localhost/phpinfo.php.

e. Look for the word ***cURL*** on the page displayed.

Related Topics

- What is cURL? How do I get cURL?
- What is OpenSSL? How do I get OpenSSL?

What is OpenSSL?

OpenSSL is well-known, widely respected open source software that provides encryption/decryption, SSL communications, and digital signature capabilities. OpenSSL is based on the SSLeay library developed by Eric A. Young and Tim J. Hudson. The OpenSSL toolkit is licensed under a Apache-style license, which basically means that you are free to use it for commercial and non-commercial purposes. If you make changes to the source code, however, you are obligated to make your changes public. See the OpenSSL web site for more information at http://www.openssl.org/. There are FAQs available there that answer many of the OpenSSL usage questions.

HOW TO GET OPENSSL

We recommend the following course of action:

1. First, check to see if the secure Web server you are using already has OpenSSL installed on it. If you are using an ISP, ask your ISP whether this software is installed on your Web server. If it is installed, find out what version is installed.
2. If OpenSSL is installed, find out whether it is installed as SHARED libraries. If it is not, reinstall OpenSSL and make sure you install them as shared libraries.
3. Make sure you are using the latest version. OpenSSL is frequently updated. For best results, you should ensure you always have the latest version of OpenSSL on your Web server. If the version installed on your server is NOT the latest version, upgrade the software (or ask your ISP to upgrade to the latest version).
4. If OpenSSL is not already installed on your Web server, your next best move is to download the source code from http://www.openssl.org/source. Instructions for installing the source code are located at: http://www.linkpoint.com/support/. Click on **FAQs**, then on **How to Build and Install OpenSSL from Source** under **Additional Materials**.
5. OpenSSL must be installed to use the API. You will need root permissions on your Web server to install OpenSSL. If you do not have root permissions on your system (and if the latest version of OpenSSL is not already installed), ask your ISP or system administrator to install (or upgrade) it for you.

Top

Do I need OpenSSL?

You will need OpenSSL before you can use the API.

Top

OpenSSL is huge! Do I really need all those files?

No, you don't need all of them to use the API. Depending on which software language you choose and the operating system on your Web server, you will need two to four files. The required files for each operating system are listed in the ***OpenSSL Download and Installation Recommendations*** located in the Support/FAQs section on linkpoint.com/support. The binary versions of these files can be downloaded from that page. If you choose to use source code, see ***How to Build and Install OpenSSL from Source*** in the same location.

Top

Do I need the OpenSSL executable file?

No, you do not need to install the executable. The LinkPoint software does not execute OpenSSL, rather it calls functions defined within the OpenSSL libraries (or dlls).

Top

Where do I get OpenSSL again?

http://www.linkpoint.com/support/index_openssl.html for the binaries

http://www.openssl.org for the source code.

Top

What is cURL?

cURL (or simply just 'cURL') is a library for getting or sending files using URL syntax. The name is a play on 'Client for URLs', originally with URL spelled in uppercase to make it obvious that it deals with URLs.

cURL is open source code that supports a range of common Internet protocols, currently including: HTTP, HTTPS, FTP, FTPS, GOPHER, LDAP, DICT, TELNET and FILE. cURL exists, compiles, builds and runs under a wide range of operating systems, including all modern Unixes (and a bunch of older ones too), Windows, Amiga, BeOS, OS/2, OS X, QNX, etc.

HOW TO GET cURL

cURL is already available on some Web servers, so we recommend a similar course of action that you used with OpenSSL:

1. First, check to see if your Web server has cURL installed. If you are using Red Hat Linux, cURL is most likely included.
2. If it is installed, check the version number.
3. If it is not a compatible version, download the appropriate cURL binaries or source code and install it.
4. If you choose to build it from source, make sure you install OpenSSL first, then cURL. Also make sure you enable SSL in the installation.

Please note: If you are building the files from source, make sure you enable SSL in the installation. You can download cURL source code or binaries from the cURL Web site at http://cURL.haxx.se/download.html. Installation and usage instructions are also available on the cURL Web site.

E-mail Settings

Once an order is processed successfully, the system automatically sends both the merchant and the customer an e-mail receipt, which will appear to come directly from your online store. This feature can be disabled.

If the customer has ordered a soft good, the receipt includes download information. If the customer has ordered a hard good, when the product has been marked “shipped,” the system automatically sends the customer an additional message stating that the product has been shipped.

The e-mail receipt seen by the customer is created based on the items ordered and the elements of your merchant configuration file (for example, the name of the store).

CONFIGURING E-MAIL RECEIPTS

Disabling E-Mail Receipts

To disable the customer and/or merchant e-mail receipts function, notify Support by e-mail with your request.

Customizing the Receipt

You have the option to include a block of text that is appended to the end of each customer e-mail receipt (e.g., appending a message like “Test Store, Inc . – One Stop Shopping for Logo Merchandise!”). If you want to use this feature, save the text message you would like appended in a file named contact.txt, using any text editor. E-mail the file to Support with your Store Number (or Storename).

SAMPLE E-MAIL RECEIPT

This is an example of a customer receipt e-mail.

> From: Joe Merchant
> Subject: Your Online Order
> Company: Test Store, Inc.
> Reference Number: 266.266.266.266-894386536-16
>
>> Item 1:
>> Part #: 1
>> Quantity: 1
>> Description: Logo T-Shirt
>> Price: $20.00
>> Color: Red
>> Size: Extra Large
>> Item Total: $20.00
>>
>> Item 2:
>> Part #: 2

Quantity: 2
Description: Bubble Pen
Price: $1.00
Item Total: $2.00

Subtotal: $22.00
Tax: $1.82
Shipping: $5.00
Total: $28.82

Test Store, Inc.
1234 Main Street
Elm, TX 77777
Phone: (512)-555-1212

Processing Transactions Manually

If you need to process transactions manually (that is, by keying in the information), you can use the virtual point-of-sale terminal available in LinkPoint® Central (LPC).

To use it, log into LinkPoint Central at https://www.linkpointcentral.com, click on **POS**, then key in the information for the transaction.

You can post-authorize several transactions simultaneously by using LPC Reports:

1. Click on **Reports** in the **Main Menu Bar**, then on **View Orders**.
2. Choose the criteria for the transactions you wish to post-authorize, then click the **Submit Query** button to bring up the report.
3. Once you are viewing the report, select orders by clicking the associated checkbox in the left-most column of the table.
4. Click on the **Work with Selected Orders** button at the bottom of the page.
5. Select the option to **Confirm Shipment of Merchandise**, then click on the **Submit Query** button.
6. A confirmation page will appear.

Similarly, you can void transactions that have not yet been processed by going to the **Credit Card Batches** or **Check Batches** report, then clicking on Current Batch. Follow a similar procedure to void any transactions you need to void.

For more assistance on using LinkPoint Central, click on **Help** within LinkPoint Central. This will bring up full instructions for using LPC.

How Transactions Are Processed

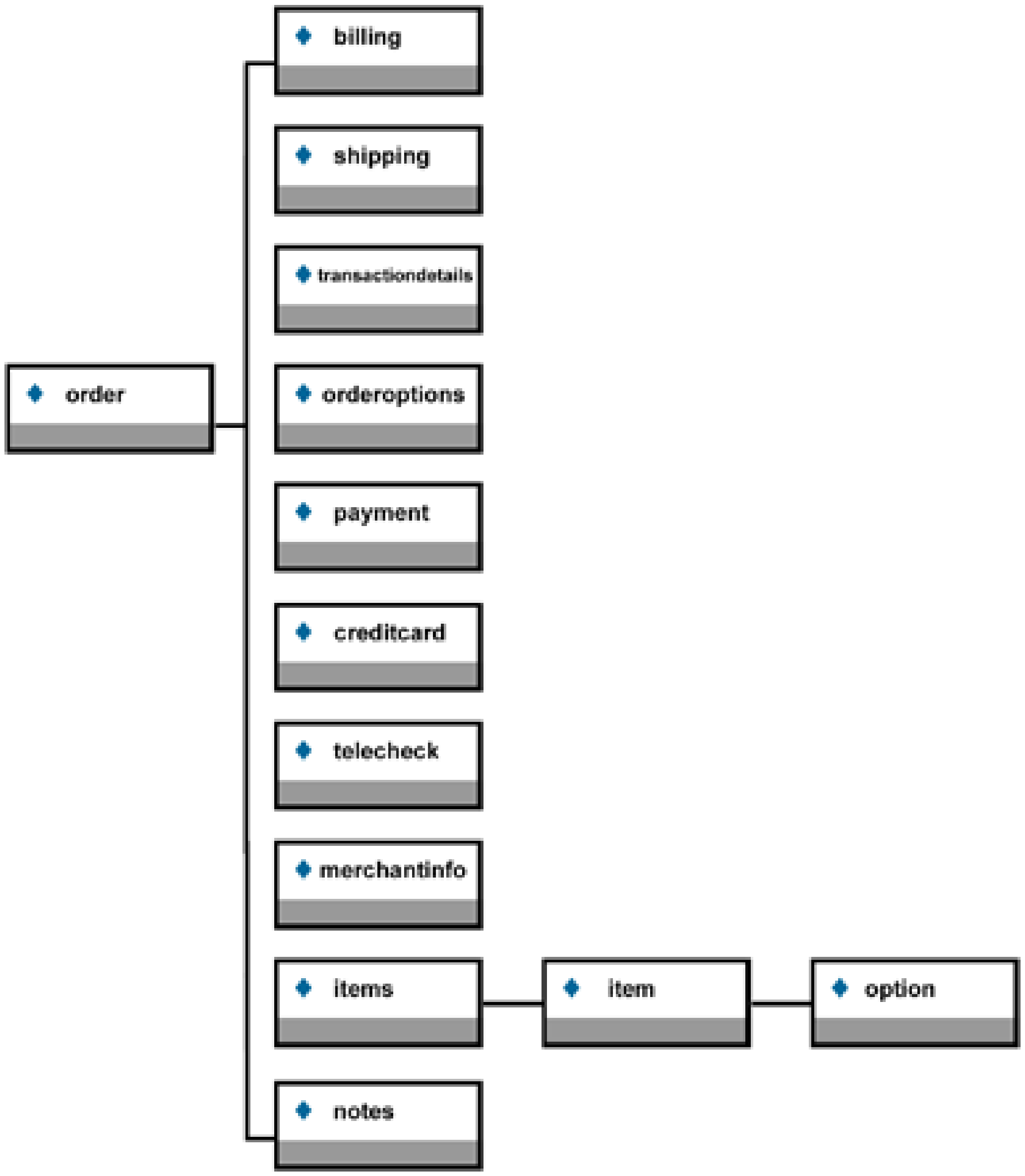
Regardless of which programming language the merchant uses to integrate with the Gateway, the Gateway uses XML to describe each transaction. The XML is often generated by the API software, but in many cases, the merchant has the option to generate the XML stream and send it to the Gateway in XML format.

The extensibility of XML helps to ensure that future modifications to the gateway transaction server have minimal impact on the merchant's implementation. For example, in the future, if a new payment type were to be offered, the ***creditcard*** node in the XML could be replaced by the tag structure of the new payment type with little or no modification to the surrounding application. For instance, to support checks, a ***check*** node would simply replace the ***creditcard*** node. Likewise, the ***cardnumber*** tag would be replaced by a ***micr*** tag. The key is that if the merchant does not wish to support the ***check*** capability, no changes would be necessary to their implementation.

To process a transaction, the client application makes an SSL connection to the gateway server, sends a transaction in the form of an XML request and waits for the response. The response will be in the form <response>, and it contains a set of response fields that may vary depending on the type of transaction requested and whether the transaction is successful.

The XML request includes various XML entities to provide the server with inputs to the function to be performed. Some of these entities are required for any transaction, while others are optional and are specific for particular transaction types. See Required Entities and the Minimum Required Fields section for further information on which entities and fields are required for each transaction type.

The diagram below shows the entities that can make up a typical order.


order
billing
shipping
transactiondetails
orderoptions
payment
creditcard
telecheck
merchantinfo
items
item
option
notes

Rules for Processing VirtualCheck Transactions

VirtualCheck is a means of processing automated clearing house (ACH) transactions via the Internet. ACH transactions debit a customer's account and transfers the funds to a merchant's account. There are specific rules that apply when using the Internet to initiate a debit to a consumer's bank account that the merchant must follow.

In all cases, the merchant must provide the customer with a receipt detailing the transaction. If the merchant has not disabled the API e-mail receipts, the customer will automatically receive an e-mail receipt that contains all required information. If the merchant chooses not to use this receipt, the merchant must take steps to provide a receipt (printed, faxed, or e-mailed) to the customer prior to the check settlement date.

The rules for authorization differ depending on whether the transaction is:

- e-commerce,
- retail,
- mail order, or
- telephone order.

See below for important obligations of the merchant for each type of transaction.

Once the merchant has obtained written authorization from the consumer to debit the account and submits the transaction for processing, an immediate response will result. If the check is **immediately declined** because of the consumer's credit, an error message containing the following text will be returned in the field **r_error**.

> *We are sorry that we cannot accept your check at this time. Our decision is based, in whole or in part, on information provided to us by TeleCheck. We encourage you to call TeleCheck at 1-877-678-5898 or write TeleCheck Customer Care at P.O. Box 4513, Houston, TX 77210-4513. Please provide TeleCheck your driver's license number and the state where it was issued, and the complete banking numbers printed on the bottom of your check. Under the Fair Credit Reporting Act, you have the right to a free copy of your information held in TeleCheck's files within 60 days from today. You may also dispute the accuracy or completeness of any information in TeleCheck's consumer report. TeleCheck did not make the adverse decision to not accept your check and is unable to explain why this decision was made.*

If this error message appears, the **merchant is responsible for providing the text message in its entirety** to the consumer. The merchant should either seek another form of payment or choose not to accept the order from the customer.

E-commerce Transactions

When processing e-commerce check transactions, the merchant must obtain electronic authorization from the consumer to debit the account. Before the merchant Web site or system submits the payment for processing, it must present the customer with an authorization form where consent language is displayed, along with an **Authorize** button and a **Cancel** button. The **Authorize** button continues with the transaction--the **Cancel** button may present the customer with other payment options, but it cannot continue to process the check transaction.

The authorization form must include:

1. The merchant's DBA name
2. The amount of the transaction
3. The date
4. Consent language; that is, text that specifically states that the customer is authorizing the merchant to debit the customer's bank account.
5. An **Authorize** button and a **Cancel** button

Sample Authorization Form (E-Commerce)

Date

By clicking on the **Authorize** button below, I authorize ***[merchant name]*** to initiate an electronic debit to my bank account in the amount of ***[order total]***.

This authorization is to remain in full force and effect unless I provide written notification to ***[merchant name]*** within an appropriate timeframe as to allow ***[merchant name]*** to act on it.

If you wish to cancel this transaction, click the **Cancel** button.

Retail and Mail Order Transactions

When the transaction is either a retail (face to face) order or a mail order transaction, the merchant is required to obtain written authorization from the consumer to debit the account. The merchant must save the written records for four years from the date of the authorization in case of a later dispute.

The authorization must include:

1. The merchant's DBA name
2. The amount of the transaction
3. The date
4. Consent language
5. The consumer's name
6. A line for the consumer to sign

In the case of a mail order transaction, the merchant may fax or e-mail the authorization form to the customer, if needed.

Sample Authorization Form (Retail or Mail Order)

Date

I authorize ***[merchant name]*** to initiate an electronic debit to my bank account in the amount of ***[order total]***.

This authorization is to remain in full force and effect unless I provide written notification to ***[merchant name]*** within an appropriate timeframe as to allow ***[merchant name]*** to act on it.

____
Print name

____
Signature

Telephone Order Transactions

Requirements for a telephone order include the same information and consent language-the difference is that the merchant has the option to tape record the customer's authorization and retain the recording as proof of authorization for a period of four years.

If the merchant is not recording the conversation, the merchant must send written notification to the consumer confirming the verbal authorization prior to settlement. The notification must include the following information. e-mail receipts fulfill the requirement to send written notification, but if the merchant chooses not to use them, please note that the notification must include the following information. E-mail or fax notification will suffice. If the authorization is VERBAL, the merchant must use similar language (in the conversation with the customer) to the following.

Sample VERBAL Authorization (Telephone Order)

By answering YES, you are authorizing ***[merchant name]*** to initiate an electronic debit to your bank account in the amount of ***[order total]*** on or after ***[today's date]***.

This authorization is to remain in full force and effect unless you provide written notification to ***[merchant name]*** within an appropriate timeframe as to allow ***[merchant name]*** to act on it.

Do you authorize this transaction? (Please answer Yes or No.)

Sample Authorization Form (Telephone Order)

Date

Dear ***[full customer name]***,

This notice is to confirm your verbal authorization given on ***[date of verbal authorization]*** for ***[merchant name]*** to initiate an electronic debit to your bank account in the amount of ***[order total]***.

This authorization will remain in full force and effect unless you provide written notification to ***[merchant name]*** within an appropriate timeframe as to allow ***[merchant name]*** to act on it.

Sincerely,
[Merchant DBA]

Credit Card Transaction Types

The transaction type you use for a credit card transaction depends on whether you want to confirm that the customer has available funds (**Authorize Only**), to capture funds on a customer’s card (**Sale**), or to charge the customer’s card for an amount previously authorized (**Ticket Only** or **Forced Ticket**). Sale transactions can be made **Recurring** (a.k.a. Periodic Billing). You can also return money to a customer, either against an existing order on the Gateway (a **Return**) or against an existing order from outside the Gateway (a **Credit**). These same transaction types apply for credit cards and for purchasing cards. Additional information about each transaction type is provided below

To make changes to a **Recurring** credit card **Sale** transaction, you can either modify it using the API, or see your **Reports**. These recurring transactions are referred to as **Periodic Bills** in **Reports**.

No funds are transferred during any of these transactions; funds are transferred only after your batch of transactions is settled (this is set up to occur automatically once a day).

- **Sale**: immediately charges a customer’s credit card. A Sale transaction can be made Recurring.
- **Authorize Only**: reserves funds on a customer's credit card. Authorize Only does not charge the card until you perform a Ticket Only transaction and/or confirm shipment of the order (using an option available in **Reports**). Note that authorization reserves funds for varying periods, depending on the issuing credit card company's policy. The period may be as little as three days or as long as several months. For your protection we strongly suggest that you confirm shipment as soon as possible after authorization.
- **Ticket Only**: a post authorization. Captures the funds from an **Authorize Only** transaction, reserving funds on the customer’s card for the amount specified. Funds are transferred when your batch of transactions is settled. If you enter a larger total in the Post-Authorization transaction than was specified for the **Authorize Only** transaction, the Post-Authorization transaction may be declined. If you enter a smaller amount than was authorized, an adjustment is made to the Authorization to reserve only the smaller amount of funds on the customer’s card for the transaction.
- **Forced Ticket**: a forced post authorization. This transaction type is used similarly to a **Ticket Only** transaction, except it is specifically for authorizations you obtained over the phone (not through LinkPoint Central). It requires a reference number (or approval code) that you should have received when you did the phone authorization.
- **Return**: returns funds to a customer’s credit card against an existing order on the Gateway. To perform a return, you need the order number (which you can find in your **Reports**). If you perform a Return of the full order amount, the order will appear in your Reports with a transaction amount of 0.00.
- **Credit**: returns funds to a customer’s credit card for orders where you do not have a Gateway order number. This transaction is intended for returns against orders processed outside the Gateway. (Credit transactions are marked as **Returns** in your **Reports**.) Credit transactions are not enabled on some merchant accounts--if you need this functionality, please contact support.
- **Void**: voids the transaction. Only transactions in the current batch which have not yet been submitted for final processing can be voided.

VirtualCheck Transaction Types

Currently, there are only two transaction types for a VirtualCheck transaction: SALE and VOID.

- **Sale**: Initiates a debit from the consumer's account. Money is transferred to the merchant account after a short holding period.
- **Void**: voids the transaction. Only transactions in the current batch which have not yet been submitted for processing can be voided.

Sending a Transaction to the Gateway

No matter which type of transaction you are doing, the basic steps to process it are the same. Credit card, virtual check, ESD, shipping and tax calculations all follow this same process. To process a transaction, follow the steps shown below.

1. First, instantiate the **LinkPointTransaction** object.
2. Build an XML request, either directly, or by using the **LPOrderPart** object to build XML fragments and to combine them into a complete order, ready to be sent for processing.
3. Use the **LinkPointTransaction** ***send()*** method to send the transaction to the Gateway for processing.
4. Evaluate the values returned in the **response** entity. If there are errors, they will be returned in the **r_error** field.

Where to Go from Here

- Before you start building your XML request, make sure you review the How Transactions are Processed section to determine which entities and fields are required for your transaction.
- Examples of each of these steps (in computer code form) are shown in the **Sample Code** section.
- Information on the entities and data fields to include for each transaction are included in the **Entities and Data Fields** section.
- Look for information on Gateway responses and error messages/codes in the Responses from the Gateway section.

Using the Shipping Calculator

The Shipping Calculator is an optional service that enables you to set precise rules for calculating shipping charges.

NOTE: Skip this topic if you use a third-party product or other method of calculating shipping costs. You may simply send your calculated shipping charges by populating the **shipping** field in the **payment** entity.

IMPLEMENTING THE SHIPPING CALCULATOR

To use the shipping calculator module, you must first create a shipping and carrier file to be housed on the Gateway server. Once you've created your shipping file, send it to support, along with the appropriate merchant store number.

The shipping calculator uses the shipping address and other information sent in the **shipping** entity along with the appropriate pricing data defined in the shipping file to calculate the charges.

CREATING A SHIPPING FILE

The shipping file is a plain text file that consists of multiple sets of programming code called zone type and zone definition lines. An example of how these lines might appear in a shipping file is shown below.

```
zone type line
zone definition line
zone definition line
zone type line
zone definition line
```

The fields within both types of lines go together to define the shipping charges. The zone type line describes the general shipping scheme, such as whether costs are based on item count, weight or price. The zone definition line gives specific parameters on pricing for each element in that pricing scheme.

One or more zone definition lines must immediately follow each zone type line. Use zone definition to set shipping prices based on specific geographic areas and/or types of carriers to determine where price breaks occur. The fields within each line of code are separated by double colons. For fields with multiple values, use commas (countries, states) or single colons (range definitions, prices).

Creating Zone Type Lines

Each zone-type line is formatted with three fields: the tag name, a calculation code and merchant-created range definitions (see below).

```
zone type::calculation method::range1:range2...
```

You can create as many zone type lines as you need for your business. You can use a separate zone type line for different shipping-cost calculations (e.g., total weight or total cost of an order), for separate freight or air transport carrier methods, or for division of the world shipping-zone prices.

Step 1. Enter the following tag name, followed by two colons.

zone type::

Step 2. Determine how to charge customers for shipping your products (see Selecting the Calculation Method).

Enter an applicable code number after the tag name, followed by double colons, with no spaces (e.g., 1 for the total cost of the items or 3 for the total weight of order).

zone type::1::
zone type::3::

Step 3. Create quantity ranges that share common pricing. Enter each range, followed by a single colon or a comma (e.g., 1-3, 4-5, or over 6 items, 1-24, 25-50, or over 51 pounds).

zone type::1::1-3,4-5,6+
zone type::3::1-24,25-50,51+

Selecting the Calculation Method

There are five choices for calculating the shipping charges. Select the applicable calculation method(s) for your business from the list below. Enter one code number, after the tag name, for each zone-type line.

Method Number	Description
1	Charges are based on the total number of items
2	Calculates charges based on each item, then totaled
3	Charges are based on the total weight of the order
4	Charges are based on the weight of each item, then totaled
5	Charges are based on the total price of the order

Assigning Ranges

A **range** is defined as a value or a set of values, representing all items within a predetermined category, which use the same shipping charge. A range can be a single number (e.g., 1), two numbers separated by a hyphen (e.g., 2-7) or a number followed by a plus sign (e.g., 8+). You can specify a range to include one piece or hundreds of items; a gram, many ounces or tons. Also, an infinite number of ranges can be specified, but the number of ranges within a zone type line must correlate exactly with the number of prices in the following zone definition lines.

The following restrictions apply:

- Range definitions must be contiguous - no numbers may be skipped.
- Range definitions must start with the integer 1.
- The last range defined in each line must include a "+" sign.

Creating Zone Definition Lines

A zone-definition line specifies data that is required by the preceding zone type line of code. Several fields are specific to each business and must be created by the merchant, including the zone name, the shipping carrier code and the shipping-cost codes for each range. See the example below.

```
zone name::country::carrier::range cost::range cost
```

Step 1. Create zone names for each shipping situation (see Assigning Zone Names). Enter a zone name (e.g., domestic), followed by two colons.

```
northamerica::
```

Step 2. Select the applicable countries for your zone name, followed by double colons. Use the 2-digit country codes specified in the Country Codes section.

```
northamerica::US,MX,CA::
```

For the US only, enter each applicable 2-letter state code after the country code, followed by two colons.

```
westcoast::US::CA,OR,WA,HI::
```

Step 3. Determine the different shipping methods for your business. Enter one merchant-defined shipping carrier code (e.g., ground carrier is code 1) only.

```
northamerica::US,MX,CA::1::
```

Step 4. Determine the shipping cost for each range you specified in the zone-type line (see Assigning Shipping Charges to Ranges). Enter the applicable shipping cost, followed by a colon or a comma (e.g., 1-3 items, costs $25.00 total to ship, 4-5 items cost $40.00, 6 and over $75).

```
zone type::1::1-3,4-5,6+
northamerica::US::MX::CA::1::25,40,75
```

NOTE: Each shipping cost value in the zone definition line must match up with a range in the zone type line.

Assigning Zone Names

You determine the zone name for each zone definition line. Each name is an alphabetic string containing less than 20 letters and cannot include blank spaces.

Specifying the Shipping Carriers

If you offer different types of shipping (courier, overnight, two day, ground transport), the zone definition line can list a shipping carrier option in the form of an integer. This will allow you to charge different amounts for premium shipping services.

Assigning Shipping Charges to Ranges

The zone definition contains the actual charges for shipping items in the range(s) specified by the preceding zone type. Merchants determine the charges for their products.

Tips for Using the Zone Definition

The following rules apply when you are creating zone-definition code:

- If you are shipping internationally, the U.S. state codes in a zone-definition line are ignored.
- If shipping prices are the same for all U.S. states, you do not need to name the states individually.
- If you have a few exceptions for shipping (such as AK and HI), you can define a zone for them and include the remaining states in a nonspecific U.S. zone.
- Any number of zone-definition lines may follow a zone-type line.
- The zone name and range charges must have values; all other fields can be blank.
- When the shipping calculator looks for a shipping file match, a blank field (such as carrier type) is treated as a match.

CALCULATING SHIPPING CHARGES

The **shipping** entity is the data structure used to convey the shipping address, price, weight, item count and carrier-type data to the shipping calculator for computing the shipping charge.

To calculate shipping charges, **ordertype** should be set to ***calcshipping***. The shipping computation can be based on the number of items, the weight, the carrier, or the order total, so to properly calculate shipping charges, you should pass the appropriate information (for your shipping calculation method) in the **shipping** entity; that is **items**, **weight**, **carrier**, and **total**. You should also pass **state** and **country**, although **country** will default to US if not passed.

The calculated value for shipping charges will be returned in the **r_shipping** tag.

KEEPING TRACK OF SHIPPING

Regardless if you use the API shipping calculator or another method, if shipping is to be included in the order amount that you submit for approval, you must transfer the shipping amount to the Gateway. Make sure you set the value of the **shipping** data field in the **payment** entity to the calculated shipping charges for this order.

Using the Tax Calculator

The Tax Calculator module is an optional service that provides real-time calculation of U.S. sales tax to be charged on orders, including state and/or municipal sales tax.

NOTE: Skip this section if you use another method of calculating tax costs. You may simply send your calculated tax charges by populating the **tax** field in the **payment** entity. However, if you use another means of calculating tax, the tax statistics in your Reports will contain no data.

IMPLEMENTING THE TAX CALCULATOR

To use the tax calculator module, you must create a *fulltax* line that is inserted and housed in the merchant's configuration file on the Gateway. You must send the *fulltax* line to support to load it onto the Gateway.

Creating a Fulltax Line

The *fulltax* line provides information needed for the tax module to calculate sales tax for an order. The line includes entries only for states where sales tax must be charged. Entries are separated by a comma, which may be followed by a space.

Example:

    fulltax: TX 8.25, AL 7.00, FL 7.00, UT mun

Most entries in the list consist of the two-digit code for the state (for codes refer to U.S. State Codes) followed by a space and the tax rate to be charged for that state, like this:

    TX 8.25

If the tax to be charged includes municipal tax, the listing is the two-digit state code followed by the designation mun, like this:

    UT mun

Municipal taxes are calculated according to the SALESTAX.TXT file residing on the Gateway server. The SALESTAX.TXT file is updated quarterly to ensure accuracy.

CALCULATING TAXES

To calculate sales tax, **ordertype** should be set to ***calctax***. The tax computation is based on the shipping state and the shipping zip code, so to properly calculate sales tax, you should pass the state and zip code where the goods are to be shipped. That is, in the **shipping** entity, **state** and **zip** should be present.

If the shipping state is present, but the shipping zip code is not, the Gateway will return an error. If you do not pass either the shipping state or zip code, it assumes the billing address is the shipping address and uses the billing state and zip code as the basis for tax computations.

The calculated value for sales tax will be returned in the **r_tax** tag.

KEEPING TRACK OF TAX

Regardless if you use the API tax calculator or another method, if sales tax is to be included in the order amount that you submit for approval, you must transfer the tax amount to the Gateway. Make sure you set the value of the **tax** data field in the **payment** entity to the calculated sales tax charges for this order.

Using the Electronic Softgood Download Module

OVERVIEW

The Electronic Softgoods Download (ESD) module automates the Internet-based sale of soft goods. A soft good is sometimes referred to as a digital good and is any file that can be saved on a hard drive, such as software, digital images or music. Products can be ordered, paid for and delivered by electronic download.

The ESD module provides merchant protection by creating a hidden URL from which the customer downloads a purchase. Each download URL is unique to each sale, and the soft good file at that URL is automatically removed after a successful download (as verified by byte-by-byte metering). This way, the customer can return to download the purchase after obtaining a better Internet connection if it fails during a download attempt, but the soft good is inaccessible to others who have not purchased it.

The ESD process uses an ordinary Web browser and server software and standard Internet protocols, such as standard MIME types, so that browsers will recognize files. For instance, typing text/html tells the browser to display HTML. Typing video/mpeg is recognized as a movie, and the browser spawns the correct viewer. These types are set based on the extension of the file, e.g., html, jpg, mpg, mp3. There is no need to encapsulate or modify the soft good before it can be sold, and the system is easily integrated into virtual storefront and electronic catalog packages.

Before you can use the ESD module, you must arrange to upload the file to the Gateway server. Please contact support to arrange for transferring your softgood files.

ESD TYPES

ESD works with the Payment module and includes two distinct download types: ***softgood*** and ***key***.

Softgood Type

The ***softgood*** type works by creating a URL from which the customer can download the purchased file. The download URL is delivered to the customer by means of the onscreen transaction confirmation and in an e-mail receipt.

NOTE: The unique URL is a symbolic link that points to the master file. During the download process, the system measures the delivery of the soft good, byte by byte. Once the customer has successfully downloaded the purchased product, the URL is automatically removed to prevent additional downloads.

Should the download fail for some reason, the URL remains active until the customer is able to reconnect to the Internet and to the URL and successfully download the product. The URL for a softgood download created by the module is in this form:

https://secureserver/cgi-bin/esddownload/storename/orderid/dirname/filename

This is an example of a download URL:

https://secureserver.linkpoint.com/cgi-bin/esddownload/samplestore/266.266.266.266

Monitoring Customer Download

Merchants can check the status of a customer soft good download in the Reports, which is described in the Help files.

To view the status in Reports, follow the **View Orders** link from the main Reports menu, generate an **Orders Received** report, then select an order number to display its Order Number Receipt. In the Order Number Receipt, if an item listed is a soft good and the soft good has not yet been downloaded, the **Download Time** field displays “Not downloaded yet.”

If the soft good item has been successfully downloaded, these three fields are displayed:

- **Download Time** — time the download began.
- **Download IP** — IP address to which the file was downloaded.
- **Download Rate** — rate (in Kilobytes/second) at which the download occurred.

Keygen Executable Type

The ***key*** type is a mechanism to download a generated key (frequently a serial number) that enables a customer to use a “locked” file downloaded by another means. Typically, the file is freely downloadable from the merchant’s Web site, but the means to open the file must be purchased from the merchant.

The unique key is created using a keygen executable program provided by the merchant. The ESD keygen binary for your store must reside on the gateway secure server. The binary is called when an item requiring the key is sold. The binary generates output, such as text or an HTML link.

The following data fields can be passed to the key generation utility inside the ESD module by the API:

Entity	Data Field	Notes
billing	**name**	Customer name is required if the **esdtype** is ***key***.
item	**serial**	The serial number (or key) associated with the keygen file. Only used if the if the **esdtype** is ***key***

If you pass the serial number (that is, the **serial** data field in the **item** entity), it is the first argument passed to the key generation application; otherwise, the customer name (the **name** field in the **billing** entity) is the first argument passed. Distribution of the generated key is handled as directed by the keygen executable and according to the settings for the ***sendclientkey:*** and ***sendmerchantkey:*** file tags in the store’s merchant configuration file. Contact support to change your *sendclientkey* and/or *sendmerchantkey* settings. The key is distributed to the customer using e-mail and/or returned to the storefront application for display through the customer’s browser.

Merchant Configuration File Tag	Description	Settings
sendclientkey:	Designates whether or not the key code for unlocking the downloaded software will be included in the *customer's* e-mail receipt.	***0*** : will not be included ***1*** : will be included
sendmerchantkey:	Designates whether or not the key code for unlocking the downloaded software will be included in the *merchant's* e-mail receipt.	***0*** : will not be included ***1*** : will be included

REQUIRED FIELDS FOR ESD TRANSACTIONS

The fields which you must assign values for an ESD transaction are:

item sub-entity			
item Field Name	Data Type	Description	Required?
id	string	Item ID number	Required for each item passed
description	string	Description of this item	Required for ESD
price	string	Price of this item	Required for each item passed
quantity	string	Quantity of this item to include in this order	Required for each item passed
serial	string	Serial number associated with this item	Required for ESD if **esdtype** is set to ***key***
esdtype	string	For electronic softgood download (ESD) items, this tag indicates which type of ESD item it is: ***softgood*** if softgood or ***key*** if a keygen executable.	Required for ESD
softfile	string	The name of the softgood file or keygen executable to be downloaded	Required for ESD items

MAKING ESD PRODUCTS AVAILABLE

The files you intend to make available as electronic downloads from your storefront must reside on the Gateway SSL-secure server. Send all softgood files to support for loading onto the server.

ESD ERRORS

If for some reason the download link cannot be created or the key cannot be generated, one of the following ESD error messages will be returned:

- Could not create ESD URL.
- Could not generate key.

If you receive an ESD error message, contact support to resolve the situation because a SALE transaction was already performed, charging the customer's credit card for the product.

Required Entities

The table below shows which XML entities are used for each transaction type or function to be performed. See the **Entities and Data Fields** section for information about the individual entity structures and data fields in each entity.

XML Entity	Credit Card Transaction	Recurring Credit Card Transaction	VirtualCheck Transaction	Shipping Calculation	Sales Tax Calculation
merchantinfo	Required	Required	Required	Required	Required
orderoptions	Required	Required	Required	Required	Required
transactiondetails	Optional	Optional	Optional	Optional	Optional
payment	Required	Required	Required	Required	Required
creditcard	Required	Required	N/A	N/A	N/A
telecheck	N/A	N/A	Required	Required	N/A
shipping	Optional	Optional	Optional	Required	Required
billing	Required	Optional	Required	N/A	N/A
periodic	N/A	Required	Not Available	N/A	N/A
items	Optional	Optional	Not Available	N/A	N/A
item	Optional	Optional	Not Available	N/A	N/A
option	Optional	Optional	Not Available	N/A	N/A
notes	Optional	Optional	Optional	N/A	N/A

Minimum Required Fields for a Credit Card Transaction

The table below shows the absolute minimum required entities and data fields to perform a credit card transaction. You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities for a credit card transaction are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **creditcard**

Also, if it is a card-not-present transaction (e.g., mail order, telephone order, or internet order), you should also include the **billing** entity for address verification (AVS). Most card-present transactions (i.e., retail, face-to-face) do not require AVS. The one exception is self-service stations such as gas pumps or ticket stations where you should include the **zip** data field for a partial AVS.

f you are using ASP, .NET/Visual Basic, or .NET/C#, all **billing** fields will be preceded with a lower-case "b" (e.g., **name** would be **bname**). If you are using PERL, PHP, ASP, .NET/Visual Basic or .NET/C#, all **shipping** fields will be preceded with a lower-case "s" (e.g., **name** would be **sname**).

In the **creditcard** entity, you must either pass **cardnumber**, **cardexpmonth**, and **cardexpyear** (all three fields) OR **track**.

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			

ordertype	string	The type of transaction. The possible values are ***SALE***, ***PREAUTH*** (for an Authorize Only transaction), ***POSTAUTH*** (for a Forced Ticket or Ticket Only transaction), ***VOID***, ***CREDIT***, ***CALCSHIPPING*** (for shipping charges calculations) and ***CALCTAX*** (for sales tax calculations). See the Transaction Types section for additional information.	Required
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
creditcard entity			
cardnumber	numeric	The consumer's credit card number	Required for credit card transactions if ***track*** is not provided
cardexpmonth	integer from 1 to 12	The numeric expiration month of the credit card	Required for credit card transactions if ***track*** is not provided
cardexpyear	integer from 00 to 99	The two-digit expiration year of the credit card	Required for credit card transactions if ***track*** is not provided
track	string	Contains the data swiped from the credit card track 1 or track 2.	Required for credit card transactions if ***cardnumber***, ***cardexpmonth*** and ***cardexpyear*** are not provided
transactiondetails entity			

oid	string	The Order ID to be assigned to this transaction. For ***SALE*** and ***PREAUTH***, this field must be unique. For ***VOID***, ***CREDIT***, and ***POSTAUTH***, this field must be a valid Order ID from a prior ***SALE*** or ***PREAUTH*** transaction. For a Forced Ticket (that is a **POSTAUTH** where the authorization was given over the phone), the **oid** field is not required, but the **reference_number** field is required.	Required for credit card ***VOID***, ***CREDIT***, and ***POSTAUTH*** transactions
reference_number	string	Used for Forced Ticket transactions where a reference number was obtained over the phone. Set the reference number value to the word *NEW* (all caps) + the reference number given over the phone. For example, if the reference number given over the phone was ***123456***, you would set **reference_number** to ***NEW123456***.	Required for Forced Ticket transactions only.
billing entity			
addrnum	string	The numeric portion of the street address	Required for AVS
zip	string	Billing zip or postal code	Required for AVS and for Partial AVS.

The XML stream for a minimum credit card SALE transaction would look like this:

```
<!-- Minimum Required Fields for a Credit Card SALE-->
<order>
     <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>1234567</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>SALE</ordertype>
    </orderoptions>
    <payment>
        <chargetotal>12.99</chargetotal>
    </payment>
    <creditcard>
```

```
        <cardnumber>4111-1111-1111-1111</cardnumber>
        <cardexpmonth>03</cardexpmonth>
        <cardexpyear>05</cardexpyear>
    </creditcard>
</order>
```

Minimum Required Fields for a VirtualCheck Transaction

The required fields for a VirtualCheck transaction depend on the type of transaction it is. VirtualCheck orders are classified as pre-authorized (paper-based orders; e.g., retail or mail order transactions), telephone (i.e., orders taken over the phone), or web orders (i.e., orders taken via e-mail or over the Internet). There are also specific rules that apply to each type of transaction; please see *Rules for Processing VirtualCheck Transactions*.

Required Fields (VirtualCheck)

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			
ordertype	string	The type of transaction. The possible values for VirtualChecks are SALE or VOID only. Other transaction types do not apply for VirtualChecks. See the ***VirtualCheck Transaction Types*** section for additional information.	Required
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
telecheck entity			
account	string	The customer's bank account number	Required for all check transactions
routing	string	The transit routing number for the customer’s bank	Required for all check transactions
bankname	string	The name of the customer’s bank	Required for VirtualCheck transactions

bankstate	string	The state that the customer's bank is located in	Required for VirtualCheck transactions. Use one of the US State Codes.
dl	string	Driver's license number of the customer	Required for check orders taken over the telephone or the web
dlstate	string	The state where the customer's driver's license was issued	Required for check orders taken over the telephone or the web. Use one of the US State Codes.
void	integer	To void an unsettled check, pass a 1 in this tag. Otherwise, do not include this data field.	Required for check VOID transactions
billing entity			
name	string	This should be the customer's name as it appears on the payment account.	Required for all VirtualCheck transactions
address1	string	The 1st line of the customer's street address	Required for check orders taken over the telephone.
city	string	Billing city	Required for check orders taken over the telephone.
state	string	Billing state. For international addresses, you can use this field to hold the province or territory, as applicable.	Required for check orders taken over the telephone. Use one of the US State Codes.
zip	string	Billing zip or postal code	Required for check orders taken over the telephone.
phone	string	Billing phone number	Required for check orders taken over the telephone.
email	string	E-mail address	Required if you wish to send the customer an e-mail receipt.
transactiondetails entity			

transactionorigin	string	The source of the transaction. The possible values are ECI (if the order was received via e-mail or Internet), MAIL (mail order), TELEPHONE (telephone order) and RETAIL (face to face). For VirtualCheck transactions, mail order and telephone order transactions are treated differently and the merchant should take care to identify them separately. For a VirtualCheck order, you should **NOT** set the **transactionorigin** to ***MOTO***.	Required for retail, mail, or telephone orders. Defaults to ECI.

XML Stream (VirtualCheck)

The XML stream for a minimum VirtualCheck SALE transaction would look like this:

```
<!-- Minimum Required Fields for a VirtualCheck SALE-->
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>1234567</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>SALE</ordertype>
    </orderoptions>
    <payment>
        <chargetotal>12.99</chargetotal>
    </payment>
    <telecheck>
        <!-- Customer's Driver's license # and DL state.-->
        <dl>120381698</dl>
        <dlstate>CA</dlstate>
        <!-- Transit routing number for the customer's bank -->
        <routing>123456789</routing>
        <!-- Customer's bank account number -->
        <account>2139842610</account>
        <!-- Is this a business or consumer (personal) account? personal = pc, business = bc -->
        <accounttype>pc</accounttype>
        <account>2139842610</account>
        <!-- Bank name and 2-letter bank state -->
        <bankname>MyBank</bankname>
        <bankstate>CA</bankstate>
    </telecheck>
    <billing>
```

```
        <name>Bill Johnson</name>
        <address1>123 Broadway</address1>
        <city>Camarillo</city>
        <state>CA</state>
        <zip>93010</zip>
        <phone>8051234567</phone>
        <email>bjohnson@somewhere.com</email>
    </billing>
    <transactiondetails>
        <!-- Required for Retail, Mail, or Telephone orders only -->
        <transactionorigin>TELEPHONE</transactionorigin>
    </transactiondetails>
</order>
```

Minimum Required Fields to Calculate Shipping

The table below shows the absolute minimum required entities and data fields to calculate shipping charges. You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities needed to calculate shipping charges are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **shipping**

Before you can calculate shipping charges, the payment gateway needs information from you on how to calculate shipping. You must first create a shipping file that describes the calculation method and amounts to charge. There are several different shipping methods available. Contact support to send your shipping file to the gateway. See the chapter entitled *Using the Shipping Calculator* for more information on shipping files.

To calculate shipping charges, **ordertype** should be set to ***calcshipping***. The shipping computation can be based on the number of items, the weight, the carrier, or the order total, so to properly calculate shipping charges, you should pass the appropriate information (for your shipping calculation method) in the **shipping** entity; that is **items**, **weight**, **carrier**, and **total**. You should also pass **state**. If you do not pass **country**, it will default to ***US***.

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			

ordertype	string	The type of transaction. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations). See the Transaction Types section for additional information.	Required. Set to ***CALCSHIPPING*** to calculate shipping charges.
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
shipping entity			
state	string	Shipping state. For international addresses, you can use this field to hold the province or territory, as applicable.	Required for shipping and tax calculations
weight	decimal	The total weight of the order to be shipped. Used when the shipping calculation method is based on the total weight of the order.	Required for shipping methods 3 and 4
items	decimal	Total number of items in the order. Used when the shipping calculation method is based on the number of items in the order. See Using the Shipping Calculator for more information.	Required for shipping methods 1, 2, and 4
carrier	integer	The carrier to be used to ship the order (e.g., Ground, Overnight, etc.). Used when the merchant is specifying carrier. The merchant should assign integer values to each carrier the merchant uses.	Required if merchant is using carrier types.
total	decimal	The order total before the shipping charges are added.	Required for shipping method 5

The XML stream for a shipping calculation would look like this:

```
<!-- Minimum Required Fields to Calculate Shipping Charges -->
```

```
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>YOURSTORE</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>CALCSHIPPING</ordertype>
    </orderoptions>
     <shipping>
        <!-- Include the factors needed for your shipping method -->
        <carrier>1</carrier>
        <weight>2.00</weight>
        <items>2</items>
        <total>15.00</total>
        <state>TX</state>
    </shipping>
</order>
```

Minimum Required Fields to Calculate Sales Tax

The table below shows the absolute minimum required entities and data fields to calculate sales tax. You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities needed to calculate sales tax are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **shipping**

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			
ordertype	string	The type of transaction. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations). See the Transaction Types section for additional information.	Required. Set to ***CALCTAX*** to calculate tax charges.
payment entity			
subtotal	double	The transaction subtotal, before shipping and tax	Required for tax calculations
shipping entity			
state	string	Shipping state. For international addresses, you can use this field to hold the province or territory, as applicable.	Required for shipping and tax calculations

The XML stream for a tax calculation would look like this:

```
<!-- Minimum Required Fields to Calculate Sales Tax on an Order -->
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>YOURSTORE</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>CALCTAX</ordertype>
    </orderoptions>
    <shipping>
        <carrier>1</carrier>
        <weight>1.000000</weight>
        <zip>91504</zip>
        <total>26.99</total>
        <state>CA</state>
    </shipping>
    <payment>
        <subtotal>26.99</subtotal>
    </payment>
</order>
```

Minimum Required Fields for a Recurring Credit Card Transaction

The table below shows the absolute minimum required entities and data fields to perform a recurring credit card transaction (a.k.a., periodic bill). You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities for a recurring credit card transaction are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **creditcard**
- **periodic**

In the **creditcard** entity, you must either pass **cardnumber**, **cardexpmonth**, and **cardexpyear** (all three fields) OR **track**.

Also, if it is a card-not-present transaction (e.g., mail order, telephone order, or internet order), you should also include the **billing** entity for address verification (AVS). Most card-present transactions (i.e., retail, face-to-face) do not require AVS. The one exception is self-service stations such as gas pumps or ticket stations where you should include the **zip** data field for a partial AVS.

If you are using ASP, .NET/Visual Basic, or .NET/C#, all **billing** fields will be preceded with a lower-case "b" (e.g., **name** would be **bname**). If you are using PERL, PHP, ASP, .NET/Visual Basic or .NET/C#, all **shipping** fields will be preceded with a lower-case "s" (e.g., **name** would be **sname**).

At this time, recurring transactions can only be run in ***LIVE*** mode, so please ensure that for a recurring transaction that the **transactionorigin** field in the **transactiondetails** entity is NOT set to any other value.

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			

ordertype	string	The type of transaction. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations). See the Transaction Types section for additional information.	Required
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
creditcard entity			
cardnumber	numeric	The consumer's credit card number	Required for credit card transactions if ***track*** is not provided
cardexpmonth	integer from 1 to 12	The numeric expiration month of the credit card	Required for credit card transactions if ***track*** is not provided
cardexpyear	integer from 00 to 99	The two-digit expiration year of the credit card	Required for credit card transactions if ***track*** is not provided
track	string	Contains the data swiped from the credit card track 1 or track 2.	Required for credit card transactions if ***cardnumber***, ***cardexpmonth*** and ***cardexpyear*** are not provided
periodic entity			
action	string	Tells which action to take regarding the periodic transaction. Possible values are ***SUBMIT*** (to submit the recurring transaction for processing), ***MODIFY*** (to edit a previously submitted recurring transaction), or ***CANCEL*** (to cancel a previously submitted recurring transaction).	Required for all recurring transactions
installments	integer from 0 to 99	Identifies how many recurring payments to charge the customer	Required for all recurring transactions

periodicity	string	Tells how often to charge the payment. Possible values are: ● ***monthly*** (to charge once a month), ● ***bimonthly*** (to charge every other month), ● ***weekly*** (to charge once a week), ● ***biweekly*** (to charge once every other week), ● ***yearly*** (to charge once a year), ● ***daily*** (to charge once a day), ● or you can use the code ***XN***, which means every ***N*** days/weeks/months. ***X*** can be ***d*** for day, ***m*** for month, or ***y*** for year. ***N*** can be any integer from 1 through 999. Example: ***m3*** means charge the customer once every 3 months.	Required for all recurring transactions
startdate	string	Tells the date to begin charging the recurring payments. Should be in the format YYYYMMDD, where YYYY is the four-digit year, MM is the 2-digit month, and DD is the 2-digit day. Example: 20040910 means September 10, 2004. If you wish to start payments immediately, set this tag to ***immediate***	Required for all recurring transactions
billing entity			
addrnum	string	The numeric portion of the street address	Required for AVS
zip	string	Billing zip or postal code	Required for AVS and for Partial AVS.

The XML stream for a recurring credit card transaction would look like this:

```
<!-- Minimum Required Fields for a Recurring Credit Card Sale (Periodic Bill)-->
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>1234567</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>SALE</ordertype>
    </orderoptions>
    <payment>
```

```
        <chargetotal>12.99</chargetotal>
    </payment>
    <creditcard>
        <cardnumber>4111-1111-1111-1111</cardnumber>
        <cardexpmonth>03</cardexpmonth>
        <cardexpyear>05</cardexpyear>
    </creditcard>
    <periodic>
        <!-- Submits a recurring transaction charging the card 3 times, once a month, starting today -->
        <action>SUBMIT</action>
        <installments>3</installments>
        <threshold>3</threshold>
        <!-- If you don't want it to start today, pass a date in the format YYYYMMDD -->
        <startdate>immediate</startdate>
        <periodicity>monthly</periodicity>
    </periodic>
</order>
```

Minimum Required Fields for a Level 2 Purchasing Card Transaction

The table below shows the absolute minimum required entities and data fields to perform a Level 2 purchasing card transaction. You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities for a Level 2 purchasing card transaction are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **transactiondetails**
- **creditcard**

In the **creditcard** entity, you must either pass **cardnumber**, **cardexpmonth**, and **cardexpyear** (all three fields) OR **track**.

Also, if it is a card-not-present transaction (e.g., mail order, telephone order, or internet order), you should also include the **billing** entity for address verification (AVS). Most card-present transactions (i.e., retail, face-to-face) do not require AVS. The one exception is self-service stations such as gas pumps or ticket stations where you should include the **zip** data field for a partial AVS.

If you are using ASP, .NET/Visual Basic, or .NET/C#, all **billing** fields will be preceded with a lower-case "b" (e.g., **name** would be **bname**). If you are using PERL, PHP, ASP, .NET/Visual Basic or .NET/C#, all **shipping** fields will be preceded with a lower-case "s" (e.g., **name** would be **sname**).

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			

ordertype	string	The type of transaction. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations). See the Transaction Types section for additional information.	Required
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
tax	double	Sales tax dollar amount	Required for Level 2 purchasing card transactions only. If tax is $0.00 for a purchasing card transaction, set this field to 0.
transactiondetails entity			
taxexempt	string	An indicator field representing whether the order is tax exempt. It should be set to Y if the order is tax exempt, or N if not.	Required for Level 2 purchasing card transactions
ponumber	string	The purchase order number, department number, or other customer-supplied number that applies to this transaction. One P.O. number can apply to multiple orders. This value MUST be supplied by the customer, not the merchant.	Required for Level 2 purchasing card transactions
creditcard entity			
track	string	Contains the data swiped from the credit card track 1 or track 2.	Required for credit card transactions if ***cardnumber***, ***cardexpmonth*** and ***cardexpyear*** are not provided
cardnumber	numeric	The consumer's credit card number	Required for credit card transactions if ***track*** is not provided

cardexpmonth	integer from 1 to 12	The numeric expiration month of the credit card	Required for credit card transactions if ***track*** is not provided
cardexpyear	integer from 00 to 99	The two-digit expiration year of the credit card	Required for credit card transactions if ***track*** is not provided
billing entity			
addrnum	string	The numeric portion of the street address	Required for AVS
zip	string	Billing zip or postal code	Required for AVS and for Partial AVS.

The XML stream for a level 2 purchasing card transaction would look like this:

```
<!-- An Example Level 2 Purchasing Card SALE -->
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>1234567</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>SALE</ordertype>
    </orderoptions>
    <payment>
        <!-- Tax is required for purchasing cards. If the tax is $0.00, pass a value of 0 for tax -->
        <tax>0.32</tax>
        <chargetotal>47.32</chargetotal>
    </payment>
    <creditcard>
        <cardnumber>4111-1111-1111-1111</cardnumber>
        <cardexpmonth>03</cardexpmonth>
        <cardexpyear>05</cardexpyear>
    </creditcard>
    <transactiondetails>
        <!-- If there is no PO Number for this order, pass a department code or other value, but make sure
the value you pass is supplied by the customer -->
        <ponumber>1203A-G4567</ponumber>
        <!-- If the purchase is tax exempt, pass a value of Y for taxexempt-->
        <taxexempt>N</taxexempt>
    </transactiondetails>
</order>
```

Minimum Required Fields for AVS and CVM

To take advantage of the fraud prevention features of the Address Verification Service (AVS) and the Card Code (commonly called CVV, CVV2, CVC2 or CVM), there are just a couple extra fields you need to pass to the Gateway. Look for the AVS and CVM results in the **r_avs** response field.

The table below shows the absolute minimum required entities and data fields to perform a credit card transaction using these two features. You may use any or all of the data fields available, but you **must** include the entities and fields listed below.

The minimum required entities for a credit card transaction using AVS and CVM are:

- **merchantinfo**
- **orderoptions**
- **payment**
- **creditcard**
- **billing**

In the **creditcard** entity, you must either pass **cardnumber**, **cardexpmonth**, and **cardexpyear** (all three fields) OR **track**.

If you are using ASP, .NET/Visual Basic, or .NET/C#, all **billing** fields will be preceded with a lower-case "b" (e.g., **name** would be **bname**). If you are using PERL, PHP, ASP, .NET/Visual Basic or .NET/C#, all **shipping** fields will be preceded with a lower-case "s" (e.g., **name** would be **sname**).

See the table below for a list of required data fields for each entity.

Field Name	Data Type	Description	Required?
merchantinfo entity			
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten- digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
orderoptions entity			

ordertype	string	The type of transaction. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations). See the Transaction Types section for additional information.	Required
payment entity			
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required
creditcard entity			
cardnumber	numeric	The consumer's credit card number	Required for credit card transactions if ***track*** is not provided
cardexpmonth	integer from 1 to 12	The numeric expiration month of the credit card	Required for credit card transactions if ***track*** is not provided
cardexpyear	integer from 00 to 99	The two-digit expiration year of the credit card	Required for credit card transactions if ***track*** is not provided
track	string	Contains the data swiped from the credit card track 1 or track 2.	Required for credit card transactions if ***cardnumber***, ***cardexpmonth*** and ***cardexpyear*** are not provided
cvmvalue	integer from 000 to 999	3-digit numeric valued typically printed on the signature panel on the back of the credit card. See Card Codes section for more information.	Required for CVM (CVV2, CVC2)
cvmindicator	string	Indicates whether CVM was supplied and, if not, why. The possible values are “provided”, “not_provided”, “illegible”, “not_present”, and “no_imprint”.	Required if ***cvmvalue*** is passed
transactiondetails entity			

oid	string	The Order ID to be assigned to this transaction. For SALE and PREAUTH, this field must be unique. For VOID, CREDIT, and POSTAUTH, this field must be a valid Order ID from a prior SALE or PREAUTH transaction.	Required for credit card VOID, CREDIT, and POSTAUTH transactions
billing entity			
name	string	This should be the customer's name as it appears on the payment account.	Required for AVS and fraud name blocking
addrnum	string	The numeric portion of the street address	Required for AVS
zip	string	Billing zip or postal code	Required for AVS

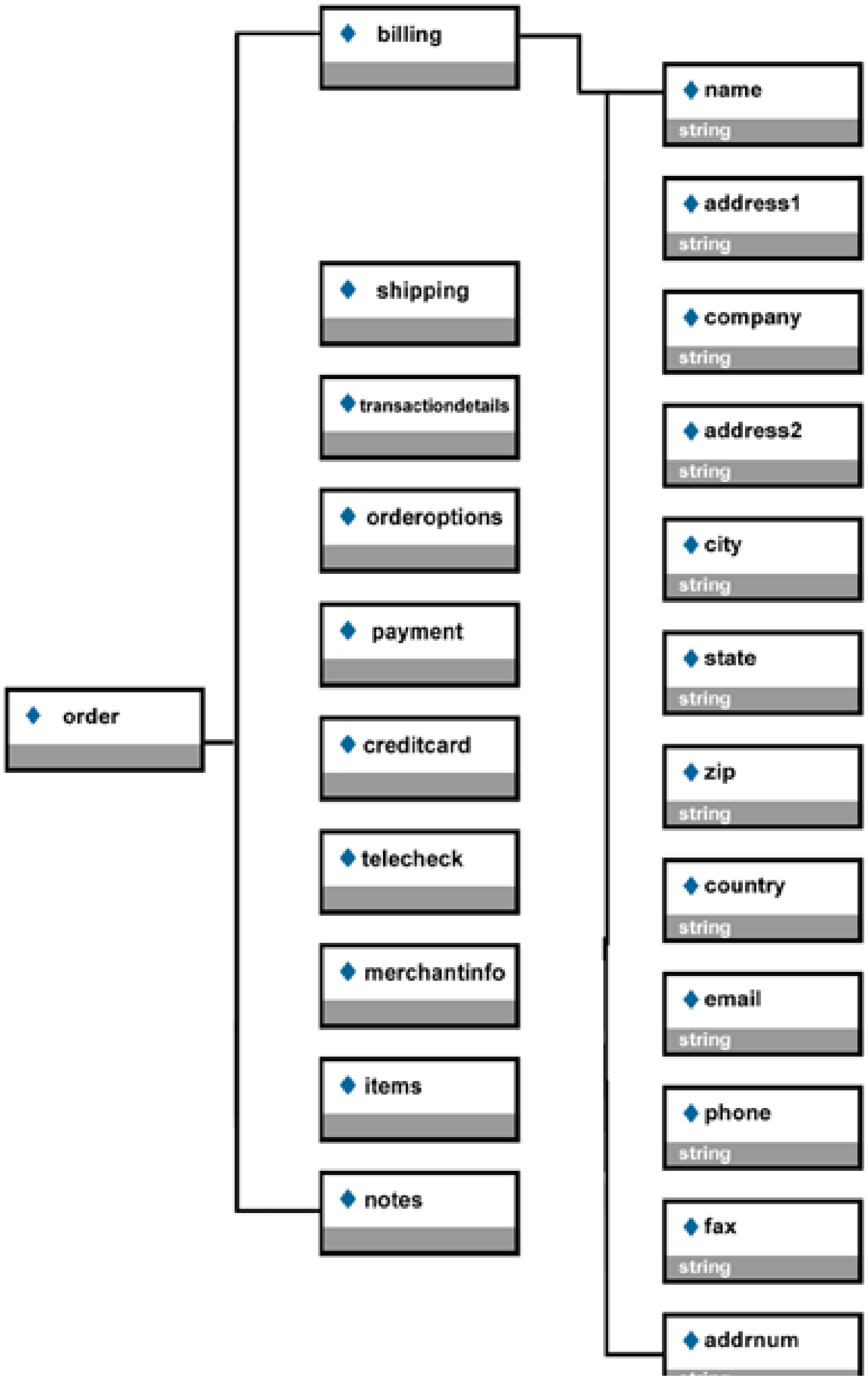
The XML stream for a credit card SALE transaction using AVS and Card Code fraud prevention would look like this:

```
<!-- An Example Credit Card SALE With Minimum Fields required for AVS and Card Code fraud
prevention measures-->
<order>
    <merchantinfo>
        <!-- Replace with your STORE NUMBER or STORENAME-->
        <configfile>1234567</configfile>
    </merchantinfo>
    <orderoptions>
        <ordertype>SALE</ordertype>
    </orderoptions>
    <payment>
        <chargetotal>47.32</chargetotal>
    </payment>
    <creditcard>
        <cardnumber>4111-1111-1111-1111</cardnumber>
        <cardexpmonth>03</cardexpmonth>
        <cardexpyear>05</cardexpyear>
        <!-- CVM is the three-digit security code usually found on the signature line on the back of the
card -->
        <cvmvalue>123</cvmvalue>
        <cvmindicator>provided</cvmindicator>
    </creditcard>
    <billing>
        <!-- Required for Address Verification -->
        <addrnum>123</addrnum>
        <zip>87123</zip>
    </billing>
</order>
```

The Billing Entity

The **billing** entity is used to describe the billing information for the customer. It contains thirteen data fields: name, company, address1, address2, city, state, zip, country, email, phone, fax, and addrnum.

The diagram below shows the **billing** entity data structure.



string

Billing Data Fields

The table below describes each data field in the **billing** entity. If you are using ASP, .NE Visual Basic, or .NET/C#, all billing fields will be preceded with a lower-case "b" (e.g., **name** would be **bname**).

billing Field Name	Data Type	Description	Required?
name	string	This should be the customer's name as it appears on the payment account.	Required for AVS and fraud name blocking
company	string	Company name	Not required. If the company name has an ampersand (&) in it, replace the & with the word "***and***" or the HTML escape character ***&amp;***.
address1	string	The 1st line of the customer's address	Not required for credit card transactions. Required for VirtualCheck transactions if the order was taken over the telephone.
address2	string	The 2nd line of the customer's address	Not required
city	string	Billing city	Not required. Required for VirtualCheck transactions if the order was taken over the telephone.
state	string	Billing state. For international addresses, you can use this field to hold the province or territory, as applicable. For US states, use one of the US State Codes.	Not required. Required for VirtualCheck transactions if the order was taken over the telephone.
zip	string	Billing zip or postal code	Required for AVS. Required for VirtualCheck transactions if the order was taken over the telephone.
country	string	Billing country. If passed, must be a valid country code. See Country Codes.	Not required, but post-authorizations may require it
phone	string	Billing phone number	Required for VirtualCheck transactions if the order was taken over the telephone.
fax	string	Fax number	Not required

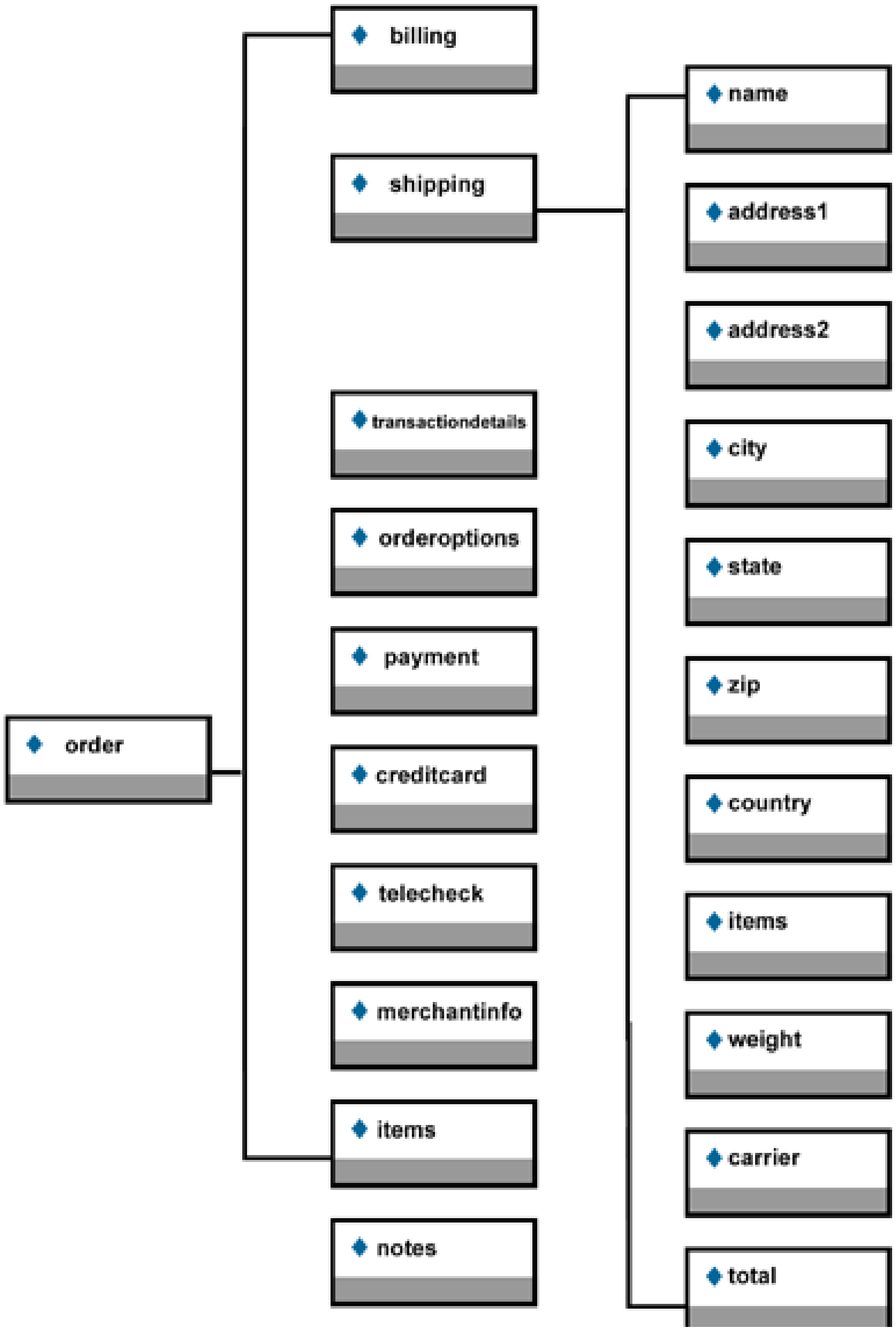
email	string	E-mail address	Required if you wish to send the customer an e-mail receipt.
addrnum	string	The numeric portion of the street address	Required for AVS

The Shipping Entity

The **shipping** entity contains the shipping address and shipping calculation information, which contains the following data fields:

- For shipping addresses: **name**, **address1**, **address2**, **city**, **state**, **zip**, and **country**.
- For shipping calculations: **items**, **weight**, **carrier**, **total**

The diagram below shows the **shipping** entity data structure.



Shipping Data Fields

The table below describes each data field in the **shipping** entity. If you are using PERL, PHP, ASP, .NET/Visual Basic or .NET/C#, all shipping fields will be preceded with a lower-case "s" (e.g., **name** would be **sname**).

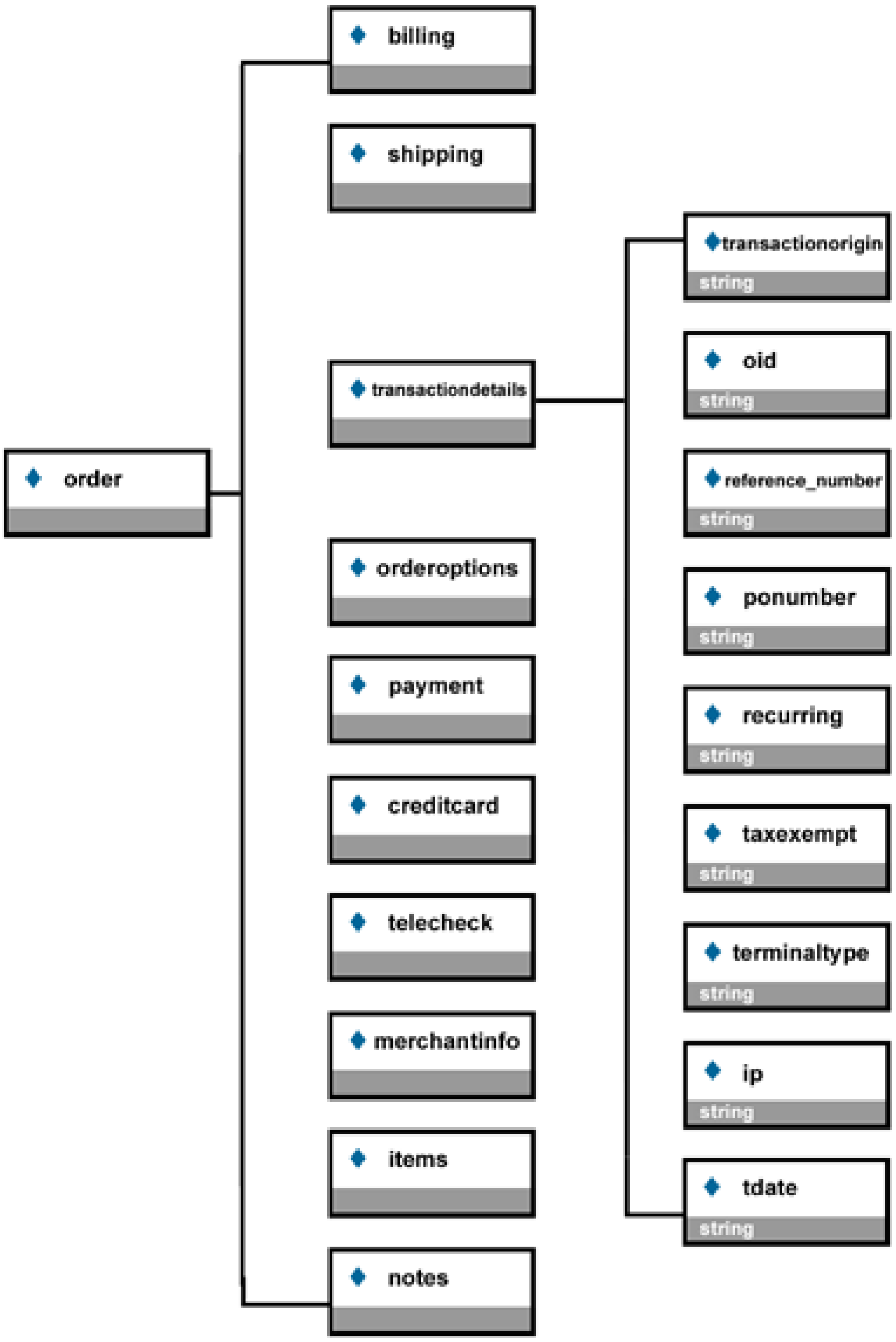
shipping Field Name	Data Type	Description	Required?
name	string	The name of the person to ship to.	Not required.
address1	string	The 1st line of the shipping address	Not required
address2	string	The 2nd line of the shipping address	Not required
city	string	Shipping city	Not required
state	string	Shipping state. For international addresses, you can use this field to hold the province or territory, as applicable. For US states, use one of the US State Codes.	Required for shipping and tax calculations
zip	string	Shipping zip or postal code.	Not required
country	string	Shipping country. If passed, must be a valid country code. See Country Codes.	Not required, but post-authorizations may require it.
weight	decimal	The total weight of the order to be shipped. Used when the shipping calculation method is based on the total weight of the order.	Required for shipping methods 3 and 4

items	decimal	Total number of items in the order. Used when the shipping calculation method is based on the number of items in the order. See Using the Shipping Calculator for more information.	Required for shipping methods 1, 2, and 4
carrier	integer	The carrier to be used to ship the order (e.g., Ground, Overnight, etc.). Used when the merchant is specifying carrier. The merchant should assign integer values to each carrier the merchant uses.	Required if merchant is using carrier types.
total	decimal	The order total before the shipping charges are added.	Required for shipping method 5

The Transaction Details Entity

The purpose of the **transactiondetails** entity is to communicate all the general information about the payment transaction. The data fields that it may contain are transactionorigin, oid, reference_number, ponumber, recurring, taxexempt, terminaltype, ip, and tdate.

The diagram below shows the **transactiondetails** entity data structure.



Transaction Details Data Fields

The table below describes each data field in the **transactiondetails** entity.

transactiondetails Field Name	Data Type	Description	Required?
transactionorigin	string	The source of the transaction. The possible values are ***ECI*** (if the order was received via e-mail or Internet), ***MAIL*** (mail order), ***MOTO*** (mail order / telephone order), ***TELEPHONE*** (telephone order) and ***RETAIL*** (face to face). For credit card transactions, mail order and telephone order transactions are treated the same. However, for VirtualCheck transactions, mail order and telephone order transactions are treated differently and the merchant should take care to identify them separately.	Required for retail, mail, and telephone orders. Defaults to ***ECI***.
oid	string	The Order ID to be assigned to this transaction. For ***SALE*** and ***PREAUTH***, this field must be unique. For ***VOID***, ***CREDIT***, and ***POSTAUTH***, this field must be a valid Order ID from a prior ***SALE*** or ***PREAUTH*** transaction. For a Forced Ticket (that is a **POSTAUTH** where the authorization was given over the phone), the **oid** field is not required, but the **reference_number** field is required.	Required for credit card ***VOID***, ***CREDIT***, and ***POSTAUTH*** transactions

ponumber	string	The purchase order number, department number, or other customer-supplied number that applies to this transaction. One P.O. number can apply to multiple orders.	Required for Level 2 purchasing card transactions
taxexempt	string	An indicator field representing whether the order is tax exempt. It should be set to Y if the order is tax exempt, or N if not.	Required for Level 2 purchasing card transactions
terminaltype	string	The type of terminal that is sending the transaction. Set the value to ***STANDALONE*** for a point-of-sale credit card terminal, ***POS*** for an electronic cash register or integrated POS system, ***UNATTENDED*** for a self-service station such as a gas station terminal, vending machine, self-service ticket booth or other self checkout station, or ***UNSPECIFIED*** for e-commerce, general, CRT, or other applications.	Not required, but transactions may be downgraded if not passed properly. Default is ***UNSPECIFIED***.
ip	string	The IP address of the consumer	Not required, but needed for fraud blocking by IP address.
reference_number	string	Used for Forced Ticket transactions where a reference number was obtained over the phone. Set the reference number value to the word *NEW* (all caps) + the reference number given over the phone. For example, if the reference number given over the phone was ***123456***, you would set **reference_number** to ***NEW123456***.	Required for Forced Ticket transactions only

recurring	string	A flag to indicate whether the transaction is a recurring transaction or not. If this transaction is recurring (and if the merchant is not using the recurring processor provided here), this flag must be set to ***Yes***.	Defaults to ***No***. Set to ***Yes*** if the transaction is part of a recurring series of transactions and you are NOT using the periodic billing option of the API.
tdate	string	This field is returned with every successful transaction. If you need to run a void (or other transaction) against an existing order ID, you may need to pass the tdate to uniquely identify the specific transaction you wish to void. See below for example.	Not required, but may be needed for some transactions. See example below.

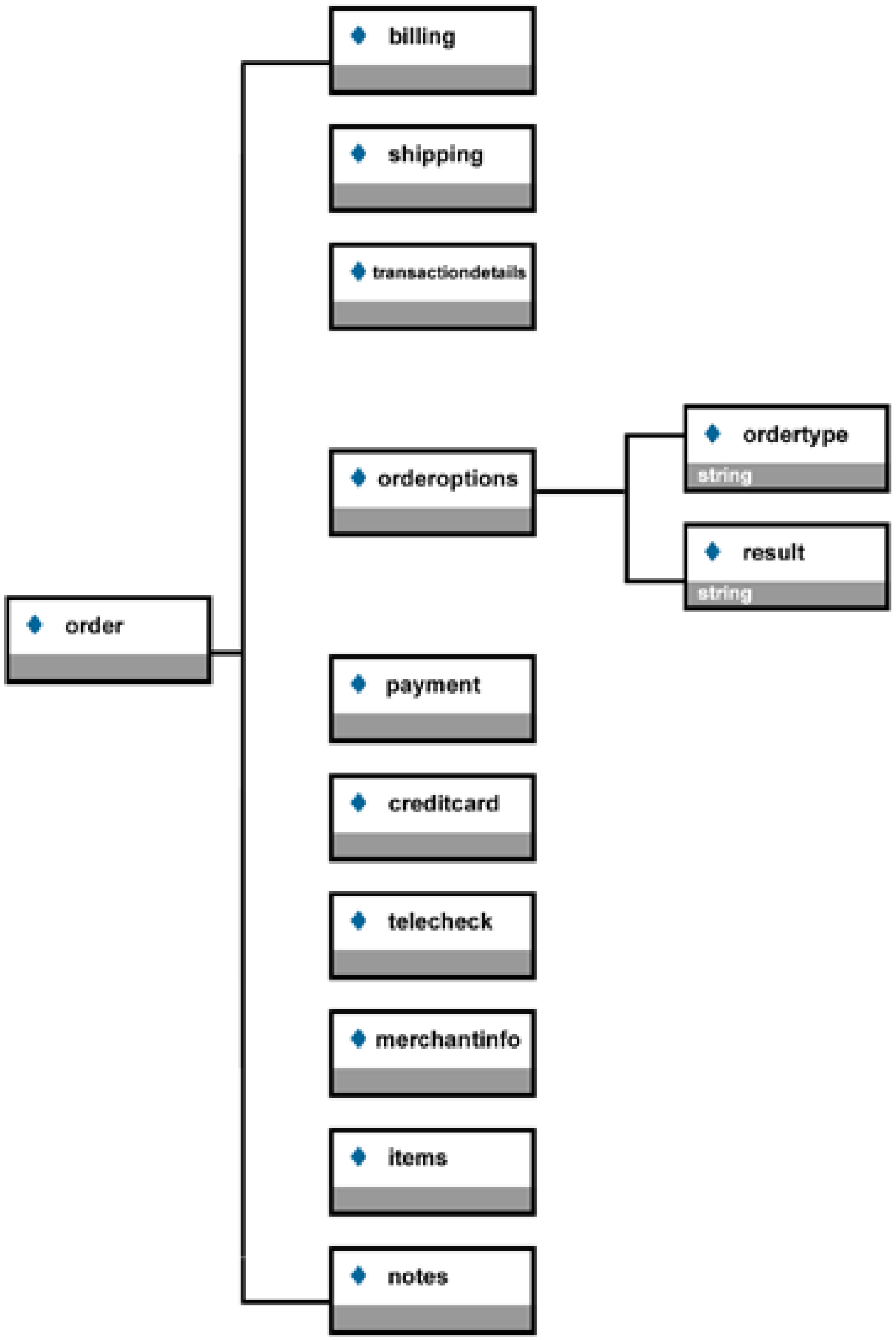
A tdate example

1. Say you run a Sale transaction for $10. The order ID (or oid) for your Sale transaction is 12309. The tdate returned is 9023.
2. Then you run a Credit transaction for $5 against the same order ID (12309) to return a portion of the customer's money. The tdate returned for the Credit transaction is 2134.
3. But just after you've run the Credit, you realize you made a mistake--you didn't need to credit the transaction.
4. So now you need to void the Credit, but if you just used the Order ID (12309), the Gateway would void the entire order (both the Sale AND the Credit).
5. To void ONLY the Credit, you must pass the ***tdate*** for the Credit transaction (2134) in addition to the ***oid*** (12309). The ***tdate*** uniquely identifies the Credit transaction as the transaction you wish to void.

The Order Options Entity

The **orderoptions** entity contains specific instructions on how to process the order. The data fields it can contain are ***ordertype*** and ***result***.

The diagram below shows the **orderoptions** entity data structure.



Order Options Data Fields

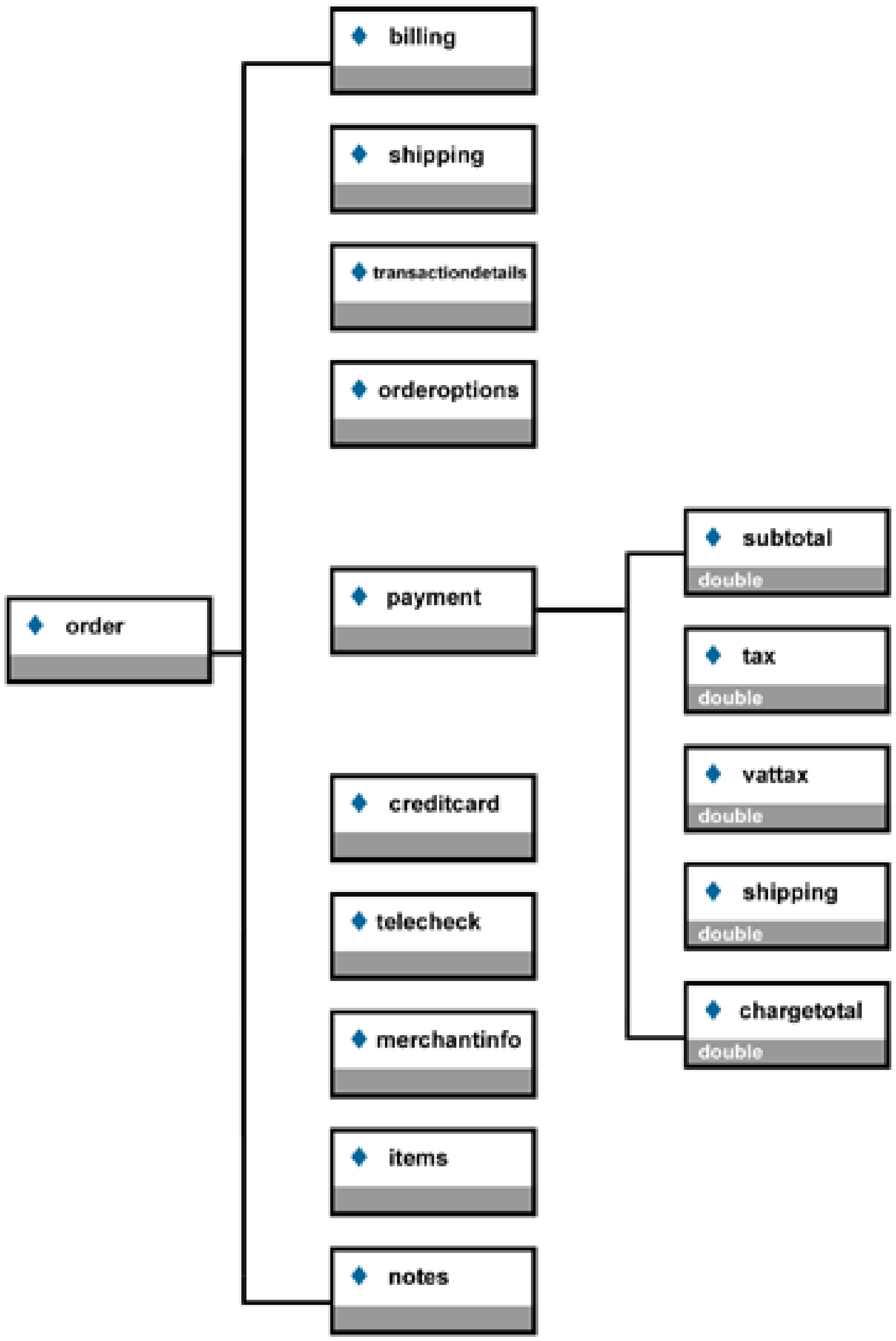
The table below describes each data field in the **orderoptions** entity.

orderoptions Field Name	Data Type	Description	Required?
ordertype	string	The type of transaction. The possible values are ***SALE***, ***PREAUTH*** (for an Authorize Only transaction), ***POSTAUTH*** (for a Forced Ticket or Ticket Only transaction), ***VOID***, ***CREDIT***, ***CALCSHIPPING*** (for shipping charges calculations) and ***CALCTAX*** (for sales tax calculations). See the Transaction Types section for additional information.	Required
result	string	This field puts the account in live mode or test mode. Set to ***LIVE*** for live mode, ***GOOD*** for an approved response in test mode, ***DECLINE*** for a declined reponse in test mode, or ***DUPLICATE*** for a duplicate response in test mode.	Not required. Defaults to LIVE.

The Payment Entity

The **payment** entity contains the transaction totals. The data fields available are subtotal, tax, vattax, shipping, and chargetotal.

The diagram below shows the **payment** entity data structure.



Payment Data Fields

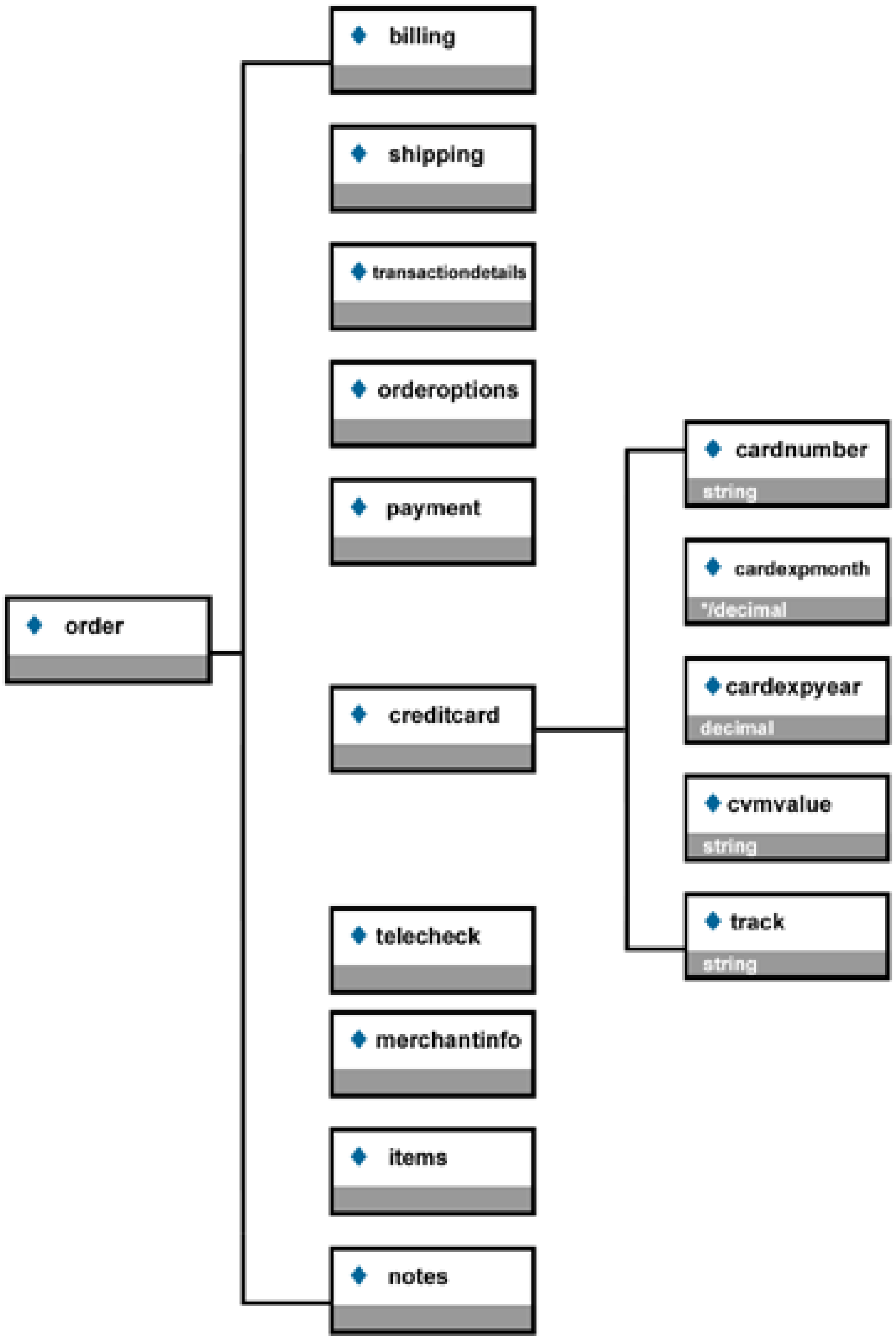
The table below describes each data field in the **payment** entity.

payment Field Name	Data Type	Description	Required?
subtotal	double	The transaction subtotal, before shipping and tax	Not required
tax	double	Sales tax dollar amount	Required for Level 2 purchasing card transactions only. If tax is $0.00 for a purchasing card transaction, set this field to 0.
vattax	double	Amount of VAT (typically, only tax or vattax is used). This is a tax sometimes applied to international orders.	Not required
shipping	double	Shipping charge dollar amount	Not required
chargetotal	double	The total dollar amount of this transaction, including subtotal, tax, and shipping	Required

The Credit Card Entity

The **creditcard** entity contains details about the credit card order. The data fields available are cardnumber, cardexpmonth, cardexpyear, cvmvalue, cvmindicator, and track.

The diagram below shows the **creditcard** entity data structure.



Credit Card Data Fields

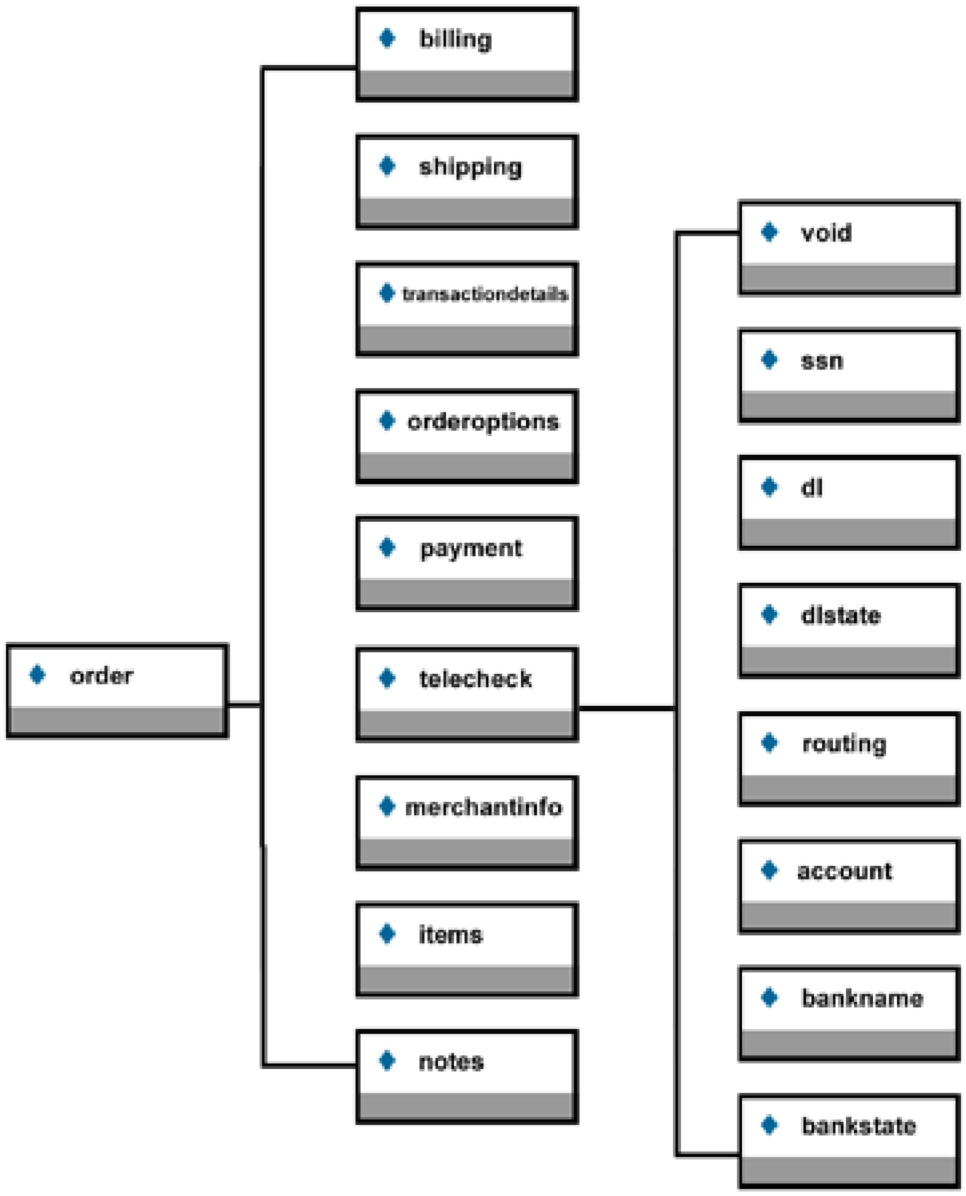
The table below describes each data field in the **creditcard** entity.

creditcard Field Name	Data Type	Description	Required?
cardnumber	numeric	The consumer's credit card number	Required for credit card transactions if ***track*** is not provided
cardexpmonth	integer from 1 to 12	The numeric expiration month of the credit card	Required for credit card transactions if ***track*** is not provided
cardexpyear	integer from 00 to 99	The two-digit expiration year of the credit card	Required for credit card transactions if ***track*** is not provided
cvmvalue	integer from 000 to 999	3-digit numeric valued typically printed on the signature panel on the back of the credit card. See Card Codes section for more information.	Not required, but recommended for fraud prevention
cvmindicator	string	Indicates whether CVM was supplied and, if not, why. The possible values are “provided”, “not_provided”, “illegible”, “not_present”, and “no_imprint”.	Required if ***cvmvalue*** is passed
track	string	Contains the data swiped from the credit card track 1 or track 2.	Required for credit card transactions if ***cardnumber***, ***cardexpmonth*** and ***cardexpyear*** are not provided

The TeleCheck Entity

The telecheck entity contains information required by VirtualCheck to process the check payment. The data fields are void, ssn, dl, dlstate, routing, account, bankname and bankstate.

The diagram below shows the **telecheck** entity data structure.



VirtualCheck Data Fields

The table below describes each data field in the **telecheck** entity. The **telecheck** entity contains the data fields needed for a VirtualCheck transaction.

In addition to the fields listed as required below, if the order was taken over the telephone, the following fields from the **billing** entity are also required.

- name
- address1
- city
- state
- zip
- phone

If this order is NOT an e-commerce order (i.e., if it was a retail, mail or telephone order), you must also pass the **transactionorigin** data field in the **transactiondetails** entity.

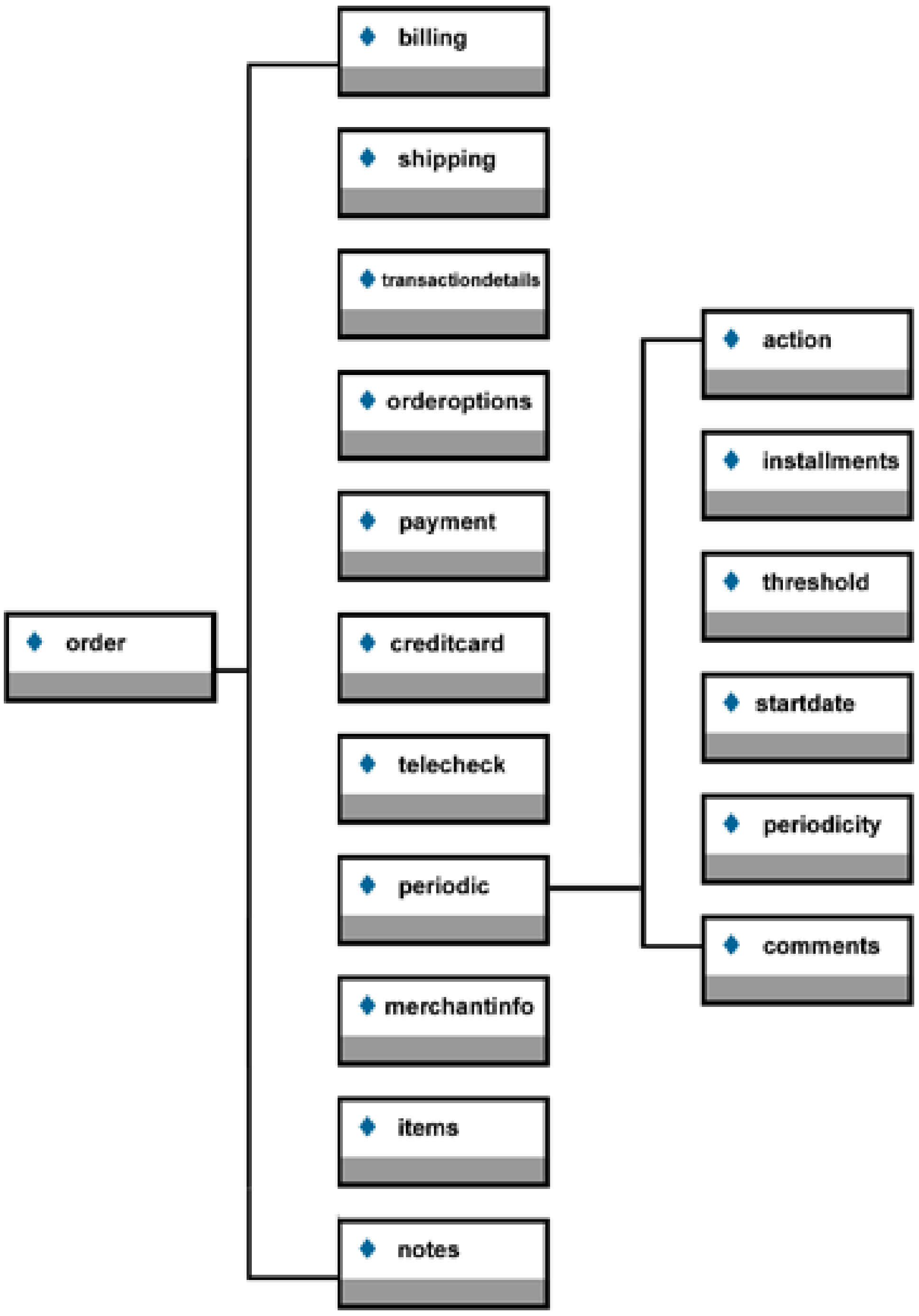
telecheck Field Name	Data Type	Description	Required?
routing	string	The transit routing number for the customer’s bank	Required for all check transactions
account	string	The customer's bank account number	Required for all check transactions
checknumber	string	The check number from the customer's check. A VirtualCheck transaction does not use the check number; i.e., the customer can still use the paper check for another purchase.	Optional
bankname	string	The name of the customer’s bank	Required
bankstate	string	The state that the customer’s bank is located in. Set the value to one of the US State Codes.	Required
dl	string	Driver’s license number of the customer	Required for check orders taken over the telephone or the web
dlstate	string	The state where the customer’s driver's license was issued. Set the value to one of the US State Codes.	Required for check orders taken over the telephone or the web

void	integer	To void an unsettled check, pass a 1 in this tag. Otherwise, do not include this data field.	Required for check VOID transactions
accounttype	string	The type of account used for the purchase. Possible values are: ● "pc" for personal checking ● "ps" for personal savings ● "bc" for business checking ● "bs" for business savings	Required for all VirtualCheck SALES
ssn	string	Social security number of the customer	Optional, for identity verification purposes

The Periodic Entity

The **periodic** entity contains information about a recurring transaction. The data fields are action, installments, threshold, startdate, periodicity, and comments. Currently, recurring transactions are only applicable to credit card payments—check payments cannot be made recurring with this entity.

The diagram below shows the **periodic** entity data structure.



Periodic Data Fields

The table below describes each data field in the **periodic** entity.

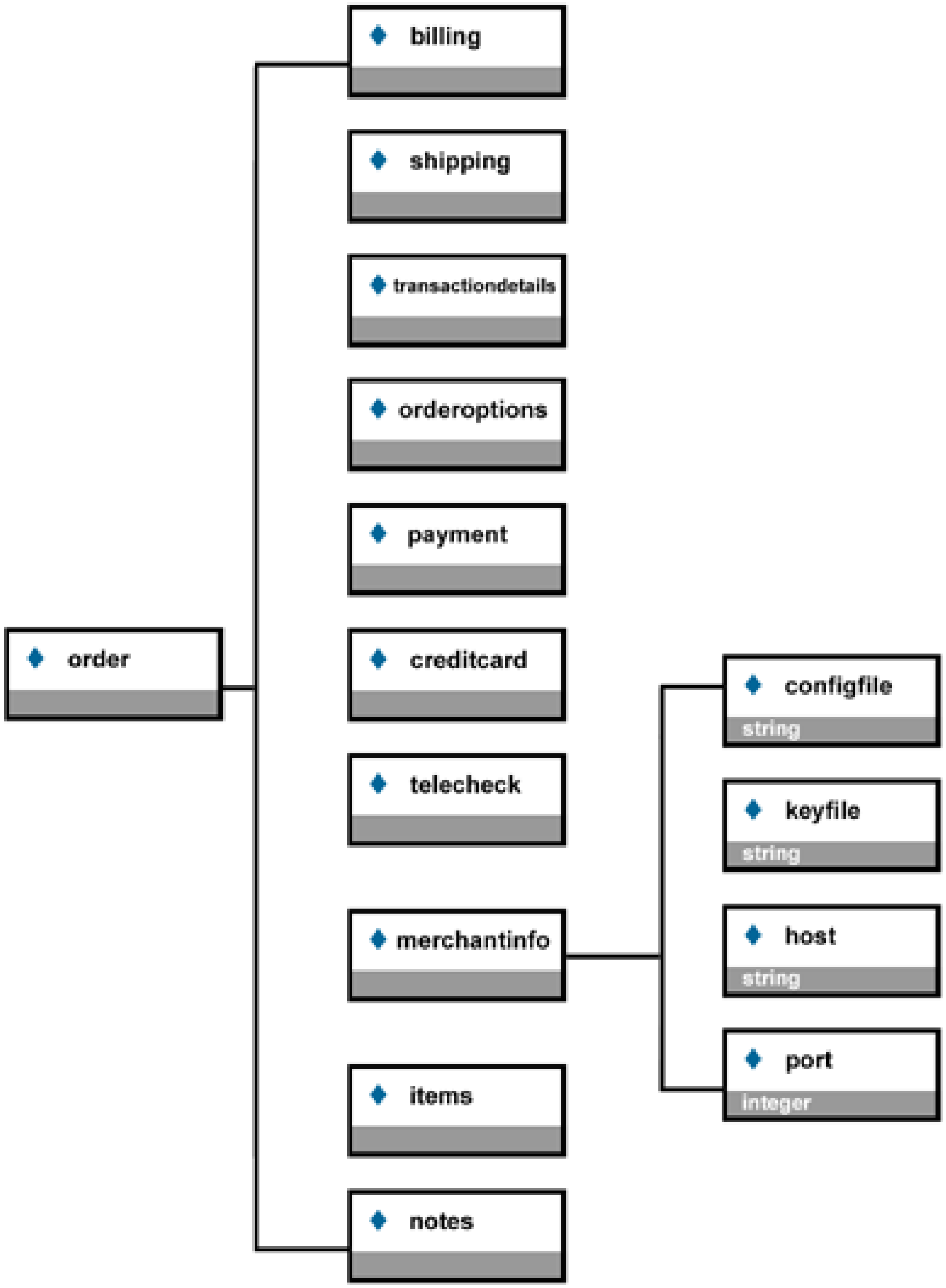
periodic Field Name	Data Type	Description	Required?
action	string	Tells which action to take regarding the periodic transaction. Possible values are SUBMIT (to submit the recurring transaction for processing), MODIFY (to edit a previously submitted recurring transaction), or CANCEL (to cancel a previously submitted recurring transaction).	Required for all recurring transactions
installments	integer from 0 to 99	Identifies how many recurring payments to charge the customer	Required for all recurring transactions
threshold	integer from 1 to 5	Tells how many times to retry the transaction (if it fails) before contacting the merchant.	Not required. If not supplied, it defaults to 3.
startdate	string	Tells the date to begin charging the recurring payments. Should be in the format YYYYMMDD, where YYYY is the four-digit year, MM is the 2-digit month, and DD is the 2-digit day. Example: 20040910 means September 10, 2004. If you wish to start payments immediately, set this tag to ***immediate***	Required for all recurring transactions
periodicity	string	Tells how often to charge the payment. Possible values are: ● ***monthly*** (to charge once a month), ● ***bimonthly*** (to charge every other month), ● ***weekly*** (to charge once a week), ● ***biweekly*** (to charge once every other week), ● ***yearly*** (to charge once a year), ● ***daily*** (to charge once a day), ● or you can use the code ***XN***, which means every ***N*** days/weeks/months. ***X*** can be ***d*** for day, ***m*** for month, or ***y*** for year. ***N*** can be any integer from 1 through 999. Example: ***m3*** means	Required for all recurring transactions

		charge the customer once every 3 months.	
comments	string	Lets you enter optional comments about the transaction	Not required

The Merchant Info Entity

The **merchantinfo** entity contains details about the merchant. This entity is required for every transaction or function call to the Gateway. The four merchantinfo data fields are configfile (i.e., the merchant store number), keyfile, host, and port.

The diagram below shows the **merchantinfo** data structure.



Merchant Info Data Fields

The table below describes each data field in the **merchantinfo** entity.

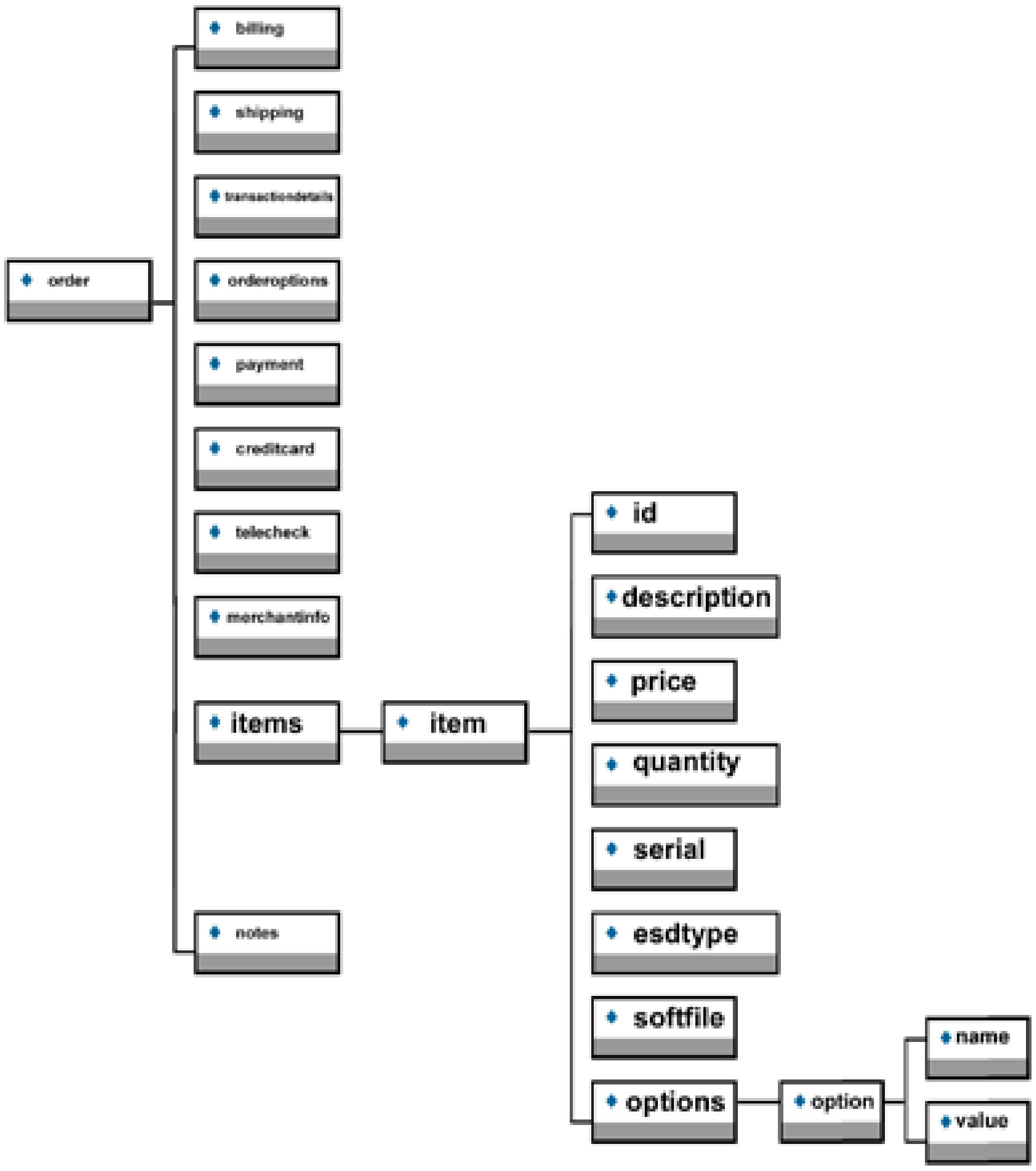
merchantinfo Field Name	Data Type	Description	Required?
configfile	string	This field should contain the merchant store name or number, which is generally a six- to ten-digit number assigned at merchant account setup. If this is a test account, the store name may be a text string.	Required for ALL transactions and function calls.
keyfile	string	This field contains the path and filename of the digital certificate (or PEM file) issued for a given store.	Not required. Information is provided to the Gateway when setting up the SSL connection.
host	string	This is the URL or IP address of the Gateway Server. This information is provided in the merchant's Welcome e-mail.	Not required. Information is provided to the Gateway when setting up the SSL connection.
port	integer	The port that the application uses to communicate to the Payment Gateway. Used only on the client side.	Not required. Information is provided to the Gateway when setting up the SSL connection.

The Items Entity

The **items** entity contains one or more **item** sub-entities.

Each **item** sub-entity contains up to seven data fields (id, description, price, quantity, serial, esdtype, softfile) and may also contain an **options** entity.

The **options** sub-entity can then contain one or more **option** sub-entities, as reflected in the following diagram. Each option sub-entity can contain two data fields (name and value).



An Items Example

The items entity can contain one or more item sub-entities. Each **item** sub-entity can contain up to seven data fields, plus an **options** sub-entity. Each **options** sub-entity can contain one or more **option** entities, which can each contain two data fields. This structure is shown below.

items entity
one or more **item** sub-entities [7 data fields]
options sub-entity
one or more **option** sub-entities [2 data fields]

A Sample Order

Now let's say you are a merchant selling bowling-related products. You might receive an order like this:

1. 2 T-shirts. Red. XL.
2. 1 Bowling Ball. Weight: 12 lb.
3. 1 eBook on How to Bowl

To describe this order using the **items** entity, you might describe the order as shown in the diagram below.

Data Fields for Each Item

All the data fields used in the **item** sub-entity are listed in the table below.

item sub-entity			
item Field Name	Data Type	Description	Required?
id	string	Item ID number	Required for each item passed
description	string	Description of this item	Required for ESD items. Optional for other items.
price	string	Price of this item	Required for each item passed
quantity	string	Quantity of this item to include in this order	Required for each item passed
serial	string	Serial number associated with this item	Not required
esdtype	string	For electronic softgood download (ESD) items, this tag indicates which type of ESD item it is: set the value to ***softgood*** if it's a softgood file or ***key*** if it's a keygen executable.	Required for ESD items only
softfile	string	The name of the softgood file or keygen executable to be downloaded	Required for ESD items only

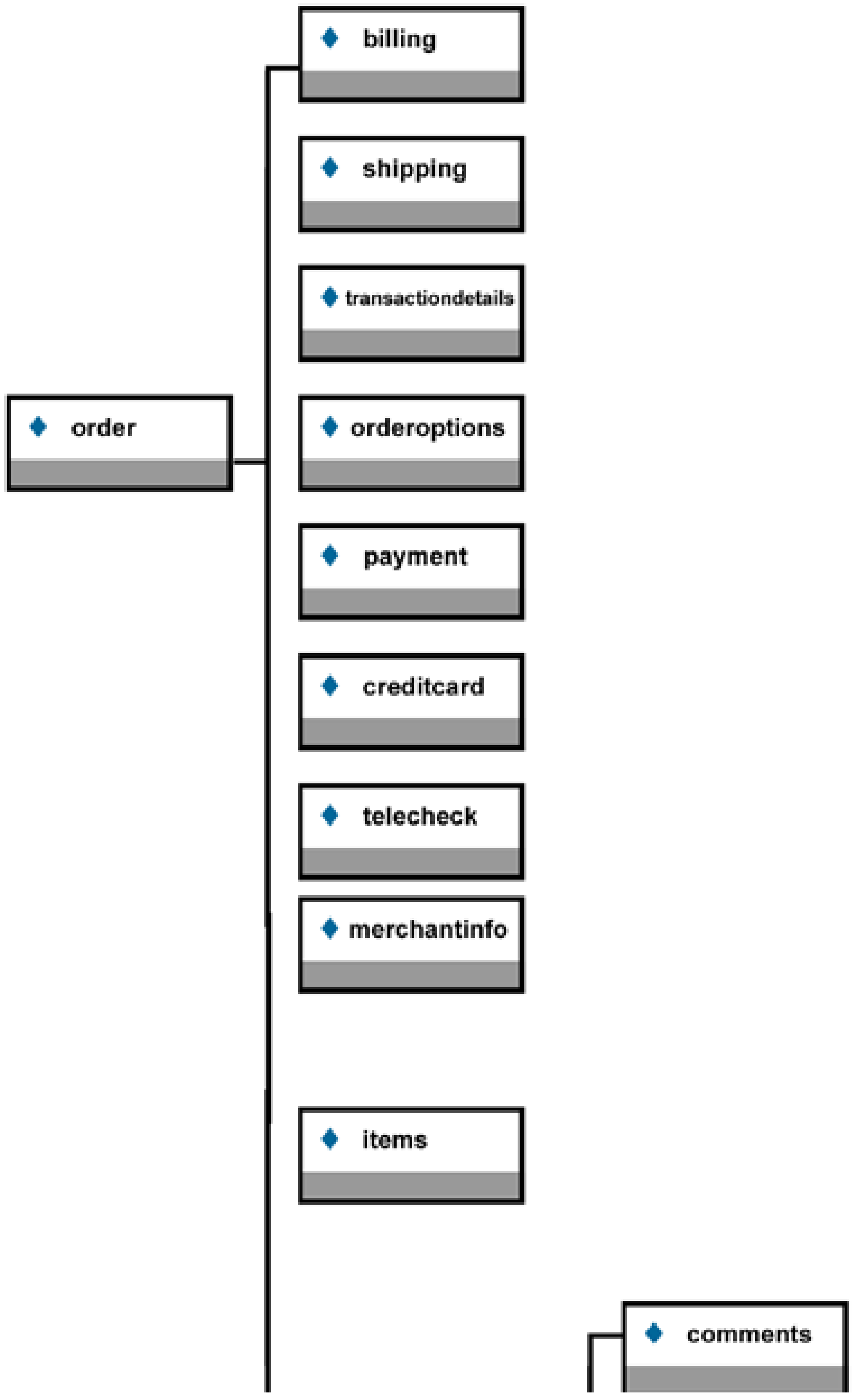
Data Fields for Each Option

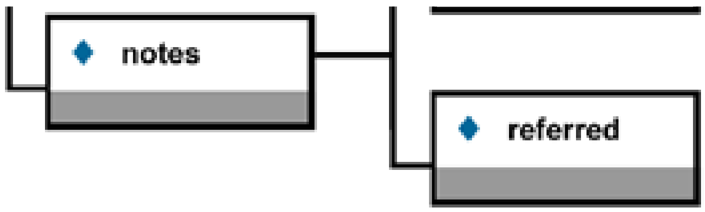
All the data fields used in the **option** sub-entity are listed in the table below.

option sub-entity			
option Field Name	Data Type	Description	Required?
name	string	Name of the option	Required for each option passed
value	string	Value of the option for this order	Required for each option passed

The Notes Entity

The **notes** entity contains optional comments about the transaction. It may contain two data fields, comments and referred, as reflected in the following diagram.




notes
referred

Notes Data Fields

The table below describes each data field in the **notes** entity.

notes Field Name	Data Type	Description	Required?
comments	string	For optional comments about the transaction	Not required
referred	string	Intended for merchants to track how the customer was referred to them. Often a URL is passed in this field.	Not required

US State Codes

When passing a value for a US State in the **billing** or **shipping** entity, you must use the 2-digit codes specified below.

A	Alabama	AL	Alaska	AK
	Arizona	AZ	Arkansas	AR

C	California	CA	Colorado	CO
	Connecticut	CT		

D	Delaware	DE	District of Columbia	DC

F	Florida	FL	

G	Georgia	GA	

H	Hawaii	HI	

I	Idaho	ID	Illinois	IL
	Indiana	IN	Iowa	IA

K	Kansas	KS	Kentucky	KY

L	Louisiana	LA	

M	Maine	ME	Maryland	MD
	Massachusetts	MA	Michigan	MI
	Minnesota	MN	Mississippi	MS
	Missouri	MO	Montana	MT

N	Nebraska	NE	Nevada	NV
	New Hampshire	NH	New Jersey	NJ
	New Mexico	NM	New York	NY
	North Carolina	NC	North Dakota	ND

O	Ohio	OH	Oklahoma	OK
	Oregon	OR		

P	Pennsylvania	PA	

R	Rhode Island	RI	

S	South Carolina	SC	South Dakota	SD

T	Tenessee	TN	Texas	TX

U	Utah	UT	

V	Vermont	VT	Virgina	VA

W	Washington	WA	Wisconsin	WI
	West Virginia	WV	Wyoming	WY

Country Codes

Whenever passing a value for **country** in the **shipping** or **billing** entity, you must use the following two-letter abbreviations. Countries are grouped by these continents:

- Africa
- Antarctica
- Asia
- Australia
- Caribbean
- Central America
- Europe
- Middle East
- North America
- South America
- Others

AFRICA					
Algeria	DZ	Benin	BJ	Burkina Faso	BF
Burundi	BI	Cameroon	CM	Cape Verde	CV
Central African Republic	CF	Chad	TD	Comoros	KM
Congo	CG	Cote D'Ivoire	CI	Djibouti	DJ
Equatorial Guinea	GQ	Eritrea	ER	Ethiopia	ET
Egypt	EG	Gabon	GA	Gambia	GM
Ghana	GH	Guinea	GN	Guinea-Bissau	GW
Kenya	KE	Lesotho	LS	Liberia	LR

Madagascar	MG	Mali	ML	Mauritania	MR
Mayotte	YT	Morocco	MA	Mozambique	MZ
Malawi	MW	Namibia	NA	Niger	NE
Nigeria	NG	Reunion	RE	St. Helena	SH
Sao Tome and Principe	ST	Senegal	SN	Sierra Leone	SL
Somalia	SO	South Africa	ZA	Sudan	SD
Swaziland	SZ	Tanzania	TZ	Togo	TG
Uganda	UG	Western Sahara	EH	Zaire	ZR
Zambia	ZM	Zimbabwe	ZW		

ANTARCTICA		
Antarctica	AQ	

ASIA					
Afghanistan	AF	Bangladesh	BD	Bhutan	BT
Brunei	BN	Cambodia	KH	China	CN
Hong Kong	HK	India	IN	Indonesia	ID
Japan	JP	Kazakhstan	KZ	Kyrgyzstan	KG
Laos	LA	Macau	MO	Malaysia	MY

Maldives	MV	Mongolia	MN	Nepal	NP
Pakistan	PK	Philippines	PH	Republic of Korea	KR
Russia	RU	Seychelles	SC	Singapore	SG
Sri Lanka	LK	Taiwan	TW	Tajikistan	TJ
Thailand	TH	Turkmenistan	TM	Uzbekistan	UZ
Vietnam	VN				

AUSTRALIA					
American Samoa	AS	Australia	AU	Federated States of Micronesia	FM
Fiji	FJ	French Polynesia	PF	Guam	GU
Kiribati	KI	Marshall Islands	MH	Nauru	NR
New Caledonia	NC	New Zealand	NZ	Northern Mariana Islands	MP
Palau	PW	Papua New Guinea	PG	Pitcairn	PN
Solomon Islands	SB	Tonga	TO	Tuvalu	TV
Vanuatu	VU				

CARIBBEAN					
Anguilla	AI	Antiqua and Barbuda	AG	Aruba	AW

Bahamas	BS	Barbados	BB	Bermuda	BM
British Virgin Islands	VI	Cayman Islands	KY	Dominica	DM
Dominican Republic	DO	Grenada	GD	Guadeloupe	GP
Haiti	HT	Jamaica	JM	Martinique	MQ
Netherlands Antilles	AN	Puerto Rico	PR	St. Kitts and Nevis	KN
St. Lucia	LC	St. Vincent and the Grenadines	VC	Trinidad and Tobago	TT
Turks and Caicos Islands	TC				

CENTRAL AMERICA					
Belize	BZ	Costa Rica	CR	El Salvador	SV
Guatemala	GT	Honduras	HN	Nicaragua	NI
Panama	PA				

EUROPE					
Albania	AL	Andorra	AD	Armenia	AM
Austria	AT	Azerbaijan	AZ	Belarus	BY
Belgium	BE	Bulgaria	BG	Croatia	HR
Cyprus	CY	Czech Republic	CZ	Denmark	DK

Estonia	EE	Faroe Islands	FO	Finland	FI
France	FR	Georgia	GE	Germany	DE
Gibraltar	GI	Greece	GR	Greenland	GL
Hungary	HU	Iceland	IS	Ireland	IE
Italy	IT	Latvia	LV	Liechtenstein	LI
Lithuania	LT	Luxembourg	LU	Malta	MT
Metropolitan France	FX	Moldova	MD	Netherlands	NL
Norway	NO	Poland	PL	Portugal	PT
Romania	RO	Slovakia	SK	Slovenia	SI
Spain	ES	Svalbard and Jan Mayen Islands	SJ	Sweden	SE
Switzerland	CH	The former Yugoslav Republic of Macedonia	MK	Turkey	TR
Ukraine	UA	United Kingdom	GB	Vatican City	VA
Yugoslavia	YU				

MIDDLE EAST					
Israel	IL	Jordan	JO	Kuwait	KW
Lebanon	LB	Oman	OM	Qatar	QA

Saudi Arabia	SA	Syria	SY	United Arab Emirates	AE
Yemen	YE				

NORTH AMERICA					
Canada	CA	Mexico	MX	United States	US

SOUTH AMERICA					
Argentina	AR	Bolivia	BO	Brazil	BR
Chile	CL	Colombia	CO	Equador	EC
Falkland Islands	FK	French Guiana	GF	Guyana	GY
Paraguay	PY	Peru	PE	Peru	PE
Suriname	SR	Uruguay	UY	Venezuela	VE

OTHERS					
Bahrain	BH	Bouvet Islands	BV	British Indian Ocean Territory	IO

Response Fields

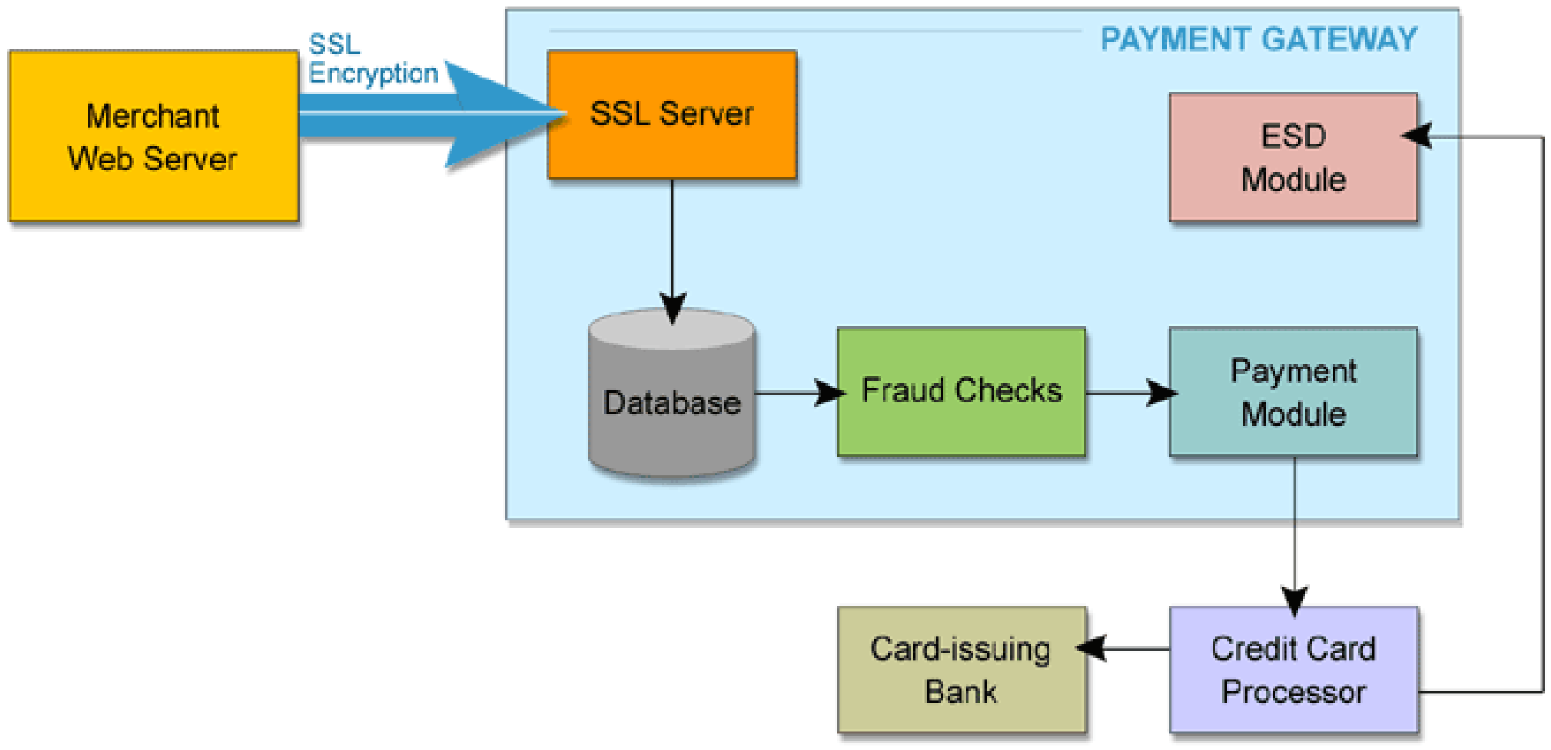
The Gateway's transaction engine will return a set of response codes for each transaction that has successfully communicated with the gateway through the SSL communications link. The response codes are returned in a set of XML tags, which are prefaced with an "r_".

The following table represents the possible set of response tags. Note that if an error occurs, the response may contain only the **r_error** tag, although it is possible that other tags will be returned if additional information is available.

Response Field	Description
r_avs	The Address Verification System (AVS) response for this transaction. The first character indicates whether the contents of the **addrnum** tag match the address number on file for the billing address. The second character indicates whether the billing zip code matches the billing records. The third character is the raw AVS response from the card-issuing bank. The last character indicates whether the **cvmvalue** was correct and may be "M" for Match, "N" for No Match, or "Z" if the match could not determined. See the sections entitled *Using Address Verification* and *Using the Card Code* for additional information on using this information to help you combat fraud.
r_ordernum	The order number associated with this transaction.
r_error	Any error message associated with this transaction.
r_approved	The result of the transaction, which may be APPROVED, DECLINED, or FRAUD.
r_code	The approval code for this transaction.
r_message	Any message returned by the processor; e.g., "CALL VOICE CENTER".
r_time	The time and date of the transaction server response.
r_ref	The reference number returned by the credit card processor.
r_tdate	A server time-date stamp for this transaction. Used to uniquely identify a specific transaction where one order number may apply to several individual transactions. See the *Transaction Details Data Fields* section for further information and an example of tdate.
r_tax	The calculated tax for the order, when the ordertype is ***calctax***.
r_shipping	The calculated shipping charges for the order, when the ordertype is ***calcshipping***.

Where and Why Errors Occur

There are many different response messages or codes you may receive from the Gateway. All response error or decline messages are passed in the **r_error** data field returned by the Gateway. Some responses indicate an error; some are returned with a declined transaction as explanation for the decline. To understand the error messages you get, it helps to understand the process a transaction goes through when you submit it to the gateway.



The Transaction Process

The transaction process follows this general path. This description assumes an e-commerce application, but the general transaction flow is roughly the same for retail or mail order/telephone order (MO/TO) transactions.

1. The merchant's or commerce service provider's Web server sends transaction information submitted by a customer at an online store to the SSL server.
2. The SSL server decrypts the data and writes it to the database.
3. If the shipping or tax calculators are used, they are invoked immediately after the transaction information is received by the SSL server.
4. Next, fraud and error checks are performed on the data.
5. If the information passes the fraud and error checks, the transaction is passed on to the payment lease line to the credit card processor.
6. If the ESD module is used, the ESD process is invoked when the transaction is passed to the credit card processor.
7. Once the credit card processor responds, successful transaction information is run back through the gateway, ending with the transaction being logged to the database and verification being sent to the customer and merchant.

Throughout this process, however, errors and/or declines can occur at any point. Error codes are generated when transactions cannot be completed because of machine error, incomplete data types or connection/transmission problems.

Decline values result when a transaction is declined for any reason, including when a transaction fails a fraud check. Error codes and decline values are returned in the **r_error** field. We have broken up the error messages into categories. If you are encountering an error message, please see the associated *Errors* section of this document.

Fraud and Error Checks

Fraud and error checks are performed by the Gateway Fraud Protection module. Fraud messages and declines occur when the information submitted by the customer is either incomplete/incorrect or because the information has failed one of the Gateway fraud checks. See the *Fraud Decline Messages* section for further information on what you may encounter when a transaction is declined because of fraud.

Payment Handler Interface

Unless it is halted because an error is found, transaction processing starts in the Gateway server, passes to the payment handler, then to a card processor, then to a card-issuing bank, then all the way back through the same stages. Successful or not, every transaction ultimately produces a return value, which may indicate that the transaction completed and was approved, completed and was declined or did not complete because it failed a specific error check at any point in the process.

See the following *Errors* sections for further information on messages or codes that may be returned by the Gateway.

Merchant Information Errors

These types of errors occur when the merchant information sent to the Gateway is incomplete or erroneous or when additional setup is required on the Gateway that has not yet been completed. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
Merchant ID number not specified. or **ReqErr_InvalidConfigfile**	Valid store number (storename) is required for all transactions.	Make sure you are passing the store number (storename) provided in the merchant's Welcome e-mail.	**configfile**	**merchantinfo**
ReqErr_InvalidKeyfile	The merchant's digital certificate filename or path is not found.	Check the location of the certificate file (.PEM file) and compare against the path and file name provided when setting up the SSL connection to the Gateway.	**--**	**--**
ReqErr_InvalidHost	The host name was not supplied or is invalid.	Check the host name provided when setting up the SSL connection to the Gateway.	**--**	**--**
ReqErr_InvalidPort	The port number was not supplied or is invalid.	Check the port number provided when setting up the SSL connection to the Gateway. It should be 1129.	**--**	**--**

ConfigErr_Merchant	Unable to open merchant configuration file or the merchant configuration file does not exist.	Ensure you have an active store on the Gateway. Contact support and ask them to check your config file.	**--**	**--**
ConfigErr_Shipping	Unable to open shipping configuration file or the merchant's shipping file does not exist.	Ensure you have provided a shipping file or shipping parameters to Support and that there is a valid, accessible shipping file on the Gateway for this merchant.	**--**	**--**
ConfigErr_Missing	There is information missing in the merchant's configuration file on the Gateway.	Ensure you have an active store on the Gateway. Contact support, ask them to check the information in your config file and ensure that all required data is in the file.	**--**	**--**

ConfigErr_Certificate	Unable to open or parse the merchant's digital certificate file.	Ensure you properly copied the certificate provided in the merchant's Welcome e-mail and saved it with a .PEM extension. If you are sure the certificate is correct, contact support for further assistance.	**--**	**--**
AuthErr_Certificate	There is an invalid contact name in the merchant's digital certificate.	Check the contact name for the certificate.	**--**	**--**

Credit Card Related Errors

These errors occur when incomplete or erroneous credit card information is passed for a credit card transaction. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
Credit card expiration month must be selected. or **OrderErr_Invalid_Expmonth**	Either no expiration month for the credit card was received or the value received was outside of the allowed range.	Ensure you are passing a 2-digit number from 01 through 12.	**cardexpmonth**	**creditcard**
Credit card expiration year must be selected. or **OrderErr_InvalidExpyear**	Either no expiration year for the credit card was received or the value received was outside of the allowed range.	Ensure you are passing a 2-digit number from 00 through 99.	**cardexpyear**	**creditcard**
Credit card number must be filled in. or **OrderErr_NoCardnumber**	No credit card number was received by the Gateway.	Ensure you are passing a nonempty credit card number. Check the validity of the number. Spaces and/or dashes are optional.	**cardnumber**	**creditcard**

This is not a valid credit card. Please try another card. or **OrderErr_InvalidCardnumber**	The card number isn't valid.	Prompt the customer to re-enter the card number or try a different card.	**cardnumber**	**creditcard**
Card number has too few digits. or **OrderErr_CardnumberShort**	The credit card number provided has fewer digits than allowed for that card type.	Check the number of digits for each credit card to make sure it's the correct length or prompt the customer to re-enter the card number.	**cardnumber**	**creditcard**
Card number has too many digits. or **OrderErr_CardnumberLong**	The credit card number provided has more digits than allowed for that card type.	Check the number of digits for each credit card to make sure it's the correct length or prompt the customer to re-enter the card number.	**cardnumber**	**creditcard**
Credit card number must be filled in.	No credit card number was provided for a credit card transaction.	Ensure that the **creditcard** field is not empty when submitting a credit card transaction to the Gateway.	**cardnumber**	**creditcard**

This credit card appears to have expired.	The expiration date provided is in the past.	Compare the expiration date provided with the current month and year. If it occurs in the past, prompt the customer that the credit card appears to have expired.	**cardexpmonth** and **cardexpyear**	**creditcard**
OrderErr_UnsupportedCardType	The credit card type is not supported.	Check which credit card types the merchant supports. The Gateway supports Visa, MasterCard, Diner's Club, JCB, Discover, and American Express. Ask the customer to choose a credit card of a type that is supported by both the merchant and the Gateway.	**cardnumber**	**creditcard**

OrderErr_CardnumberMismatch	The credit card number does not match. This may occur when doing a credit, post-auth, or void transaction where the credit card number supplied in the second transaction does not match the card number provided in the first transaction.	Check the card number in the original transaction and make sure you are populating the **cardnumber** field correctly.	**cardnumber**	**creditcard**

Customer Information Errors

These error responses occur when the customer information supplied for a given transaction is incomplete or erroneous. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
Invalid e-mail address.	E-mail address provided is not in the right format.	Ensure you are passing a valid e-mail address in the format someone@domain.com.	**email**	**billing**
E-mail address must be filled in. or **OrderErr_InvalidEmail**	A valid e-mail address is required for this transaction.	Ensure you are passing a valid e-mail address in the format someone@domain.com.	**email**	**billing**
The ZIP code given does not match up with the city and state given.	The zip code appears to be in a different city and state than the one specified.	Prompt the customer to re-enter the zip code or check the city and state.	**zip**	**billing** or **shipping**
ZIP/postal code not found in state.	The zip code provided is not valid for the US state value received.	Check the values for state and zip code. Prompt the customer to re-check the data entered.	**zip**	**billing** or **shipping**
State must be selected.	State is a required field for this transaction.	Make sure you are passing a valid state code.	**state**	**billing** or **shipping**

Name must be filled in.	Customer's name is required for this transaction.	Make sure a value is passed in the name field.	**name**	**billing**
IP address given does not match up with the city and state given. or **OrderErr_InvalidIP**	IP address or location is invalid.	Check that you are passing a valid customer IP address.	**ip**	**transactiondetails**
OrderErr_InvalidBname	Customer's name is required for this transaction.	Make sure a value is passed in the **name** field.	**name**	**billing**
OrderErr_InvalidBzip	Customer's billing zip code is required for this transaction.	Make sure a valid value is passed in the zip code field.	**zip**	**billing**

Order-Related Errors

These error responses occur when the order data is incomplete, erroneous, or when follow-on transactions for the same order do not match up with previous transactions. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
OrderErr_InvalidChargetype	Order charge type specified is not valid	Make sure a valid value is passed in the **ordertype** field. The possible values are SALE, PREAUTH (for an Authorize Only transaction), POSTAUTH (for a Forced Ticket or Ticket Only transaction), VOID, CREDIT, CALCSHIPPING (for shipping charges calculations) and CALCTAX (for sales tax calculations).	**ordertype**	**orderoptions**
OrderErr_InvalidResult	Order result specified is not valid	Make sure a valid value is passed in the **result** field. Set to LIVE for live mode, GOOD for an approved response in test mode, DECLINE for a declined reponse in test mode, or DUPLICATE for a duplicate response in test mode. If you are not testing (that is, if you are running your store "live"), you do not need to pass the result field.	**result**	**orderoptions**

OrderErr_PostAuthRef	No reference number was given for a POSTAUTH transaction.	Make sure a valid reference number is passed in the **reference_number** field. If you are unsure about the reference number, you can look it up in your online transaction reports. If you are doing a POSTAUTH for a PREAUTH obtained over the phone, provide the reference number given with the approved authorization.	**reference_number**	**transactiondetails**
OrderErr_InvalidResult	Order result specified is not valid	Make sure a valid value is passed in the **result** field. Set to LIVE for live mode, GOOD for an approved response in test mode, DECLINE for a declined reponse in test mode, or DUPLICATE for a duplicate response in test mode. If you are not testing (that is, if you are running your store "live"), you do not need to pass the result field.	**result**	**orderoptions**

OrderErr_VoidSaleRef	No reference number was given for a void sale transaction.	Make sure a valid reference number is passed in the **reference_number** field. If you are unsure about the reference number, you can look it up in your online transaction reports. If you are doing a VOID for a PREAUTH obtained over the phone, provide the reference number given with the approved authorization.	**reference_number**	**transactiondetails**
OrderErr_PostAuthOid	No order number was given for a POSTAUTH transaction.	Make sure the corresponding order ID number from the PREAUTH transaction is provided when you attempt the POSTAUTH transaction.	**oid**	**transactiondetails**
Total charge amount ($_.__) did not match sum of the subtotal, shipping and tax amounts. or **OrderErr_ChargeTotal**	The subtotal plus the shipping and tax amounts do not total up to the total amount of the order. **subtotal** + **tax** + **vattax** + **shipping** MUST = **chargetotal**	Verify that the total charge amount entered matches the sum of the subtotal, shipping and tax amounts.	**subtotal**, **tax**, **vattax**, **shipping**, **chargetotal**	**payment**

OrderErr_CreditAmount	Tried to credit more than the original amount of the sale.	Make sure the amount you are crediting is less than the original amount of the sale. If other credits have already been run against this order, check that the total credits are not more than the original amount of the sale.	**chargetotal**	**payment**
OrderErr_UniqueOid	The order number given must be unique.	Verify that the order number system used assigns a unique number to each order.	oid	**transactiondetails**
OrderErr_RefNumMismatch	The reference number provided does not match the PREAUTH.	Check the reference number for the PREAUTH transaction.	**reference_number**	**transactiondetails**
OrderErr_NoAmount	No amount provided.	Verify that the **chargetotal** amount is provided and is a valid numeric value. May also apply to **tax** field if the tax field is required for this transaction.	chargetotal	**payment**
OrderErr_AmountTooHigh	The amount specified exceeds the order total.	Check that the amount specified for **tax**, **shipping**, and **vattax** are all less than the **chargetotal**. Check that the **subtotal** + **tax** + **vattax** + **shipping** = **chargetotal**	**reference_number**	**transactiondetails**
OrderErr_AmountTooLow	The amount is less than one dollar.	Verify that the **chargetotal** amount is more than one dollar.	chargetotal	**payment**

OrderErr_NoTransactionFound	No corresponding transaction was found to void.	Occurs for VOID transactions. Check that the order ID number is valid. Check that you are voiding a transaction that has not yet settled. If the transaction has already settled, you will need to do a CREDIT transaction, rather than a VOID.	**oid**	**transactiondetails**
OrderErr_VoidBeforePostAuth	No post-authorization has been run to void.	Verify that the transaction you are trying to void has been post-authorized. If it has not, you do not need to void it, since an AUTH transaction will not result in charges to a card without a corresponding POSTAUTH.	--	--
OrderErr_NoPreAuth	No corresponding PREAUTH transaction was found.	Occurs for POSTAUTH transactions. Check that the order ID number is valid. Check that you are running the POSTAUTH transaction against an approved PREAUTH.	**oid**	**transactiondetails**
OrderErr_NoApprovedPreAuth	No corresponding approved PREAUTH transaction was found.	Occurs for POSTAUTH transactions. Check that the order ID number is valid. Check that you are running the POSTAUTH transaction against an approved PREAUTH.	--	--

OrderErr_ApprovedPostAuth	An approved POSTAUTH transaction was found. The transaction appears to have been post-authorized already.	Occurs for POSTAUTH transactions. Check that the order ID number is valid. Check that you are not running duplicate POSTAUTH transactions.	**oid**	**transactiondetails**
OrderErr_DeclinedAuth	The original AUTH was not approved.	Occurs for POSTAUTH transactions. Check that the order ID number is valid. Check that you are running the POSTAUTH transaction against an approved PREAUTH.	**oid**	**transactiondetails**

Items and Options Errors

These errors may occur when describing items and options in an order if invalid or incomplete data is passed. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
ItemErr_InvalidItemid	An item identifier must be supplied.	Check that the **id** field is not empty.	**id**	**item**
ItemErr_InvalidDescription	The item description supplied is invalid.	Check the **description** field.	**description**	**item**
ItemErr_InvalidPrice	The value supplied for the item's price is null, empty or invalid.	Check that the **price** field is populated with a valid number.	**price**	**item**
ItemErr_InvalidQuantity	The value supplied for the item's quantity is null, empty or invalid.	Check that the **quantity** field is populated with a valid number.	**quantity**	**item**
ItemErr_InvalidEsdtype	The value supplied for the esdtype is invalid.	If the item is an electronic softgood download item, check that the **esdtype** field is populated with one of the following: ***softgood*** if it is a softgood or ***key*** if it is a keygen executable.	**esdtype**	**item**

ItemErr_InvalidSoftfile	There is no softfile corresponding to the file specified.	Ensure you have provided the file for download to support for loading on the Gateway. Check with support whether the file has been loaded to the Gateway.	**softfile**	**item**
ItemErr_MaxOptions	The maximum number of options has been exceeded for this item.	You will have to use fewer options for this item.	**--**	**options**
OrderErr_DuplicateItems	There are duplicate item numbers found in the order.	Ensure you send only one **item** entity for each item in the order. Ensure that the item **id** within the order are different. No two item ids in the same order should be the same.	**id**	**item**
OrderErr_Datasize	The transaction item information is over the maximum limit.	You have exceeded the maximum number of items allowed for one order. You can break the order up into two or more orders if needed.	**--**	**item**
OptionErr_InvalidOption	The **name** field for this option must be populated.	Ensure that, for every option sent, each **option** entity contains a **name** that is not null or empty.	**name**	**option**

OptionErr_InvalidChoice	The **value** field for this option must be populated.	Ensure that, for every option sent, each **option** entity contains a **value** that is not null or empty.	**value**	**item**

Fraud Decline Messages

When a specific transaction is declined because of the risk of fraud, the following messages may be returned in the **r_error** data field.

Response Message	Description
The host you are ordering from has been blocked.	The merchant has blocked the customer's host.
The credit card you are using has been blocked.	The merchant has blocked the customer's credit card number.
The domain of your host has been blocked.	The merchant has blocked the customer's domain name.
The class C subnet for this IP has been blocked.	The merchant has blocked the customer's Class C address.
The name that was entered has been blocked.	The merchant has temporarily blocked the customer's name.
The host you are ordering from has been temporarily blocked.	The merchant has temporarily blocked the customer's host.
The credit card you are using has been temporarily blocked.	The merchant has temporarily blocked the customer's credit card.
The purchase amount exceeds the merchant approved limit.	The total amount of the order exceeds the maximum purchase limit set by the merchant.
Merchant transaction limit is less than the amount requested for that transaction.	The total amount of the order is less than the minimum purchase limit set by the merchant.
DUPLICATE	A Transaction for the identical dollar amount and the identical credit card was processed within the last X hours. X is generally 24 hours, but can be altered by changing the duplicate lockout time in the merchant's fraud protection settings.

Other Decline or Error Response Messages

There are many other reasons that an error or decline may occur. You may receive any of the following messages in the **r_error** field if a validation, transmission, or other error occurs.

Connection Errors

These errors may occur if the wrapper is unable to connect to the Gateway XML processor.

Response Message	Description	Possible Course of Action
Creating socket failed.	Cannot create a socket to connect to the Gateway XML processor.	Check your host name, digital certificate, and port number. These items are included in the merchant's Welcome e-mail. If the host name, certificate, and port number are correct, retry the transaction.
Unable to connect to server.	The TCP/IP connection to the Gateway XML server failed.	Check your host name, digital certificate, and port number. These items are included in the merchant's Welcome e-mail. If the host name, certificate, and port number are correct, retry the transaction.
Failed to resolve host	Unknown host name	Check your host name. The host name is identified in the merchant's Welcome e-mail.
Cannot find/load cert/key file.	The client certificate file is missing or invalid	Check that you properly copied the digital certificate from the merchant's Welcome e-mail into a .PEM file on the Web server. Check the path to the file against the path and file name you provided to the Gateway.

Unable to connect to SSL server.	Cannot establish SSL connection	Check your host name, digital certificate, and port number. These items are included in the merchant's Welcome e-mail. If the host name, certificate, and port number are correct, retry the transaction.
Cannot find/load cert/key file.	The client certificate file is missing or invalid	Check that you properly copied the digital certificate from the merchant's Welcome e-mail into a .PEM file on the Web server. Check the path to the file against the path and file name you provided to the Gateway.
SSL read failed.	One of the servers (either merchant-side or Gateway-side) has dropped the SSL connection.	Check your host name, digital certificate, and port number. These items are included in the merchant's Welcome e-mail. If the host name, certificate, and port number are correct, retry the transaction.
Request rejected by SSL server.Make sure you supplied valid certificate.	The Gateway server cannot verify the client server's credentials.	Check that you properly copied the digital certificate from the merchant's Welcome e-mail into a .PEM file on the Web server. Check the path to the file against the path and file name you provided to the Gateway.

Other Errors

These errors may occur after the merchant (or CSP) Web server has made a connection to the Gateway server. In this case, the transmission or validity error is between the Gateway server and the processor.

Response Message	Description	Possible Course of Action

A file being accessed for opening does not exist.	A file is accessed for writing, and it cannot be written to.	Retry the transaction.
Error timeout waiting for response.	The process timed out waiting for a response from the credit card processor. This message results when the credit card processor does not respond to the leased line transmission of the transaction data within 60 seconds.	Retry the transaction
NetErr_Connect	Unable to connect to SSL Server	Check network configuration and digital certificate. If all seems okay on your side, contact support.
NetErr_SSL	Unrecoverable SSL error. Connection closed.	Retry the transaction.
NetErr_Decode	Unable to decode received message	Check network configuration and digital certificate. If all seems okay on your side, contact support.
ServerErr_Resource	Server unable to allocate required resources.	Retry the transaction.
ServerErr_Database	Server encountered a database error.	Check network configuration and digital certificate. If all seems okay on your side, retry the transaction or contact support.
ServerErr_Encode	Server unable to encode client response	Retry the transaction.

ServerErr_Decode	Server unable to decode client response.	Check network configuration and digital certificate. If all seems okay on your side, retry the transaction or contact support.
ServerErr_Module	Unsupported server module	--
CoreErr_Trunc	Results truncated.	--
CoreErr_Field_Format	Data contains invalid characters.	Check the format of your data fields.
CoreErr_Alloc	Unable to allocate memory.	--
CoreErr_InvalidHandle	Uninitialized/invalid handle supplied as an argument.	--
CoreErr_InvalidField	Unknown field name.	Check your data field names.
CoreErr_UnknownError	Error with unknown cause.	--
ReqErr_InvalidRequest	Request context must be allocated.	Contact support
OrderErr_Lock	Unable to set transaction lock on order.	Retry the transaction.
OrderErr_Unlock	Unable to remove transaction lock on order.	Retry the transaction.
OrderErr_Data	Error processing order data.	Retry the transaction.

Shipping Calculation Errors

These errors may occur when using the Gateway to do shipping charge calculations if invalid data is passed or if the merchant's shipping file is not properly configured for shipping calculations on the Gateway. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
ShipErr_InvalidCriteria	Total, items or weight must be specified to calculate the shipping charges. The data required depends on the shipping calculation method specified in the merchant's shipping file.	Check the shipping file for shipping method and provide valid data fields for that calculation method.	**weight**, **items**, and/or **weight**, depending on calculation method	**shipping**
ShipErr_InvalidCountry	The ship-to country code must be a valid 2-letter country code.	Ensure you are using a valid 2-letter country code as listed in the Country Codes section.	**country**	**shipping**
ShipErr_InvalidState	The ship-to state code must be a valid 2-letter US state code.	Ensure you are using a valid 2-letter US state code as listed in the State Codes section.	**state**	**shipping**
ShipErr_InvalidSyntax	The shipping file syntax is incorrect.	Check that the shipping file is properly formatted. Resubmit to support if necessary. See the *Using the Shipping Calculator* section for further information.	**--**	**--**

Tax Calculation Errors

These errors may occur when using the Gateway to do tax calculations if invalid data is passed or if the merchant's config file is not properly configured for tax calculations on the Gateway. You may receive any of the following messages in the **r_error** field when this occurs.

Response Message	Description	Possible Cause	Populate this Data Field	In this Entity
TaxErr_InvalidZip	The ship-to zip code passed is invalid.	Check the zip code provided.	**zip**	**shipping**
TaxErr_InvalidState	The ship-to state passed is invalid.	Ensure you are using a valid 2-letter US state code as listed in the State Codes section.	**state**	**shipping**
TaxErr_InvalidTotal	The total amount to be taxed is 0 or less.	Check the zip code provided.	**zip**	**shipping**
TaxErr_InvalidFullTax	There is an invalid fulltax value in the merchant's config file on the Gateway OR the config file is not set up to process tax calculations.	Check with support that the merchant's config file is set up with the appropriate values for tax calculations.	**--**	**--**

VirtualCheck Errors

The errors below may be returned from the API when a validation or transmission error occurs in processing a VirtualCheck transaction.

Error Code	Error Message	Comments
Status code 0	duplicate txn, stop	Two identical transactions were received. This is the second, which will not be processed.
Status code 10	Invalid parameters	Check input parameter format and validity. Check that all required parameters were included.
Status code 20	Dig Sig invalid	Check digital certificate for merchant site
Status code 30	User Cancelled auth -- Or -- incomplete data	User cancelled the transaction or not all required fields were provided
Status code 40	User account error -- Or -- We are sorry that we cannot accept your check at this time. Our decision is based, in whole or in part, on information provided to us by TeleCheck. We encourage you to call TeleCheck at 1-877-678-5898 or write TeleCheck Customer Care at P.O. Box 4513, Houston, TX 77210- 4513. Please provide TeleCheck your driver's license number and the state where it was issued, and the complete banking numbers printed on the bottom of your check. Under the Fair Credit Reporting Act, you have the right to a free copy of your information held in TeleCheck's files within 60 days from today. You may also dispute the accuracy or completeness of any information in TeleCheck's consumer report. TeleCheck did not make the adverse decision to not accept your check and is unable to explain why this decision was made.	If this text block is included with the response, the merchant has a legal obligation to display it to the consumer.

Status code 50	System not available	Contact customer support if problem persists.
LP-8996	Non-live TeleCheck transactions are not supported.	VirtualCheck account not set up yet. Check with your sales agent or merchant services on account status.
LP-001 to LP-008		Contact customer support if problem persists.
32001	CheckErr: Invalid order data.	
32002	CheckErr: Invalid check data.	
32003	CheckErr: Invalid request.	
32004	CheckErr: Invalid account type.	
32005	CheckErr: Invalid transit routing.	
32006	CheckErr: Invalid MICR.	
32007	CheckErr: Invalid check number.	
32008	CheckErr: Invalid check comment.	
32009	CheckErr: Routing number does not match.	
32010	CheckErr: Check order type is wrong.	
32011	CheckErr: Invalid check order data.	
32012	CheckErr: Error inserting order.	
32013	CheckErr: Error inserting transaction.	
32014	CheckErr: Error inserting batch transaction.	
32015	CheckErr: Unable to verify check processing status.	
32016	CheckErr: Error deleting check batch entry.	
32017	CheckErr: Check sent for processing.	You may be trying to void a check that has been sent for processing.
32018	CheckErr: Error voiding check.	
32019	CheckErr: Error updating transaction.	

Other Text-Based Messages that May Occur

- Your session has expired.
- You have exceeded the maximum number of authorization attempts. Please choose another payment option.
- We are unable to verify your checking account information. Please review the information you entered to ensure that all information is correct, then click **Authorize**.
- We are unable to verify your checking account information because your bank account may not be set up to handle electronic funds transfers through the Automated Clearinghouse (ACH). Please contact your bank to determine whether this account accepts Automated Clearinghouse (ACH) transactions. If you have another checking account, you may change your routing and account number information below.
- Please enter a First Name.
- Please enter a valid First Name.
- Please enter a Last Name.
- Please enter a valid Last Name.
- Please enter an Address.
- Please enter a valid Address (Address Line 1).
- Please enter a valid Address (Address Line 2).
- Please enter a City.
- Please enter a valid City.
- Please enter your entire Driver's License or State ID Number.
- Please make sure all the information is entered correctly and submit your request again. You may also select another payment option.
- Please enter a valid Driver's License or State ID Number.
- Please select the state where your Driver's License or State ID was issued.
- Please select a State.
- Please enter a ZIP Code.
- Please enter a valid ZIP Code.
- Please enter an Email.
- Please enter a valid Email.
- Please enter a Home Phone Number.
- Please enter a valid Home Phone Number.
- Please enter the name of your Bank.
- Please enter valid information for the name of your Bank.
- Please select a Bank State.
- Please enter a Routing Number.
- Please enter a valid Routing Number.

- Please enter a Checking Account Number.
- Please enter a valid Checking Account Number.

Requesting a Test Account

To request a test account, please fill out our teststore form or contact us at teststore@linkpoint.com.

Once your test store is set up, the test store will be valid for 30 days. For additional usage after 30 days, you may request an extension. If no extension is requested, the test store may be permanently removed any time on or after the 31st day. Maintenance on the test server may be performed without prior or adequate notice to the users. We will notify users when possible of any maintenance or outages on the staging (test) server.

Test store policy

Test stores may NOT be used for the following purposes:

- To reverse engineer the system to understand or breach security of the LinkPoint Gateway.
- To copy any of LinkPoint International products.
- To abuse the staging server in any way.

LinkPoint International will frequently monitor test store usage. Any breach of the above policies will result in suspension or permanent removal of a test store.

Support

Feel free to contact us at teststore@linkpoint.com for any test store related questions. Contact support@linkpoint.com for LinkPoint technical support.

Going Live

When you apply for a live store it will be on a different server and will have a different store number. You will need to change your host name and store number in your code. Also, when you want to go live, make sure the **result** field in the **orderoptions** entity is set to ***LIVE***.

Password

Ignore the section in your welcome e-mail about calling us for your initial password for your test store. Your initial password will be 12345678. To protect your test account, you should change this password on initial login.

Sign In / Login

LinkPoint Central (LPC) provides you with a virtual terminal and order management functions. On the staging server, LPC is available at https://staging.linkpt.net. To log in, enter:

- Your Storename: (a 10-digit number for test accounts),
- Your Username: (a 6-digit number), and

- Your Password: (initial password is 12345678)

LinkPoint Basic

To set up your LinkPoint Basic test account:

- The storename parameter will be a 10-digit number on the staging server.
- The posting url will be https://staging.linkpt.net/lpc/servlet/lppay
- The Admin url will be https://staging.linkpt.net/admin/xxxxxxxxxx/hlpadmin , where xxxxxxxxxxx is your 10-digit store number.

LinkPoint Select API

- Set the **configfile** data field to your 10-digit test store number.
- The **host** will be ***staging.linkpt.net***.
- The **keyfile** is your digital certificate (cert and key including dashes) included at the bottom of "Welcome to LinkPoint Select API" e-mail. Save this certificate to a file on your Web server with a .pem extension. The keyfile for JavalinkAPI users will have to be split and converted to .der file using PEM2DER included in the Javalink API download.
- The **port** is ***1129*** for all APIs, except the AIX API, the Coldfusion custom tag, and the Javalink API, which use ***1139***.

Downloading Software

For all API and Wrapper downloads visit https://www.linkpoint.com/viewcart/index.html

For sales please call 1.888.477.3611 option 1.

Testing with your LIVE Account

To perform tests with your live account, set the **result** field in the **orderoptions** entity to ***GOOD*** (for an approved response), ***DECLINE*** (for a declined response), or ***DUPLICATE*** (for a duplicate response).

We do not recommend performing tests with your LIVE account other than those necessary to validate correct functionality of your live account. Please request a test account if you need to do extensive testing.

When you are through testing, make sure you set the **result** field in the **orderoptions** entity to ***LIVE***.

Test Credit Cards

For testing purposes, you can use any of the card numbers listed below. These are test card numbers that will not result in any charges to the card. Use these card numbers with any expiration date in the future.

- **American Express®**: 371111111111111
- **Discover®**: 6011-1111-1111-1111
- **JCB®**: 311111111111111
- **MasterCard®**: 5111-1111-1111-1111
- **MasterCard**: 5419-8400-0000-0003
- **Visa®**: 4111-1111-1111-1111

About Chargebacks

A **chargeback** is a forced refund to the customer via the merchant's bank account. Chargebacks can occur with any type of business, but it is more prevalent for Internet businesses because of the increased chance of fraud. Each fraudulent credit card transaction usually results in a chargeback. Merchants new to the Internet often begin by accepting fraud and the resulting chargebacks as part of the price of doing business. Then they regret this after the bank terminates their account for too many chargebacks.

Credit card associations penalize merchant banks for chargebacks. Naturally, the bank passes the fines on to the responsible merchant, and these penalties can be severe.

While consumers are provided with a certain degree of protection if their credit card numbers are stolen and misused, Internet merchants are fully liable for all transactions because Internet transactions are classified as "card-not-present."

There are several ways to help you prevent chargebacks:

Address Verification

The first thing you should pay attention to is the Address Verification System (AVS) code. When a credit card transaction is submitted for processing, the customer's address is checked against the card-issuing bank's address on file. The transaction results page will return an AVS code, which will tell you whether the address and zip code matched or not. We encourage you to use AVS codes to help you prevent costly chargebacks. See AVS Codes for more information.

Blocking and Limiting

There are other ways to prevent fraud. If you suspect certain transactions might be fraudulent, you can block further purchases by blocking credit card numbers, persons' names, domain names, IP addresses or Class C addresses from purchasing at your store. You can also limit the amount that any customer can spend at your store by setting a maximum purchase amount, and you can set how long automatic lockouts and duplicate lockouts will continue to be blocked. See Preventing Fraud for more information.

Card Codes

The card code is a 3 or 4-digit number usually found on the back of the credit card. It is one more measure credit card companies are taking to help you ensure that the transaction is not fraudulent. Even though the system to check card codes for all types of cards is not yet in place, we encourage you to get in the habit of filling in the card code field. See Using Card Codes for more information.

Using Address Verification

For non-swiped transactions, the Gateway provides Address Verification System (AVS) codes to help protect you from costly chargebacks and fraud. Also, some credit cards (including Discover®, MasterCard®, and Visa®) require the use of AVS when you are processing card-not-present (i.e., MO/TO or e-commerce) transactions.

Whenever you perform a credit card **Sale** or **Authorize Only** transaction, the Gateway verifies the customer's address provided against the address that the card-issuing bank has on file for the customer. In order to take advantage of AVS, you need to pass the first line of the customer's billing address and the zip code to the Gateway.

The AVS code tells you how well the two addresses match. If the transaction is approved, you will find the 3-digit AVS code in the middle of the **Approval Code**. A typical transaction result code might look like this. The AVS code is highlighted.

0097820000019564:**YNA**M:12345678901234567890123:

AVS compares the numeric portion of the street address and the zip code against the information on file with the card-issuing bank. If the street address number matches, the 3-digit AVS code will begin with a **Y**. If it doesn't, the AVS code will begin with an **N**. If the zip code matches, the second digit will be a **Y**; if not, it will be an **N**. If there is insufficient information to do a comparison, an **X** will appear in the first and/or second digit. An **N** indicates a higher probability of fraud. Please see the table below for a list of codes and their meanings.

You will only get an AVS code if the transaction was approved, regardless of whether the addresses match. If you get an AVS code indicating that the address and/or zip code do not match, it is up to you to decide whether you wish to accept the risk and ship the goods to the customer to complete the transaction.

It is important to know that AVS has some limitations, because this may impact your decision-making about how to treat bad verification results:

- The AVS system isn't always reliable; bad results can be triggered unnecessarily because people move, or because some people report five-digit zip codes and some report nine-digit zip codes. This may generate a response stating that the address matches, but the zip code does not match.
- The AVS system doesn't generally handle addresses outside the U.S., so if you decide to ship only to addresses with good AVS results, you will rule out most international orders.

However, checking the AVS response on all transactions greatly reduces your exposure to risk.

Declines can be handled politely to keep the door open to legitimate orders by displaying a message similar to: "We are unable to process your credit card payment at this time. If you still wish to purchase this product or service, please call us at 1-800….". At this time the merchant can obtain more information from the customer to verify why the address didn't match such as recently moved, city changed zip codes, etc. The merchant can also ensure their product is shipped via registered mail with a return signed receipt to ensure it was received by the proper person.

AVS Code Meanings

Response codes can seem rather cryptic, but the table below shows what they mean.

AVS Code	DESCRIPTION
YY*	Address matches, zip code matches
YN*	Address matches, zip code does not match
YX*	Address matches, zip code comparison not available
NY*	Address does not match, zip code matches
XY*	Address comparison not available, zip code matches
NN*	Address comparison does not match, zip code does not match
NX*	Address does not match, zip code comparison not available
XN*	Address comparison not available, zip code does not match
XX*	Address comparisons not available, zip code comparison not available
(*) -- This is the one character response code sent by the authorizing bank and it varies by card type (e.g., Y,Z,A,N,U,R,S,E,G are valid responses for Visa®; Y,Z,A,N,X,W,U,R,S are valid for MasterCard®; Y,Z,A,N,U,R,S are valid for American Express® and A,Z,Y,N,W,U are valid for Discover®).	

Using the Card Code

Mail order and telephone order (MO/TO) and other card-not-present transactions have higher fraud rates than face-to-face transactions. To help reduce fraud in the card-not-present environment, credit card companies have introduced a card code program. Visa® calls this code Card Verification Value (CVV)--MasterCard® calls it Card Validation Code (CVC)).

What is a card code?

The card code is a three- or four- digit security code that is printed on the back of cards. The number typically appears at the end of the signature panel. This program helps validate that a genuine card is being used during a transaction. All MasterCard cards, both credit and debit, were required to contain CVC2 by January 1, 1997; all Visa cards must contain CVV2 by January 1, 2001.

How does the card code work?

You send the card code to the Gateway when processing an order. The Gateway then compares the card code against the code on file with the card-issuing bank. Results of this comparison show in the transaction approval code.

Why should I use a card code?

To help combat fraud, card-not-present merchants (those who receive orders via mail order, telephone order, or the Internet) should always enter a card code (if on the card) when processing an authorization. For retail transactions, you may wish to enter the card code printed on the card to ensure that the card was not fraudulently reproduced. By using the card code results along with the Address Verification Service (AVS), you can make more informed decisions about whether to accept transactions.

Where do I find card code comparison results?

A typical transaction result code might look like this. The card code result is highlighted.

0097820000019564:YNAM:12345678901234567890123:

The last alphabetic character in the middle (M) is a code indicating whether the card code matched the card-issuing bank's code. An "M" indicates that the code matched. This code may or may not be present, depending on whether the card code was passed and the service was available for the type of card used. Below is a table showing all the possible return codes and their meanings.

Value	Meaning
M	Card Code Match
N	Card code does not match

P	Not processed
S	Merchant has indicated that the card code is not present on the card
U	Issuer is not certified and/or has not provided encryption keys
	A blank response should indicate that no code was sent and that there was no indication that the code was not present on the card.

What about American Express® and Discover®? Don't they have card codes too?

Yes, American Express and Discover do have card codes printed on their cards. The Gateway does not currently support American Express or Discover card codes, so the card code response will be blank from an American Express or Discover card. We do encourage you to get in the habit of entering card codes from all cards, however.

Using Blocking and Limits

If certain individuals are continuously hurting your business with costly chargebacks, you can block them from purchasing at your store. We provide you the capability to block credit card numbers, names, domain names, and IP or Class C addresses from purchasing at your store. Two other capabilities available are the power to set a limit on the maximum purchase that can be made at your store, and the ability to set auto lockout and duplicate lockout times.

To change your fraud settings, log into your online account, select **Admin** in the **Main Menu Bar**, then click on **Fraud Settings** in the **Left Menu Box**. Fraud settings are sequentially ordered, so once you've finished with one, you can proceed to the next fraud setting. There is also a quick navigation bar at the top of each Fraud Setting page, which allows you to jump immediately to any Fraud Setting.

Your Fraud Settings

Here is a list of all the fraud settings you have control over.

- Blocking Credit Card Numbers
- Blocking Names
- Blocking Domain Names
- Blocking Class C and IP Addresses
- Setting a Maximum Purchase Limit
- Setting Auto lockout times
- Setting Duplicate lockout times

Sample Code to Pass XML to the PHP Module

The code shown below demonstrates how to process a Sale by passing an XML stream to the Gateway. The transaction shown is a credit card sale, but the same method is applicable to any type of transaction. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

# PASS_XML.pl - example of how to send and recieve an XML string
#
# In this sample, the merchant would be doing their own XML encoding/decoding
#
# This sample does only a minimal SALE transaction, but it can be
# used as an example of passing in larger XML strings for more complex
# transactions. Any of the included LinkPoint XML sample files could be
# passed in as an XML string here.
#
# Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
#
# This software is the proprietary information of LinkPoint International, Inc.
# Use is subject to license terms.


 include"lphp.php";
 $mylphp=new lphp;


/*  The formatting of the XML in this sample is only for example
 purposes and human-readability; in real life it would typically
 be all one long unbroken line. */
 $xml ="
 <order>
```

```
    <orderoptions>
      <result>GOOD</result>
      <ordertype>SALE</ordertype>
    </orderoptions>
    <merchantinfo>
      <configfile>1234567</configfile> <!-- CHANGE THIS TO YOUR STORE
NUMBER -->
    </merchantinfo>
    <creditcard>
      <cardnumber>4111111111111111</cardnumber>
      <cardexpmonth>12</cardexpmonth>
      <cardexpyear>08</cardexpyear>
    </creditcard>
    <payment>
      <chargetotal>1.03</chargetotal>
    </payment>
  </order>";

  $myorder["host"]   = "secure.linkpt.net";
  $myorder["port"]   = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # change this to the name and location
of your certificate file
  $myorder["xml"]   = $xml;
// $myorder["debugging"] = "true"; # for development only; not intended for
production use



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder);    # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods

  if (strlen($result) < 2)  # no response
  {
    $result = "<r_error>Could not execute curl.</r_error>";
  }
  echo "Response: $result\n";

  # Process the XML from here....




  # Or OPTIONALLY - you could convert XML response to a readable array
  preg_match_all ("/<(.*?)>(.*?)\</", $result, $outarr, PREG_SET_ORDER);

  $n = 0;
  while (isset($outarr[$n]))
  {
    $retarr[$outarr[$n][1]] = strip_tags($outarr[$n][0]);
    $n++;
  }

  # and then look at it like this
  while (list($key, $value) = each($retarr))
    echo "$key = $value \n";
  # and use the hash elements that you need
?>
```

Sample Code to Pass XML Directly to the PHP Module

The code shown below demonstrates how to process a Sale by passing an XML stream to the Gateway. The transaction shown is a credit card sale, but the same method is applicable to any type of transaction. This sample demonstrates bypassing the LPHP.PHP module that is used by most of the PHP samples, and sending an XML string DIRECTLY to the shared library LIBLPHP.SO which then uses LIBLPSSL.SO to communicate with the Gateway. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
#!/usr/bin/php

<?php

# PASS_XML_LIB.php - example of how to send and recieve an XML string
#
# In this sample, the merchant would be doing their own XML encoding/decoding
#
# This sample demonstrates bypassing the LPHP.PHP module that is used
# by most of the LinkPoint PHP samples, and sending an XML string DIRECTLY
# to the shared library LIBLPHP.SO which then uses LIBLPSSL.SO to communicate
# with LSGS.
#
# An appropriate use of this code would be in a PHP installation where the
# merchant wants to do their own XML encoding/decoding and the server's PHP
# installation does not support the built-in CURL methods.
#
# This sample does only a minimal SALE transaction, but it can be
# used as an example of passing in larger XML strings for more complex
# transactions. Any of the included LinkPoint XML sample files could be
# passed in as an XML string here.
#
# Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
```

```
#
# This software is the proprietary information of LinkPoint International, Inc.
# Use is subject to license terms.


# If the library is at loaded at startup in PHP.INI, we don't need this call
  dl("liblphp.so");

  $xml ="
<order>
<orderoptions>
<result>GOOD</result>
<ordertype>SALE</ordertype>
</orderoptions>
<merchantinfo>
<configfile>123456</configfile> <!-- CHANGE THIS TO YOUR STORE NUMBER
-->
</merchantinfo>
<creditcard>
<cardnumber>4111111111111111</cardnumber>
<cardexpmonth>12</cardexpmonth>
<cardexpyear>08</cardexpyear>
</creditcard>
<payment>
<chargetotal>1.03</chargetotal>
</payment>
</order>";

  $cert  = "./123456.pem"; # change this to the name and location of your certificate file
  $host  = "secure.linkpt.net";
  $port  = 1129;

  // send transaction to liblphp.so
  $retstg = send_stg($xml, $cert, $host, $port);

  # look at the xml that comes back
  print ("response: $retstg\n\n");

  if (strlen($retstg) < 2) # no response
  {
    $retstg = "<r_error>Could not execute curl.</r_error>";
  }

  # Process the XML from here
```

```
 # Or OPTIONALLY - you could convert XML to a readable array
 preg_match_all ("/<(.*?)>(.*?)\</", $retstg, $outarr, PREG_SET_ORDER);

 $n = 0;
 while (isset($outarr[$n]))
 {
   $retarr[$outarr[$n][1]] = strip_tags($outarr[$n][0]);
   $n++;
 }

 # and then look at it like this
 while (list($key, $value) = each($retarr))
   echo "$key = $value \n";
 # and use the hash elements that you need

?>
```

Sample Code to Pass XML Directly to the PHP Module

The code shown below demonstrates how to process a Sale by passing an XML stream to the Gateway. The transaction shown is a credit card sale, but the same method is applicable to any type of transaction. This sample demonstrates using the PHP built-in cURL methods to send an XML string DIRECTLY to the Gateway, bypassing lphp.php, liblphp.so and liblpssl.so. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

# PASS_XML_DIRECT.php - example of how to send and recieve an XML string
# directly to LSGS
#
# This sample demonstrates using the PHP built-in CURL methods to send an
# XML string DIRECTLY from this sample program to LSGS, bypassing lphp.php,
# liblphp.so and liblpssl.so.
#
# We assume that PHP has been compiled to support the CURL methods.
# To see if your server's PHP supports CURL methods, enter the command
# "php -m" to see a listing of supported modules.
#
# In this program, the merchant would be doing their own XML encoding/decoding
#
# This sample does only a minimal SALE transaction, but it can be
# used as an example of passing in larger XML strings for more complex
# transactions. Any of the included LinkPoint XML sample files could be
# passed in as an XML string here.
#
# Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
#
# This software is the proprietary information of LinkPoint International, Inc.
# Use is subject to license terms.
# Build a simple transaction XML string
# This XML formatting is for human readability only
# (In real life, this would be all one long unbroken string)
  $xml ="
<order>
<orderoptions>
<result>GOOD</result>
<ordertype>SALE</ordertype>
</orderoptions>
<merchantinfo>
<configfile>123456</configfile> <!-- CHANGE THIS TO YOUR STORE NUMBER
-->
</merchantinfo>
<creditcard>
<cardnumber>4111111111111111</cardnumber>
<cardexpmonth>12</cardexpmonth>
<cardexpyear>08</cardexpyear>
</creditcard>
<payment>
<chargetotal>1.03</chargetotal>
</payment>
</order>";

  $host  = "secure.linkpt.net";
  $port  = 1129;
  $cert  = "./YOURCERT.pem"; # change this to the name and location of your
certificate file


  $hoststring = "https://".$host.":".$port."/LSGSXML";

  # use PHP built-in curl functions
  $ch = curl_init ();
  curl_setopt ($ch, CURLOPT_URL,$hoststring);
  curl_setopt ($ch, CURLOPT_POST, 1);
  curl_setopt ($ch, CURLOPT_POSTFIELDS, $xml); # the string we built above
  curl_setopt ($ch, CURLOPT_SSLCERT, $cert);
  curl_setopt ($ch, CURLOPT_RETURNTRANSFER, 1);
#  curl_setopt ($ch, CURLOPT_SSL_VERIFYHOST, 0);
#  curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, 0);

//  curl_setopt ($ch, CURLOPT_VERBOSE, 1);  // optional - verbose debug output
                    // not for production use
  # send the string to LSGS
  $result = curl_exec ($ch);

  if (strlen($result) < 2) # no response
  {
    $result = "<r_error>Could not execute curl.</r_error>";
  }



  # look at the xml that comes back
  print ("response: $result\n\n");

  # Process the XML from here....



  # Or OPTIONALLY - you could convert XML to an array
  preg_match_all ("/<(.*?)>(.*?)\</", $result, $outarr, PREG_SET_ORDER);

  $n = 0;
  while (isset($outarr[$n]))
  {
    $retarr[$outarr[$n][1]] = strip_tags($outarr[$n][0]);
    $n++;
  }

  # and then look at it like this
  while (list($key, $value) = each($retarr))
    echo "$key = $value \n";
  # and use the hash elements that you need

?>
```

Sample Forms for the PHP Module

To help you with your Web site development, we have included two demonstration forms. PHP_FORM_MIN.html sends a post to the PHP_FORM_MIN.php script, which sends transactions to the Gateway. PHP_FORM_MAX.html posts to the PHP_FORM_MAX.php script. Both of these scripts call functions in the lpPHP.pm module.

Depending on your server setup, the scripts that process this form, PHP_FORM_MAX.php and PHP_FORM_MIN.php and the PHP module may need to be located in a cgi-bin directory, and the path in the "action" method of the HTML forms will need to be adjusted accordingly.

The form fields have been pre-filled with test data only for convenience.

If you copy this code for your own use, .phpease make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if ap.phpicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

PHP_FORM_MIN.html

```
<!------------------------------------------------------------------------------
* PHP_FORM_MIN.html - A form processing example showing the minimum
* number of possible fields for a credit card SALE transaction.
*
* This page passes form data passed to the script PHP_FORM_MIN.php.
*
* Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
*
* This software is the proprietary information of LinkPoint International, Inc.
* Use is subject to license terms.
*



<html><head><title>PHP_FORM_MIN.html sample program</title></head><body>

<br>
<form name="form1" method="post" action="PHP_FORM_MIN.php">

<br><br>
<table>
  <tr>
```

```
    <td align="right">ordertype</td>
    <td><input name="ordertype" value="SALE"></td>
  </tr>
  <tr>
    <td align="right">amount</td>
    <td><input name="chargetotal" value="9.99"></td>
  </tr>
  <tr>
    <td align="right">cardnumber</td>
    <td><input name="cardnumber" value="4111111111111111"></td>
  </tr>
  <tr>
    <td align="right">cardexpmonth</td>
    <td><input name="cardexpmonth" value="01"></td>
  </tr>
  <tr>
    <td align="right">cardexpyear</td>
    <td><input name="cardexpyear" value="05"></td>
  </tr>
  <tr>
    <td align="right"> debugging</td>
    <td><input type=checkbox name="debugging"></td>
  </tr>
  <tr>
    <td align="right"> verbose output</td>
    <td><input type=checkbox name="verbose"></td>
  </tr>
  <tr>
    <td> </td>
    <td> <input type="submit"></td>
  </tr>
</table>
</form>

</font></body></html>
```

PHP_FORM_MIN.php

```
<?
echo"<html><head><title>PHP_FORM_MIN.php LinkPoint Sample </title></head><body><br>";

/*
<!---------------------------------------------------------------------------------
* PHP_FORM_MIN.php - A form processing example showing the minimum
* number of possible fields for a credit card SALE transaction.
*
* This script processes form data passed in from PHP_FORM_MIN.html
*
*
* Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
*
* This software is the proprietary information of LinkPoint International, Inc.
* Use is subject to license terms.
*

    This program is based on the sample SALE_MININFO.php

    Depending on your server setup, this script may need to
    be placed in the cgi-bin directory, and the path in the
    calling file PHP_FORM_MIN.html may need to be adjusted
    accordingly.

    NOTE: older versions of PHP and in cases where the PHP.INI
    entry is NOT "register_globals = Off", form data can be
    accessed simply by using the form-field name as a varaible
    name, eg. $myorder["host"] = $host, instead of using the
    global $_POST[] array as we do here. Passing form fields
    as demonstrated here provides a higher level of security.

---------------------------------------------------------------------------------->
*/
  include"lphp.php";
  $mylphp=new lphp;

  # constants
  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of your
certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  # form data
  $myorder["cardnumber"] = $_POST["cardnumber"];
  $myorder["cardexpmonth"] = $_POST["cardexpmonth"];
  $myorder["cardexpyear"] = $_POST["cardexpyear"];
  $myorder["chargetotal"] = $_POST["chargetotal"];
  $myorder["ordertype"] = $_POST["ordertype"];

  if ($_POST["debugging"])
    $myorder["debugging"]="true";

# Send transaction. Use one of two possible methods
# %response = $lperl->process($myorder); # use shared library model
%response = $lperl->curl_process($myorder); # or use curl methods

  if ($result["r_approved"] != "APPROVED") // transaction failed, print the reason
  {
    print "Status: $result[r_approved]<br>\n";
    print "Error: $result[r_error]<br><br>\n";
  }
  else  // success
  {
    print "Status: $result[r_approved]<br>\n";
    print "Transaction Code: $result[r_code]<br><br>\n";
  }

# if verbose output has been checked,
# print complete server response to a table
  if ($_POST["verbose"])
  {
    echo "<table border=1>";

    while (list($key, $value) = each($result))
    {
      # print the returned hash
      echo "<tr>";
      echo "<td>" . htmlspecialchars($key) . "</td>";
      echo "<td><b>" . htmlspecialchars($value) . "</b></td>";
      echo "</tr>";
    }

    echo "</TABLE><br>\n";
  }
?>

</body></html>
```

PHP_FORM_MAX.html

```
<!---------------------------------------------------------------------------------
* php_form_max.html - A form processing example showing many of the possible fields
*
* Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
*
* This software is the proprietary information of LinkPoint International, Inc.
* Use is subject to license terms.
*

    This program is based on the sample SALE_MAXINFO.php
    See comments in that file for many field descriptions.

    Depending on your server setup, this script may need to
    be placed in the cgi-bin directory, and the path in the
    calling file PHP_FORM_MAX.html may need to be adjusted
    accordingly.

    NOTE: older versions of PHP and in cases where the PHP.INI
    entry is NOT "register_globals = Off", form data can be
    accessed simply by using the form-field name as a varaible
    name, eg. $myorder["host"] = $host, instead of using the
    global $_POST[] array as we do here. Passing form fields
    as demonstrated here provides a higher level of security.


---------------------------------------------------------------------------------->



<html><head><title>PHP_FORM_MAX.php LinkPoint Sample</title></head><body>
<br>
<form name="form1" method="post" action="PHP_FORM_MAX.php">

<table>
  <tr>
    <td>
      <table>
      <tr> <th colspan=2>Transaction Details</th></tr>

      <tr>
        <td align="right">ordertype</td>
        <td><input name="ordertype" value="SALE"></td>
      </tr>
      <tr>
  <td align="right">result</td>
  <td><input name="result" value="LIVE"></td>
</tr>
<tr>
  <td align="right">transactionorigin</td>
  <td><input name="transactionorigin" value="ECI"></td>
</tr>
<tr>
  <td align="right">oid</td>
  <td><input name="oid" value="12345678-A345"></td>
</tr>
<tr>
  <td align="right">ponumber</td>
  <td><input name="ponumber" value="09876543-Q1234"></td>
</tr>

<tr>
  <td align="right">taxexempt</td>
  <td><input name="taxexempt" value="NO"></td>
</tr>
<tr>
  <td align="right">terminaltype</td>
  <td><input name="terminaltype" value="UNSPECIFIED"></td>
</tr>

<tr>
  <td align="right">ip</td>
  <td><input name="ip" value="123.123.123.123"></td>
</tr>

<tr> <th colspan=2>Totals</th> </tr>

<tr>
  <td align="right">subtotal</td>
  <td><input name="subtotal" value="12.99"></td>
</tr>
<tr>
  <td align="right">tax</td>
  <td><input name="tax" value="0.34"></td>
</tr>
<tr>
  <td align="right">shipping</td>
  <td><input name="shipping" value="1.45"></td>
</tr>
<tr>
  <td align="right">vattax</td>
```

```
    <td><input name="vattax" value="0.00"></td>
  </tr>
  <tr>
    <td align="right">chargetotal</td>
    <td><input name="chargetotal" value="14.78"> </td>
  </tr>

  <tr><th colspan=2>Card Info.</th></tr>

  <tr>
    <td align="right">cardnumber</td>
    <td><input name="cardnumber" value="4111111111111111"></td>
  </tr>
  <tr>
    <td align="right">cardexpmonth</td>
    <td><input name="cardexpmonth" value="01"></td>
  </tr>
  <tr>
    <td align="right">cardexpyear</td>
    <td><input name="cardexpyear" value="04"></td>
  </tr>
  <tr>
    <td align="right">cvmindicator</td>
    <td><input name="cvmindicator" value="provided"></td>
  </tr>
  <tr>
    <td align="right">cvmvalue</td>
    <td><input name="cvmvalue" value="123"></td>
  </tr>

  <tr> <th colspan=2>Item & Options</th></tr>

  <tr>
    <td align="center" colspan=2>Item 1</td>
  </tr>
  <tr>
    <td align="right">id</td>
    <td><input name="id" value="123456-A98765"></td>
  </tr>
  <tr>
    <td align="right">description</td>
    <td><input name="description" value="Logo T-Shirt"></td>
  </tr>
  <tr>
    <td align="right">quantity</td>
    <td><input name="quantity" value="1"></td>
  </tr>
  <tr>
    <td align="right">price</td>
    <td><input name="price" value="12.99"></td>
  </tr>
  <tr>
    <td align="center" colspan=2>Item 1 - Option 1</td>
  </tr>
  <tr>
    <td align="right">name</td>
    <td><input name="name1" value="Color"></td>
  </tr>
  <tr>
    <td align="right">value</td>
    <td><input name="value1" value="Red"></td>
  </tr>

  <tr>
    <td align="center" colspan=2>Item 1 - Option 2</td>
  </tr>
  <tr>
    <td align="right">name</td>
    <td><input name="name2" value="Size"></td>
  </tr>
  <tr>
    <td align="right">value</td>
    <td><input name="value2" value="XL"></td>
  </tr>

</table>
</td>

<td>                </td>

<td>
  <table>
  <tr> <th colspan=2>Billing Info.</th></tr>
  <tr>
    <td align="right">name</td>
    <td><input name="name" value="Joe Customer"></td>
  </tr>
  <tr>
    <td align="right">company</td>
    <td><input name="company" value="SomeWhere, Inc."></td>
  </tr>
  <tr>
<td align="right">address1</td>
<td><input name="address1" value="123 Broadway"></td>
</tr>
<tr>
<td align="right">address2</td>
<td><input name="address2" value="Suite 23"></td>
</tr>
<tr>
<td align="right">city</td>
<td><input name="city" value="Moorpark"></td>
</tr>
<tr>
<td align="right">state</td>
<td><input name="state" value="CA"></td>
</tr>
<tr>
<td align="right">country</td>
<td><input name="country" value="US"></td>
</tr>
<tr>
<td align="right">phone</td>
<td><input name="phone" value="8051234567"></td>
</tr>
<tr>
<td align="right">fax</td>
<td><input name="fax" value="8059876543"></td>
</tr>
<tr>
<td align="right">email</td>
<td><input name="email" value="joe.customer@somewhere.com"></td>
</tr>
<tr>
<td align="right">zip</td>
<td><input name="zip" value="87123"></td>
</tr>
<tr>
<td align="right">addrnum</td>
<td><input name="addrnum" value="123"></td>
</tr>

<tr> <th colspan=2>Shipping Info.</th></tr>
<tr>
<td align="right">sname</td>
<td><input name="sname" value="Joe Customer"></td>
</tr>
<tr>
```

```
      <td align="right">saddress1</td>
      <td><input name="saddress1" value="123 Broadway"></td>
    </tr>
    <tr>
      <td align="right">saddress2</td>
      <td><input name="saddress2" value="Suite 23"></td>
    </tr>
    <tr>
      <td align="right">scity</td>
      <td><input name="scity" value="Moorpark"></td>
    </tr>
    <tr>
      <td align="right">sstate</td>
      <td><input name="sstate" value="CA"></td>
    </tr>
    <tr>
      <td align="right">szip</td>
      <td><input name="szip" value="12345"></td>
    </tr>
    <tr>
      <td align="right">scountry</td>
      <td><input name="scountry" value="US"></td>
    </tr>

    <tr> <th colspan=2>Misc.</th></tr>

    <tr>
      <td colspan=2>comments<br><textarea name="comments">Repeat customer. Ship 
immediately.</textarea></td>
    </tr>

    <tr>
      <td colspan=2>referred<br><textarea name="referred">Saw ad on Web site.</textarea></td>
    </tr>

    <tr><td> </td></tr>
    <tr><td> </td></tr>

    <tr>
      <td align="right"><input type=checkbox name="verbose"></td>
      <td> verbose output </td>
    </tr>
    <tr>
      <td align="right"><input type=checkbox name="debugging"></td>
      <td> debugging </td>
    </tr>
```

```
     <tr><td> </td></tr>
     <tr><td> </td></tr>

     </table>
   </td>
  </tr>

  <tr>
   <td colspan=3 align="middle"> <input type="submit" ></td>
  </tr>
</table>

</body></html>
```

PHP_FORM_MAX.php

```
<?
echo"<html><head><title>PHP_FORM_MIN.php LinkPoint Sample </title></head><body><br>";
/*

<!-------------------------------------------------------------------------------
* PHP_FORM_MAX.php - A form processing example showing many of the possible fields
*
* This script processes form data passed in from PHP_FORM_MAX.html
*
*
* Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
*
* This software is the proprietary information of LinkPoint International, Inc.
* Use is subject to license terms.
*


    This program is based on the sample SALE_MAXINFO.php
    See comments in that file for many field descriptions.

    Depending on your server setup, this script may need to
    be placed in the cgi-bin directory, and the path in the
    calling file PHP_FORM_MAX.html may need to be adjusted
    accordingly.

    NOTE: older versions of PHP and in cases where the PHP.INI
    entry is NOT "register_globals = Off", form data can be
    accessed simply by using the form-field name as a varaible
    name, eg. $myorder["host"] = $host, instead of using the
```

```
   global $_POST[] array as we do here. Passing form fields
   as demonstrated here provides a higher level of security.

------------------------------------------------------------------------------------>
*/
  include"lphp.php";
  $mylphp=new lphp;

  # constants
  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of your
certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  # transaction details
  $myorder["ordertype"] = $_POST["ordertype"];
  $myorder["result"] = $_POST["result"];
  $myorder["transactionorigin"] = $_POST["transactionorigin"];
  $myorder["oid"] = $_POST["oid"];
  $myorder["ponumber"] = $_POST["ponumber"];
  $myorder["taxexempt"] = $_POST["taxexempt"];
  $myorder["terminaltype"] = $_POST["terminaltype"];
  $myorder["ip"] = $_POST["ip"];

  # totals
  $myorder["subtotal"] = $_POST["subtotal"];
  $myorder["tax"] = $_POST["tax"];
  $myorder["shipping"] = $_POST["shipping"];
  $myorder["vattax"] = $_POST["vattax"];
  $myorder["chargetotal"] = $_POST["chargetotal"];

  # card info
  $myorder["cardnumber"] = $_POST["cardnumber"];
  $myorder["cardexpmonth"] = $_POST["cardexpmonth"];
  $myorder["cardexpyear"] = $_POST["cardexpyear"];
  $myorder["cvmindicator"] = $_POST["cvmindicator"];
  $myorder["cvmvalue"] = $_POST["cvmvalue"];

  # BILLING INFO
  $myorder["name"] = $_POST["name"];
  $myorder["company"] = $_POST["company"];
  $myorder["address1"] = $_POST["address1"];
  $myorder["address2"] = $_POST["address2"];
  $myorder["city"] = $_POST["city"];
  $myorder["state"] = $_POST["state"];
```

```
 $myorder["country"] = $_POST["country"];
 $myorder["phone"] = $_POST["phone"];
 $myorder["fax"] = $_POST["fax"];
 $myorder["email"] = $_POST["email"];
 $myorder["addrnum"] = $_POST["addrnum"];
 $myorder["zip"] = $_POST["zip"];

 # SHIPPING INFO
 $myorder["sname"] = $_POST["sname"];
 $myorder["saddress1"] = $_POST["saddress1"];
 $myorder["saddress2"] = $_POST["saddress2"];
 $myorder["scity"] = $_POST["scity"];
 $myorder["sstate"] = $_POST["sstate"];
 $myorder["szip"] = $_POST["szip"];
 $myorder["scountry"] = $_POST["scountry"];
 # SHIPPING CALCULATION
 $myorder["sweight"] = $_POST["sweight"];
 $myorder["sitems"] = $_POST["sitems"];
 $myorder["scarrier"] = $_POST["scarrier"];
 $myorder["stotal"] = $_POST["stotal"];

 # MISC
 $myorder["comments"] = $_POST["comments"];
 $myorder["referred"] = $_POST["referred"];

 # ITEMS AND OPTIONS
 # there are several ways to pass items and options; see sample SALE_MAXINFO.php

 $myorder["items"][item1]["id"] = $_POST["id"];
 $myorder["items"][item1]["description"] = $_POST["description"];
 $myorder["items"][item1]["quantity"] = $_POST["quantity"];
 $myorder["items"][item1]["price"] = $_POST["price"];

 $myorder["items"][item1]["option1"]["name"] = $_POST["name1"];
 $myorder["items"][item1]["option1"]["value"] = $_POST["value1"];
 $myorder["items"][item1]["option2"]["name"] = $_POST["name2"];
 $myorder["items"][item1]["option2"]["value"] = $_POST["value2"];

 if ($_POST["debugging"])
   $myorder["debugging"]="true";



# Send transaction. Use one of two possible methods
# %response = $lperl->process($myorder); # use shared library model
%response = $lperl->curl_process($myorder); # or use curl methods
```

```
  if ($result["r_approved"] != "APPROVED") // transaction failed, print the reason
  {
    print "Status: $result[r_approved]<br>\n";
    print "Error: $result[r_error]<br><br>\n";
  }
  else  // success
  {
    print "Status: $result[r_approved]<br>\n";
    print "Transaction Code: $result[r_code]<br><br>\n";
  }

# if verbose output has been checked,
# print complete server response to a table
  if ($_POST["verbose"])
  {
    echo "<table border=1>";

    while (list($key, $value) = each($result))
    {
      # print the returned hash
      echo "<tr>";
      echo "<td>" . htmlspecialchars($key) . "</td>";
      echo "<td><b>" . htmlspecialchars($value) . "</b></td>";
      echo "</tr>";
    }

    echo "</table><br>\n";
  }
?>

</body></html>
```

Sample Code to Process a Sale (Minimum Info) Using PHP

The code shown below demonstrates how to process a credit card sale. This example includes only the minimum fields required for a SALE transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

/***********************************************************************\

  SALE_MININFO.php - Minimum Required Fields for a Credit Card SALE


  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\***********************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["result"] = "LIVE"; # For a test, set result to GOOD, DECLINE, or
```

```
DUPLICATE
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";
  $myorder["chargetotal"] = "9.99";

  $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["zip"] = "12345"; # Required for AVS. If not provided, transactions will
downgrade.
//  $myorder["debugging"] = "true"; # for development only - not intended for production
use



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder);    # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {  // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  // Look at returned hash & use the elements you need //
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a Sale (Maximum Info) Using PHP

The code shown below demonstrates how to process a credit card sale. This example includes samples of all the commonly used fields you can use in a SALE transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

/*************************************************************************\

  SALE_MININFO.php - Minimum Required Fields for a Credit Card SALE


  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location
of your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["result"] = "LIVE"; # For test transactions, set to GOOD, DECLINE, or
```

```
DUPLICATE
  $myorder["transactionorigin"] = "MOTO"; # For credit card retail txns, set to RETAIL,
for Mail order/telephone order, set to MOTO, for e-commerce, leave out or set to ECI
  $myorder["oid"] = "12345678-A345"; # Order ID number must be unique. If not set,
gateway will assign one.
  $myorder["ponumber"] = "09876543-Q1234";
  $myorder["taxexempt"] = "N";
  $myorder["terminaltype"] = "UNSPECIFIED"; # Set terminaltype to POS for an
electronic cash register or integrated POS system, STANDALONE for a point-of-sale
credit card terminal, UNATTENDED for a self-service station, or UNSPECIFIED for
e-commerce or other applications
  $myorder["ip"] = "123.123.123.123";

  $myorder["subtotal"] = "12.99";
  $myorder["tax"] = "0.34";
  $myorder["shipping"] = "1.45";
  $myorder["vattax"] = "0.00";
  $myorder["chargetotal"] = "14.78";

  # CARD INFO
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";
  $myorder["cvmindicator"] = "provided";
  $myorder["cvmvalue"] = "123";

  # BILLING INFO
  $myorder["name"] = "Joe Customer";
  $myorder["company"] = "123 Broadway";
  $myorder["address1"] = "Suite 23v";
  $myorder["address2"] = "";
  $myorder["city"] = "Moorpark";
  $myorder["state"] = "CA";
  $myorder["country"] = "US";
  $myorder["phone"] = "8051234567";
  $myorder["fax"] = "8059876543";
  $myorder["email"] = "joe.customer@somewhere.com";
  $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["zip"] = "87123"; # Required for AVS. If not provided, transactions will
downgrade.

  # SHIPPING INFO
  $myorder["sname"] = "Joe Customer";
  $myorder["saddress1"] = "123 Broadway";
  $myorder["saddress2"] = "'Suite 23";
```

```
 $myorder["scity"] = "Moorpark";
 $myorder["sstate"] = "CA";
 $myorder["szip"] = "12345";
 $myorder["scountry"] = "US";

 # ITEMS AND OPTIONS
 $items = array (
   'id'       => '123456-A98765',
   'description'   => 'Logo T-Shirt',
   'quantity'     => '1',
   'price'      => '12.99',
   'options' => array
        (
        'name' => 'Color',
        'value'=> 'Red'
        ),
   'options2' => array
       (
       'name' => 'Size',
       'value' => 'XL'
       )
   );

 $myorder["items"][0] = $items; # put array of items into hash

 # you could also submit this same item w/ options like this:
/*
 $myorder["items"][item1]["id"] = "123456-A98765";
 $myorder["items"][item1]["description"] = "Logo T-Shirt";
 $myorder["items"][item1]["quantity"] = "1";
 $myorder["items"][item1]["price"] = "12.99";

   # pass options in like this:
   $myorder["items"][item1]["option1"]["name"] = "Color";
   $myorder["items"][item1]["option1"]["value"] = "Red";
   $myorder["items"][item1]["option2"]["name"] = "Size";
   $myorder["items"][item1]["option2"]["value"] = "XL";

   # or you could alternately pass the same options in like this:
#    $myorder["items"][item1][0]["name"] = "Color";
#    $myorder["items"][item1][0]["value"] = "Red";
#    $myorder["items"][item1][1]["name"] = "Size";
#    $myorder["items"][item1][1]["value"] = "XL";
*/
```

```
  # MISC
  $myorder["comments"] = "Repeat customer. Ship immediately.";
  $myorder["referred"] = "Saw ad on Web site.";
//  $myorder["debugging"] = "true"; # for development only - not intended for
production use


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  # (if you're in web space, look at response like this):
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a PreAuth Transaction Using PHP

The code shown below demonstrates how to process a PreAuthorization Transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*****************************************************************************\

  PREAUTH.php - Minimum Required Fields for a PreAuth

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*****************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "PREAUTH";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["zip"] = "12345";  # Required for AVS. If not provided, transactions will
downgrade.


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a PostAuth Transaction Using PHP

The code shown below demonstrates how to process a PostAuthorization Transaction (that is, to complete a PreAuthorization with a valid order ID). The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*****************************************************************************\

  POSTAUTH.php - Minimum Required Fields for a PostAuth

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*****************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "POSTAUTH";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  $myorder["oid"] = "12345678-A345"; # Must be a valid order ID from a prior Sale or
PreAuth


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code for Using AVS and CVM with PHP

The code shown below demonstrates how to process a credit card sale taking advantage of both the AVS and the CVM fraud protection measures. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  AVS_CVM.php - An Example Credit Card SALE With Minimum Fields required for
AVS and Card Code fraud prevention measures

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

$myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  $myorder["cvmindicator"] = "provided";
  $myorder["cvmvalue"] = "123";  # CVM is the three-digit security code usually found
on the signature line on the back of the card
  $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["zip"] = "12345";  # Required for AVS. If not provided, transactions will
downgrade.



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to run a FORCED TICKET Transaction using PHP

The code shown below demonstrates how to process a PostAuthorization for a Voice Authorization (a Forced Ticket transaction). The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/************************************************************************\

  FORCED_TICKET.php - Minimum Required Fields for a FORCED TICKET
Transaction

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "POSTAUTH";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  $myorder["reference_number"] = "NEW987654"; # The reference number given over
the phone


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a Return Using PHP

The code shown below demonstrates how to process a credit card return. To perform a return, you must have a valid order ID (**oid**) from a prior SALE or PREAUTH/POSTAUTH. The example uses a hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*********************************************************************\

  RETURN.php - Minimum Required Fields for a Credit Card CREDIT

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*********************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location
of your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  # Amount returned must be less than or equal to the order amount.
  # If there is more than one return against this order, make sure
  #the total of the returns doesn't exceed the original sale amount.
  $myorder["chargetotal"] = "12.99";
  $myorder["ordertype"] = "CREDIT";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  #Must be a valid order ID from a prior Sale
  $myorder["oid"] = "12345678-A345";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to perform a Credit Card VOID Transaction Using PHP

The code shown below demonstrates how to process a credit card VOID transaction. You can only void a credit card SALE transaction if the associated credit card batch has not yet been processed (that is, the SALE was performed the same day as the VOID. VirtualCheck transactions are automatically sent to the banking system at the end of each day. No VOIDs are allowed once the transaction is in the banking system.) You will need the Order ID (**oid**) from the SALE transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*********************************************************************\

  VOID.php - Minimum Required Fields for a Credit Card VOID

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*********************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location
of your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "VOID";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  #Must be a valid order ID from a prior Sale or PreAuth
  $myorder["oid"] = "12345678-A345";


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to run a Credit Card Transaction using Items (Including ESD) with PHP

The code shown below demonstrates how to process a Sale using the Items and Options feature. One of the items included in the example is an Electronic Softgood Download (ESD) item. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

/*********************************************************************************\

  ITEMS_W_ESD.php - Required Fields for an Example Credit Card SALE with ITEMS

Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*********************************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of your
certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
```

```
 $myorder["subtotal"] = "45.98";
 $myorder["tax"] = "0.32";
 $myorder["shipping"] = "1.02";
 $myorder["chargetotal"] = "47.32";
 $myorder["cardnumber"] = "4111-1111-1111-1111";
 $myorder["cardexpmonth"] = "01";
 $myorder["cardexpyear"] = "05";
 $myorder["name"] = "Joe Customer"; # If you use ESD items, name is required
 $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will downgrade.
 $myorder["zip"] = "87123"; # Required for AVS. If not provided, transactions will downgrade.


 # ITEMS AND OPTIONS
 $item0 = array (
   'id'       => '123456-A98765',
   'description'   => 'Logo T-Shirt',
   'quantity'     => '1',
   'price'      => '12.99',
   'serial'    => '0987654321',
   'options' => array
        (
        'name' => 'Color',
        'value'=> 'Red'
        ),
   'options2' => array
       (
       'name' => 'Size',
       'value' => 'XL'
       )
   );

 $myorder["items"][0] = $item0; # put array of items/options into hash

 # ITEM w/ ESD
$item1 = array (
   'id'       => '123456-B98765',
   'description'   => 'Blast-Em Software',
   'quantity'     => '1',
   'price'      => '12.99',
   'serial'    => '0987654321',
'esdtype' => 'Esdtype_Softgood',
'softfile' => 'blastemgame.exe',
   );

 $myorder["items"][1] = $item1; # put item array into hash
```

```
  # you could also submit the same item w/options like this:
/*
  $myorder["items"][item1]["id"] = "123456-A98765";
  $myorder["items"][item1]["description"] = "Logo T-Shirt";
  $myorder["items"][item1]["quantity"] = "1";
  $myorder["items"][item1]["price"] = "12.99";

    # pass options in like this:
    $myorder["items"][item1]["option1"]["name"] = "Color";
    $myorder["items"][item1]["option1"]["value"] = "Red";
    $myorder["items"][item1]["option2"]["name"] = "Size";
    $myorder["items"][item1]["option2"]["value"] = "XL";

    # or you could alternately build arrays like this:
#    $myorder["items"][item1][0]["name"] = "Color";
#    $myorder["items"][item1][0]["value"] = "Red";
#    $myorder["items"][item1][1]["name"] = "Size";
#    $myorder["items"][item1][1]["value"] = "XL";
*/



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  # (if you're in web space, look at response like this):
    echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to run a Level 2 Purchasing Card Transaction with PHP

The code shown below demonstrates how to process a Sale including all the data elements needed for a Level 2 purchasing card transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php


/*********************************************************************\

  L2PURCHASING_CARD.php - An Example Level 2 Purchasing Card SALE


  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*********************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
```

```
 $myorder["tax"] = "0.32";
 $myorder["chargetotal"] = "47.32";
 $myorder["cardnumber"] = "4111-1111-1111-1111";
 $myorder["cardexpmonth"] = "01";
 $myorder["cardexpyear"] = "05";
 $myorder["chargetotal"] = "9.99";
 $myorder["ponumber"] = "1203A-G4567";
 $myorder["taxexempt"] = "N";


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
 $result = $mylphp->curl_process($myorder); # use curl methods


 if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
 {
   print "Status: $result[r_approved]\n";
   print "Error: $result[r_error]\n";
 }
 else
 {          // success
   print "Status: $result[r_approved]\n";
   print "Code: $result[r_code]\n";
   print "OID: $result[r_ordernum]\n\n";
 }

/*
 # Look at returned hash & use the elements you need #
 while (list($key, $value) = each($result))
 {
   echo "$key = $value\n";

 #if you're in web space, look at response like this:
    echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
 }
*/
?>
```

Sample Code to run a Batch of Transactions using PHP

The code shown below demonstrates how to send multiple credit card transactions to the Gateway for processing. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

/*************************************************************************\

# MULTIPLE.php - Sample of multiple processing
#
# This sample demonstrates running an array of PREAUTH transactions.
#
# This could equally well be a series of POSTAUTHS or RETURNS if the
# OIDs match those previously approved.

Note: if you are using the shared object LIBLPHP.SO and are getting
PHP warings "Function registration failed..." while processing batches,
you can add this entry to your php.ini file:
      extension=liblphp.so
and then comment out the lines to load the library at LIBLPHP.PHP
module at about line # 57.

# Copyright 2003 LinkPoint International, Inc. All Rights Reserved.
#
# This software is the proprietary information of LinkPoint International, Inc.
# Use is subject to license terms.

\*************************************************************************/
```

```
  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "PREAUTH";
  $myorder["result"] = "good"; # For a test, set result to GOOD, DECLINE, or
DUPLICATE
//  $myorder["debugging"] = "true";


  # build array of individual orders
  $order0 = array (
    'chargetotal' => '1.11',
    'cardnumber' => '4111-1111-1111-1111',
    'cardexpmonth' => '01',
    'cardexpyear' => '04',
    'addrnum' => '123', # Required for AVS. If not provided, transactions will downgrade.
    'zip' => '12345', # Required for AVS. If not provided, transactions will downgrade.
#    'oid' => '12345678-A1'
    );

  $myorder["orders"][0] = $order0; # put array into hash

  $order1 = array (
    'chargetotal' => '2.22',
    'cardnumber' => '4111-1111-1111-1111',
    'cardexpmonth' => '01',
    'cardexpyear' => '02',
    'addrnum' => '123', # Required for AVS. If not provided, transactions will downgrade.
    'zip' => '23456', # Required for AVS. If not provided, transactions will downgrade.
#    'oid' => '12345678-B2'
    );

  $myorder["orders"][1] = $order1; # put array into hash

  $order2 = array (
    'chargetotal' => '3.33',
    'cardnumber' => '4111-1111-1111-1111',
    'cardexpmonth' => '01',
    'cardexpyear' => '04',
    'addrnum' => '123', # Required for AVS. If not provided, transactions will downgrade.
```

```
   'zip' => '34567', # Required for AVS. If not provided, transactions will downgrade.
#    'oid' => '12345678-C3'
   );

 $myorder["orders"][2] = $order2; # put array into hash


 # PROCESS ORDERS ARRAYS #

 while (list ($key, $val) = each ($myorder))
 {
   if (is_array($val))
   {
     $orders_array = $val;

     while (list ($akey, $aval) = each ( $orders_array))
     {
       $oa2 = $aval;

       # CONSTANTS
       $txn["host"] = $myorder["host"];
       $txn["port"] = $myorder["port"];
       $txn["keyfile"] = $myorder["keyfile"];
       $txn["configfile"] = $myorder["configfile"];
       $txn["debugging"] = $myorder["debugging"];

       # INDIVIDUAL ORDER ELEMENTS
       $txn["chargetotal"] = $oa2[chargetotal];
       $txn["cardnumber"] = $oa2[cardnumber];
       $txn["cardexpmonth"] = $oa2[cardexpmonth];
       $txn["cardexpyear"] = $oa2[cardexpyear];
       $txn["addrnum"] = $oa2[addrnum];
       $txn["zip"] = $oa2[zip];
       $txn["oid"] = $oa2[oid];

      # Send transaction. Use one of two possible methods #
       $result = $mylphp->process($txn);    # use shared library model
     //  $result = $mylphp->curl_process($txn); # use curl methods

       if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
       {
         print "Status: $result[r_approved]\n";
         print "Error: $result[r_error]\n";
       }
       else
       {  // success
```

```
          print "Status: $result[r_approved]\n";
          print "Code: $result[r_code]\n";
          print "OID: $result[r_ordernum]\n";
        }
      }
    }
  }
?>
```

Sample Code to Set Up a New Recurring Credit Card Transaction Using PHP

The code shown below demonstrates how to process a Recurring Credit Card Transaction (Periodic Bill). The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  PB_NEW.php - Minimum Required Fields for a Recurring Credit Card Sale (Periodic Bill)

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "9.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";
  $myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["zip"] = "12345"; # Required for AVS. If not provided, transactions will
downgrade.
  $myorder["debugging"] = "true"; # for development only - not intended for production
use

  # periodic fields
  $myorder["action"] = "SUBMIT";
  $myorder["installments"] = "3";
  $myorder["threshold"] = "3";
  $myorder["startdate"] = "immediate";
  $myorder["periodicity"] = "monthly";


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Modify a Recurring Credit Card Transaction Using PHP

The code shown below demonstrates how to modify a Recurring Credit Card Transaction (Periodic Bill). The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*********************************************************************\

  PB_MODIFY.php - Minimum Required Fields to Modify a Recurring Credit Card Sale
(Periodic Bill)

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*********************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
```

```
 $myorder["cardexpyear"] = "05";

 # PERIODIC FIELDS

 # Set action to MODIFY to modify the periodic bill. You may modify ONLY the
parameters in the periodic entity
 $myorder["action"] = "MODIFY";
 # Changes will be in effect as of the time specified. No periodic bills will be sent until
the startdate specified here. If you don't want it to start today, pass a date in the format
YYYYMMDD
 $myorder["startdate"] = "immediate";
 # Specifies a recurring transaction charging the card 5 times, once a month.
 $myorder["installments"] = "5";
 $myorder["threshold"] = "3";
 $myorder["periodicity"] = "monthly";
 # You MUST pass a valid Order ID for an EXISTING periodic bill to identify which bill
you want to modify -->
 $myorder["oid"] = "111.456.87.03-O986646-A123";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
 $result = $mylphp->curl_process($myorder); # use curl methods



 if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
 {
  print "Status: $result[r_approved]\n";
  print "Error: $result[r_error]\n";
 }
 else
 {         // success
  print "Status: $result[r_approved]\n";
  print "Code: $result[r_code]\n";
  print "OID: $result[r_ordernum]\n\n";
 }

/*
 # Look at returned hash & use the elements you need #
 while (list($key, $value) = each($result))
 {
  echo "$key = $value\n";

 #if you're in web space, look at response like this:
   echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
 }
```

```
*/
?>
```

Sample Code to Cancel a Recurring Credit Card Transaction Using PHP

The code shown below demonstrates how to Cancel an existing Recurring Credit Card Transaction (Periodic Bill). The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  PB_CANCEL.php - Minimum Required Fields to Cancel a Recurring Credit Card Sale
(Periodic Bill)

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";

  # periodic fields
   # Set action to CANCEL to cancel the recurring transaction
  $myorder["action"] = "CANCEL";
  # Changes will be in effect as of the time specified. No periodic bills will be sent until
the startdate specified here.
  # If you don't want it to start today, pass a date in the format YYYYMMDD
$myorder["startdate"] = "immediate";
  # Specifies a recurring transaction charging the card 5 times, once a month.
  $myorder["installments"] = "5";
  $myorder["threshold"] = "3";
  $myorder["periodicity"] = "monthly";
  # You MUST pass a valid Order ID for an EXISTING periodic bill to identify which bill
you want to modify -->
  $myorder["oid"] = "111.456.87.03-O986646-A123";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
```

```
?>
```

Sample Code to Process a Retail Swipe Transaction Using PHP

The code shown below demonstrates how to process a Retail SALE Transaction where the credit card information is swiped with a credit card reader. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/************************************************************************\

  RETAIL_SWIPE.php - Minimum Required Fields for a Credit Card SALE

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["transactionorigin"] = "RETAIL";
  # Set terminaltype to POS for an electronic cash register or integrated POS system,
  # STANDALONE for a point-of-sale credit card terminal,
  # UNATTENDED for a self-service station,
  # or UNSPECIFIED for e-commerce or other applications
  $myorder["terminaltype"] = "POS";
  $myorder["chargetotal"] = "12.99";
  $myorder["track"] = "12983610236l4908";

  # You should not do AVS for a retail swiped transaction, unless it is an UNATTENDED
terminaltype.
  # billing addrnum and billing zip are not required because we're not doing AVS
  #$myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  #$myorder["zip"] = "12345";  # Required for AVS. If not provided, transactions will
downgrade.



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a Retail Keyed Transaction Using PHP

The code shown below demonstrates how to process a Retail SALE Transaction where the credit card information is keyed in. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*****************************************************************************\

  RETAIL_KEYED.php - Minimum Required Fields for a Credit Card SALE

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*****************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["transactionorigin"] = "RETAIL";
  # Set terminaltype to POS for an electronic cash register or integrated POS system,
  # STANDALONE for a point-of-sale credit card terminal,
  # UNATTENDED for a self-service station,
  # or UNSPECIFIED for e-commerce or other applications
  $myorder["terminaltype"] = "POS";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";

  # You should not do AVS for a retail swiped transaction, unless it is an UNATTENDED
terminaltype.
  # billing addrnum and billing zip are not required because we're not doing AVS
  #$myorder["addrnum"] = "123"; # Required for AVS. If not provided, transactions will
downgrade.
  #$myorder["zip"] = "12345";  # Required for AVS. If not provided, transactions will
downgrade.



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Process a Retail UNATTENDED Transaction Using PHP

The code shown below demonstrates how to process a Retail SALE Transaction where the credit card terminal is unattended (that is, a self-service station such as a gas station or ticket booth). In this case, you should do a partial AVS where you collect the zip code only. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  RETAIL_PARTIAL_AVS.php - Minimum Required Fields for a Credit Card SALE

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "12.99";
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "03";
  $myorder["cardexpyear"] = "05";
  $myorder["chargetotal"] = "12.99";

  $myorder["transactionorigin"] = "RETAIL";
  # Set terminaltype to POS for an electronic cash register or integrated POS system,
  # STANDALONE for a point-of-sale credit card terminal,
  # UNATTENDED for a self-service station,
  # or UNSPECIFIED for e-commerce or other applications
  $myorder["terminaltype"] = "UNATTENDED";

  #For an UNATTENDED terminaltype, do a partial AVS: zip code only, no addrnum
  $myorder["zip"] = "12345";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods



  if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {         // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to perform a VirtualCheck SALE Transaction Using PHP

The code shown below demonstrates how to process a VirtualCheck SALE transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  VCHECK_SALE.php - Minimum Required Fields for a VirtualCheck SALE

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["chargetotal"] = "9.99";

  # Customer's Driver's license # and DL state
  $myorder["dl"] = "120381698";
  $myorder["dlstate"] = "CA";

  # Transit routing number for the customer's bank
  $myorder["routing"] = "123456789";

  # Customer's bank account number
  $myorder["account"] = "2139842610";

  # Is this a business or personal account? personal = pc, business = bc
  $myorder["accounttype"] = "pc";

# Bank name and 2-letter bank state
  $myorder["bankname"] = "MyBank";
  $myorder["bankstate"] = "CA";

  # Additional required VCHECK fields
  $myorder["name"] = "Joe Customer";
  $myorder["address1"] = "123 Broadway";
  $myorder["city"] = "Moorpark";
  $myorder["state"] = "CA";
  $myorder["zip"] = "12345";
  $myorder["phone"] = "8051234567";
  $myorder["email"] = "joe.customer@somewhere.com";  # optional field


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($result["r_approved"] != "SUBMITTED")  // transaction failed, print the reason
  {
    print "Approved: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else // success
  {
    print "Approved: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to perform a VirtualCheck VOID Transaction Using PHP

The code shown below demonstrates how to process a VirtualCheck VOID transaction. You can only void a VirtualCheck SALE transaction if it has not yet been sent for processing (that is, the SALE was performed the same day as the VOID. VirtualCheck transactions are automatically sent to the banking system at the end of each day. No VOIDs are allowed once the transaction is in the banking system.) You will need the Order ID (**oid**) from the SALE transaction. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  VCHECK_VOID.php - Minimum Required Fields for a VirtualCheck VOID

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "VOID";
  $myorder["chargetotal"] = "12.99"; # You must match previous transaction amount
  $myorder["oid"] = "0987654321"; # You must have a valid order ID from a previous
VCheck SALE transaction that has not yet been processed
  $myorder["voidcheck"] = "true";


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods

  if ($result["r_approved"] != "SUBMITTED")  // transaction failed, print the reason
  {
    print "Status: $result[r_approved]\n";
    print "Error: $result[r_error]\n";
  }
  else
  {          // success
    print "Status: $result[r_approved]\n";
    print "Code: $result[r_code]\n";
    print "OID: $result[r_ordernum]\n\n";
  }

/*
  # Look at returned hash & use the elements you need #
  while (list($key, $value) = each($result))
  {
    echo "$key = $value\n";

  #if you're in web space, look at response like this:
     echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<BR>\n";
  }
*/
?>
```

Sample Code to Calculate Shipping Charges Using PHP

The code shown below demonstrates how to use the shipping calculator. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/***************************************************************************\

  SHIPPING.php - An Example shipping calculation

  NOTE: your store must be explicitly set up before doing shipping calculations

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\***************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number
  $myorder["ordertype"] = "CALCSHIPPING";

  # Shipping fields
  $myorder["scarrier"] = "2";
  $myorder["sweight"] = "1.2";
  $myorder["sitems"] = "1";
  $myorder["stotal"] = "12.99";
  $myorder["sstate"] = "CA";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods

  if ($result["r_shipping"])
  {
    print "Shipping: $result[r_shipping]\n";
  }
  else
  {
    print "Error: $result[r_error]\n";
  }
?>
```

Sample Code to Calculate Sales Tax Using PHP

The code shown below demonstrates how to use the Tax calculator. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php
/*************************************************************************\

  TAX.php - An Example tax calculation

  NOTE: your store must be explicitly set up before doing tax calculations

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\*************************************************************************/



  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of
your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "CALCTAX";
  $myorder["subtotal"] = "12.99";
  # Shipping fields
  $myorder["stotal"] = "12.99";
  $myorder["sstate"] = "CA";
  $myorder["szip"] = "91367";



# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder); # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods

  if ($result["r_tax"])
  {
    print "Tax: $result[r_tax]\n";
  }
  else
  {
    print "Error: $result[r_error]\n";
  }
?>
```

Sample Code to Process a Sale (with extra arguments) Using PHP

The code shown below demonstrates how to process a credit card sale. This example demonstrates various optional arguments that can passed into the PHP module.. The example uses a PHP hash table. See the comments within the code for further explanation.

If you copy this code for your own use, please make sure you change the values to be specific to your merchant store. That is, at a minimum, make sure you change:

1. "**1234567**" to your merchant store number (or storename) as specified in the merchant's Welcome e-mail,
2. "**secure.linkpt.net**" to the host name specified in your Welcome e-mail (if applicable), and
3. "**./YOURCERT.pem**" to the location and file name of the digital certificate, which should be saved on your Web server with a ***.pem*** extension.

Sample Code

```
<?php

/************************************************************************\

  XTRA_ARGS.php - Simple SALE transaction demonstrating various optional
  arguments that can passed into the LPHP.PHP module

  Copyright 2003 LinkPoint International, Inc. All Rights Reserved.

  This software is the proprietary information of LinkPoint International, Inc.
  Use is subject to license terms.

\************************************************************************/


  include"lphp.php";
  $mylphp=new lphp;

  $myorder["host"] = "secure.linkpt.net";
  $myorder["port"] = "1129";
  $myorder["keyfile"] = "./YOURCERT.pem"; # Change this to the name and location of your certificate file
  $myorder["configfile"] = "1234567"; # Change this to your store number

  $myorder["ordertype"] = "SALE";
  $myorder["result"] = "GOOD"; # For a test, set result to GOOD, DECLINE, or DUPLICATE
  $myorder["cardnumber"] = "4111-1111-1111-1111";
  $myorder["cardexpmonth"] = "01";
  $myorder["cardexpyear"] = "05";
  $myorder["chargetotal"] = "9.99";


/* FIELD NAME:    cbin - Curl Binary
DEFAULT VALUE:    false
EXPLANATION:    Curl binary - tells the module to use curl binary executable, as opposed to the
        PHP built-in curl methods. This should be used only when (often older)
        PHP version does not support curl directly so we have to shell out.
EXAMPLE USAGE:    */
  $myorder["cbin"] = "true";


/* FIELD NAME:    cpath - Curl Path
DEFAULT VALUE: /usr/bin/curl
EXPLANATION: Use this if curl binary executable is in a nonstandard location
         Only to be used in conjunction with "cbin" argument.
EXAMPLE USAGE:    */
//  $myorder["cpath"] = "/usr/local/bin/curl";


/* FIELD NAME:    cargs - Curl Arguments
DEFAULT VALUE:    -m 300 -s -S
EXPLANATION:    Curl arguments - only if using binary curl ( cbin above).
         You need to enter the complete string if you over-ride the default.
         For more info: http://curl.haxx.se/docs/manpage.html.
Note: *don't use this field unless you have a specific reason and
         know exactly what you're doing*
EXAMPLE USAGE:    To decrease the timeout to 2 minutes (-m) and disable certificate
         verification (-k) use this string:*/
  $myorder["cargs"] = "-m 120 -k -s -S";


/* FIELD NAME:    xml - use XML processing
DEFAULT VALUE:    none
EXPLANATION:    If this field is present, we are passing an xml string directly
         into lphp.php and the server response will also be in xml format.
         Hash values passed in will be ignored, except for host, port and keyfile.
EXAMPLE USAGE:    See the LinkPoint sample "PASS_XML.php" for example usage. */
//  $myorder["xml"] =
"<order><orderoptions><result>GOOD</result><ordertype>SALE</ordertype></orderoptions><merchantinfo><configfile>909001</configfile></merchantinfo><creditcard><cardnumber>4111111111111111</cardnumber><cardexpmonth>12</cardexpmonth><cardexpyear>08</cardexpyear></creditcard><payment><chargetotal>1.03</chargetotal></payment></order>";


/* FIELD NAME:    debugging
DEFAULT VALUE:    false
EXPLANATION:    Prints values as they are processed by module. It also turns on
         verbose connection output if curl methods are being used.
         Debugging is for development only - not intended for production use
EXAMPLE USAGE:    */
  $myorder["debugging"] = "true";


/* FIELD NAME:    webspace
DEFAULT VALUE:    true
EXPLANATION:    When viewing PHP output through a web browser, all xml tags are
         removed. This PHP call allows you to view debugging output with XML tag
         names. If running PHP directly from the command line (not in browser),
         set this field to false so as to see complete xml output.
EXAMPLE USAGE:    */
```

```
//  $myorder["webspace"] = "false";


# Send transaction. Use one of two possible methods #
//  $result = $mylphp->process($myorder);    # use shared library model
  $result = $mylphp->curl_process($myorder); # use curl methods


  if ($myorder["xml"])  // we're processing xml string response
  {
    # break it into array
    preg_match_all ("/<(.*?)>(.*?)\</", $result, $outarr, PREG_SET_ORDER);

    $n = 0;
    while (isset($outarr[$n]))
    {
      $retarr[$outarr[$n][1]] = strip_tags($outarr[$n][0]);
      $n++;
    }

    # and then look at it like this
    while (list($key, $value) = each($retarr))
    {
      if($myorder["htmlspecialchars"] == "false")
      {
        echo "$key = $value\n";
      }
      else
      {
        echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<br>\n";
      }
    }
  }
  else    // we're processing a hash response
  {
    if ($result["r_approved"] != "APPROVED")  // transaction failed, print the reason
    {
      print "Approved: $result[r_approved]\n";
      print "Error: $result[r_error]\n";
    }
    else
    {  // success
      print "Approved: $result[r_approved]\n";
      print "Code: $result[r_code]\n";
      print "OID: $result[r_ordernum]\n\n";
    }

    // Look at returned hash & use the elements you need //
/*    while (list($key, $value) = each($result))
    {
      if(!$myorder["webspace"]){
        echo "$key = $value\n";}
      else{
        echo htmlspecialchars($key) . " = " . htmlspecialchars($value) . "<br>\n";}
    }
*/
  }
?>
```

API Frequently Asked Questions (FAQs)

How to troubleshoot the following errors:

What is the API?

The API is a tool for the merchant that needs a custom commerce solution. It is an application programming interface (API) used by merchants to build complex Web sites or other custom systems that process payments. There are also tax and shipping calculators and an Electronic Softgood Download module available for use with the API. The API enables full-featured, highly secure and reliable e-commerce Web sites and custom retail implementations. The API includes LinkPoint Central for comprehensive store management.

Top

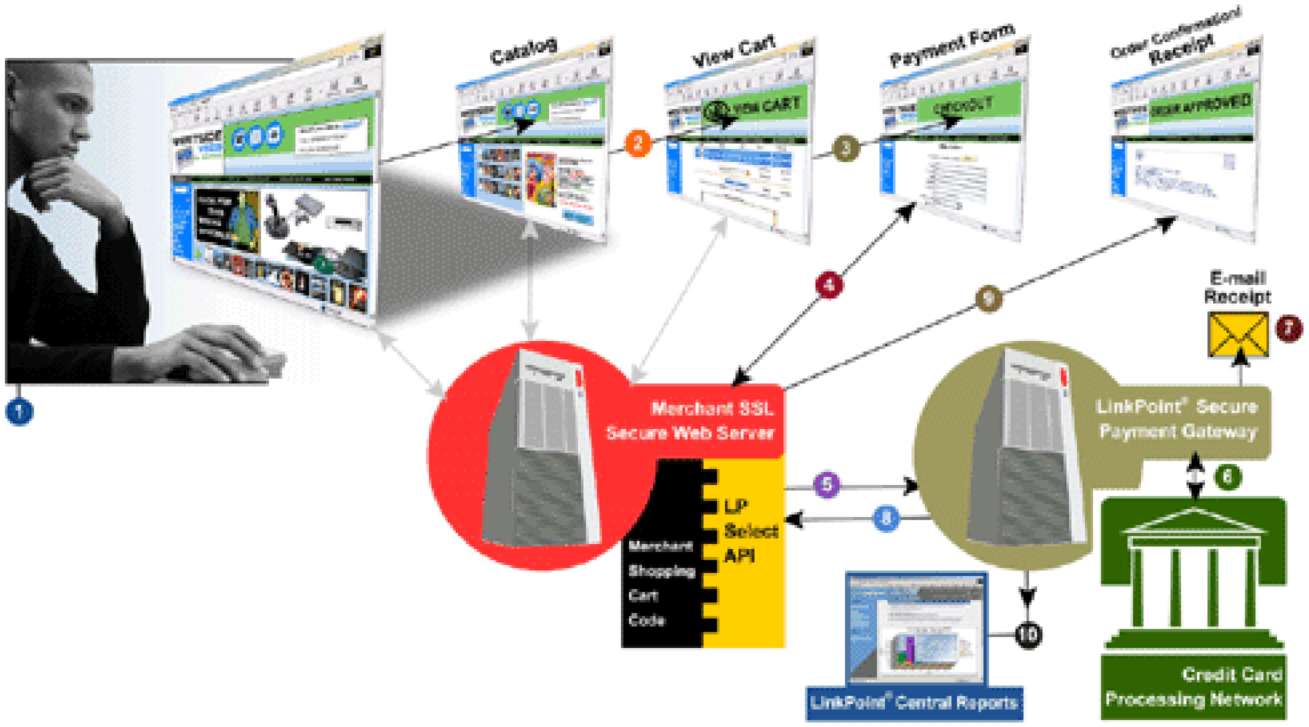
How does the API work?

A merchant uses the API software modules to build the payment solution that fits the merchant's unique needs. Modules offered are:

- The payment module
- The shipping calculator module
- The tax calculator module
- The ESD module
- The VirtualCheck module
- The Recurring payments module

Merchants can pick and choose modules as needed. For credit card payments, all of the interactions with credit card processors and financial institutions are resolved and managed through the API payment module. For electronic check payments, all of the interactions with financial institutions are resolved and managed through the API VirtualCheck module.

Once a merchant has integrated the API payment module into a Web site, a customer can purchase items on the merchant's Web site and all payment details are processed automatically. Merchants can then review transaction activity by visiting LinkPoint Central. Let's follow a simple transaction as it travels from the customer's order to approval.


Catalog
View Cart
Payment Form
Order Confirmation/ Receipt
E-mail Receipt
Merchant SSL Secure Web Server
LinkPoint® Secure Payment Gateway
LP Select API
Merchant Shopping Cart Code
LinkPoint® Central Reports
Credit Card Processing Network
1
2
3
4
5
6
7
8
9
10

1. A customer selects items for purchase from a merchant's online store over the Internet.
2. The customer clicks on the **View Cart** button and finalizes the order.
3. The customer clicks on the **Check Out** button and proceeds to the merchant's payment page.
4. The ordering information is received by the merchant's CSP or by the merchant's own server (self-hosting).
5. The merchant's customized software receives the information, uses the tax and shipping calculators as needed, calculates the order total and passes relevant order and payment information to the API payment module. The API payment module funnels the order data through the secure sockets layer (SSL) pipeline to the LinkPoint Secure Payment Gateway.
6. The LinkPoint Secure Payment Gateway calls and transmits the data through a dedicated, secure

frame relay system to the banking network. (Once the transaction is approved and settled, funds are withdrawn from the credit card-issuing bank and deposited into the merchant bank.)

7. A confirmation e-mail is sent to the merchant and the purchaser when the transaction is completed.
8. The merchant's Web server receives the transaction response information. This generally happens within six seconds.
9. The merchant's Web server displays the transaction results to the customer.
10. The merchant uses LinkPoint Central reports to review transactions and mark items as shipped (e.g., ready for settlement).

Top

What security measures are used?

The Gateway provides data encryption, server authentication, message integrity and optional client authentication for a TCP/IP connection by the Secure Sockets Layer (SSL) protocol. SSL is a protocol developed by Netscape® to provide secure transmission of private information that is sent over the Internet. This protocol uses public and private key pairs to encrypt data. The public and private keys are dissimilar and each pair is unique; therefore, the keys possessed verify the sender's identity. The public key is distributed to the merchant, ISP or CSP in the form of a digital certificate, which contains information that can verify the key holder identity and the key validity. The private key is kept confidential and remains on the Payment Gateway. If data is encrypted with the private key, only the public key can decrypt it. If data is encrypted with the public key, only the private key can decrypt it. This process prevents the data from being compromised while in transit.

Top

What do I need to know to use the API?

You need to have some programming skills to use the API. Any of the following languages can be used to invoke the API:

- C/C++
- C#
- ASP, Visual Basic, or VBScript
- PERL
- PHP
- .Net
- XML
- Java^{TM}

Top

How do I get the API and/or wrapper software?

All the LinkPoint software and related documentation is downloadable from the downloads section of http://www.linkpoint.com. You will need a live merchant account with a LinkPoint-compatible merchant service provider or a LinkPoint test account before you can use the LinkPoint software.

Top

Do you offer a Software Development Kit (SDK)?

We do not offer SDKs, but we offer the software modules in many different programming languages.

Top

What fraud protection measures are included?

Fraud protection measures included with all LinkPoint payment services are:

- Address Verification System (AVS)
- Card verification schemes (CVV2 and CVC2)
- Merchant-configurable fraud protection. Merchants can block customer names, IP or class C addresses, domain names, and specific credit card numbers. Merchants can also set automatic lockout times. These are set in the **Admin** section of LinkPoint Central.

For more information on each of these fraud protection capabilities, please refer to the LinkPoint Central Help files or the LPC user manual.

Top

How can I change my fraud protection settings?

Log into https://www.linkpointcentral.com. Click on **Admin**, then on **Fraud Settings**. Please refer to the LinkPoint Central Help files or user manual for further instructions.

Top

Is there a way for me to review transaction activity for my store?

Yes. Log into https://www.linkpointcentral.com. Click on **Reports** and choose your preferred report. Please refer to the LinkPoint Central Help files or user manual for further instructions.

Top

What is included in the welcome e-mail?

When your application is approved, you will receive an introductory e-mail titled "Welcome to LinkPoint Select API". The following information is included in this e-mail:

- your DBA store name
- your store number (represented by a 10-Digit number) which is required to obtain your password
- your User ID number
- your secure Host Name and Port Number
- your digital certificate (embedded in text within the e-mail or at a URL to which you are directed).
- contact information for questions or problems

Top

What if I lose my welcome e-mail?

Welcome e-mails can be resent if the original e-mails are lost. Please contact support@linkpoint.com to resend your welcome e-mail.

Top

How do I get a pem file?

The owner of the account should have recieved a Welcome e-mail when the mercahnt account was opened. You must have a LinkPoint Select API account in order to recieve this e-mail. If you lost this e-mail, you will have to call your merchant account provider to have the e-mail resent at 1-800 456-5989 X4100 (human interaction is a security requirement). At the very end of the message (embedded in the e-mail text) will be the digital certificate, which is your pem file. Follow the instructions given in the e-mail exactly to save the digital certificate into a file on your Web server with a ***.pem*** extension. Note the location (path) to the pem file.

Top

How do I create my pem file?

At the bottom of the Welcome e-mail is a copy of your digital certificate. If you are using an ISP, a copy of this e-mail should have been e-mailed to your ISP. This certificate should be saved as a file on your Web server with a .pem extension. Copy all characters beginning with the line that states, "-----BEGIN RSA PRIVATE KEY-----" through the end of this e-mail where it states, "-----END CERTIFICATE-----" into a file with a ***.pem*** extension onto your Web server. Note the path to the pem file.

Top

What is a digital certificate? How do I get mine? Is that the only certificate I need for my store?

A digital certificate is a key used on the Internet to identify a company or organization and secure transactions between two parties. When you sign up for the LinkPoint Select API service, you receive a digital certificate. The LinkPoint Select API software uses this certificate to send SSL-secured payment transactions to the LinkPoint Secure Payment Gateway.

The certificate you receive with the LinkPoint Select API solution is used ONLY for LinkPoint Select API purposes--it does not make your Web server secure. You will need to obtain a digital certificate from a certificate authority in order to secure your Web server. Your Web server should be properly secured whenever collecting sensitive information from your customers.

Top

Can you send me the digital certificate?

The digital certificate is sent in the Welcome e-mail to the merchant and the merchant's ISP or CSP. The Welcome e-mail can be resent upon request. For security purposes, we can only send the certificate to those parties authorized on the merchant account. If you are a developer working for a merchant and have not received the digital certificate, you should ask the merchant to forward the "Welcome to LinkPoint Select API" e-mail to you. Make sure you ask the merchant to send the message as an attachment to ensure that no extra characters are inserted into the digital certificate.

Top

How do I install the digital certificate?

While the installation is fairly simple, it varies depending on the scenario. Basically, you need to copy the digital certificate on the welcome letter as a whole; do not break up the RSA and the Certificate-copy it in one piece.

Follow these instructions carefully:

- Start copying exactly at the first dash of -------Begin RSA and end exactly after the last dash of END CERTIFICATE----- with no extra spaces included.
- Paste it in a text utility and save the file with a .Pem extension. We recommend you use the store number.pem; e.g., 1234123456.pem.
- Review the file name to ensure it has a .pem extension. Make sure you computer view displays all file extensions with no hidden extensions (1234123456.pem.txt will not work). In the Windows® operating system, under My Computer, select View- Options-View (tab) Hidden files -mark the Show all files and uncheck the box for Hide file Extensions for known file types).
- If you have a hosted site, you will need to find out from your Web hosting company the directory and folder to which you must upload the file via FTP. Some hosts have an area located in the shopping cart where you copy and paste the file instead of using FTP.
- If you host a site on your own Web server, then the file needs to go in the file system where your Web server can access it.

Top

I'm using Java™ and I've saved my certificate as a PEM file, but it's not working. What do I do?

Java requires the PEM file to be in pkcs12 format. To convert your PEM file to pkcs12 format, run the following from a command prompt:

```
openssl pkcs12 -export -in YOURPEM.pem -inkey YOURPEM.pem -out
YOURPEM.p12 -passout pass:YOURPASS -name "YOURNAME"
```

Where:

- YOURPEM - the name of your PEM file
- YOURPASS - any password
- YOURNAME - any arbitrary name

For example:

```
openssl pkcs12 -export -in 1234567.pem -inkey 1234567.pem -out 1234567.p12 -passout
pass:987654321 -name "LinkPoint"
```

The output *.p12 file and the password are used to pass as parameters for JLinkPointTransaction object If you have a problem converting your PEM file, please contact support.

Top

Which shopping carts are integrated with the LinkPoint Select API solution?

Please see the list of shopping carts on http://www.linkpoint.com for the latest list of shopping carts. If you do not see the cart you want on the list, please contact us at apisupport@linkpoint.com. New carts are not added to the list until they have completed the certification process-your cart may already have been integrated, but not yet added to the list.

Top

What if I lose my password?

If you lose or forget your password, please contact us at 1 (877) 229-8739 or at support@linkpoint.com to reset your password.

Top

How can I test my store before going live?

There is a test server available for merchant testing purposes. Please go to https://staging.linkpt.net/cgi-bin/teststore to request a test store. Test accounts are completely separate from your live LinkPoint accounts. If you have any questions about test stores, please e-mail teststore@linkpoint.com.

Top

Can I set up recurring payments using LinkPoint Select API software? How about payments in installments?

Yes, you can use the LinkPoint Select API periodic billing module to set up daily, weekly, monthly, or annual payments and/or payments made at regular intervals in equal installments. You need to pass the fields in the **periodic** entity to set up a recurring payment.

Top

How do I do API tax calculations?

To use the tax calculator module, you must first create a fulltax line and send it to us for loading to the server. See Using the Tax Calculator for instructions on creating the fulltax line. Please save the file in text format and send it to support@linkpoint.com.

Top

How do I do API shipping calculations?

To use the shipping calculator module, you must create a shipping and carrier file to be housed on the LinkPoint Secure Payment Gateway. The shipping calculator matches the addressing and shipping carrier information from the shipping context with the appropriate pricing data as defined in your shipping file. Please save the file in text format and send it to support@linkpoint.com. See Using the Shipping Calculator for further instructions.

Top

What are the absolute 'required' fields for LinkPoint Select API?

The required fields vary by the type of transaction you wish to perform. Please see the ***Minimum Required Fields*** section that applies to the type of transaction you need.

Top

We want to be taken off of test mode and set to live mode? How do we handle this?

If you have been testing with a "live" store, then set the **result** field in the **orderoptions** entity to ***LIVE***. If you have been testing with a test store, you need to apply for a live API store and set the result parameter to ***LIVE***. You will also need to change your host name from the staging server host name to the production server host name. You will need to replace your test store number with your live store number. Please refer to the Welcome e-mail for the live store for the production host name and store number.

Top

What does AVS stand for?

The LinkPoint Secure Payment Gateway gives you Address Verification System (AVS) codes to help protect you from costly chargebacks and fraud. Whenever you perform a credit card Sale or Authorize Only transaction, the Gateway verifies the customer's address you entered on the Point of Sale page against the address that the card-issuing bank has on file for the customer. The AVS code tells you how well the two addresses match. If the transaction is approved, you will find the 3-digit alphabetic AVS code in the Approval Code (following the approval number and the reference number) on your Transaction Result page. A typical transaction result code might look like this. The AVS code is highlighted.

0097820000019564:**YNA**M:12345678901234567890123:

AVS compares the numeric portion of the street address and the zip code. If both the zip code and street address match, the 3-digit AVS code will begin with a Y. If they do not match, the AVS code will begin with an N. An N indicates a higher probability of fraud. The second character of the code will tell you more about the AVS results (see the table below). You will only get an AVS code if the transaction was approved, regardless of whether the addresses match. If you get an AVS code indicating that the address and/or zip code do not match, it is up to you to decide whether you wish to accept the risk and ship the goods to the customer to complete the transaction.

AVS Code	DESCRIPTION
YY*	Address matches, zip code matches
YN*	Address matches, zip code does not match
YX*	Address matches, zip code comparison not available
NY*	Address does not match, zip code matches
XY*	Address comparison not available, zip code matches
NN*	Address comparison does not match, zip code does not match
NX*	Address does not match, zip code comparison not available

XN*	Address comparison not available, zip code does not match
XX*	Address comparisons not available, zip code comparison not available
(*) -- This is the one character response code sent by the authorizing bank and it varies by card type (e.g., Y,Z,A,N,U,R,S,E,G are valid responses for Visa®; Y,Z,A,N,X,W,U,R,S are valid for MasterCard®; Y,Z,A,N,U,R,S are valid for American Express® and A,Z,Y,N,W,U are valid for Discover®).	

Top

How do I set up AVS?

You don't need to do anything to set up AVS other than to send the following fields in the **billing** entity. The AVS response will be automatically included in the transaction response code (as described above).

- addrnum
- zip

Top

What is CVV2? CVC2?

To help reduce fraud in the card-not-present environment, credit card companies have introduced a card code program. Visa calls this code Card Verification Value (CVV)--MasterCard calls it Card Validation Code (CVC)). The card code is a three- or four- digit security code that is printed on the back of cards. The number typically appears at the end of the signature panel. This helps validate that a genuine card is being used during a transaction, especially in situations like mail orders, telephone orders or Internet orders where the card is not present. All MasterCard cards, both credit and debit, were required to contain CVC2 by January 1, 1997; all Visa cards must contain CVV2 by January 1, 2001. By using the card code results along with the Address Verification Service (AVS), you can make more informed decisions about whether to accept transactions.

A typical transaction result code might look like this. The card code result is highlighted.

0097820000019564:YNA**M**:12345678901234567890123:

The last alphabetic character in the middle (M) is a code indicating whether the card code matched the card-issuing bank's code. An "M" indicates that the code matched. This code may or may not be present, depending on whether the card code was passed and the service was available for the type of card used. Below is a table showing all the possible return codes and their meanings.

Value	Meaning
M	Card Code Match
N	Card code does not match
P	Not processed
S	Merchant has indicated that the card code is not present on the card
U	Issuer is not certified and/or has not provided encryption keys
	A blank response should indicate that no code was sent and that there was no indication that the code was not present on the card.

Top

How do I set up CVV?

You don't need to do anything to set up CVV other than to send the following fields in the **creditcard** entity. The CVV response will be automatically included in the transaction response code (as described above).

- cvmvalue
- cvmindicator

Top

Is there a sample credit card number that I can use to test with my store?

Yes. There are several:

- Visa: 4111111111111 (begin with 4 and 13 digits long total)
- MasterCard: 5111111111111111 (begin with 5 and 16 digits long total)
- MasterCard: 5419840000000003 (begin with 5 and 16 digits long total)
- Amex: 371111111111111 (begin with 37 and 15 digits long total)
- Discover: 6011111111111111 (begin with 60 and 16 digits long total)
- JCB®: 311111111111111 (begin with 3 and 15 digits long total)

Top

How do I use the LinkPoint Select API electronic softgood downloads (ESD) module?

You need to send us the file that will be downloaded after purchase. We take that file and post it on the LinkPoint Secure Payment Gateway. Please include all required information necessary, including Store Number. We will send you confirmation when it has been posted. Send files to support@linkpoint.com. Please allow 72 hours for posting ESD files. See the ESD chapter for further instructions.

Top

Is there a size limit on ESD files?

There is no size limit at this time. However, if you are sending very large files, please contact support@linkpoint.com to arrange an alternate method of file transfer.

Top

How do I get access to LinkPoint Central?

When you sign up for a LinkPoint Basic or LinkPoint Select API account, you will automatically get access to the LinkPoint Central service. Log in at https://www.LinkPointCentral.com with your store number, user ID and password.

Top

Our Web server has moved. Do you need to update your records with our new IP address?

Your IP address change does not affect your API account, so we do not need to update our records.

Top

Which credit cards can I accept with the LinkPoint Select API solution?

Visa, MasterCard, Discover, American Express, JCB, and Diner's Club®

Top

Is there any way to check authorization of a credit card without having those funds reserved?

No, but depending on the bank, the funds are only reserved for 3-10 days.

Top

What computer platforms are supported?

It depends on the specific language you are using, but, in general, the following platforms are supported.

- Windows NT® (Versions 4.0)
- BSDI® (Versions 3.1 & 3.3 & 4.0)
- FreeBsd® (Versions 3.1 & 3.3)
- SUN Solaris® (Versions 2.6 &2.7)
- HP-UX® (Versions 10.2 & 11.0)
- Linux® (Versions 5.2 & 6.x)

Top

How do I run a test transaction?

Run a transaction with the **result** field in the **orderoptions** entity set to "GOOD", "DECLINE", or "DUPLICATE".

Top

Troubleshooting

Potential reasons for each error are listed under the error message below.

Top

Unable to open merchant configuration file.

Potential reasons for this error:

- The store number or storename is incorrect. (This is where the store number from the Welcome e-mail should appear.)
- The store number is not present. (Make sure the store number is in the correct place.)
- The host name is wrong. (The host name should be "secure.linkpt.net" or the host name EXACTLY as specified in your Welcome e-mail.)
- The configuration file is missing on the gateway. To test this, we need a copy of your .pem file. Send your questions and .pem file to support@linkpoint.com.

Top

Unable to open/parse client certificate file.

Potential reasons/fixes for this error:

- Make sure the .PEM file is in the correct location.
- Make sure that everything in the .PEM file (from ----RSA KEY--- to ---End Certificate---) was copied correctly.
- The certificate file could be on an unsupported platform.
- Check the host name (the correct server name is "secure.linkpt.net" or the host name EXACTLY as specified in your Welcome e-mail).
- Check port (the correct port is "1129").
- Check the permissions on the file. Make sure the file is not set to "READ ONLY".

Top

Unable to authorize payment: Unable to connect to SSL server.

Potential reasons/fixes for this error:

- Make sure port 1129 is not blocked in the firewall or router.
- Make sure the secure server is set to the host name specified in your Welcome e-mail and is on port 1129.
- Check to see if our port requires unblocking because the Web server is blocking our port.

Top

N: socket write error (Time out waiting for response)

Potential reasons/fixes for this error:

- Unable to get out of the firewall.
- Make sure that port 1129 is open.
- Your server or our server may be down.

Top

Not an ELF account: (cannot authorize transaction)

Make sure that your operating system is supported. **Unsupported systems include BSDI 4.01 and Digital UNIX.**

Top

Unable to open shipping configuration file

The shipping calculator is an optional module that requires a little extra configuration from your side. Once you have created this file, e-mail it to us at support@linkpoint.com and we can have it put on the server. Until this file is in place, you will be unable to use the shipping calculator. See Using the Shipping Calculator for further instructions.

Top

Request context must be allocated

You may getting this error because you are passing a 4-digit year (e.g., "2005"). You need to pass a 2-digit month and year (MM & YY) for the card expiration date.

Top

Invalid contact name in merchant certificate

This error is caused when the certificate/pemfile does not match the copy on our side. Please check the following:

1. Be sure you are using the correct certificate/pemfile assigned for your merchant account. If you have more then one merchant account, or a even a test account, please verify you are using the correct certificate. If you do not have a pem file, please see How do I get a pem file?.
2. Make sure you are using the correct configfile/storename. If you use the incorrect storename the certificate/pemfile will not match.
3. Be sure you have created the pemfile correctly. Refer to How do I create my pem file?.
4. Check with your merchant provider that your account is still active.
5. If all the above information is correct, you will want to send a copy of your certificate/pemfile to LinkPoint support (support@linkpoint.com), who can verify whether the certificate is good.

   Top

Getting LinkPoint Support

LinkPoint has a variety of support options for our products, including searchable online help files, user manuals, frequently asked questions (FAQs), e-mail and telephone support. We recommend you look through our help files, FAQs, and user documentation to see if the answer to your question is answered there.

If you have questions about your merchant account, please contact your merchant account provider.

If you would like to purchase LinkPoint products or services, please contact one of our authorized resellers. For a list of authorized resellers, please visit www.linkpoint.com.

If you are experiencing issues with your gateway account, please consult the Gateway Status page at http://www.linkpoint.com/gatewaystatus.html to see if your issue is already being worked.

If you have consulted the help files and documentation and/or the Gateway Status page and still cannot find the answer to your question, we recommend e-mailing our support group for a response typically within 48 hours. If you need immediate assistance, use telephone support.

:: Type	**:: LinkPoint Product Support**
E-mail Support	apisupport@linkpoint.com
Telephone Support	(888) 477-3611

Support Hours: Monday through Friday, 8 a.m. to 5 p.m., Pacific standard time (PST).

Glossary of Common Internet Commerce Terms

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

ACH	ACH is an abbreviation for Automated Clearing House. ACH allows merchants to accept payments from a consumer's checking or savings account.
Account Number	The account number for a checking or savings account is a unique number that identifies the customer's account. The account number will show on the bottom of the check, usually after (but sometimes before) the transit routing number. You may also see the check number on the bottom of the check--the two are usually separated by a non-alphabetic, non-numeric symbol.
Acquiring Bank	The financial institution that settles card transactions for a merchant.
Address Verification Service (AVS)	A system that compares the numeric portion of the customer's street address and the zip code against the information on file with the card-issuing bank. The Gateway provides an AVS code in each approved transaction result code that tells you how well the two addresses match. If they match, there is a lower probability of fraud. If there is a discrepancy in either the address or zip code, the probability of fraud is higher. Merchants can use AVS codes to help protect themselves from chargebacks and fraud.
Antivirus	Antivirus (or "anti-virus") software is a class of program that searches your Software hard drive and floppy disks for any known or potential viruses. Because of the risk of computer viruses doing harm to your computer files, antivirus software is recommended for all Internet users.
Authorize Only	The transaction type that reserves funds on a customer's credit card (performs an authorization). See also *Authorization*.
Authorization	Reserves funds on a customer's credit card. Authorization does not charge the card until you perform a Ticket Only transaction and/or confirm shipment of the order. Authorization reserves funds for varying periods, depending on the card-issuing bank's policy. The period may be as little as three days or as long as several months.

Batch	A group of credit card or check transactions that are submitted together to the Gateway for settlement. On the gateway, batches are submitted automatically once a day.
Browser	Short for Web browser, a software application used to locate and display Web pages. The two most popular browsers are Netscape® and Microsoft® Internet Explorer®. Both of these are graphical browsers, which means that they can display graphics as well as text. In addition, most modern browsers can present multimedia information, including sound and video, though they require plug-ins for some formats.
Cable Modem	A means for consumers and businesses to access the Internet by connecting their computers to a TV cable network through a device called a cable modem. This type of service is widely available and does not tie up a separate phone line; however, if the cable network is shared with many other Internet subscribers, Internet access speed may go down.
Card-issuing Bank	The financial institution or bank that issues a credit or purchasing card to a business or consumer.
Card Code	To help reduce fraud in the card-not-present environment, credit card companies have introduced a card code program. Visa calls this code Card Verification Value (CVV)-MasterCard calls it Card Validation Code (CVC)). The card code is a three- or four- digit security code that is printed on the back of cards. The number typically appears at the end of the signature panel. This program helps validate that a genuine card is being used during a transaction. To help combat fraud, card-not-present merchants should always enter a card code (if on the card) when processing an authorization. The Gateway will compare the card code against the code on file with the card-issuing bank. Results of this comparison will show in the transaction approval code. By using the card code results along with the Address Verification Service (AVS), you can make more informed decisions about whether to accept transactions.
Chargeback	A chargeback is a forced refund to the customer via the merchant's bank account. Chargebacks can occur with any type of business, but it is more prevalent for Internet businesses because the card is not present, so there is an increased chance for fraud. Each fraudulent credit card transaction usually results in a chargeback. Credit card associations penalize merchant banks for chargebacks. Naturally, the bank passes the fines on to the responsible merchant, and these penalties can be severe.

Check Number	The check number simply gives a unique number to each check. The check number is always found in the upper right hand corner of the check. With Internet check acceptance, the check number may be reused by the consumer. The check number is only provided as a reference to process the ACH transaction.
Credit Card	MasterCard®, Visa®, or other formal organization or network that sets rules, association protocols, etc., that allow for merchants to accept credit cards and for financial instutitions to offer credit cards to individuals and corporations.
CSP	Commerce Service Provider. CSPs supply businesses with the tools and services they need to buy and sell products and services over the Internet and manage their online enterprises. CSPs can generally host a secure Web site that could be connected to a secure payment gateway for selling products or services over the Web.
CVC2	See Card Code
CVV2	See Card Code
DDA Number	The account number for a particular bank account. See Account Number.
Dial Up Connection	A means of accessing the Internet by using a computer modem and telephone line. The speed of dial up connections is usually slower than other Internet access options.
Digital Certificate	A digital certificate is an electronic means that establishes the merchant's credentials when doing business on the Web. It is an encrypted set of information issued by an Internet certification authority such as Thawte. Digital certificates are required for merchants who choose to use the API or a wrapper. For other products, the merchant does not need a digital certificate.
Domain Name	A name that identifies one or more IP addresses. For example, the domain name microsoft.com represents about a dozen IP addresses. Domain names are used in URLs to identify particular Web pages. For example, in the URL http://www.linkpoint.com/index.html, the domain name is linkpoint.com.
DSL	DSL (Digital Subscriber Line) is a technology for bringing fast Internet service to homes and small businesses over ordinary copper telephone lines.

E-Commerce	Electronic commerce. The term commonly used for conducting business online (buying and selling products or services over the Internet).
Electronic Check Acceptance	Electronic check acceptance. Offered with some POS terminals, but not available with Gateway accounts. The Gateway uses Internet Check Acceptance (see ICA), not ECA. With ECA, the check is electronically submitted as a check. The check is no longer usable and the paper check must be voided. The customer signs and receives a paper receipt. ECA services may include a check guarantee service. ECA is used for retail payments only. .
Field	A field is a named area on a Web form or software application that has a purpose and usually a fixed size. In a form that you fill out on a Web site, each box that asks you for information is a text entry field.
Firewall	A firewall is a set of related programs, located at a network gateway server, that protects the resources of a private network from users of other networks.
Forced Ticket	A forced post authorization transaction. This transaction type is used similarly to a Ticket Only transaction, except it is specifically for authorizations you obtained over the phone. It requires a reference number (or approval code) that you should have received when you did the phone authorization.
Gateway	A payment gateway is a system that enables payments over the Internet. When a customer orders something from a merchant, the merchant has the order information, the customer has the payment information, and each of them have their own accounts with banking institutions. The gateway collects all the information and sends it to the right places to enable the payment to go from the customer's credit card of bank account to the merchant's account. You can think of a payment gateway as a messenger that sends information between the consumer, the merchant, the acquirer (e.g., the merchant's bank), and the consumer's bank (i.e., for credit card payments, the card-issuing bank).
HTML	Short for HyperText Markup Language, a markup language used to structure text and multimedia documents and to set up hypertext links between documents, used extensively on the World Wide Web. Other than manually entering transactions using the virtual POS terminal, HTML is the simplest way to send payment transactions to the Gateway.

HTTP	Short for HyperText Transfer Protocol, the underlying protocol, or computer language, used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and tells Web servers and browsers what to do when a user clicks on something. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.
Hyperlink	An element in a document (or Web page) that links to another place in the same page or to an entirely different page. Typically, you click on the hyperlink to follow the link. Hyperlinks are the most essential ingredient of the World Wide Web.
ICA	Internet Check Acceptance. This is the type of check service provided on the Gateway. ICA uses the Automated Clearing House (ACH) to debit the consumer’s account. The account information is entered in an online payment form, and no check is used (i.e., the check is still usable). The customer may or may not sign a payment form. In either case, the merchant needs a documented record of the customer's authorization to debit the account. ICA includes an electronic receipt (via e-mail). There is no check guarantee service with ICA. Typically used for mail order/telephone order (MO/TO) or e-commerce transactions, but may also be used for retail.
Input	The information provided to a computer system for processing. When you enter information into a Web form, you are providing input. Gateway products require input to process a transaction.
Internet	The Internet is a massive network of networks, a networking infrastructure. It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet. Information that travels over the Internet does so via a variety of computer languages known as protocols.
IP address	Short for Internet Protocol address. An IP address is a number that is used to identify a specific computer on a network or on the Internet. The format of an IP address is written as four numbers separated by periods. Each number can be zero to 255. For example, 1.160.10.240 could be an IP address.

ISP	Short for Internet Service Provider, a company that provides access to the Internet. For a monthly fee, the service provider gives you a software package, username, password and access phone number. Equipped with a modem, you can then log on to the Internet and browse the World Wide Web, and send and receive e-mail. In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company's networks to the Internet. An ISP may also host a company's or person's Web site.
LAN	Local Area Network. A computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN).
Log In	To enter into a computer the information required to begin a session. Also sometimes referred to as Log On. Users log into Gateway reports by going to the login URL and entering their store name, user ID, and password. Once they've logged in, they have secure access to the virtual point-of-sale terminal, as well as gateway reports, customization, and admin features.
Log Off	To enter into a computer the command to end a session or leave the current application. Also referred to as Log Out. To log out of the Gateway reports/admin/customization, merchants should click on the Log Out button on the Main Menu Bar. To prevent unauthorized users from accessing their account, merchants should always log off and close the browser window when they are done using the Gateway functions.
Network	A group of two or more computer systems linked together.
Password	A sequence of characters that you must enter to gain access to a file, application, or computer system. Users must enter a password in a Web form to access Gateway reports, admin, and customization functions. We recommend that users change their password frequently and do not share it with anyone to prevent unauthorized access to their Gateway accounts.
PDF File	An Adobe® Acrobat® file, commonly used for text and graphic files transmitted over the Internet. Internet users need an Adobe Acrobat viewer to open a PDF file, which can be downloaded for free at http://www.adobe.com. Gateway user manuals are available as PDF files.

Periodic Billing	The capability to charge customers on a recurring basis according to merchant-defined rules. Gateway products allow a merchant to charge a customer's card in exchange for products and services one or more times every day, week, month, or year.
POS	Point of Sale. The time at which the consumer is purchasing the product from the merchant and the merchant is processing the payment transaction. Point of sale is commonly used to refer to the payment terminals or software that merchants use to process the payment transaction. The Gateway offers a virtual POS terminal for merchant use at the point of sale.
Plug-In	A hardware or software module that adds a specific feature or service to a larger system. For example, there are number of plug-ins for the Netscape Navigator browser that enable it to display different types of audio or video files.
Protocol	A language used for computers to talk to one another. There are several different protocols. HTTP is the protocol used on the Web.
Purchasing Card	A purchasing card is a corporate card used by some companies for their business purchases. When a customer pays for goods or services using a purchasing card, the following information must be included with the order information. This information is optional for a regular credit card transaction. ● An indication of whether the order is tax exempt. ● The amount of tax applied to the order. If the order is tax exempt, the tax amount should be zero. ● A purchase order number associated with this order. One purchase order can apply to several individual orders, to allow for delivery of goods over time. If there isn't a purchase order associated with this order, the customer must supply some value for this field-the value could be a department number, expense code, project number, etc.
Recurring	To occur at a stated interval, or according to some regular rule. LinkPoint Internet products allow a merchant to charge a customer's card on recurring intervals. Merchants can set up charges for goods or services one or more times every day, week, month, or year.
Return	Gives back funds to a customer's credit card against an existing order on the Gateway system. To perform a return, you need the order number (which you can find in your Reports). If you perform a Return of the full order amount, the order will appear in your Reports with a transaction amount of 0.00.

Sale	A sale transaction immediately charges a customer's credit card when the batch of transactions is closed.
Settlement	The completion of a payment transaction. When a transaction is settled, it has been funded and the funds deposited in the merchant account.
SSH	Secure Shell is a secure means to move files from one machine to another. It provides secure communications over insecure channels. SSH protects a network from attacks. An attacker who has managed to take over a network can only force SSH to disconnect. He or she cannot play back the traffic or hijack the connection when encryption is enabled.
SSL	Secure Sockets Layer, a protocol (or language) for transmitting secure information (such as credit card data) via the Internet. SSL works by encrypting the data that's transferred over the connection. Both Netscape Navigator and Internet Explorer support SSL, and many Web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with *https:* instead of *http:*
Store Name	Also called storename or Store Number. A six- to ten-digit number that is needed to identify the merchant. The store name (store number) is given to the merchant in the Welcome E-mail. Merchants need the store name (number), user ID and password to access the virtual point-of-sale terminal, as well as reports, admin, and customization functions. The store name is also needed for using the API and other Gateway products.
Submit	To send information to a system. On a Web form, you submit the form (send the information) by clicking on a button at the bottom of the form.
Ticket Only	A post authorization. Captures the funds from an Authorize Only transaction, reserving funds on the customer's card for the amount specified. Funds are transferred when your batch of transactions is settled.
Transit Routing Number	The transit routing number is a number to identify at which bank the customer has the account. As you see on the Point of Sale page when you click on Pay by Check, you can easily locate the transit routing number between the two **:|** symbols.
URL	Abbreviation for Uniform Resource Locator, the address for documents and other pages on the World Wide Web. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located.

User ID	On Gateway accounts where there are multiple users, each individual user will be assigned a User ID. The user will need this User ID, along with the store name and password, to log into reports, admin, and customization functions and to access the virtual point-of-sale terminal.
Virtual	Not a tangible object you can pick up and hold in your hands. The term virtual is often used on the Web to denote a Web-based program that functions similarly to a physical device or system. For example, a virtual point-of-sale terminal is a Web-based computer program that performs the same functions as a real point-of-sale terminal.
Void	To cancel a payment transaction. Merchants can void transactions prior to settlement. Once the transaction has settled, the merchant has to perform a return or credit to reverse the charges and credit the customer's card.
WAN	A computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LANs). Computers connected to a wide-area network are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites. The largest WAN in existence is the Internet.
Web	The World Wide Web, or simply Web, is a way of accessing information over the medium of the Internet. It is an information-sharing model that is built on top of the Internet. The Web uses the HTTP protocol, one of the computer languages spoken over the Internet, to transmit data. Web services, which use HTTP to allow applications to communicate in order to exchange business logic, use the Web to share information. The Web also uses browsers, such as Internet Explorer or Netscape, to access Web documents called Web pages that are linked to each other via hyperlinks. Web documents can also contain graphics, sounds, text and video.
Web Server	At its core, a Web server is a computer with specialized software installed that serves content (or Web pages) to a Web browser by loading a file from a disk and serving it across the network to a user's Web browser. The browser and server talk to each other using HTTP.

XML	XML is the Extensible Markup Language, which is a universal format for the representation of documents and data. It is designed to improve the functionality of the Web and software applications by providing a more flexible and adaptable way to identify information. It is considered extensible because it is not a fixed format, but is actually a 'meta-language' — a language for describing other languages. XML follows the standards set in the ISO 8879 document—the international standard describing meta-languages. XML documents are hierarchical in nature and are made up of entities or nodes, meaning that an XML entity may contain data or other entities.