

Direct Post Method (DPM)

Developer Guide

March 2012

Authorize.Net
a CyberSource solution

Authorize.Net LLC ("Authorize.Net") has made efforts to ensure the accuracy and completeness of the information in this document. However, Authorize.Net disclaims all representations, warranties and conditions, whether express or implied, arising by statute, operation of law, usage of trade, course of dealing or otherwise, with respect to the information contained herein. Authorize.Net assumes no liability to any party for any loss or damage, whether direct, indirect, incidental, consequential, special or exemplary, with respect to (a) the information; and/or (b) the evaluation, application or use of any product or service described herein.

Authorize.Net disclaims any and all representation that its products or services do not infringe upon any existing or future intellectual property rights. Authorize.Net owns and retains all right, title and interest in and to the Authorize.Net intellectual property, including without limitation, its patents, marks, copyrights and technology associated with the Authorize.Net services. No title or ownership of any of the foregoing is granted or otherwise transferred hereunder. Authorize.Net reserves the right to make changes to any information herein without further notice.

Authorize.Net Trademarks

Advanced Fraud Detection Suite™

Authorize.Net®

Authorize.Net Your Gateway to IP Transactions™

Authorize.Net Verified Merchant Seal™

Automated Recurring Billing™

eCheck.Net®

FraudScreen.Net®

The logo for Authorize.Net, featuring the brand name in a large, blue, serif font. Below it, the tagline "a CyberSource solution" is written in a smaller, grey, sans-serif font. The entire logo is set against a light grey rectangular background.

Authorize.Net
a CyberSource solution

Contents

Revision History 6

Chapter 1	Introduction	7
	Other Integration Methods	7
	AIM	7
	SIM	7
	Conceptual Overview	8
	DPM Minimum Requirements	10
	Managing Integration Settings	11
	Features of DPM	12
	eCheck.Net®	13
	Developer Support	13
	Software Development Kits	14

Chapter 2	Transaction Data Requirements	15
	Credit Card Transaction Types	15
	Authorization and Capture	15
	Authorization Only	16
	Prior Authorization and Capture	16
	Capture Only	17
	Credit	17
	Void	17
	Partial Authorization	17
	Using the Merchant Interface	18

Chapter 3	Submitting Transactions	19
	Generating a Unique Transaction Fingerprint	19
	Custom Transaction Fingerprint Code	19
	The Transaction Key	21
	Minimum Required Fields	21
	Merchant-defined fields	27

Chapter 4	Receipt Options	29
	Relay Response	29
	Tips for Using Relay Response	31
	Email Receipt	31

Chapter 5	Additional API Fields	33
	Transaction Information	33
	Itemized Order Information	34
	Additional Customer Information	35

Chapter 6	Transaction Response	37
	Fields in the Payment Gateway Response	37
	Using the MD5 Hash Feature	42
	Response for Duplicate Transactions	43
	DPM Relay Response	44
	DPM Transaction Response Versions	44
	Version 3.0	44
	Version 3.1	45
	Upgrading the Transaction Version	45
	Response Code Details	45
	Response Codes	46
	Response Reason Codes and Response Reason Text	46
	Example of a Response for Partial Authorization Transactions	58

Chapter 7	Test Transactions	60
	Testing to Generate Specific Transaction Results	61

Chapter 8	Using the .NET SDK	63
------------------	---------------------------	-----------

Chapter 9	Using the Java SDK	66
------------------	---------------------------	-----------

Chapter 10	Using the PHP SDK	72
-------------------	--------------------------	-----------

Chapter 11	Using the Ruby SDK	76
-------------------	---------------------------	-----------

Appendix A **Fields by Transaction Type** 79

Minimum Required Fields 79

Required Fields for Additional DPM Features 80

Best Practice Fields 80

Appendix B **API Fields** 82

Revision History

Publish Date	Updates
March 2012	Made minor edits to formatting and grammar.
January 2011	Corrected code examples.
November 2010	Corrected sample code for checkout_form.jsp. Made minor editorial changes.
October 2010	Release of Ver 1.0 Direct Post Method (DPM) Developer Guide. Minor formatting changes and additional clarifying content in Sections 1 and 5.

Introduction

This guide describes the Web development required to submit credit card transactions to the Authorize.Net gateway for authorization and settlement using the Direct Post Method (DPM).

DPM offers site customization while using Authorize.Net for help with PCI compliance. The Authorize.Net Payment Gateway handles data submission while keeping Authorize.Net virtually transparent. The merchant's website collects data and responds to customers with a merchant-designed receipt page.

The security of a DPM transaction is ensured by a unique digital signature or "fingerprint" that is sent with each transaction. Authorize.Net uses this fingerprint to authenticate both the merchant and the transaction. Sample code for this function is available for free from the Authorize.Net Developer Center:

<http://developer.authorize.net>

Other Integration Methods

AIM

The Advanced Integration Method (AIM) is designed for merchants who need a highly customizable payment form (for example, complete control of look and feel and the ability to keep the customer on their website during the entire checkout process) or for those integrating a standalone business application. When using AIM, the developer can maintain a continuous session with the customer during a transaction. For more information about AIM, see the *Advanced Integration Method (AIM) Developer Guide*:

<http://developer.authorize.net/guides/AIM/>

SIM

The Server Integration Method (SIM) is a hosted payment option that offers the user site customization while relying on Authorize.Net for PCI compliance. SIM is an ideal

integration solution because merchants are not required to collect, transmit, or store sensitive cardholder information in order to process transactions. SIM does not require merchants to purchase and install a Secure Sockets Layer (SSL) digital certificate, reducing the complexity of securely handling and storing cardholder information and simplifies compliance with the Payment Card Industry (PCI) Data Security Standard. For more information about SIM, see the *SIM Developer Guide*:

<http://developer.authorize.net/guides/SIM>

Conceptual Overview

With DPM, customers' payment data is posted directly to Authorize.Net, bypassing the merchant server. Upon authorization, Authorize.Net returns coded values to the merchant server for verification, then generates a receipt for display within the merchant page. Customer billing data posts directly to Authorize.Net without being stored on the merchant server, while the merchant still retains control over the checkout process.

Authorize.Net sends a POST of the transaction result to the relay URL in DPM. DPM then redirects the client browser to the merchant's server. This way, the URL in the client's browser shows the merchant's server and not Authorize.Net. The redirect uses JavaScript if available on the client browser, and a meta refresh tag if it is not.

The following diagram and table describe the architecture and flow of control for a DPM transaction:

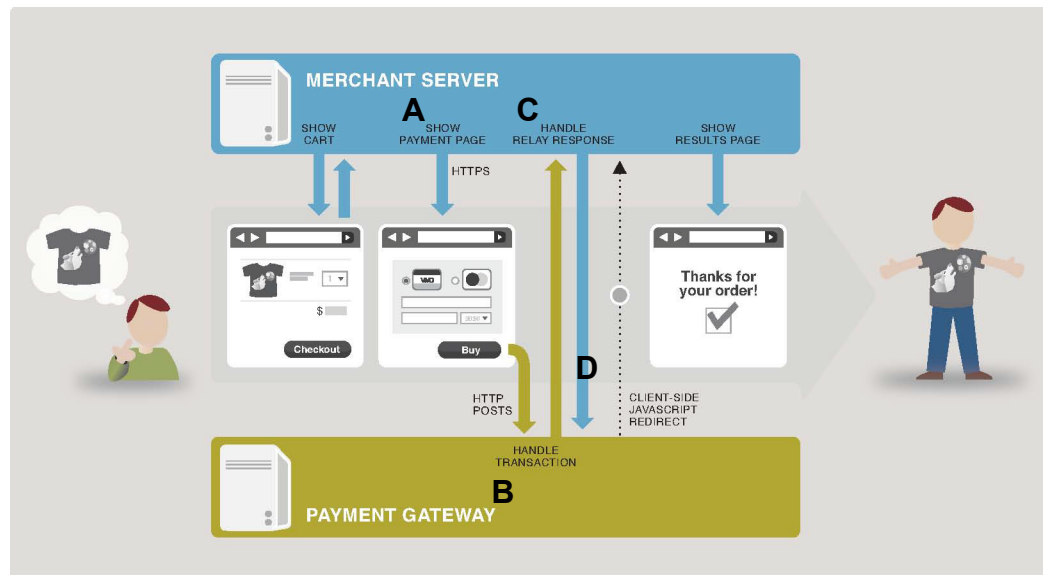


Table 1 Transaction Flow


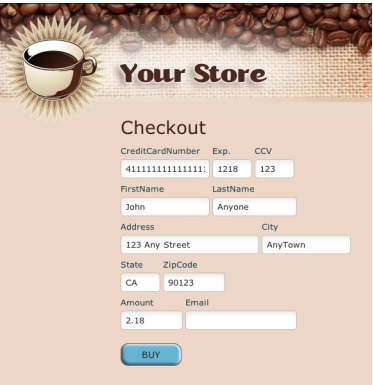
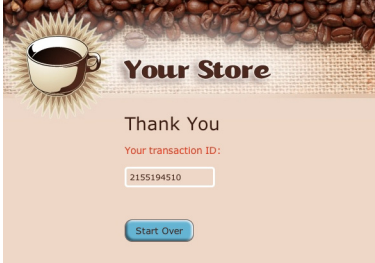
Label	User Experience	Description
A		<p>When a customer submits an order on the shopping cart, the merchant's server generates a payment form (for example, checkout_form.jsp) to collect the customer's payment and shipping information.</p>
B		<p>On Submit, customer billing data posts directly to Authorize.Net, bypassing the merchant server.</p> <p>This post can include both hidden (merchant-supplied) and customer data fields. The developer can use any of the DPM form fields and can create merchant-defined data fields. "Merchant-defined fields," page 27 identifies and defines common form fields and specifies the constraints on merchant-defined fields.</p> <p>The posted form includes an x_relay_url value (containing the URL to which Authorize.Net will post transaction results upon completion).</p> <p>This is not a typical relay URL; that is, it contains no content for display in the client browser. Instead, it returns a code snippet that redirects the client browser to a URL on the merchant's server. The redirect maintains the merchant server's URL in the client browser address bar throughout the transaction.</p>

Table 1 Transaction Flow (Continued)

Label	User Experience	Description
C/D		<p>When the merchant server receives the HTTP POST from Authorize.Net, it validates the hash values and logs the order. Authorize.Net expects a return from this HTTP POST and displays the result:</p> <ul style="list-style-type: none"> ■ On failure (an HTTP response code other than 200-OK), an error message ■ On success, the Authorize.Net server returns the code snippet generated by the merchant's relay URL that forces the redirection of the client browser to the merchant server. This uses JavaScript if available on the client browser and a <i>meta refresh</i> tag if it is not. For example: <pre><html> <head> <script type='text/javascript' charset='utf-8'> window.location='http://YOUR_ SERVER.COM/receipt.jsp'; </script> <noscript> <meta http-equiv='refresh' content='1;url=http://YOURSERVER.COM/ receipt.jsp'> </noscript> </head> <body></body> </html></pre> <p>The redirect keeps the URL in the client browser pointed to the merchant's server. (The redirect should also contain enough information about the transaction so that the merchant's server can display something sensible to the user.)</p>

**Note**

The fields documented in the following sections describe only the minimum required fields. If you wish to submit additional fields (such as billing address), refer to the documentation for the Server Integration Method (SIM).

DPM Minimum Requirements

Before you begin, check with the merchant to make sure that the following DPM requirements have already been met. We recommended that you work closely with the merchant to ensure that any other business and website requirements are included in their DPM integration (for example, bank requirements, processor requirements, or website design preferences).

- The merchant must have a U.S.-based merchant bank account that allows Internet transactions.
- The merchant must have an e-commerce (Card Not Present) Authorize.Net Payment Gateway account.
- The merchant's website or hosting provider must have server scripting or CGI capabilities such as ASP.Net, PHP, Java, or Ruby.
- The merchant must be able to store payment gateway account data securely (for example, API Login ID or Transaction Key).

**Note**

Merchants should avoid storing any type of sensitive cardholder information. However, in the event that a merchant or third party must store sensitive customer business or payment information, compliance with industry standard storage requirements is required. See the *Developer Security Best Practices White Paper* for Guidelines: <http://www.authorize.net/files/developerbestpractices.pdf>

Managing Integration Settings

When integrating the merchant's website with the payment gateway, you should be aware that most settings for a merchant's integration can be configured and managed in one of two ways:

- Included in the transaction request on a per-transaction basis by means of the application programming interface (API), as described in this guide.
- Configured in the Merchant Interface and applied to all transactions.

**Important**

The Merchant Interface at <https://secure.authorize.net> is a secure website where merchants can manage their payment gateway account settings, including their website integration settings. We recommend that you review the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/> for information on managing the merchant's payment gateway integration using the Merchant Interface.

Transaction settings submitted in the transaction request override transaction settings configured in the Merchant Interface. However, please be aware that some integration settings must be configured in the Merchant Interface. To help the merchant maintain a robust integration, review the integration settings that can be configured in the Merchant Interface with the merchant and determine which integration settings can be posted on a per-transaction basis, and which should be configured in the Merchant Interface. See "[Fields by Transaction Type](#)," [page 79](#) for a list of fields the payment gateway recommends be submitted on a per-transaction basis.

Features of DPM

In addition to basic transaction processing, DPM provides merchants with several features for configuring transaction security options and further customizing their customers' checkout experience. These features are listed in the table below. Please take a few moments to discuss these with your merchant and select features they would like to include in their integration.

Table 2 Features of DPM

Feature	Description	Requirements
Address Verification Service (AVS) Filter	This feature allows merchants to compare the billing address submitted by the customer for the transaction with the address on file at the card issuing bank. Filter settings in the Merchant Interface allow the merchant to reject transactions based on the AVS response received.	To implement AVS, the merchant must require the Address and ZIP Code fields on the payment gateway hosted payment form. For more information about AVS, please see the <i>Merchant Integration Guide</i> : http://www.authorize.net/support/merchant .
Card Code Verification (CCV) Filter	This feature allows merchants to compare the card code submitted by the customer for the transaction with the card code on file at the card issuing bank. Filter settings in the Merchant Interface allow the merchant to reject transactions based on the CCV response received.	To implement CCV, the merchant must require the Card Code field on the payment form, and settings must be configured in the Security Settings section of the Account Settings menu in the Merchant Interface or submitted on a per-transaction basis. For more information about CCV, please see the <i>Merchant Integration Guide</i> at: http://www.authorize.net/support/merchant .
Itemized Order Information	This feature allows merchants to submit details for items purchased. This information is included in the merchant transaction confirmation email, in the Transaction Details for the transaction, and in QuickBooks download reports in the Merchant Interface.	To implement Itemized Order Information, the line item field must be submitted on a per-transaction basis. See " Itemized Order Information ," page 34 for details.

Table 2 Features of DPM (Continued)

Feature	Description	Requirements
Email Receipt	This feature allows merchants to opt for an automatic email receipt to be sent by the payment gateway to their customers.	To configure the payment gateway email receipt, the merchant must require the customer email address on their custom payment form, and settings must be configured in the Email Receipts section of the Settings menu in the Merchant Interface or submitted on a per-transaction basis. See " Receipt Options ," page 29 for details.

eCheck.Net[®]

In addition to processing credit card transactions, the payment gateway also supports electronic check transactions with our exclusive eCheck.Net[®] product. Please contact the merchant to determine whether eCheck.Net is enabled for their payment gateway account or if they would like to sign up. **In the event that eCheck.Net is enabled, you will need to ensure that the merchant's website integration supports all eCheck.Net field requirements.** Please see the *eCheck.Net Developer Guide* at <http://developer.authorize.net/guides/echeck.pdf> for more information.

Developer Support

There are several resources available to help you successfully integrate a merchant website or other application to the Authorize.Net Payment Gateway.

The Developer Center at <http://developer.authorize.net> provides test accounts, sample code, FAQs, and troubleshooting tools. You can also find information and support in the developer community forums at <http://community.developer.authorize.net/>.

If you can't find what you need in the Developer Center, our Integration Team is available to answer your questions by email at developer@authorize.net. (Our Integration Team can only assist with support requests specifically about the Authorize.Net application programming interface (API) and/or services.)

Be sure to read our *Developer Security Best Practices White Paper* at <http://www.authorize.net/files/developerbestpractices.pdf> for information on how to maximize the security and reliability of your merchant integration solutions.

If you have any suggestions about how we can improve or correct this guide, please email documentation@authorize.net.

Software Development Kits

Authorize.Net offers software development kits (SDKs) that present an alternate object-oriented model, in several popular languages. To use these SDKs, the merchant's transaction version must be set to 3.1. The SDK performs the core payment activities (such as error handling and parsing, network communication, and data encoding) behind the scenes.

The SDK provides utility methods to help developers build payment flows for each of the integration methods. You can download the SDKs:

<http://developer.authorize.net/downloads/>

Transaction Data Requirements

The payment gateway supports several credit card transaction types for transactions submitted by means of DPM.

To see a table listing minimum form field requirements for posting credit card transaction requests to the payment gateway, see ["Minimum Required Fields," page 21](#).

Credit Card Transaction Types

This section describes the credit card transaction types supported by the payment gateway and their specific field requirements. Talk to your merchant about how their business plans to submit transaction so that you can integrate their payment gateway account to support their business processes.

For example, are they submitting transactions mainly through an e-commerce website? Do they need to integrate a custom application to allow call center representatives to enter mail order/telephone order (MOTO) transactions? Would they like the ability to verify the availability of funds on a customer's credit card account at the time of purchase and then charge the credit card at the time they ship the order?



Note

Some of the field requirements listed in this section for each credit card transaction type are *in addition* to the minimum field requirements already set forth above for ALL transactions submitted to the payment gateway. For a list of all fields that are required for each credit card transaction type, please see [Appendix A, "Fields by Transaction Type," on page 79](#).

Authorization and Capture

Authorization with Auto Capture (`Auth_Capture`) is the default transaction type in the Virtual Terminal. If no `x_type` variable is submitted with a website transaction request, the type defaults to `Auth_Capture`. This type of transaction is completely automatic; the transaction is submitted to your processor for authorization and, if approved, is placed in Unsettled Transactions already set to Capture. The transaction will settle with your next batch settlement. Settlement occurs every 24 hours, within 24 hours of the time specified in your Settings menu, under Transaction Cutoff Time.

The unique field requirement for an authorization and capture is:

```
<INPUT TYPE=HIDDEN NAME="x_type" VALUE="AUTH_CAPTURE">
```

Authorization Only

This transaction type is sent for authorization only. When an Authorization Only (Auth_Only) transaction is submitted, it is sent to your processor for authorization. If approved, the transaction is placed Unsettled Transactions status with a status of Authorized/Pending Capture. The authorization places the funds on hold with the customer's bank, but until the transaction is captured, the funds transfer process does not occur. This type of transaction is not sent for settlement until you submit a credit card transaction type Prior Authorization and Capture or until you submit the transaction for capture manually in the Merchant Interface. Authorization Only can be useful in situations where you need to make a sale but won't be able to ship merchandise for several days; you can authorize the transaction to ensure the availability of funds, then, once you have shipped, you can capture the transaction to obtain the funds.

Authorization Only transactions are only kept Unsettled Transactions status for 30 days. After that, their transaction status changes to Expired, and the funds will NOT be transferred. To capture a transaction, you can manually log on to the Merchant Interface and go to the Unsettled Transactions window. From there, you can use the Group Capture filter to capture multiple transactions at once, or click on the individual Transaction ID of the transaction you would like to capture, and the next screen will provide a Capture button. From a website or billing application, you can submit the x_type variable with a value of Prior_Auth_Capture to capture the transaction.

The unique field requirement for an Authorization Only transaction:

```
<INPUT TYPE=HIDDEN NAME="x_type" VALUE="AUTH_ONLY">
```



Note

Merchants who use DPM can configure the hosted payment form to submit either Authorization and Capture or Authorization Only transactions. Please check with the merchant regarding their preferences regarding which of these credit card transaction types should be used for their website.

Prior Authorization and Capture

This transaction type is used to complete an Authorization Only transaction that is successfully authorized through the payment gateway. Credits can be manually processed through the Virtual Terminal or can be submitted from a website or billing application.

If this transaction type is required, we recommend that the merchant process the transactions by logging on to the Merchant Interface directly or by using a desktop application that uses AIM.

Capture Only

Capture Only transactions are used when you already have an authorization from a bank. To use this type of transaction, you need an authorization code from the card issuer (usually a five- or six digit number). For example, if you called Visa directly and obtained an authorization over the phone, you would need to submit a Capture Only transaction to start the funds transfer process. You can manually submit a Capture Only transaction from your Virtual Terminal by selecting Capture Only, or from a website or billing application by including the following variables with your transaction request:

- `x_type` (Capture_Only)
- `x_Auth_Code` (the five- or six digit code provided by the card issuer)

Credit

This transaction type is used to refund a customer for a transaction that was originally processed and successfully settled through the payment gateway. Credits can be manually processed through the Virtual Terminal or can be submitted from a website or billing application.

Credit transactions cannot be processed using DPM. If this transaction type is required, the merchant should process the transactions by logging on to the merchant interface directly or by using a desktop application that uses AIM.

Void

This transaction type is used to cancel an existing transaction that has a status of Authorized/Pending Capture or Captured/Pending Settlement. Settled transactions cannot be voided (issue a Credit to reverse such charges). The DPM API does not support Void transactions.

You can manually void transactions from the Unsettled Transactions screen of the Merchant Interface. From there, you can use the Group Void filter toward the top of your screen to void multiple transactions at once, or click on the individual Transaction ID of the transaction you would like to void; the next screen will provide a Void button.

If this transaction type is required, we recommend that the merchant process the transactions by logging on to the Merchant Interface directly or by using a desktop application that uses AIM.

Partial Authorization

A partial authorization, or *split tender*, order is one in which two or more transactions are used to cover the total amount of the order.

The merchant must indicate that they are able to handle the extra processing either by selecting the Partial Authorization option in the Account settings of the Merchant Interface, or by sending `x_allow_partial_auth=true` with each transaction. Without this flag, the transaction would be handled as any other, and would be either fully authorized or declined due to lack of funds on the card.

When the first transaction is successfully authorized for a partial amount of the order, a split tender ID is generated and returned in the response. This ID must be passed back with each of the remaining transactions of the group, using `x_split_tender_id=<value>`. If you include both a split tender ID and a transaction ID on the same request, an error results.

If successfully authorized, all transactions in the group are held until the final transaction of the group is successfully authorized, unless the merchant has indicated either by input parameter or default configuration that the transactions should not be held.

The following fields are returned in the relay response data sent to the merchant's URL. The data they contain are in all prepaid card responses.

- `x_prepaid_requested_amount`—this is the amount requested.
- `x_split_tender_id`—this is the Split Tender ID provided when the first partial authorization transaction was issued.
- `x_split_tender_status`—indicates whether or not the transaction is complete.
- `x_card_type`—the card type.

Using the Merchant Interface

The Merchant Interface enables merchants to manage transactions, capture Authorize Only transactions, void transactions, and issue refunds. These transaction types can also be managed automatically by means of the API if you are integrating a custom application to the payment gateway. However, for many integrations, these transaction types can be more conveniently and easily managed in the Merchant Interface.

For more information on submitting transactions in the Merchant Interface, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant> or click **Help** in the top right corner of the Merchant Interface.

Submitting Transactions

The standard payment gateway application programming interface (API) consists of required and additional optional form fields that can be submitted to the payment gateway for real-time transaction processing.

Generating a Unique Transaction Fingerprint

The transaction authentication used by the Direct Post Method is a “transaction fingerprint,” or a hash of merchant- and transaction-specific information using the HMAC-MD5 (MD5 RFC 1321 with a 128-bit hash value) hashing algorithm. The HMAC-MD5 is used only for generating the unique transaction fingerprint. The transaction fingerprint must be generated on a per-transaction basis by a server-side script on the merchant’s Web server and inserted into the transaction request.

There are two options for developing the fingerprint generation script:

- You can develop a custom script yourself using the API fields information in this section, OR
- You can use Authorize.Net sample code available for free from our Developer Center at <http://developer.authorize.net>.

Custom Transaction Fingerprint Code

If you choose to develop custom code for generating the transaction fingerprint, the following table represents the field requirements for the transaction fingerprint. The form fields inserted into the transaction request by the fingerprint generation use the syntax:

```
<INPUT TYPE="HIDDEN" NAME="x_name_of_field" VALUE="value of the field" />
```

Table 3 Field requirements for the transaction fingerprint

FIELD NAME	VALUE	FORMAT	NOTES
x_fp_hash	The unique transaction fingerprint	N/A	<p>The fingerprint is generated using the HMAC-MD5 hashing algorithm on the following field values:</p> <ul style="list-style-type: none"> ■ API Login ID (x_login) ■ The sequence number of the transaction (x_fp_sequence) ■ The timestamp of the sequence number creation (x_fp_timestamp) ■ Amount (x_amount) <p>Field values are concatenated and separated by the “^” character.</p>
x_fp_sequence	The merchant-assigned sequence number for the transaction	Numeric	The sequence number can be a merchant-assigned value, such as an invoice number or any randomly generated number.
x_fp_timestamp	The timestamp at the time of fingerprint generation.	UTC time in seconds since January 1, 1970	<p>Coordinated Universal Time (UTC) is an international atomic standard of time (sometimes referred to as GMT). Using a local time zone timestamp will cause fingerprint authentication to fail.</p> <p>If the fingerprint is more than one hour old or more than 15 minutes into the future, it is rejected.</p>

The transaction fingerprint that is submitted in x_fp_hash is generated using an HMAC-MD5 hashing algorithm on the following field values:

- Step 1** API Login ID (x_login)
- Step 2** Sequence number (x_fp_sequence)
- Step 3** UTC timestamp in seconds (x_fp_timestamp)

**Note**

Be sure that the merchant server’s system clock is set to the proper time and time zone.

- Step 4** Amount (x_amount)

**Note**

The amount used to generate the fingerprint must reflect the final amount of the transaction. To avoid any discrepancy, it is strongly recommended that you generate the fingerprint at a point in the checkout process when the amount can no longer be changed.

When generating the fingerprint, input values must be provided to the script in the field order listed above and concatenated by the “^” character. All trailing spaces must be removed from input values. If the fingerprint is generated using any other field order, authentication will fail and the transaction will be rejected.

Example Fingerprint Input Field Order

```
"authnettest^789^67897654^10.50^"
```

Please note the required trailing “^” character

The Transaction Key

The cryptographic key used in the HMAC calculation is the merchant’s unique Transaction Key, which is a payment-gateway generated, 16-character value that can be obtained by the merchant in the Merchant Interface. For more information, please see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant>.

**Important**

The merchant’s Transaction Key is highly sensitive and should only be known by the payment gateway and the merchant. For this reason it is vital that the Transaction Key is stored securely and separately from the merchant’s Web server. In addition, please note that the merchant’s API Login ID will be visible in the source for the payment form request, but the Transaction Key should never be visible.

Example of the call to generate the transaction fingerprint

```
Fingerprint = HMAC-MD5
("authnettest^789^67897654^10.50^", "abcdefgh12345678")
```

**Note**

This code is intended to be an example only, and will not necessarily work if you paste it into your application.

Minimum Required Fields

The following table represents the minimum fields required for submitting a credit card transaction request to the payment gateway using AIM. The data fields are name/value pairs with the following syntax:

```
<INPUT TYPE="HIDDEN" NAME="x_name_of_field" VALUE="value of the field" />
```

The following table lists the fields that can be configured in the Merchant Interface or submitted in the transaction request string. The form fields are submitted using the syntax:

```
<INPUT TYPE=HIDDEN NAME="x_name_of_field" VALUE="value of the field">
```

Table 4 Payment form fields

Field Name	Value	Format	Notes
PAYMENT INFORMATION			
Recurring Billing Transaction (x_recurring_billing)	The recurring billing status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicating marker used by merchant account providers to identify transactions which originate from merchant hosted recurring billing applications. This value is not affiliated with Automated Recurring Billing.
ORDER INFORMATION			
Invoice Number (x_invoice_num)	The merchant assigned invoice number for the transaction	Up to 20 characters (no symbols)	The invoice number must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be included on the hosted payment form, the attribute View must be configured for this field in the Merchant Interface payment form settings.
Description (x_description)	The transaction description	Up to 255 characters (no symbols)	The description must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.
BILLING INFORMATION			
First Name (x_first_name)	The first name associated with the customer's billing address	Up to 50 characters (no symbols)	

Table 4 Payment form fields (Continued)

Field Name	Value	Format	Notes
Last Name (x_last_name)	The last name associated with the customer's billing address	Up to 50 characters (no symbols)	
Company (x_company)	The company associated with the customer's billing address	Up to 50 characters (no symbols)	
Address (x_address)	The customer's billing address	Up to 60 characters (no symbols)	Required if the merchant would like to use the Address Verification Service filter. For more information on AVS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant Required for Zero Dollar Authorizations for Visa verification transactions.
City (x_city)	The city of the customer's billing address	Up to 40 characters (no symbols)	
State (x_state)	The state of the customer's billing address	Up to 40 characters (no symbols) or a valid two-character state code	
ZIP Code (x_zip)	The ZIP code of the customer's billing address	Up to 20 characters (no symbols)	Required if the merchant would like to use the Address Verification Service filter. For more information on AVS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant Required for Zero Dollar Authorizations for Visa verification transactions.
Country (x_country)	The country of the customer's billing address	Up to 60 characters (no symbols)	
Phone (x_phone)	The phone number associated with the customer's billing address	Up to 25 digits (no letters) Ex. (123)123-1234	

Table 4 Payment form fields (Continued)

Field Name	Value	Format	Notes
FAX (x_fax)	The fax number associated with the customer's billing address	Up to 25 digits (no letters) Ex. (123)123-1234	
Email (x_email)	The customer's valid email address	Up to 255 characters Ex. janedoe@customer.com	The email address to which the customer's copy of the email receipt is sent when Email Receipts is configured in the Merchant Interface. The email is sent to the customer only if the email address format is valid. For more information about Email Receipts, please see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/
Customer ID (x_cust_id)	The merchant assigned customer ID	Up to 20 characters (no symbols)	The unique identifier to represent the customer associated with the transaction. The customer ID must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.
SHIPPING INFORMATION			
First Name (x_ship_to_first_name)	The first name associated with the customer's shipping address	Up to 50 characters (no symbols)	
Last Name (x_ship_to_last_name)	The last name associated with the customer's shipping address	Up to 50 characters (no symbols)	
Company (x_ship_to_company)	The company associated with the customer's shipping address	Up to 50 characters (no symbols)	
Address (x_ship_to_address)	The customer's shipping address	Up to 60 characters (no symbols)	

Table 4 Payment form fields (Continued)

Field Name	Value	Format	Notes
City (x_ship_to_city)	The city of the customer's shipping address	Up to 40 characters (no symbols)	
State (x_ship_to_state)	The state of the customer's shipping address	Up to 40 characters (no symbols) or a valid two-character state code	
ZIP Code (x_ship_to_zip)	The ZIP code of the customer's shipping address	Up to 20 characters (no symbols)	
Country (x_ship_to_country)	The country of the customer's shipping address	Up to 60 characters (no symbols)	
ADDITIONAL SHIPPING INFORMATION (Level 2 Data)			
Tax (x_tax)	The valid tax amount OR delimited tax information	When submitting delimited tax information, values must be delimited by a bracketed pipe < >	The tax amount charged OR when submitting this information by means of the HTML Form POST, delimited tax information including the sales tax name, description, and amount is also allowed. The total amount of the transaction in x_amount must <i>include</i> this amount.
	tax item name< >		The tax item name.
	tax description< >		The tax item description.
	tax amount	The dollar sign (\$) is not allowed when submitting delimited information.	The tax item amount. The total amount of the transaction in x_amount must <i>include</i> this amount.
Example:	<INPUT TYPE="HIDDEN" name="x_tax" VALUE="Tax1< >state tax< >0.0625">		
Freight (x_freight)	The valid freight amount OR delimited freight information	When submitting delimited freight information, values must be delimited by a bracketed pipe < >	The freight amount charged OR when submitting this information by means of the HTML Form POST, delimited freight information including the freight name, description, and amount is also allowed. The total amount of the transaction in x_amount must <i>include</i> this amount.
	freight item name< >		The freight item name.
	freight description< >		The freight item description.

Table 4 Payment form fields (Continued)

Field Name	Value	Format	Notes
	freight amount	The dollar sign (\$) is not allowed when submitting delimited information.	The freight amount. The total amount of the transaction in <code>x_amount</code> must <i>include</i> this amount.
Example:	<code><INPUT TYPE="HIDDEN" name="x_freight" VALUE="Freight1< >ground overnight< >12.95></code>		
Duty (<code>x_duty</code>)	The valid duty amount OR delimited duty information	When submitting delimited duty information, values must be delimited by a bracketed pipe <code>< ></code>	The duty amount charged OR when submitting this information by means of the HTML Form POST, delimited duty information including the duty name, description, and amount is also allowed. The total amount of the transaction in <code>x_amount</code> must <i>include</i> this amount.
	duty item name <code>< ></code>		The duty item name.
	duty description <code>< ></code>		The duty item description.
	duty amount	The dollar sign (\$) is not allowed when submitting delimited information.	The duty amount. The total amount of the transaction in <code>x_amount</code> must <i>include</i> this amount.
Example:	<code><INPUT TYPE="HIDDEN" name="x_duty" VALUE="Duty1< >export< >15.00></code>		
Tax Exempt (<code>x_tax_exempt</code>)	The tax exempt status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates whether the transaction is tax exempt.
Purchase Order Number (<code>x_po_num</code>)	The merchant assigned purchase order number	Up to 25 characters (no symbols)	The purchase order number must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.

**Important**

If the merchant chooses to use the standard payment gateway security features, Address Verification Service (AVS) and Card Code Verification (CCV), they need to require the customer's card code and billing address information on the payment form. These requirements must be configured in the Payment Form setting in the Merchant Interface. For more information about AVS and CCV, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

Delimited duty, freight, and tax information and all line item information are not returned in the transaction response or in the merchant confirmation email. This information is displayed only on the Transaction Detail page in the Merchant Interface. You can retrieve this information from the [Transaction Details API](#).

The Invoice Number and Customer ID must be created dynamically or provided on a per-transaction basis. The payment gateway does not perform this function.

Example Submitting itemized order information

```
x_line_item=item1<|>golf balls<|><|>2<|>18.95<|>Y&x_line_item=
item2<|>golf bag<|>Wilson golf carry bag, red<|>1<|>39.99<|>Y&
x_line_item=item3<|>book<|>Golf for Dummies<|>1<|>21.99<|>Y&
```

Merchant-defined fields

Merchants can also define special fields to further customize the information included with a transaction. Merchant-defined fields are any fields that are not recognized by the payment gateway as standard application programming interface (API) fields.

For example, the merchant might want to provide a field in which customers provide specific shipping instructions and product color information. All you need to do is submit a custom field name and any accompanying text with the transaction request string—for example, *shipping_instructions* and *product_color*.

Standard payment gateway fields that are misspelled are treated as merchant-defined fields.



Merchant-defined data fields are not intended to and must not be used to capture personally identifying information. Accordingly, the merchant is prohibited from capturing, obtaining, and/or transmitting any personally identifying information in or by means of the merchant-defined data fields. Personally identifying information includes, but is not limited to, name, address, credit card number, social security number, driver's license number, state-issued identification number, passport number, and card verification numbers (CVV, CVC2, CVV2, CID, CVN). If Authorize.Net discovers that Merchant is capturing and/or transmitting personally identifying information by means of the merchant-defined Data fields, whether or not intentionally, Authorize.Net will immediately suspend the merchant's account, which will result in a rejection of any and all transaction requests submitted by the merchant after the point of suspension.

Receipt Options

DPM provides a Relay Response feature to communicate the transaction results to the customer. The Relay Response feature of DPM enables the merchant to create a custom receipt page using transaction results returned by the payment gateway. The custom receipt page is then relayed to the customer's Web browser.

Relay Response

Relay Response does not redirect the end user to your server but relays your specified Relay URL to the end user through your receipt page instead of displaying our default receipt page. If you would like to redirect the end user to your server, provide a link on your Relay URL for this purpose.

The following table describes form fields that can be submitted to configure Relay Response. These settings can also be configured in the Merchant Interface. For more information about configuring Relay Response in the Merchant Interface, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

The form fields are submitted using the syntax:

```
<INPUT TYPE=HIDDEN NAME="x_name_of_field" VALUE="value of the field">
```

Table 5 Configuring Relay Response

Filed Name	Value	Format	Notes
x_relay_response	The request for a relay response	TRUE	This field instructs the payment gateway to return transaction results to the merchant by means of an HTML form POST to the merchant's Web server for a relay response.

Table 5 Configuring Relay Response (Continued)

Filed Name	Value	Format	Notes
x_relay_url	The URL on the merchant's website to which the payment gateway posts transaction results for a relay response	Any valid URL Including name/value pairs in the URL (anything after a "?") is not recommended	If this field is submitted, the payment gateway validates the URL value against the Relay Response URL configured in the Merchant Interface. If the URL submitted does not match the URL configured in the Merchant Interface, the transaction is rejected. If no value is submitted in the HTML Form POST, the payment gateway posts transaction results to the URL configured in the Merchant Interface.

The following code is an example of including the Relay Response request in the HTML Form POST.

**Note**

The code included in this document is pseudo code. Because code varies based on Web programming language, do not copy and paste sample code but rather use it as a guide. Additional sample code is available for download from the Authorize.Net Developer Center at <http://developer.authorize.net>.

Example Payment Form Request Including Relay Response Request

```

<!--#INCLUDE FILE= "simlib.asp"-->
<FORM METHOD=POST ACTION=
"https://secure.authorize.net/gateway/transact.dll">
    <% ret = InsertFP (APIloginid, sequence, amount, txnkey) %>
<INPUT TYPE=HIDDEN NAME="x_login" VALUE="the merchant's API Login ID">
<INPUT TYPE=HIDDEN NAME="x_version" VALUE="3.1">
<INPUT TYPE=HIDDEN NAME="x_method" VALUE="CC">
<INPUT TYPE=HIDDEN NAME="x_show_form" VALUE="PAYMENT_FORM">
<INPUT TYPE=HIDDEN NAME="x_amount" VALUE="9.95">
<INPUT TYPE=HIDDEN NAME="x_relay_response" VALUE="TRUE">
<INPUT TYPE=HIDDEN NAME="x_relay_url" VALUE="Any valid URL">
<INPUT TYPE=SUBMIT VALUE="Click here for the secure payment form">
</FORM>

```

When Authorize.Net sends a Relay Response to the merchant's server, and the server does not respond positively within 10 seconds, the connection times out and an error is generated for the transaction.

**Note**

All Web traffic to and from Authorize.Net must be on ports 80 and 443.

Tips for Using Relay Response

The Relay Response URL should be a script that can parse the transaction results posted from the payment gateway. The URL can be a plain HTML page if a static response is desired for every transaction. However, in this case the merchant's Web server should be configured to allow an HTML Form POST to a plain HTML page.

Do not rely on HTTP headers for including customer information such as cookies. When the response is relayed to the customer's browser, HTTP headers are replaced.

Keep in mind that the relay response is rendered on the payment gateway server. Custom receipt pages must incorporate absolute URLs.

Redirects or frames in the relay script are not recommended because the information might not be transferred properly.

Email Receipt

Merchants can choose to send a payment-gateway-generated email receipt to customers who provide an email address with their transaction. The email receipt includes a summary and results of the transaction. To the customer, this email appears to be sent from the merchant contact that is configured as the Email Sender in the Merchant

Interface. (For more information about the Email Sender setting, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.)

To send the payment gateway-generated customer email receipt, submit the API fields that appear in the following table with the transaction request string. These settings can also be configured in the Merchant Interface.

Submit the form fields using the syntax:

```
x_name_of_field=value of t.he field&
```

DPM provides a Relay Response feature to communicate the transaction results to the customer. The Relay Response feature of DPM enables the merchant to create a custom receipt page using transaction results returned by the payment gateway. The custom receipt page is then relayed to the customer's Web browser

In addition, the merchant can receive a transaction confirmation email from the payment gateway at the completion of each transaction, which includes order information and the results of the transaction. Merchants can sign up for confirmation emails in the Merchant Interface. For more information, please see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

Additional API Fields

The following tables describe additional API fields that can be submitted by means of the transaction request to the payment gateway. The form fields are submitted using the syntax:

```
<INPUT TYPE=HIDDEN NAME="x_name_of_field" VALUE="value of the field">
```

Transaction Information

The following fields contain optional or conditional transaction-specific information.

Table 6 Fields Containing Transaction-Specific Information

Field Name	Value	Format	Notes
x_version	The merchant's transaction version	3.0, 3.1	<p>Indicates to the system the set of fields that will be included in the response:</p> <p>3.0 is the default version.</p> <p>3.1 allows the merchant to utilize the Card Code feature, and is the current standard version.</p> <p>We recommend that you submit this field on a per-transaction basis, particularly if you are using Relay Response. For more information, see "Relay Response," page 29 and Appendix A, "Fields by Transaction Type," on page 79.</p>
x_method	The payment method	CC or ECHECK	<p>The method of payment for the transaction, CC (credit card), or ECHECK (electronic check). If left blank, this value defaults to CC.</p> <p>For more information about eCheck.Net transaction requirements, see the <i>eCheck.Net Developer Guide</i> at http://developer.authorize.net/guides/echeck.pdf.</p>

Table 6 Fields Containing Transaction-Specific Information (Continued)

Field Name	Value	Format	Notes
x_test_request	The request to process test transactions	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates if the transaction should be processed as a test transaction. See " Test Transactions ," page 60 for more information.
x_duplicate_window	The window of time after the submission of a transaction during which a duplicate transaction cannot be submitted	Any value between 0 and 28800 (no comma)	Indicates in seconds the window of time after a transaction is submitted during which the payment gateway will check for a duplicate transaction. The maximum time allowed is 8 hours (28800 seconds). If a value less than 0 is sent, the payment gateway will default to 0 seconds. If a value greater than 28800 is sent, the payment gateway will default to 28800. If no value is sent, the payment gateway will default to 2 minutes (120 seconds). If this field is present in the request with or without a value, an enhanced duplicate transaction response is sent. See " Response for Duplicate Transactions ," page 43 for more information.

Itemized Order Information

Based on their respective business requirements, merchants can choose to submit itemized order information with a transaction. Itemized order information is not submitted to the processor and is currently not returned with the transaction response. This information is displayed on the Transaction Detail page and in the QuickBooks download file reports in the Merchant Interface.



Note

The value for the x_line_item field can include delimited item information. In this case, line item values must be included in the order listed below.

Table 7 Itemized Order Information

Field Name	Value	Format	Notes
x_line_item	Any string	Line item values must be delimited by a bracketed pipe < >	Itemized order information.
	Item ID< >	Up to 31 characters	The ID assigned to an item.
	< >item name< >	Up to 31 characters	A short description of an item.
	< >item description< >	Up to 255 characters	A detailed description of an item.
	< >itemX quantity< >	Up to two decimal places Must be a positive number	The quantity of an item.
	< >item price (unit cost)< >	Up to two decimal places Must be a positive number	Cost of an item per unit, <i>excluding</i> tax, freight, and duty. The dollar sign (\$) is not allowed when submitting delimited information.
	< >itemX taxable	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates whether the item is subject to tax.

The merchant can submit up to 30 distinct line items containing itemized order information per transaction. For example:

Example Submitting Itemized Order Information

```
<INPUT TYPE="HIDDEN" name="x_line_item" VALUE="item1<|>golf
balls<|><|>2<|>18.95<|>Y">
<INPUT TYPE="HIDDEN" name="x_line_item" VALUE="item2<|>golf bag<|>Wilson
golf carry bag, red<|>1<|>39.99<|>Y">
<INPUT TYPE="HIDDEN" name="x_line_item" VALUE="item3<|>book
<|>Golf for Dummies<|>1<|>21.99<|>Y">
```



Note

For Prior Authorization and Capture transactions, if line item information was submitted with the original transaction, adjusted information can be submitted if the transaction changed. If no adjusted line item information is submitted, the information submitted with the original transaction will apply.

Additional Customer Information

The following fields describe additional customer information that can be submitted with each transaction.

Table 8 Additional Customer Information

Field Name	Value	Format	Notes
x_customer_ip	The customer's IP address	Up to 15 characters (no letters) Example: 255.255.255.255	IP address of the customer initiating the transaction. If this value is not passed, it will default to 255.255.255.255. This field is required when using the Fraud Detection Suite™ (FDS) IP Address Blocking tool. For more information about FDS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .

Transaction Response

When Relay Response is configured, the transaction response that is returned to the merchant from the payment gateway is a set of fields that provides information about the status of a transaction—whether it was accepted or declined—as well as information included in the transaction request.

The merchant server can parse data in the transaction response and customize the message to display to the customer. Transaction results are also provided in the merchant confirmation email, customer email receipt (if configured), and on the Transaction Detail page for the transaction in the Merchant Interface.

Fields in the Payment Gateway Response

The following table lists the fields returned in the response from the payment gateway.

Please note that the transaction response fields are not necessarily sent in the exact order listed here. Developers are encouraged to use the name of the field in order to locate the correct response. If your code expects transaction response fields in a particular order, future updates to the DPM API may cause unexpected results from your code.

Table 9 Fields in the Payment Gateway Response

Field Name	Value	Format	Notes
x_response_code	The overall status of the transaction	1 = Approved 2 = Declined 3 = Error 4 = Held for Review	
x_response_reason_code	A code that represents more details about the result of the transaction	Numeric	See "Response Code Details," page 45 for a listing of response reason codes.

Table 9 Fields in the Payment Gateway Response (Continued)

Field Name	Value	Format	Notes
x_response_reason_text	A brief description of the result, which corresponds with the response reason code	Text	You can generally use this text to display a transaction result or error to the customer. However, review " Response Code Details ," page 45 to identify any specific texts you do not want to pass to the customer.
x_auth_code	The authorization or approval code	6 characters	
x_avs_code	The Address Verification Service (AVS) response code	<p>A = Address (Street) matches, ZIP does not</p> <p>B = Address information not provided for AVS check</p> <p>E = AVS error</p> <p>G = Non-U.S. Card Issuing Bank</p> <p>N = No Match on Address (Street) or ZIP</p> <p>P = AVS not applicable for this transaction</p> <p>R = Retry – System unavailable or timed out</p> <p>S = Service not supported by issuer</p> <p>U = Address information is unavailable</p> <p>W = Nine digit ZIP matches, Address (Street) does not</p> <p>X = Address (Street) and nine digit ZIP match</p> <p>Y = Address (Street) and five digit ZIP match</p> <p>Z = Five digit ZIP matches, Address (Street) does not</p>	<p>Indicates the result of the (AVS) filter.</p> <p>For more information about AVS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/.</p>
x_trans_id	The payment gateway assigned identification number for the transaction	When x_test_request is submitted, this value will be 0	This value must be used for any follow-on transactions such as a CREDIT, PRIOR_AUTH_CAPTURE or VOID.
x_invoice_num	The merchant assigned invoice number for the transaction	Up to 20 characters (no symbols)	

Table 9 Fields in the Payment Gateway Response (Continued)

Field Name	Value	Format	Notes
x_description	The transaction description	Up to 255 characters (no symbols)	
x_amount	The amount of the transaction	Up to 15 digits	
x_method	The payment method	CC or ECHECK	
x_type	The type of credit card transaction	AUTH_CAPTURE, AUTH_ONLY, CREDIT, PRIOR_AUTH_CAPTURE, VOID	
x_account_number	Last 4 digits of the card provided	Alphanumeric (XXXX6835)	
x_card_type	Visa, MasterCard, American Express, Discover, Diners Club, JCB	Text	
x_split_tender_id	Value that links the current authorization request to the original authorization request. This value is returned in the reply message from the original authorization request	Alphanumeric	This is returned in the reply message for the first transaction that receives a partial authorization.
x_prepaid_requested_amount	Amount requested in the original authorization	Numeric	This is present if the current transaction is for a prepaid card or if a splitTenderId was sent.
x_prepaid_balance_on_card	Balance on the debit card or prepaid card	Numeric	This is present if the current transaction is for a prepaid card or if a splitTenderId was sent.
x_cust_id	The merchant assigned customer ID	Up to 20 characters (no symbols)	
x_first_name	The first name associated with the customer's billing address	Up to 50 characters (no symbols)	
x_last_name	The last name associated with the customer's billing address	Up to 50 characters (no symbols)	

Table 9 Fields in the Payment Gateway Response (Continued)

Field Name	Value	Format	Notes
x_company	The company associated with the customer's billing address	Up to 50 characters (no symbols)	
x_address	The customer's billing address	Up to 60 characters (no symbols)	
x_city	The city of the customer's billing address	Up to 40 characters (no symbols)	
x_state	The state of the customer's billing address	Up to 40 characters (no symbols) or a valid two-character state code	
x_zip	The ZIP code of the customer's billing address	Up to 20 characters (no symbols)	
x_country	The country of the customer's billing address	Up to 60 characters (no symbols)	
x_phone	The phone number associated with the customer's billing address	Up to 25 digits (no letters) Ex. (123)123-1234	
x_fax	The fax number associated with the customer's billing address	Up to 25 digits (no letters) Ex. (123)123-1234	
x_email	The customer's valid email address	Up to 255 characters	
x_ship_to_first_name	The first name associated with the customer's shipping address	Up to 50 characters (no symbols)	
x_ship_to_last_name	The last name associated with the customer's shipping address	Up to 50 characters (no symbols)	
x_ship_to_company	The company associated with the customer's shipping address	Up to 50 characters (no symbols)	
x_ship_to_address	The customer's shipping address	Up to 60 characters (no symbols)	

Table 9 Fields in the Payment Gateway Response (Continued)

Field Name	Value	Format	Notes
x_ship_to_city	The city of the customer's shipping address	Up to 40 characters (no symbols)	
x_ship_to_state	The state of the customer's shipping address	Up to 40 characters (no symbols) or a valid two-character state code	
x_ship_to_zip	The ZIP code of the customer's shipping address	Up to 20 characters (no symbols)	
x_ship_to_country	The country of the customer's shipping address	Up to 60 characters (no symbols)	
x_tax	The tax amount charged	Numeric	Delimited tax information is not included in the transaction response.
x_duty	The duty amount charged	Numeric	Delimited duty information is not included in the transaction response.
x_freight	The freight amount charged	Numeric	Delimited freight information is not included in the transaction response.
x_tax_exempt	The tax exempt status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	
x_po_num	The merchant assigned purchase order number	Up to 25 characters (no symbols)	
x_MD5_Hash	The payment gateway generated MD5 hash value that can be used to authenticate the transaction response.		For more information about creating an MD5 hash value, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .
x_cvv2_resp_code	The card code verification (CCV) response code	M = Match N = No Match P = Not Processed S = Should have been present U = Issuer unable to process request	Indicates the result of the CCV filter. For more information about CCV, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .

Table 9 Fields in the Payment Gateway Response (Continued)

Field Name	Value	Format	Notes
x_cavv_response	The cardholder authentication verification response code	Blank or not present = CAVV not validated 0 = CAVV not validated because erroneous data was submitted 1 = CAVV failed validation 2 = CAVV passed validation 3 = CAVV validation could not be performed; issuer attempt incomplete 4 = CAVV validation could not be performed; issuer system error 5 = Reserved for future use 6 = Reserved for future use 7 = CAVV attempt – failed validation – issuer available (U.S.-issued card/non-U.S acquirer) 8 = CAVV attempt – passed validation – issuer available (U.S.-issued card/non-U.S. acquirer) 9 = CAVV attempt – failed validation – issuer unavailable (U.S.-issued card/non-U.S. acquirer) A = CAVV attempt – passed validation – issuer unavailable (U.S.-issued card/non-U.S. acquirer) B = CAVV passed validation, information only, no liability shift	The cardholder authentication programs are not applicable to DPM.

Using the MD5 Hash Feature

The MD5 Hash feature enables you to authenticate that a transaction response is securely received from Authorize.Net. The payment gateway creates the MD5 hash using the following pieces of account and transaction information as input:

- MD5 Hash value
- API Login ID (x_login)
- Transaction ID (x_trans_id)
- Amount (x_amount)

The MD5 Hash value is a random value configured by the merchant in the Merchant Interface. It should be stored securely separately from the merchant's Web server. For more information on how to configure this value, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

**Note**

MD5 Hash values are returned in transaction responses even when the merchant has not configured a value in the Merchant Interface.

For example, if the MD5 Hash value configured by the merchant in the Merchant Interface is *wilson*, the API Login ID is *myAPIloginid*, the Transaction ID is 987654321, and the amount is \$1.00, then the field order used by the payment gateway to generate the MD5 Hash would be as follows.

Example MD5 Hash Input Field Order

```
wilsonmyAPIloginid9876543211.00
```

**Note**

The value passed back for x_amount is formatted with the correct number of decimal places used in the transaction. For transaction types that do not include a transaction amount, the amount used by the payment gateway to calculate the MD5 Hash is 0.00.

To authenticate the MD5 Hash returned by the payment gateway in the transaction response, create a script that can receive and parse the transaction response, call the merchant's MD5 Hash value, and run the MD5 algorithm on the same fields listed above. If the result matches the MD5 hash returned by the payment gateway, the transaction response is successfully authenticated.

Response for Duplicate Transactions

The DPM API enables you to specify the window of time after a transaction is submitted during which the payment gateway checks for a duplicate transaction (based on credit card number, invoice number, amount, billing address information, transaction type, etc.) using the duplicate window field (x_duplicate_window). The value for this field can be between 0 and 28800 seconds (maximum of 8 hours).

If the transaction request does not include the duplicate window field, and the payment gateway detects a duplicate transaction within the default window of 2 minutes, the

payment gateway response will contain the response code of 3 (processing error) with a response reason code of 11 (duplicate transaction) with no additional details.

If the transaction request *does* include the duplicate window field and value, and the payment gateway detects a duplicate transaction within the window of time specified, the payment gateway response for the duplicate transaction will include the response code and response reason code listed above, as well as information about the original transaction (as outlined below).

If the original transaction was declined, and a value was passed in the duplicate window field, the payment gateway response for the duplicate transaction will include the following information for the original transaction:

- The AVS result
- The CCV result
- The transaction ID
- The MD5 hash (if this feature was used for the original transaction)

If the original transaction was approved, and a value was passed in the duplicate window field, the payment gateway response will also include the authorization code for the original transaction. All duplicate transactions submitted after the duplicate window, whether specified in the transaction request or after the payment gateway's default 2 minute duplicate window, are processed normally.

DPM Relay Response

The response from the gateway to a request by means of DPM for a Relay Response consists of a set of fields returned as a POST string to the merchant server at the location indicated in the `x_relay_url` field.

DPM Transaction Response Versions

There are two versions of the response string. The set of fields in the response differ based on the response version.

Version 3.0

The version 3.0 response contains system fields from position 1 to 38 and echoes merchant defined fields from 39 on, in the order received by the system. Version 3.0 is the Payment Gateway default.

Version 3.1

The version 3.1 response string contains 68 system fields, with field number 39 representing the Card Code (CVV2/CVC2/CID) response code. Merchant-defined fields are echoed from field 69 on. Merchants wishing to use partial authorizations or the Card Code feature must use transaction version 3.1.

Upgrading the Transaction Version

To upgrade the transaction version, follow these steps (only users with the appropriate permissions will be able to access this setting):

- Step 1** Log on to the Merchant Interface.
- Step 2** Select **Settings** from the Main Menu.
- Step 3** Click **Transaction Version** in the Transaction Response section.
- Step 4** Change the transaction version using the drop-down box.
- Step 5** Click **Submit**.



You can only upgrade to a higher transaction version. You cannot set your transaction version to a previous version.

Response Code Details

The following tables describe the response codes and response reason texts that are returned for each transaction. In addition to the information in this document, the Authorize.Net Developer Center at <http://developer.authorize.net/tools/responsereasoncode> provides a valuable tool for troubleshooting errors.

- **Response Code** indicates the overall status of the transaction with possible values of approved, declined, error, or held for review.
- **Response Reason Code** is a numeric representation of a more specific reason for the transaction status.
- **Response Reason Text** details the specific reason for the transaction status. This information can be returned to the merchant and/or customer to provide more information about the status of the transaction.

Response Codes

Table 10 Response Codes

Response Code	Description
1	This transaction has been approved.
2	This transaction has been declined.
3	There has been an error processing this transaction.
4	This transaction is being held for review.

Response Reason Codes and Response Reason Text

Table 11 Response Reason Codes and Response Reason Text

Response Code	Response Reason Code	Response Reason Text	Notes
1	1	This transaction has been approved.	
2	2	This transaction has been declined.	
2	3	This transaction has been declined.	
2	4	This transaction has been declined.	The code returned from the processor indicates that the card used needs to be picked up.
3	5	A valid amount is required.	The value submitted in the amount field did not pass validation for a number.
3	6	The credit card number is invalid.	
3	7	The credit card expiration date is invalid.	The format of the date submitted was incorrect.
3	8	The credit card has expired.	
3	9	The ABA code is invalid.	The value submitted in the x_bank_aba_code field did not pass validation or was not for a valid financial institution.
3	10	The account number is invalid.	The value submitted in the x_bank_acct_num field did not pass validation.
3	11	A duplicate transaction has been submitted.	A transaction with identical amount payment information was submitted during the Duplicate Transaction Window for the original transaction. Please see the "Response for Duplicate Transactions" section for more details.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	12	An authorization code is required but not present.	A transaction that required x_auth_code to be present was submitted without a value.
3	13	The merchant API Login ID is invalid or the account is inactive.	
3	14	The Referrer or Relay Response URL is invalid.	The Relay Response or Referrer URL does not match the merchant's configured value(s) or is absent. Applicable only to DPM and WebLink APIs.
3	15	The transaction ID is invalid.	The transaction ID value is non-numeric or was not present for a transaction that requires it (such as VOID, PRIOR_AUTH_CAPTURE, and CREDIT).
3	16	The transaction was not found.	The transaction ID sent in was properly formatted but the gateway had no record of the transaction for the gateway account used.
3	17	The merchant does not accept this type of credit card.	The merchant was not configured to accept the credit card type submitted in the transaction.
3	18	ACH transactions are not accepted by this merchant.	The merchant does not accept electronic checks.
3	19 - 23	An error occurred during processing. Please try again in 5 minutes.	
3	24	The Nova Bank Number or Terminal ID is incorrect. Call Merchant Service Provider.	
3	25 - 26	An error occurred during processing. Please try again in 5 minutes.	
2	27	The transaction resulted in an AVS mismatch. The address provided does not match billing address of cardholder.	
2	28	The merchant does not accept this type of credit card.	The Merchant ID at the processor was not configured to accept this card type.
2	29	The Paymentech identification numbers are incorrect. Call Merchant Service Provider.	
2	30	The configuration with the processor is invalid. Call Merchant Service Provider.	

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
2	31	The FDC Merchant ID or Terminal ID is incorrect. Call Merchant Service Provider.	The merchant was incorrectly set up at the processor.
3	32	This reason code is reserved or not applicable to this API.	
3	33	<i>FIELD</i> cannot be left blank.	The word <i>FIELD</i> will be replaced by an actual field name. This error indicates that a field the merchant specified as required was not filled in. Please see the Form Settings topic in the <i>Merchant Integration Guide</i> for details.
2	34	The VITAL identification numbers are incorrect. Call Merchant Service Provider.	The merchant was incorrectly set up at the processor.
2	35	An error occurred during processing. Call Merchant Service Provider.	The merchant was incorrectly set up at the processor.
3	36	The authorization was approved, but settlement failed.	
2	37	The credit card number is invalid.	
2	38	The Global Payment System identification numbers are incorrect. Call Merchant Service Provider.	The merchant was incorrectly set up at the processor.
3	40	This transaction must be encrypted.	
2	41	This transaction has been declined.	Only merchants set up for the FraudScreen.Net service would receive this decline. This code will be returned if a given transaction's fraud score is higher than the threshold set by the merchant.
3	43	The merchant was incorrectly set up at the processor. Call your Merchant Service Provider.	The merchant was incorrectly set up at the processor.
2	44	This transaction has been declined.	The card code submitted with the transaction did not match the card code on file at the card issuing bank and the transaction was declined.
2	45	This transaction has been declined.	This error would be returned if the transaction received a code from the processor that matched the rejection criteria set by the merchant for both the AVS and Card Code filters.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	46	Your session has expired or does not exist. You must log in to continue working.	
3	47	The amount requested for settlement may not be greater than the original amount authorized.	This occurs if the merchant tries to capture funds greater than the amount of the original authorization-only transaction.
3	48	This processor does not accept partial reversals.	The merchant attempted to settle for less than the originally authorized amount.
3	49	A transaction amount greater than \$[amount] will not be accepted.	The transaction amount submitted was greater than the maximum amount allowed.
3	50	This transaction is awaiting settlement and cannot be refunded.	Credits or refunds can only be performed against settled transactions. The transaction against which the credit/refund was submitted has not been settled, so a credit cannot be issued.
3	51	The sum of all credits against this transaction is greater than the original transaction amount.	
3	52	The transaction was authorized, but the client could not be notified; the transaction will not be settled.	
3	53	The transaction type was invalid for ACH transactions.	If x_method = ECHECK, x_type cannot be set to CAPTURE_ONLY.
3	54	The referenced transaction does not meet the criteria for issuing a credit.	
3	55	The sum of credits against the referenced transaction would exceed the original debit amount.	The transaction is rejected if the sum of this credit and prior credits exceeds the original debit amount.
3	56	This merchant accepts ACH transactions only; no credit card transactions are accepted.	The merchant processes eCheck.Net transactions only and does not accept credit cards.
3	57 - 63	An error occurred in processing. Please try again in 5 minutes.	
2	65	This transaction has been declined.	The transaction was declined because the merchant configured their account through the Merchant Interface to reject transactions with certain values for a Card Code mismatch.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	66	This transaction cannot be accepted for processing.	The transaction did not meet gateway security guidelines.
3	68	The version parameter is invalid.	The value submitted in x_version was invalid.
3	69	The transaction type is invalid.	The value submitted in x_type was invalid.
3	70	The transaction method is invalid.	The value submitted in x_method was invalid.
3	71	The bank account type is invalid.	The value submitted in x_bank_acct_type was invalid.
3	72	The authorization code is invalid.	The value submitted in x_auth_code was more than six characters in length.
3	73	The driver's license date of birth is invalid.	The format of the value submitted in x_drivers_license_dob was invalid.
3	74	The duty amount is invalid.	The value submitted in x_duty failed format validation.
3	75	The freight amount is invalid.	The value submitted in x_freight failed format validation.
3	76	The tax amount is invalid.	The value submitted in x_tax failed format validation.
3	77	The SSN or tax ID is invalid.	The value submitted in x_customer_tax_id failed validation.
3	78	The Card Code (CVV2/CVC2/CID) is invalid.	The value submitted in x_card_code failed format validation.
3	79	The driver's license number is invalid.	The value submitted in x_drivers_license_num failed format validation.
3	80	The driver's license state is invalid.	The value submitted in x_drivers_license_state failed format validation.
3	81	The requested form type is invalid.	The merchant requested an integration method not compatible with the AIM API.
3	82	Scripts are only supported in version 2.5.	The system no longer supports version 2.5; requests cannot be posted to scripts.
3	83	The requested script is either invalid or no longer supported.	The system no longer supports version 2.5; requests cannot be posted to scripts.
3	84	This reason code is reserved or not applicable to this API.	

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	85	This reason code is reserved or not applicable to this API.	
3	86	This reason code is reserved or not applicable to this API.	
3	87	This reason code is reserved or not applicable to this API.	
3	88	This reason code is reserved or not applicable to this API.	
3	89	This reason code is reserved or not applicable to this API.	
3	90	This reason code is reserved or not applicable to this API.	
3	91	Version 2.5 is no longer supported.	
3	92	The gateway no longer supports the requested method of integration.	
3	97	This transaction cannot be accepted.	Applicable only to DPM API. Fingerprints are only valid for a short period of time. This code indicates that the transaction fingerprint has expired.
3	98	This transaction cannot be accepted.	Applicable only to DPM API. The transaction fingerprint has already been used.
3	99	This transaction cannot be accepted.	Applicable only to DPM API. The server-generated fingerprint does not match the merchant-specified fingerprint in the x_fp_hash field.
3	100	The eCheck.Net type is invalid.	Applicable only to eCheck.Net. The value specified in the x_echeck_type field was invalid.
3	101	The given name on the account and/or the account type does not match the actual account.	Applicable only to eCheck.Net. The specified name on the account and/or the account type do not match the NOC record for this account.
3	102	This request cannot be accepted.	A password or Transaction Key was submitted with this WebLink request. This is a high security risk.
3	103	This transaction cannot be accepted.	A valid fingerprint, Transaction Key, or password is required for this transaction.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	104	This transaction is currently under review.	Applicable only to eCheck.Net. The value submitted for country failed validation.
3	105	This transaction is currently under review.	Applicable only to eCheck.Net. The values submitted for city and country failed validation.
3	106	This transaction is currently under review.	Applicable only to eCheck.Net. The value submitted for company failed validation.
3	107	This transaction is currently under review.	Applicable only to eCheck.Net. The value submitted for bank account name failed validation.
3	108	This transaction is currently under review.	Applicable only to eCheck.Net. The values submitted for first name and last name failed validation.
3	109	This transaction is currently under review.	Applicable only to eCheck.Net. The values submitted for first name and last name failed validation.
3	110	This transaction is currently under review.	Applicable only to eCheck.Net. The value submitted for bank account name does not contain valid characters.
3	120	An error occurred during processing. Please try again.	The system-generated void for the original timed-out transaction failed. (The original transaction timed out while waiting for a response from the authorizer.)
3	121	An error occurred during processing. Please try again.	The system-generated void for the original errored transaction failed. (The original transaction experienced a database error.)
3	122	An error occurred during processing. Please try again.	The system-generated void for the original errored transaction failed. (The original transaction experienced a processing error.)
3	123	This account has not been given the permission(s) required for this request.	The transaction request must include the API Login ID associated with the payment gateway account.
2	127	The transaction resulted in an AVS mismatch. The address provided does not match billing address of cardholder.	The system-generated void for the original AVS-rejected transaction failed.
3	128	This transaction cannot be processed.	The customer's financial institution does not currently allow transactions for this account.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	130	This payment gateway account has been closed.	IFT: The payment gateway account status is Blacklisted.
3	131	This transaction cannot be accepted at this time.	IFT: The payment gateway account status is Suspended-STA.
3	132	This transaction cannot be accepted at this time.	IFT: The payment gateway account status is Suspended-Blacklist.
2	141	This transaction has been declined.	The system-generated void for the original FraudScreen-rejected transaction failed.
2	145	This transaction has been declined.	The system-generated void for the original card code-rejected and AVS-rejected transaction failed.
3	152	The transaction was authorized, but the client could not be notified; the transaction will not be settled.	The system-generated void for the original transaction failed. The response for the original transaction could not be communicated to the client.
2	165	This transaction has been declined.	The system-generated void for the original card code-rejected transaction failed.
3	170	An error occurred during processing. Please contact the merchant.	Concord EFS – Provisioning at the processor has not been completed.
2	171	An error occurred during processing. Please contact the merchant.	Concord EFS – This request is invalid.
2	172	An error occurred during processing. Please contact the merchant.	Concord EFS – The store ID is invalid.
3	173	An error occurred during processing. Please contact the merchant.	Concord EFS – The store key is invalid.
2	174	The transaction type is invalid. Please contact the merchant.	Concord EFS – This transaction type is not accepted by the processor.
3	175	The processor does not allow voiding of credits.	Concord EFS – This transaction is not allowed. The Concord EFS processing platform does not support voiding credit transactions. Please debit the credit card instead of voiding the credit.
3	180	An error occurred during processing. Please try again.	The processor response format is invalid.
3	181	An error occurred during processing. Please try again.	The system-generated void for the original invalid transaction failed. (The original transaction included an invalid processor response format.)

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	185	This reason code is reserved or not applicable to this API.	
4	193	The transaction is currently under review.	The transaction was placed under review by the risk management system.
2	200	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The credit card number is invalid.
2	201	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The expiration date is invalid.
2	202	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The transaction type is invalid.
2	203	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The value submitted in the amount field is invalid.
2	204	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The department code is invalid.
2	205	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The value submitted in the merchant number field is invalid.
2	206	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The merchant is not on file.
2	207	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The merchant account is closed.
2	208	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The merchant is not on file.
2	209	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. Communication with the processor could not be established.
2	210	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The merchant type is incorrect.
2	211	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The cardholder is not on file.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
2	212	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The bank configuration is not on file
2	213	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The merchant assessment code is incorrect.
2	214	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. This function is currently unavailable.
2	215	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The encrypted PIN field format is invalid.
2	216	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The ATM term ID is invalid.
2	217	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. This transaction experienced a general message format problem.
2	218	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The PIN block format or PIN availability value is invalid.
2	219	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The ETC void is unmatched.
2	220	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The primary CPU is not available.
2	221	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. The SE number is invalid.
2	222	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. Duplicate auth request (from INAS).
2	223	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. This transaction experienced an unspecified error.
2	224	This transaction has been declined.	This error code applies only to merchants on FDC Omaha. Please re-enter the transaction.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	243	Recurring billing is not allowed for this eCheck.Net type.	The combination of values submitted for x_recurring_billing and x_echeck_type is not allowed.
3	244	This eCheck.Net type is not allowed for this Bank Account Type.	The combination of values submitted for x_bank_acct_type and x_echeck_type is not allowed.
3	245	This eCheck.Net type is not allowed when using the payment gateway hosted payment form.	The value submitted for x_echeck_type is not allowed when using the payment gateway hosted payment form.
3	246	This eCheck.Net type is not allowed.	The merchant's payment gateway account is not enabled to submit the eCheck.Net type.
3	247	This eCheck.Net type is not allowed.	The combination of values submitted for x_type and x_echeck_type is not allowed.
2	250	This transaction has been declined.	This transaction was submitted from a blocked IP address.
2	251	This transaction has been declined.	The transaction was declined as a result of triggering a Fraud Detection Suite filter.
4	252	Your order has been received. Thank you for your business!	The transaction was accepted, but is being held for merchant review. The merchant can customize the customer response in the Merchant Interface.
4	253	Your order has been received. Thank you for your business!	The transaction was accepted and was authorized, but is being held for merchant review. The merchant can customize the customer response in the Merchant Interface.
2	254	Your transaction has been declined.	The transaction was declined after manual review.
3	261	An error occurred during processing. Please try again.	The transaction experienced an error during sensitive data encryption and was not processed. Please try again.
3	270	The line item [item number] is invalid.	A value submitted in x_line_item for the item referenced is invalid.
3	271	The number of line items submitted is not allowed. A maximum of 30 line items can be submitted.	The number of line items submitted exceeds the allowed maximum of 30.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
3	288	Merchant is not registered as a Cardholder Authentication participant. This transaction cannot be accepted.	The merchant has not indicated participation in any Cardholder Authentication Programs in the Merchant Interface.
3	289	This processor does not accept zero dollar authorization for this card type.	Your credit card processing service does not yet accept zero dollar authorizations for Visa credit cards. You can find your credit card processor listed on your merchant profile.
3	290	One or more required AVS values for zero dollar authorization were not submitted.	When submitting authorization requests for Visa, the address and zip code fields must be entered.
4	295	The amount of this request was only partially approved on the given pre-paid credit card. A second credit card is required to complete the balance of this transaction.	The amount authorized is less than the requested transaction amount.
3	296	The specified SplitTenderId is not valid.	
3	297	A Transaction ID and a Split Tender ID cannot both be used in a single transaction request.	
3	300	The device ID is invalid.	The value submitted for x_device_id is invalid.
3	301	The device batch ID is invalid.	The value submitted for x_device_batch_id is invalid.
3	302	The reversal flag is invalid.	The value submitted for x_reversal is invalid.
3	303	The device batch is full. Please close the batch.	The current device batch must be closed manually from the POS device.
3	304	The original transaction is in a closed batch.	The original transaction has been settled and cannot be reversed.
3	305	The merchant is configured for auto-close.	This merchant is configured for auto-close and cannot manually close batches.
3	306	The batch is already closed.	The batch is already closed.
1	307	The reversal was processed successfully.	The reversal was processed successfully.
1	308	Original transaction for reversal not found.	The transaction submitted for reversal was not found.
3	309	The device has been disabled.	The device has been disabled.

Table 11 Response Reason Codes and Response Reason Text (Continued)

Response Code	Response Reason Code	Response Reason Text	Notes
1	310	This transaction has already been voided.	This transaction has already been voided.
1	311	This transaction has already been captured	This transaction has already been captured.
3	312	The specified security code was invalid.	The customer entered the wrong security code. A new security code will be generated, and the customer will be prompted to try again until successful.
3	313	The customer requested a new security code.	The customer requested a new security code. A new security code will be generated, and the customer will be prompted to try again until successful.
2	315	The credit card number is invalid.	This is a processor-issued decline.
2	316	The credit card expiration date is invalid.	This is a processor-issued decline.
2	317	The credit card has expired.	This is a processor-issued decline.
2	318	A duplicate transaction has been submitted.	This is a processor-issued decline.
2	319	The transaction cannot be found.	This is a processor-issued decline.

Example of a Response for Partial Authorization Transactions

If a split tender ID was passed in, then the response includes each of the following fields. Each field will hold the current transaction followed by all transactions associated with the given splitTenderID, in order from oldest to newest. A pipe ("|") character is used to separate each value. All parameters will hold the same number of values.

Example Partial Authorization Response

Field	Value
x_response_code	1 1
x_response_reason_code	1 1
x_response_reason_text	This transaction has been approved. This transaction has been approved.
x_avs_code	Y Y
x_auth_code	C1RR33 04OSH9
x_trans_id	2147801919 2147801918

Example Partial Authorization Response (Continued)

x_method	CC CC
x_card_type	American Express American Express
x_prepaid_balance_on_card	0.00
x_prepaid_requested_amount	7.53
x_account_number	XXXX0002 XXXX0002
x_cvv2_resp_code	
x_cavv_response	2
x_amount	8.0000 1.23

Test Transactions

You must test the payment gateway integration carefully before going live to ensure successful and smooth transaction processing.

Use an Authorize.Net developer test account to submit test transactions through your integration. In this environment, test transactions are posted to <https://test.authorize.net/gateway/transact.dll>. Although this is a staging environment, its behavior mimics the live payment gateway. Transactions submitted to the test environment using a developer test account are not submitted to financial institutions for authorization and are not permanently stored in the Merchant Interface, although they will appear in the Unsettled Transactions area for the test account immediately after you submit them.

In order to use this environment, you must have an Authorize.Net *developer* test account with an associated API Login ID and Transaction Key. Test transactions to this environment are accepted with these credentials only. If you do not have a developer test account, you can sign up for one at <http://developer.authorize.net>.

**Note**

You do not need to use Test Mode when testing with a developer test account. For more information about Test Mode, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

Once the integration is successfully tested in the developer *test* environment, the *merchant's* Authorize.Net Payment Gateway API Login ID and Transaction Key can be plugged into the integration for testing against the *live* environment. (Developer test account credentials are not accepted by the live payment gateway.) In this phase, testing can be done in one of two ways:

- 1 By including the `x_test_request` field with a value of "TRUE" in the HTML Form POST to <https://secure.authorize.net/gateway/transact.dll>. See the sample below.

Example Submitting the Test Request Field

```
<INPUT TYPE="HIDDEN" NAME="x_test_request" VALUE="TRUE">
```

- 2 By placing the merchant's payment gateway account in Test Mode in the Merchant Interface. New payment gateway accounts are placed in Test Mode by default. For more information about Test Mode, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>. When processing test transactions in Test Mode, the payment gateway will return a transaction ID of "0." This means you

cannot test follow-on transactions while in Test Mode. To test follow-on transactions, you can either submit “x_test_request=TRUE” as indicated above, or process a test transaction with any valid credit card number in live mode, as explained below.

**Note**

Transactions posted against live merchant accounts using either of the above testing methods are **not** submitted to financial institutions for authorization and are not stored in the Merchant Interface.

If testing in the live environment is successful, you are ready to submit live transactions and verify that they are being submitted successfully. Either remove the x_test_request field from the HTML Form Post or set it to FALSE. Or if you are using Test Mode, turn it off in the Merchant Interface. To receive a true response, you must submit a transaction using a real credit card number. You can use any valid credit card number to submit a test transaction. You will be able to void successful transactions immediately to prevent live test transactions from being processed. This can be done quickly on the Unsettled Transactions page of the Merchant Interface. We recommend that when testing using a live credit card, you use a nominal value, such as \$0.01. That way, if you forget to void the transaction, the impact will be minimal. For VISA verification transactions, you can submit a \$0.00 value instead, if the credit card processor accepts it.

**Note**

VISA verification transactions are being changed from \$0.01 to \$0.00 for all processors. For Visa transactions using \$0.00, the Bill To address (x_address) and zip code (x_zip) fields are required.

Testing to Generate Specific Transaction Results

When testing transaction results in the developer test environment as well as the production environment, you can produce a specific response reason code by submitting a test transaction using a test credit card number designed to generate specific transaction results: Visa test credit card number 422222222222. This card number is intended for testing and should only be used for that purpose. Submit the test transaction by either placing the account in Test Mode or submitting x_test_request=TRUE, with a dollar amount value equal to the response reason code you would like to produce.

For example, to test the AVS response reason code number 27, submit the test transaction with the credit card number 422222222222 and the amount 27.00.

To test the AVS or CCV responses in the live environment, you must submit live transactions with correct street address, ZIP Code, and Card Code information to generate successful responses, and incorrect street address, ZIP Code, and Card Code information to generate other responses. You can void successful transactions immediately to prevent live test transactions from being processed. This can be done quickly on the Unsettled Transactions page of the Merchant Interface. It is not possible to test the AVS or CCV responses in the developer test environment. For more information

about AVS, see the *Merchant Integration Guide* at <http://www.authorize.net/support/merchant/>.

For more information about response reason codes, see "Transaction Response," page 37.

Using the .NET SDK

The following example .NET integration assumes that your web server is on the public internet and can be accessed by means of a domain name or IP address.

**Note**

This SDK supports .NET 3.5, not .NET 4.0.

- Step 1** Download the Authorize.Net C# SDK from <http://developer.authorize.net> and unzip it on your hard drive.
- Step 2** Create a new ASP.NET MVC application and add a reference to the `AuthorizeNET.dll` from the SDK.
- Step 3** Add the `AuthorizeNET.Helpers` namespace to `web.config`.

Under `system.web/pages`:

```
<namespaces>
  <add namespace="System.Web.Mvc"/>
  ...
  <add namespace="AuthorizeNet.Helpers"/>
</namespaces>
```

- Step 4** Create a checkout form that posts directly to Authorize.Net.

You can add the following code to the `Home/Index` View to create a form that submits a test transaction to Authorize.Net (with hardcoded values). This code uses the SDK's `CheckoutFormBuilder` to create the form for you; just include `YOUR_SERVER`, `YOUR_API_LOGIN` and `YOUR_TRANSACTION_KEY`:

```
<h1><%=ViewData["message"] %></h1>
  <%using (Html.BeginSIMForm("http://YOUR_SERVER.com/home/sim",
1.99M, "YOUR_API_LOGIN", "YOUR_TRANSACTION_KEY", true)) {%>
  <%=Html.CheckoutFormInputs(true)%>
  <%=Html.Hidden("order_id", "1234") %>
  <input type = "submit" value = "Pay" />
  <%}%>
```

In this example code, the `CheckoutFormBuilder` creates the form that will post to Authorize.Net, setting the `TestMode` to `true`. It also passes an `order id` (not required by the API, but enables order identification on receipt of the response).

Step 5 Create an action to handle the response from Authorize.Net.

This example uses ASP.NET MVC; if using ASP.NET WebForms, you can put this code on `PageLoad()`.

The above code passed in the URL `http://YOUR_SERVER.com/home/sim`, so an Action called `Sim` must be created in the application's HomeController. The form POST from Authorize.net will include all of the transaction information, including an MD5 hash for validation of the post's origin.

By appending your `MERCHANT_HASH_CODE` and `YOUR_API_LOGIN` to the following code and pasting it into the HomeController, you can create an example of an action to handle the response:

```
[AcceptVerbs (HttpVerbs.Post)]
public ActionResult Sim(FormCollection post) {
    var response = new AuthorizeNET.SIMResponse(post);
    //first order of business - validate that it was Auth.net that posted
    this using
    //the MD5 hash that was passed back to us
    var isValid = response.Validate("YOUR_MERCHANT_HASH_CODE", "YOUR_API_
LOGIN");
    //if it's not valid - just send them to the home page. Don't throw -
that's how
    //hackers figure out what's wrong :)
    if (!isValid)
        return Redirect("/");
    //the URL to redirect to- this MUST be absolute
    var returnUrl = "http://YOUR_SERVER.com/?m="+response.Message;
    return
Content (AuthorizeNET.Helpers.CheckoutFormBuilders.Redirecter (returnUrl));
}
```

The example code uses SDK classes to parse the response and validate its origin using the MD5 hash. (You can find your `MERCHANT_HASH_CODE` in your merchant profile on Authorize.Net; if you created your developer account on the new developer center, your `MERCHANT_HASH_CODE` is initially blank.)

Instead of returning content for display in the client browser, DPM works by returning datagenerated by the merchant's relay URL to the merchant server, redirecting the client browser to the merchant's server. This uses JavaScript, if available on the client browser, and a `meta refresh` tag if it is not. In the last line of the above code, the SDK generates this JavaScript. This keeps the URL in the client browser pointed to the merchant's server.

The `redirecter` call in the example code calls SDK helper code that creates the JavaScript redirect. A real implementation will contain additional parameters specifying what to display to the user on the final page. (At a minimum, by passing the

TRANSACTION_ID and the *TRANSACTION_CODE*, you can send a message to the user with information about the order and whether the transaction went through.)

For simplicity, the example code redirects to the home page of the application and displays the result. The whole process occurs out of session; that is, the Authorize.Net server notifies the merchant's server of the transaction without engaging the current user's session.

Using the Java SDK

The Java integration example that follows describes the creation of three files:

- checkout_form.jsp
- relay_response.jsp
- order_receipt.jsp

These files perform the sequence of functions described in "[Conceptual Overview](#)," [page 8](#).

**Note**

The following steps assume that your web server is on the public internet and can be accessed through a domain name or IP address.

The following steps create the three files.

Step 1 Obtain an API Login ID and Transaction Key.

These keys authenticate requests to the payment gateway. To obtain them, sign up for a test account.

Step 2 Download the Java SDK from the Developer Center:

developer.authorize.net/downloads.

Place `authnet-sdk-java.jar` (found in `target`) and the `jar` files found in `/lib` into your *classpath*.

Step 3 Install the Authorize.Net Java SDK.

This gives you access to the full suite of APIs.

Step 4 Create `checkout_form.jsp`.

The following provides an example of fully functional `checkout_form.jsp` code. If you use this code as a template, be sure to populate the three variables `API_LOGIN_ID`, `TRANSACTION_KEY`, and `MERCHANT_HOST`:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
```

```

<html lang='en'>
<head>
<title>testing</title>
</head>
<body>

<%@ page import="net.authorize.sim.*"%>
<%
String apiLoginId = "API_LOGIN_ID";
String transactionKey = "TRANSACTION_KEY";
String relayResponseUrl = "http://MERCHANT_HOST/relay_response.jsp";
String amount = "1.99";
Fingerprint fingerprint = Fingerprint.createFingerprint(apiLoginId,
transactionKey,
1234567890, // random sequence used for the fingerprint amount);
long x_fp_sequence = fingerprint.getSequence();
long x_fp_timestamp = fingerprint.getTimeStamp();
String x_fp_hash = fingerprint.getFingerprintHash();
%>

<form id='secure_redirect_form_id' action='https://test.authorize.net/
gateway/transact.dll' method='POST'>
<label for='x_card_num'>Credit Card Number</label>
<input type='text' class='text' id='x_card_num' name='x_card_num'
size='20' maxlength='16' />
<br />
<label for='x_exp_date'>Expiration Date</label>
<input type='text' class='text' id='x_exp_date' name='x_exp_date' size='6'
maxlength='6' />
<br />
<label for='x_amount'>Amount</label>
<input type='text' class='text' id='x_amount' name='x_amount' size='10'
maxlength='10' readonly='readonly' value='<%=amount%>' />
<br />
<input type='hidden' name='x_invoice_num'
value='<%=System.currentTimeMillis()%>' />
<input type='hidden' name='x_relay_url' value='<%=relayResponseUrl%>' />
<input type='hidden' name='x_login' value='<%=apiLoginId%>' />
<input type='hidden' name='x_fp_sequence' value='<%=x_fp_sequence%>' />
<input type='hidden' name='x_fp_timestamp' value='<%=x_fp_timestamp%>' />
<input type='hidden' name='x_fp_hash' value='<%=x_fp_hash%>' />
<input type='hidden' name='x_version' value='3.1' />
<input type='hidden' name='x_method' value='CC' />
<input type='hidden' name='x_type' value='AUTH_CAPTURE' />
<input type='hidden' name='x_amount' value='<%=amount%>' />
<input type='hidden' name='x_test_request' value='FALSE' />
<input type='hidden' name='notes' value='extra hot please' />
<input type='submit' name='buy_button' value='BUY' />
</form>

</body>

```

```
</html>
```

**Note**

To place the `jsp` in a separate webapp container of your choice, modify `relayResponseUrl` accordingly.

Step 5 Create `relay_response.jsp`.

The following provides an example of fully functional `relay_response.jsp` code. If you use this code as a template, be sure to populate the two variables (indicated in **BOLD CAPS**).

**Note**

Unless you have explicitly set MD5-Hash in the merchant interface (using **Account > Settings > Security Settings > MD5-Hash**), leave this as an empty string.

```
<%@ page import="java.util.Map"%>
<%@ page import="net.authorize.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
var referrer = document.referrer;
if (referrer.substr(0,7)=="http://") referrer = referrer.substr(7);
if (referrer.substr(0,8)=="https://") referrer = referrer.substr(8);
if(referrer && referrer.indexOf(document.location.hostname) != 0) {
<%
  String apiLoginId = "API_LOGIN_ID";
  String receiptPageUrl = "http://MERCHANT_HOST/order_receipt.jsp";
  /*
   * Leave the MD5HashKey as is - empty string, unless you have explicitly
   * set it in the merchant interface:
   * Account > Settings > Security Settings > MD5-Hash
   */
  String MD5HashKey = "";
  net.authorize.sim.Result result =
    net.authorize.sim.Result.createResult(apiLoginId, MD5HashKey,
      request.getParameterMap());
  // perform Java server side processing...
  // ...
  // build receipt url buffer
  StringBuffer receiptUrlBuffer = new StringBuffer(receiptPageUrl);
  if(result != null) {
    receiptUrlBuffer.append("?");
    receiptUrlBuffer.append(
      ResponseField.RESPONSE_CODE.getFieldName()).append("=");
```

```

receiptUrlBuffer.append(result.getResponseCode().getCode());
receiptUrlBuffer.append("&");
receiptUrlBuffer.append(
    ResponseField.RESPONSE_REASON_CODE.getFieldName()).append("=");
receiptUrlBuffer.append(
    result.getReasonResponseCode().getResponseReasonCode());
receiptUrlBuffer.append("&");
receiptUrlBuffer.append(
    ResponseField.RESPONSE_REASON_TEXT.getFieldName()).append("=");
receiptUrlBuffer.append(
    result.getResponseMap().get(
        ResponseField.RESPONSE_REASON_TEXT.getFieldName()));

if(result.isApproved()) {
    receiptUrlBuffer.append("&").append(
        ResponseField.TRANSACTION_ID.getFieldName()).append("=");
    receiptUrlBuffer.append(result.getResponseMap().get(
        ResponseField.TRANSACTION_ID.getFieldName()));
}
}
%>
// Use Javascript to redirect the page to the receipt redirect url.
// If Javascript is not available, then the <meta> refresh tag
// will handle the redirect.
document.location = "<%=receiptUrlBuffer.toString()%>";
}
/-->
</script>
<noscript><meta http-equiv="refresh"
content="0;url=<%=receiptUrlBuffer.toString()%>"></noscript>
</body>
</html>

```

Step 6 Create `order_receipt.jsp`.

This file contains receipt information that will display to the customer. It should be accessible at the *receipt_page_url* location specified in the previous step. The following provides an example of fully functional `order_receipt.jsp` code:

```

<%@ page import="java.util.Map" %>
<%@ page import="net.authorize.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
</head>
<body>
<h1>Your Receipt Page</h1></br>
<%
// Show the confirmation data
Map<String, String[]> requestParameterMap = request.getParameterMap();
if(requestParameterMap != null && requestParameterMap.containsKey(
    ResponseField.RESPONSE_CODE.getFieldName())) {
    String transactionId = "";

```

```

if(requestParameterMap.containsKey(
    ResponseField.TRANSACTION_ID.getFieldName())) {

    transactionId = requestParameterMap.get(
        ResponseField.TRANSACTION_ID.getFieldName())[0];
}
// 1 means we have a successful transaction
if("1".equals(requestParameterMap.get(
    ResponseField.RESPONSE_CODE.getFieldName())[0])) {
%>
<h2>Success!</h2>
<h3>Your transaction ID:</h3>
<div><%=net.authorize.util.StringUtils.sanitizeString(transactionId)%>
</div>
<%
} else {
%>
<h2>Error!</h2>
<h3><%=net.authorize.util.StringUtils.sanitizeString(
requestParameterMap.get(
    ResponseField.RESPONSE_REASON_TEXT.getFieldName())[0])%>
</h3>
<table>
<tr>
<td>response code</td>
<td><%=net.authorize.util.StringUtils.sanitizeString(
requestParameterMap.get(ResponseField.RESPONSE_CODE.getFieldName())[0])%>
</td>
</tr>
<tr>
<td>response reason code</td>
<td><%=net.authorize.util.StringUtils.sanitizeString(
requestParameterMap.get(ResponseField.RESPONSE_REASON_
CODE.getFieldName())[0])%>
</td>
</tr>
</table>
</div>
<%
}
}
%>
</body>
</html>

```

Step 7 Check your work.

checkout_form.jsp, relay_response.jsp, and receipt_page.jsp must be publicly accessible on your web server and all paths contained in such variables must match actual file locations.


Step 8 Verify your Direct Post Method implementation.

From your test environment `http://MERCHANT_HOST/checkout_form.jsp`, enter credit card `4111111111111111`, any future expiration date (in the form MMDD; for example, 1120), then **Submit**.



Authorize.Net requires port 80 for communication on the `relay_response` endpoint.

You should receive a success (receipt) or error message. You can also verify payment transaction success on your transaction report:


Feedback | Contact Us | Help Log Out

Home
Tools
Reports
Search
Account

[Search](#) > Unsettled Transactions

Unsettled Transactions [Help](#)

Select an action in the **Filter by** drop-down box to capture or void multiple transactions. Select the Trans ID from the list of transactions to view details, capture, or void a specific transaction.

Filter by: ALL View

List of Unsettled Transactions
Click on a transaction ID below to view transaction details. Click on any column heading to sort.

Note: If your account is in TEST MODE, transactions submitted for capture or void will NOT actually be processed. Before submitting live transactions, verify that TEST MODE is turned off.

Filter by: ALL View

1-20 of 1611 results | [View All](#) | [Next](#)

Trans ID	Invoice Number	Trans Status	Submit Date	Customer	Card	Payment Method	Payment Amount
2155126036		Authorized/Pending Capture	15-Oct-2010 16:35:48		D	XXXX0012	USD 401.00
2155123574		Authorized/Pending Capture	15-Oct-2010 15:05:43		D	XXXX0012	USD 662.00
2155118652	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:17	Tests, Unit	V	XXXX8888	USD 1.11
2155118651	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:16	Tests, Unit	V	XXXX0027	USD 1.11



If you receive an error, verify the syntax and accuracy of the variables you entered into the files during Steps 3-5, and verify that all paths contained in such variables match actual file locations.

Using the PHP SDK

The example PHP that follows describes the creations of three files:

- `checkout_form.php`
- `relay_response.php`
- `order_receipt.php`

These files perform the sequence of functions described in "[Conceptual Overview](#)," [page 8](#).

**Note**

The following steps assume that your web server is on the public internet and can be accessed by means of a domain name or IP address.

Follow these steps to create the three files.

Step 1 Obtain an API Login ID and Transaction Key.

These keys authenticate requests to the payment gateway. To obtain them, sign up for a test account.

Step 2 Download the Authorize.Net PHP SDK and include it in your project.

This gives you access to the full suite of APIs.

Step 3 Download the PHP SDK from the Developer Center.

`developer.authorize.net/downloads`

Step 4 Save the SDK to a folder that your web server can access (for example, `/var/www` or `/htdocs`).

Step 5 Include `anet_php_sdk/AuthorizeNet.php` in your project.

Step 6 Create `checkout_form.php`.

The following provides an example of fully functional `checkout_form.php` code. If you use this code as a template, be sure to populate the three variables (indicated in **BOLD CAPS**):

```
<?php require_once 'anet_php_sdk/AuthorizeNet.php'; // The SDK
$relay_response_url = "http://YOUR_DOMAIN.com/relay_response.php"; // You
will create this file in Step 7.
$api_login_id = 'YOUR_API_LOGIN_ID';
$transaction_key = 'YOUR_TRANSACTION_KEY';
$amount = "5.99";
$fp_sequence = "123"; // Any sequential number like an invoice number.
echo AuthorizeNetDPM::getCreditCardForm($amount, $fp_sequence, $relay_
response_url,$api_login_id, $transaction_key);?>
```

**Note**

To place the php in a separate webapp container of your choosing, modify *relayResponseUrl* accordingly.

Step 7 Create `relay_response.php`.

The following provides an example of fully functional `relay_response.php` code. If you use this code as a template, be sure to populate the two variables (indicated in **BOLD CAPS**):

**Note**

Unless you have explicitly set MD5-Hash in the merchant interface (using **Account > Settings > Security Settings > MD5-Hash**), leave this as an empty string.

```
<?php require_once 'anet_php_sdk/AuthorizeNet.php'; // The SDK
$redirect_url = "http://YOUR_DOMAIN.com/receipt_page.php"; // Where the
user
will end up.
$api_login_id = 'YOUR_API_LOGIN_ID';
$md5_setting = ""; // Your MD5 Setting
$response = new AuthorizeNetSIM($api_login_id, $md5_setting);
if ($response->isAuthorizeNet())
{
if ($response->approved)
{
// Do your processing here.
$redirect_url .= '?response_code=1&transaction_id=' .
$response->transaction_id;
}
else
{
$redirect_url .= '?response_code=' . $response->response_code .
'&response_reason_text=' . $response->response_reason_text;
}
// Send the Javascript back to AuthorizeNet, which will redirect user
back to
your site.
echo AuthorizeNetDPM::getRelayResponseSnippet($redirect_url);
}
else
```

```
{
echo "Error. Check your MD5 Setting.";
}??>
```

Step 8 Create `order_receipt.php`.

This file contains receipt information that will display to the customer. It should be accessible at the `redirect_url` location specified in the previous step.

The following provides an example of fully functional `order_receipt.php` code:

```
<?php
if ($_GET['response_code'] == 1)
{
echo "Thank you for your purchase! Transaction id: "
.htmlentities($_GET['transaction_id']);}
else
{
echo "Sorry, an error occurred: " . htmlentities($_GET['response_reason_
text']);}
}??>
```

Step 9 Check your work.

`checkout_form.php`, `relay_response.php`, and `receipt_page.php` must be publicly accessible on your web server and all paths contained in such variables must match actual file locations.

Step 10 Verify your Direct Post Method implementation.


From your test environment at `http://MERCHANT_HOST/checkout_form.php`, enter credit card `4111111111111111`, any future expiration date (in the form `MMDD`), then click **Submit**.



Note

Authorize.Net requires port 80 for communication on the `relay_response` endpoint.

You should receive a success (receipt) or error message. You can also verify payment transaction success on your transaction report:


Feedback | Contact Us | Help | Log Out

Home
Tools
Reports
Search
Account

[Search](#) > Unsettled Transactions

Unsettled Transactions [Help](#)

Select an action in the **Filter by** drop-down box to capture or void multiple transactions. Select the Trans ID from the list of transactions to view details, capture, or void a specific transaction.

Filter by:

List of Unsettled Transactions
Click on a transaction ID below to view transaction details. Click on any column heading to sort.

Note: If your account is in TEST MODE, transactions submitted for capture or void will NOT actually be processed. Before submitting live transactions, verify that TEST MODE is turned off.

Filter by:

1-20 of 1611 results | [View All](#) | [Next](#)

Trans ID	Invoice Number	Trans Status	Submit Date ▼	Customer	Card	Payment Method	Payment Amount
2155126036		Authorized/Pending Capture	15-Oct-2010 16:35:48		D	XXXX0012	USD 401.00
2155123574		Authorized/Pending Capture	15-Oct-2010 15:05:43		D	XXXX0012	USD 662.00
2155118652	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:17	Tests, Unit	V	XXXX8888	USD 1.11
2155118651	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:16	Tests, Unit	V	XXXX0027	USD 1.11



Note

If you receive an error, verify the syntax and accuracy of the variables you entered into the files during Steps 3-5, and verify that all paths contained in such variables match actual file locations.

Using the Ruby SDK

The example of Ruby integration that follows describes the creation of three files:

- payment.erb
- payments_controller.rb
- receipt.erb

These files perform the sequence of functions described in "[Conceptual Overview](#)," [page 8](#).

**Note**

The following steps assume that your web server is on the public internet and can be accessed through a domain name or IP address.

Follow these steps create the three files.

Step 1 Obtain an API Login ID and Transaction Key.

These keys authenticate requests to the payment gateway. To obtain them, sign up for a test account.

Step 2 Install the Authorize.Net SDK.

Enter the following from the command line; substitute the appropriate version number for the x's:

```
sudo gem install authorize-net-1.x.x.gem
```

Step 3 For Ruby on Rails version 2.x, run the script from the command line. For Ruby on Rails version 3, skip to Step 4.

From the command line, run the following, replacing **BOLD CAPS** with their corresponding values from the local merchant environment:

```
sudo gem install rails -v '~> 2.1'
rails my_direct_post_app
cd my_direct_post_app
script/generate authorize_net_direct_post payments \
    YOUR_API_LOGIN_ID YOUR_TRANSACTION_KEY YOUR_API_LOGIN_ID
script/server
```

This process generates examples of the three required files populated with local merchant values. If you use this code as a template, be sure to populate the variables (indicated in **BOLD CAPS**).

- Step 4** For Ruby on Rails version 2.x, skip to Step 5. For Ruby on Rails version 3, follow these steps:

From the command line, enter:

```
sudo gem install rails
rails new my_direct_post_app
cd my_direct_post_app
```

Open the file *Gemfile* in the root directory of your new Ruby app and add the following line to the end of the file:

```
gem 'authorize-net'
```

Back at the command line, enter:

```
rails generate authorize_net:direct_post payments YOUR_API_LOGIN_ID YOUR_TRANSACTION_KEY YOUR_API_LOGIN_ID
rails server
```

- Step 5** Verify your Direct Post Method implementation.


From your test environment `http://MERCHANT_HOST/payments/payment`, enter credit card `4111111111111111`, any future expiration date (in the form `MMYY`), then click Submit.



Note

Authorize.Net requires port 80 for communication on the `relay_response` endpoint.

You should receive a success (receipt) or error message. You can also verify payment transaction success on your transaction report:


Feedback | Contact Us | Help **Log Out**

Home
Tools
Reports
Search
Account

Search > Unsettled Transactions

Unsettled Transactions [Help](#)

Select an action in the **Filter by** drop-down box to capture or void multiple transactions. Select the Trans ID from the list of transactions to view details, capture, or void a specific transaction.

Filter by:

List of Unsettled Transactions
Click on a transaction ID below to view transaction details. Click on any column heading to sort.

Note: If your account is in TEST MODE, transactions submitted for capture or void will NOT actually be processed. Before submitting live transactions, verify that TEST MODE is turned off.

Filter by:

1-20 of 1611 results | [View All](#) | [Next](#)

Trans ID	Invoice Number	Trans Status	Submit Date ▼	Customer	Card	Payment Method	Payment Amount
2155126036		Authorized/Pending Capture	15-Oct-2010 16:35:48		D	XXXX0012	USD 401.00
2155123574		Authorized/Pending Capture	15-Oct-2010 15:05:43		D	XXXX0012	USD 662.00
2155118652	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:17	Tests, Unit	V	XXXX8888	USD 1.11
2155118651	123124123432	Authorized/Pending Capture	15-Oct-2010 11:39:16	Tests, Unit	V	XXXX0027	USD 1.11



Note

If you receive an error, verify the syntax and accuracy of the variables you entered into the files during Step 3, and verify that all paths contained in such variables match actual file locations.

Fields by Transaction Type

This appendix provides a complete listing of all API fields that should be submitted for each transaction type supported for DPM. It is divided into the following sections:

- The minimum fields required to submit a transaction.
- Additional fields that are required in order to configure advanced features of DPM.
- “Best practice” fields, or fields that the payment gateway recommends be submitted on a per-transaction basis in order to maintain a strong connection to the payment gateway—for example, to prevent possible conflicts in the event that integration settings in the Merchant Interface are inadvertently changed.

Minimum Required Fields

The following table provides a quick reference of all API fields that are required for each transaction type supported for DPM.

Table 12 Minimum Required Fields

Type of Field	Authorization and Capture	Authorization Only	Prior Authorization and Capture*	Credit *	Void*
Merchant Information	x_login	x_login	N/A	N/A	N/A
Fingerprint Information	x_fp_hash x_fp_sequence x_fp_timestamp	x_fp_hash x_fp_sequence x_fp_timestamp	N/A	N/A	N/A
Transaction Information	x_type = AUTH_CAPTURE	x_type = AUTH_ONLY	N/A	N/A	N/A
Payment Information	x_amount	x_amount	N/A	N/A	N/A
Payment Form Configuration	x_show_form = PAYMENT_FORM	x_show_form = PAYMENT_FORM	N/A	N/A	N/A

* For Prior Authorization and Capture, Credit, and Void transactions, it is recommended that the merchant process the transactions by logging on to the Merchant Interface directly or by using a desktop application that uses AIM.

Required Fields for Additional DPM Features

The following table provides a quick reference of additional fields that are required for advanced features of DPM and that *cannot* be configured in the Merchant Interface. For example, if the merchant wants to submit itemized order information, you must submit fields in addition to the minimum required fields.

Table 13 Required Fields for Additional Features

Type of Field	Authorization and Capture	Authorization Only	Prior Authorization and Capture*	Credit *	Void*
Itemized Order Information	x_line_item	x_line_item	N/A	N/A	N/A
Relay Response Configuration	x_relay_response = TRUE x_relay_url	x_relay_response = TRUE x_relay_url	N/A	N/A	N/A
Fraud Detection Suite™ (FDS)	x_customer_ip (required only when the merchant is using the FDS IP blocking tool)	x_customer_ip (required only when the merchant is using the FDS IP blocking tool)	N/A	N/A	N/A

* For Prior Authorization and Capture, Credit, and Void transactions, it is recommended that the merchant process the transactions by logging on to the merchant interface directly, or by using a desktop application that uses AIM.

Best Practice Fields

The following table provides a quick reference of additional API fields that the payment gateway highly recommends should be submitted on a per-transaction basis in order to maintain a strong connection.

Table 14 Best Practice Fields

Type of Field	Authorization and Capture	Authorization Only	Prior Authorization and Capture*	Credit *	Void*
Transaction Information	x_version = 3.1	x_version = 3.1	N/A	N/A	N/A
Payment Form Configuration	x_header_html_payment_form x_footer_html_payment_form	x_header_html_payment_form x_footer_html_payment_form	N/A	N/A	N/A

Table 14 Best Practice Fields (Continued)

Type of Field	Authorization and Capture	Authorization Only	Prior Authorization and Capture*	Credit *	Void*
Receipt Page Configuration	x_receipt_link_method x_header_html_receipt x_footer_html_receipt	x_receipt_link_method x_header_html_receipt x_footer_html_receipt	N/A	N/A	N/A

* For Prior Authorization and Capture, Credit, and Void transactions, it is recommended that the merchant process the transactions by logging on to the merchant interface directly, or by using a desktop application that uses AIM.

API Fields

Table 15 API Fields, in Alphabetical Order

Field Name	Required	Value	Format	Notes
x_address	Optional	The customer's billing address	Up to 60 characters (no symbols)	<p>Required if the merchant would like to use the Address Verification Service security feature.</p> <p>For more information on AVS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/.</p> <p>Required for Zero Dollar Authorizations for Visa verification transactions. If you have set x_recurring_billing to TRUE, AVS is automatically turned off.</p>
x_allow_partial_Auth	optional	True, False	True, False, T, F	Set this value if the merchant would like to override a setting in the Merchant Interface.
x_amount	Required if x_type = AUTH_CAPTURE, AUTH_ONLY, CREDIT	The amount of the transaction	Up to 15 digits with a decimal point (no dollar symbol) For example, 8.95	The total amount to be charged or credited <i>including</i> tax, shipping and any other charges. The amount can either be hard coded or posted to a script.
x_auth_code			Required if x_type = CAPTURE_ONLY	The authorization code for an original transaction not authorized on the payment gateway

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_authentication_indicator			Optional	The electronic commerce indicator (ECI) value for a Visa transaction; or the universal cardholder authentication field indicator (UCAFI) for a MasterCard transaction obtained by the merchant after the authentication process.
x_card_num	Required only if x_type = CREDIT	The customer's partial credit card number	The last four digits of the credit card number only	As a hosted solution, DPM does not support the submission of full cardholder data. Ideally, CREDIT transactions should be submitted exclusively in the Merchant Interface. This field should not be passed for a DPM transaction under any other circumstance.
x_city	Optional	The city of the customer's billing address	Up to 40 characters (no symbols)	
x_company	Optional	The company associated with the customer's billing address	Up to 50 characters (no symbols)	
x_country	Optional	The country of the customer's billing address	Up to 60 characters (no symbols)	
x_cust_id	Optional	The merchant assigned customer ID	Up to 20 characters (no symbols)	The unique identifier to represent the customer associated with the transaction. The customer ID must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_customer_ip	Optional	The customer's IP address	Up to 15 characters (no letters) For example, 255.255.255.255	The IP address of the customer initiating the transaction. If this value is not passed, it will default to 255.255.255.255. This field is required when using the Fraud Detection Suite™ (FDS) IP Address Blocking tool. For more information about FDS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .
x_description	Optional	The transaction description	Up to 255 (no symbols)	The description must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.
x_delim_data	Required for DPM transactions	Set to False to implement a relay response	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	In order to implement a relay response, this field must be submitted with a value of FALSE or the merchant has to configure a relay response through the Merchant Interface. It is recommended that you submit this field on a per-transaction basis to be sure that transaction responses are returned in the correct format. This field is paired with x_relay_response. If one is set to True, the other must be set to False.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_duplicate_window	Optional	The window of time after the submission of a transaction that a duplicate transaction can not be submitted	Any value between 0 and 28800 (inclusive, no commas)	<p>Indicates in seconds the window of time after a transaction is submitted during which the payment gateway will check for a duplicate transaction. The maximum time allowed is 8 hours (28800 seconds).</p> <p>If a value less than 0 is sent, the payment gateway will default to 0 seconds. If a value greater than 28800 is sent, the payment gateway will default to 28800. If no value is sent, the payment gateway will default to 2 minutes (120 seconds).</p> <p>If this field is present in the request with or without a value, an enhanced duplicate transaction response is sent. See "Response for Duplicate Transactions," page 43 for more information.</p>
x_duty	Optional	The valid duty amount OR delimited duty information	When submitting delimited duty information, values must be delimited by a bracketed pipe < >	<p>The duty amount charged OR when submitting this information by means of the HTML Form POST, delimited duty information including the duty name, description, and amount is also allowed.</p> <p>The total amount of the transaction in x_amount must <i>include</i> this amount.</p>
		duty item name< >		The duty item name.
		duty description< >		The duty item description.
		duty amount	The dollar sign (\$) is not allowed when submitting delimited information.	<p>The duty amount.</p> <p>The total amount of the transaction in x_amount must <i>include</i> this amount.</p>
<p>Example: <INPUT TYPE="HIDDEN" name="x_duty" VALUE="Duty1< >export< >15.00"></p>				

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_email	Optional	The customer's valid email address	Up to 255 characters For example, janedoe@customer.com	The email address to which the customer's copy of the email receipt is sent when Email Receipts is configured in the Merchant Interface. The email is sent to the customer only if the email address format is valid. For more information about Email Receipts, please see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .
x_email_customer	Optional	The customer email receipt status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates whether an email receipt should be sent to the customer. If set to TRUE, the payment gateway will send an email to the customer after the transaction is processed using the customer email address submitted with the transaction. If FALSE, no email is sent to the customer. If no value is submitted, the payment gateway will look up the configuration in the Merchant Interface and send an email only if the merchant has enabled the setting. If this field is not submitted and the setting is disabled in the Merchant Interface, no email is sent. For more information about configuring Email Receipts in the Merchant Interface, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ .
x_fax	Optional	The fax number associated with the customer's billing address	Up to 25 digits (no letters) For example, (123)123-1234.	
x_first_name	Optional	The first name associated with the customer's billing address	Up to 50 characters (no symbols)	

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_footer_email_receipt	Optional	The email receipt footer	Plain text	This text will appear as the footer on the email receipt sent to the customer.
x_fp_hash	Required	The transaction unique fingerprint	N/A	<p>The fingerprint is generated using the HMAC-MD5 hashing algorithm on the following field values:</p> <p>API Login ID (x_login)</p> <p>The sequence number of the transaction (x_fp_sequence)</p> <p>The timestamp of the sequence number creation (x_fp_timestamp)</p> <p>Amount (x_amount)</p> <p>Field values are concatenated and separated by the “^” character.</p>
x_fp_sequence	Required	The merchant assigned sequence number for the transaction	Numeric	The sequence number can be a merchant assigned value, such as an invoice number or any randomly generated number.
x_fp_timestamp	Required	The timestamp at the time of fingerprint generation	UTC time in seconds since January 1, 1970	Coordinated Universal Time (UTC) is an international atomic standard of time (sometimes referred to as GMT). Using a local time zone timestamp will cause fingerprint authentication to fail.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_freight	Optional	The valid freight amount OR delimited freight information	When submitting delimited freight information, values must be delimited by a bracketed pipe < >	The freight amount charged OR when submitting this information by means of the HTML Form POST, delimited freight information including the freight name, description, and amount is also allowed. The total amount of the transaction in x_amount must <i>include</i> this amount.
		freight item name< >		The freight item name.
		freight description< >		The freight item description.
		freight amount	The dollar sign (\$) is not allowed when submitting delimited information.	The freight item amount. The total amount of the transaction in x_amount must <i>include</i> this amount.
<p>Example: <INPUT TYPE="HIDDEN" name="x_freight" VALUE="Freight1< >ground overnight< >12.95"></p>				
x_header_email_receipt	Optional	The email receipt header	Plain text	This text will appear as the header of the email receipt sent to the customer.
x_invoice_num	Optional	The merchant assigned invoice number for the transaction	Up to 20 characters (no symbols)	The invoice number must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function. Also, in order to be included on the hosted payment form, the attribute View must be configured for this field in the Merchant Interface payment form settings.
x_last_name	Optional	The last name associated with the customer's billing address	Up to 50 characters (no symbols)	

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_line_item	Optional All line item values are required when this field is submitted	Any string	Line item values must be delimited by a bracketed pipe < >	Itemized order information.
		Item ID< >	Up to 31 characters	The ID assigned to an item.
		< >item name< >	Up to 31 characters	A short description of an item.
		< >item description< >	Up to 255 characters	A detailed description of an item.
		< >itemX quantity< >	Up to two decimal places Must be a positive number	The quantity of an item.
		< >item price (unit cost)< >	Up to two decimal places Must be a positive number	Cost of an item per unit, <i>excluding</i> tax, freight, and duty. The dollar sign (\$) is not allowed when submitting delimited information.
		< >itemX taxable	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates whether the item is subject to tax.
x_login	Required	The merchant's unique API Login ID	Up to 20 characters	The merchant API Login ID is provided in the Merchant Interface and must be stored securely. The API Login ID and transaction fingerprint together provide the merchant authentication required for access to the payment gateway. See the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/ for more information.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_method	Optional	The payment method	CC or ECHECK	<p>The method of payment for the transaction, CC (credit card) or ECHECK (electronic check). If left blank, this value defaults to CC.</p> <p>For more information about eCheck.Net transaction requirements, see the <i>eCheck.Net Developer Guide</i> at http://developer.authorize.net/guides/echeck.pdf.</p>
x_phone	Optional	The phone number associated with the customer's billing address	<p>Up to 25 digits (no letters)</p> <p>For example, (123)123-1234</p>	
x_po_num	Optional	The merchant assigned purchase order number	Up to 25 characters (no symbols)	<p>The purchase order number must be created dynamically on the merchant server or provided on a per-transaction basis. The payment gateway does not perform this function.</p> <p>Also, in order to be displayed, the attribute View must be configured for this field in the Merchant Interface payment form settings.</p>
x_recurring_billing	Optional	The recurring billing status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	<p>Indicating marker used by merchant account providers to identify transactions which originate from merchant hosted recurring billing applications. This value is not affiliated with Automated Recurring Billing.</p>
x_relay_response	Optional	The request for a relay response	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	<p>This field instructs the payment gateway to return transaction results to the merchant by means of an HTML form POST to the merchant's Web server for a relay response.</p> <p>This field is paired with x_delim_data. If one is set to True, the other must be set to False.</p>

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_relay_URL	Optional	The URL on the merchant's Web site to which the payment gateway should post transaction results for a relay response	Any valid URL Including name/value pairs in the URL (anything after a "?") is not recommended	In the event that this field is submitted, the payment gateway will validate the URL value against the Relay Response URL configured in the Merchant Interface. If the URL submitted does not match the URL configured in the Merchant Interface, the transaction will be rejected. If no value is submitted in the HTML Form POST, the payment gateway will post the transaction results to the URL configured in the Merchant Interface.
x_rename	Optional	A request to rename a field	Field name on the payment form, new field name	Use this variable to replace a field name on a payment form. This does not rename the original field, it only changes the value displayed on the payment form.
x_ship_to_address	Optional	The customer's shipping address	Up to 60 characters (no symbols)	
x_ship_to_company	Optional	The company associated with the customer's shipping address	Up to 50 characters (no symbols)	
x_ship_to_country	Optional	The country of the customer's shipping address	Up to 60 characters (no symbols)	
x_ship_to_city	Optional	The city of the customer's shipping address	Up to 40 characters (no symbols)	
x_ship_to_first_name	Optional	The first name associated with the customer's shipping address	Up to 50 characters (no symbols)	
x_ship_to_last_name	Optional	The last name associated with the customer's shipping address	Up to 50 characters (no symbols)	

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_ship_to_state	Optional	The state of the customer's shipping address	Up to 40 characters (no symbols) or a valid two-character state code	
x_ship_to_zip	Optional	The ZIP code of the customer's shipping address	Up to 20 characters (no symbols)	
x_state	Optional	The state of the customer's billing address	Up to 40 characters (no symbols) or a valid two-character state code	
x_tax	Optional	The valid tax amount OR the delimited tax information	When submitting delimited tax information, values must be delimited by a bracketed pipe < >	The tax amount charged OR when submitting this information by means of the HTML Form POST, delimited tax information including the sales tax name, description, and amount is also allowed. The total amount of the transaction in x_amount must <i>include</i> this amount.
		tax item name< >		The tax item name.
		tax description< >		The tax item description.
		tax amount	The dollar sign (\$) is not allowed when submitting delimited information.	The tax item amount. The total amount of the transaction in x_amount must <i>include</i> this amount.
Example: <INPUT TYPE="HIDDEN" name="x_tax" VALUE="Tax1< >state tax< >0.09">				
x_tax_exempt	Optional	The tax exempt status	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates whether the transaction is tax exempt. The total amount of the transaction in x_amount must <i>include</i> this amount.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_test_request	Optional	The request to process test transactions	TRUE, FALSE, T, F, YES, NO, Y, N, 1, 0	Indicates if the transaction should be processed as a test transaction. See "Test Transactions," page 60 for more information.
x_trans_id	Optional	The payment gateway-assigned transaction ID of an original transaction	Numeric	Required only for CREDIT and PRIOR_AUTH_CAPTURE transactions. For more information about transaction types, see "Credit Card Transaction Types," page 15 .
x_type	Optional	The type of credit card transaction	AUTH_CAPTURE (default), AUTH_ONLY	If the value submitted does not match a supported value, the transaction is rejected. If no value is submitted in this field, the payment gateway will process the transaction as an AUTH_CAPTURE. For transaction types CREDIT, PRIOR_AUTH_CAPTURE, and VOID, it is recommended that the merchant process the transactions by logging on to the merchant interface directly, or by using a desktop application that uses AIM.

Table 15 API Fields, in Alphabetical Order (Continued)

Field Name	Required	Value	Format	Notes
x_version	Optional, but highly recommended	The merchant's transaction version	3.1	<p>The transaction version represents the set of fields that is included with the transaction response. 3.0 is the default version. 3.1 allows the merchant to utilize the Card Code feature, and is the current standard version.</p> <p>It is highly recommended that you submit this field on a per-transaction basis to be sure that the formats of transaction requests and the responses you receive are consistent, particularly if you are using Relay Response.</p> <p>For more information, see "Relay Response," page 29 and Appendix A, "Fields by Transaction Type," on page 79.</p>
x_zip	Optional	The ZIP code of the customer's billing address	Up to 20 characters (no symbols)	<p>Required if the merchant would like to use the Address Verification Service security feature.</p> <p>For more information on AVS, see the <i>Merchant Integration Guide</i> at http://www.authorize.net/support/merchant/.</p> <p>Required for Zero Dollar Authorizations for Visa verification transactions.</p>