

# JetPay<sup>SM</sup> JetCom<sup>SM</sup>

## User's Guide



# Table of Contents

|   |    |
|---|----|
| <b>Introduction</b> .....                                 | 3  |
| <b>General</b> .....                                      | 3  |
| <b>Advantages of the JetPay Payment Gateway</b> .....     | 4  |
| Software Requirements .....                               | 4  |
| <b>Processing Transactions with JetCom</b> .....          | 5  |
| <b>Invoking the JetCom Engine</b> .....                   | 5  |
| <b>Invoking the JetCom Engine, Verified by Visa</b> ..... | 5  |
| <b>CJetPayCC Processing Properties</b> .....              | 6  |
| Required Transaction Properties .....                     | 6  |
| ACH-Specific Transaction Properties .....                 | 7  |
| Optional Properties .....                                 | 8  |
| Reference Properties .....                                | 9  |
| Miscellaneous Properties .....                            | 9  |
| Lodging Extension Info .....                              | 11 |
| Restaurant Mode Info Properties .....                     | 11 |
| Verified by Visa Transactions Info .....                  | 11 |
| Proxy Bypass Properties .....                             | 13 |
| Return Properties .....                                   | 13 |
| <b>CJetPayCCMTS Processing Properties</b> .....           | 16 |
| ExecuteTransaction Method Properties .....                | 16 |
| ExecuteTransactionASP Method Properties .....             | 21 |
| <b>JetPay COM Library Configuration Editor</b> .....      | 25 |
| <b>Demo Test Bench Application</b> .....                  | 29 |
| <b>The Basics Tab</b> .....                               | 30 |
| <b>The Options Tab</b> .....                              | 32 |
| <b>The Repeat Tab</b> .....                               | 33 |
| <b>The Certify Tab</b> .....                              | 35 |
| <b>Appendix A</b> .....                                   | 37 |
| <b>CJetPayCC Properties</b> .....                         | 37 |
| <b>Appendix B</b> .....                                   | 45 |
| <b>Host Response Codes and Text</b> .....                 | 45 |
| <b>Abort Codes and Text</b> .....                         | 50 |
| <b>Error Codes and Text</b> .....                         | 51 |
| <b>Appendix C</b> .....                                   | 53 |
| <b>Test Host Response Code Generation</b> .....           | 53 |
| <b>Appendix D</b> .....                                   | 54 |
| <b>Luhn Check Calculation</b> .....                       | 54 |
| <b>Appendix E</b> .....                                   | 56 |
| <b>Valid Card BIN Ranges</b> .....                        | 56 |

# Introduction

This document describes the requirements and parameters necessary to perform secure credit card or Automated Clearinghouse (ACH) transaction processing through the Transactional Technologies JetPay<sup>SM</sup> electronic payment gateway using the JetPay JetCom<sup>SM</sup> COM Component.

---

**Note** For more detailed documentation about the vxJetCom.dll core component, consult the JetPay<sup>™</sup> Credit Card Gateway COM Library Help file (**vxJetCom.chm**). To open this file, click **Start**, point to **Programs**, point to **JetPay<sup>™</sup> Credit Card Gateway COM Library**, and click **Help**.

---

## General

The JetCom product includes a number of components that enable application development for, and transaction processing with, the JetPay system. Also included in the product are the JetCom core engine (**vxJetCom.dll**) and several tools that facilitate application development and transaction processing using that component.

JetCom is a Component Object Model (COM), or ActiveX, Dynamic Link Library (.dll) that provides a programmatic interface for processing secure credit card transactions.

By creating an instance of a JetPay object as defined by classes stored in the .dll, setting its properties, and calling its process method, you can enable any Win32 application—including Web Server apps—to post a secure credit card or an ACH transaction.

A COM component does not know or care whether it is running in secure mode or not. It is really the environment in which the component operates that makes the difference.

Therefore, this component must be implemented on a server with a Secure Sockets Layer (SSL) certificate in order for it to make a secure connection with the JetPay payment gateway, and to receive secure data back to return to the Web browser. This includes any input forms or interfaces to this component, as well as the result pages passed back to the Web browser.

The setup routine will install and register the component. Please make sure that any required components or libraries are installed and registered correctly.

## Advantages of the JetPay Payment Gateway

You do not need to install any additional language interpreters (for example, Perl) using the JetPay payment gateway. In fact, you do not need any additional software. There are no language or software dependencies; you can run the process from virtually anywhere:

- a Perl script on Apache;
- an ASP page on IIS;
- or from any Windows fat-client app.

In addition, the JetPay payment gateway lets you optionally package order and customer information with the credit card transaction. This makes it possible to quickly and easily tie reported transactions to bank statement and receipt information, which makes transaction balancing quick, easy, and verifiable. In fact, without this capability, you could easily put your company in serious jeopardy.

### Software Requirements

- Microsoft Windows 2000 with Service Pack 2 (or Microsoft NT Server 4.0 with Service Pack 6a and Option Pack)
- A Web Server (such as Internet Information Server—IIS)
- vxJetCom.dll—the JetCom Component Object Model (COM)/ActiveX component .dll
- Access to a Microsoft SQL Server 7 database (if you want to enable local transaction logging)

---

#### Note

Additional requirements not mentioned above might include whatever software you need to make secure posts.

---

# Processing Transactions with JetCom

The JetCom .dll is a standard COM/ActiveX .dll with public properties and methods. Some of the properties are for information required to complete the transaction, such as credit card number; some are for your convenience in balancing and reconciling your transactions, such as order number.

## Invoking the JetCom Engine

There are two methods that you can use to invoke the JetCom engine. First, you must create an instance of one of two objects: **CJetPayCC**, or **CJetPayCCMTS**. If you create an instance of **CJetPayCC**, you must set a number of properties, invoke **ExecuteTransaction()** method without parameters to do the work, and then check the properties after the method executes. If you create an instance of **CJetPayCCMTS**, you pass all parameters to the **ExecuteTransaction** method, and then check its properties.

---

**Note** Many parameters were left as optional for backward compatibility with earlier versions of JetCom. The best practice would be to treat all parameters as required and initialize each one—including output parameters—with a variable of the appropriate type.

---

## Invoking the JetCom Engine, Verified by Visa

There are three methods that you must use to invoke the JetCom engine. First, you must create an instance of the object: **CJetPayCC**. Then using an instance of **CJetPayCC**, you must set a number of properties, invoke **ExecuteVbVLookup()** method to make a lookup of the Card, then check the properties after the method executes and feedback the parameters and invoke the next function **ExecuteVbVAuthenticate()**, then check the properties after the method executes, with this then you can invoke the normal **ExecuteTransaction()**. Then finally check the parameters after the methods execute.

---

**Note** Many parameters are optional. The best practice would be to treat all parameters as required and initialize each one—including output parameters—with a variable of the appropriate type.

---

## CJetPayCC Processing Properties

### Required Transaction Properties

The following properties are required for *all* transactions. For ACH transactions, properties that pertain only to credit cards must still be initialized with a variable of the appropriate type to be accepted by the host. This data will be disregarded for the transaction.

- **MerchantID**—Issued by Transactional Technologies upon account setup with the merchant bank.
- **Transtype**—Must be equal to one of the following values: “SALE”; “AUTHONLY”; “VOID”; “CREDIT”; “FORCE”; “CHECK”; “VOIDACH”; “CAPT”; “ENQ”; “PING” or “REVERSAL”. The default value is “SALE”
  - SALE—Authorizes and closes a credit card charge in a single transaction.
  - AUTHONLY—The credit card limit is checked to verify that a certain amount is available (and to reserve that amount), but the card is not charged. A “CAPT” is required to complete the transaction.
  - FORCE—A credit card charge using an amount equal to or less than the amount of a previous authorized transaction.
  - VOID—This removes a credit card charge from the host before it settles the transaction. If it is not voided before settlement, a “CREDIT” transaction must be used to reverse the charge.
  - CREDIT—A credit (refund) to a customer’s credit card account.
  - CHECK—An ACH check transaction.
  - VOIDACH—An ACH void check transaction.
  - CAPT—Closing of an “AUTHONLY” transaction.

- **ENQ**—Used to check the status of a transaction.
- **PING**—This verifies the connection to the Jet Pay Server.
- **REVERSAL**—This removes a check transaction before settlement. To reverse a check after settlement, a refund must be manually sent.
- **CardNum**—Credit card number to be submitted on the transaction.
- **CardExpMonth**—Two-digit card expiration month.
- **CardExpYear**—Two-digit card expiration year. (System tested for Y2K compliance.)
- **Price**—Purchase price to be transacted—decimal point and no dollar sign. Must be greater than zero.
- **TransactionID**—Required for “FORCE” and “VOID” transactions. Optionally generated for all other transactions.
- **Approval**—Required for force transactions.

## ACH-Specific Transaction Properties

The following properties are needed to complete a successful ACH (check) transaction.

- **ABANumber**—The American Bankers Association (ABA) number for an ACH transaction.
- **CheckingAccountNumber**—The DDA number for the transaction.
- **CheckNumber**—The check number to be submitted on the transaction.
- **CheckSurcharge**—The amount of the check surcharge to be submitted on the transaction. The surcharge is included in the price and must be equal to or greater than \$0.00 (zero), but less than \$100,000.00.
- **CheckingAccountType**—The type of the account to be submitted on the transaction. The value must be equal to one of the following values: “CHECKING”; “SAVINGS” or “BUSINESSCK”. The default value is “CHECKING”.

## Optional Properties

The following properties are optional, or will default if not set:

- **CardType**—Optional. If not set, this value will be determined from the credit card number. Valid values are: “MasterCard”; “MC”; “Visa”; “American Express”; “AMEX”; “Discover”; “Disc”; “Diners Club”; and “Carte Blanche”.
- **CVV2**—Optional. These are three or four additional digits on the credit card. Using these digits provides an additional measure of security for Internet transactions.
- **Track1Data**— Track 1 Raw Data.
- **Track2Data**— Track 2 Raw Data.
- **TerminalType**— Optional for backward compatibility. Must be equal to one of the following nine values: “ECOMMERCE”; “RETAIL”; “MOTO”; ”HOTEL”; ”RESTAURANT”; ”AUTORENTAL”; ”AIRLINE”; ”PARKING”; ”QUASICASH”. The default value is “ECOMMERCE”.
  - ECOMMERCE— Indicate Electronic Commerce Trade transactions.
  - RETAIL— Indicate Retail type of transactions.
  - MOTO— Indicate Airlines Trade transactions.
  - HOTEL—Indicate Hotel Trade transactions.
  - RESTAURANT—Indicate Restaurant Trade transactions.
  - AUTORENTAL— Indicate Car Rental Trade transactions.
  - AIRLINES—Indicate the Airlines Trade transactions.
  - PARKING— Indicate Parking Trade transactions
  - QUASICASH— Indicate QuasiCash Trade transactions.
- **ReferenceNumber**— Reference Number.

## Reference Properties

You can set the following properties in order to provide a reference between order information and credit card information, which is helpful for order tracking, customer service, auditing, and reconciliation. This data may also be useful in fraud prevention.

- **BillingAddress**—Cardholder address.
- **BillingCity**—Cardholder city.
- **BillingCountry**—Cardholder country. Valid values are full alphabetic country name or ISO Country Code.
- **BillingPhone**—Cardholder telephone number.
- **BillingPostalCode**—Cardholder Zip code.
- **BillingStateProv**—Cardholder state.
- **CardName**—Cardholder name.
- **Email**—Cardholder e-mail address.
- **OrderNumber**—Order number (or any other alphanumeric reference string).
- **UserIPAddr**—Cardholder IP address.
- **UserHost**—Cardholder host name.
- **UDField1**—User-defined field.
- **UDField2**—User-defined field.
- **UDField3**—User-defined field.

## Miscellaneous Properties

- **CardExp**—Read only. Four-digit rollover of **CardExpMonth** and **CardExpYear**.
- **Test**—Flag enabling the Credit Card Number edits to be bypassed for testing. Default is to run a real transaction. Use the **TransactionTest Enum** to set valid values for the Test property.
- **Timeout**—Transaction timeout in seconds. Default is 30.
- **DispositionType**—Custom host extension disposition model on the transaction-processing engine. Do not change unless instructed to do so.

- **IndustryType**— Must be equal to one of the following nine values: “ECOMMERCE”; “RETAIL”; “MOTO”; ”HOTEL”; ”RESTAURANT”; ”AUTORENTAL”; ”AIRLINE”; ”PARKING”; ”QUASICASH”. The default value is “ECOMMERCE”.
  - ECOMMERCE— Indicate Electronic Commerce Trade transactions.
  - RETAIL— Indicate Retail type of transactions.
  - MOTO— Indicate Airlines Trade transactions.
  - HOTEL—Indicate Hotel Trade transactions.
  - RESTAURANT—Indicate Restaurant Trade transactions.
  - AUTORENTAL— Indicate Car Rental Trade transactions.
  - AIRLINES—Indicate the Airlines Trade transactions.
  - PARKING— Indicate Parking Trade transactions
  - QUASICASH— Indicate QuasiCash Trade transactions.
- **Origin**— Must be equal to one of the following five values: “INTERNET”; “POS”; “RECURRING”; ”MAIL ORDER”; ”PHONE ORDER”. The default value is “INTERNET”.
  - INTERNET—Indicates the transaction was originated from Internet.
  - RECURRING—Indicates the transactions is frequent, like the Phone Cards companies, cell phones trade, etc .
  - MAIL ORDER—Indicates the transaction was originated by mail.
  - PHONE ORDER— Indicates the transaction was originated by telephone.
- **CardPresent**—Boolean value to indicate the Card Present or not.
- **TaxAmount**—Tax Amount for the transaction.
- **UseWinInet**—Indicate the Win Inet uses.
- **UseDialUp**—Indicate the use of Dial Up instead of Ethernet.
- **CommPort**—Communication Port used during the Dial Up.
- **Portsettings**—Settings for the Port during the Dial Up.
- **AuthPhone**—Authorization Phone during the Dial Up.
- **SubscriberName**—Identify the Subscriber Name.
- **SubscriberVersion**— Identify the Subscriber Version.
- **DllName**— Identify the JetCOM Dll Component Name.
- **DllVersion**— Identify the JetCOM Dll Component Version.

## Lodging Extension Info

You can use these properties to set the Lodging information for the Hotels Trade.

- **FolioNumber**—Number of Folio.
- **CheckInDate**—Lodging check in date.
- **CheckOutDate**—Lodging check out date.
- **OrigAuthAmount**—Original authorization amount made during the check in.
- **TotalAuthAmount**—Final authorization amount made during the check out.
- **RoomRate**—Indicates the Room Rate.
- **PrestigiousProp**—Prestigious Value.
- **SpecialCode**—Special Code.
- **Extra**—Extra field.

## Restaurant Mode Info Properties

You can use these properties to set the transaction information for the Restaurant Trade.

- **TicketNumber**—Number of Ticket.
- **ServerID**—Waiter/Waitress Server ID.
- **BaseAmount**—Base Amount for the Transaction.
- **TipAmount**—Tip Amount for the Transaction.
- **TableNumber**—Table Number.

## Verified by Visa Transactions Info

You can use these properties to set the Verified by Visa information for the transactions using Visa Cards.

## Common Variables

- **MAPUrl**—MAP URL address.

- **Version**—Version identifier, currently “1.1”.
- **ProcessorID**—Acquiring Institution identification code. Typically a 6 digit BIN assigned to the acquirer by Visa.

## Lookup Message

- **Password**—Acquirer defined merchant password.
- **PurchaseCurrency**—3 digit numeric, ISO 4217 currency code for the sale amount.
- **OrderDescription**—Brief description of items purchased, limited to 125.
- **UserAgent**—The exact content of the HTTP user-agent header.
- **Recurring**—Recurring Flag to specify if the Merchant and cardholder have agreed
  - to recurring payments.
- **RecurringFrequency**—Integer value indicating the minimum number of days between authorizations. A frequency of monthly is indicated by the value 28.
- **RecurringEnd**— The date after which no further recurring authorizations should be preformed. Format YYYYMMDD.
- **Installment**—An integer indicating the maximum number of permitted authorizations for installment payments. Must be included if the merchant and cardholder have agreed to installment payments.

## Response Lookup Message

- **Enrolled**— Must be one of the following values:
  - Contains: Y-if cardholder is enrolled
  - Contains: N-if cardholder is not enrolled
  - Contains: U-if cardholder is enrolled, but authentication is Y.
- **ACSUrl**— Contains the fully qualified URL of the Issuer ACS. This will be used by the merchant to redirect the cardholder. Available if Enrolled = Y.
- **PAReq**—Contains encoded PAReq generated by the MAPS Available if Enrolled = Y

## Authenticate Message

- **PARes**—Transaction status after finish the execution of the redirect to **ACSUrl**. Y/N/U (“A” future) value Y

## Response Authenticate Message

- **Eci**— E-commerce Indicator (ECI) from PARes.
- **Xid**— Transaction Xid from PARes.
- **Cavv**— Transaction Stain from PARes (28 Character, Base 64).
- **PAResStatus**— Transaction status from PARes. Y/N/U (“A” future) value Y
- **SignatureVerification**— Signature Verification status. Y/N value.
- **ErrorNo**— Contains Error No if any processing error occurred.
- **ErrorDesc**— Contains error description corresponding to the ErrorNo.

## Host Response Message

- **VerificationResult**—Visa Status Result after invoke **ExecuteTransaction()**.

## Proxy Bypass Properties

You can use these properties to enable a secure connection to the JetPay™ payment gateway from behind a firewall that does not permit outgoing SSL connections.

- **ProxyServer**—Proxy server IP address.
- **ProxyPort**—Proxy server port.
- **UseProxy**—Flag indicating that proxy bypass functionality should be used.

## Return Properties

Use the following read-only properties to communicate transaction status information:

- **Cleared**—Boolean transaction success indicator (generated internally by the component, based on the internal error state reported by ErrMsg and on the server response reported by ActionCode). True or False.

- **CVV2Response**—Code indicating the host response for the CVV2 value submitted by the merchant. One of the following values will be returned: “M”; “N”; “P”; “S”; “U” or “Y”
  - M—CVV2 match.
  - N—CVV2 Not Match.
  - P—Unable to Process.
  - S—The CVV2 should be on the card but merchant indicates it is not.
  - U—The Issuer is not certified or has not provided Visa with encryption keys. Issuer Unregistered.
  - Y—Invalid CVV1 Track Present
- **ErrMsg**—Internal component error message if any.
- **Approval**—Authorization code, six (6) alphanumeric characters.
- **ResponseTransactionID**—Read only. Transaction ID returned from the gateway for this transaction.
- **TransactionID**—JetPay™-generated ID for this transaction.
- **ActionCode**—Contains “000” if the card is approved. Otherwise, it is a JetPay™ Server-specific error code. This means nothing to the end user, and may not be meaningful to the programmer or merchant, but this code should be logged with the other transaction data.
- **ResponseText**—Message from the JetPay™ Server, relating to the ActionCode.
- **AddressMatch**— A code indicating the address match result for a credit card transaction. One of the following values will be returned: “Y”; “N” or “X”.
  - Y—Match
  - N—No Match
  - X—Unknown
- **ZipMatch**—A code indicating the match Zip Code match result for a credit card transaction. One of the following values will be returned: “Y”; “N” or “X”.
  - Y—Match
  - N—No Match
  - X—Unknown
- **AVS**—A code indicating the Address and Zip Code match result information for a credit card transaction. One of the following values will be returned: “A”; “B”;

“C”; “D”; “Y”; “M”; “X”; “F”; “G”; “T”; “K”; “L”; “N”; “O”; “P”; “R”; “S”;  
“U”; “W”; “Z” .

- A— Address only correct. VS, MC, AMEX
- B— Address match, Zip/Postal not verified because incompatible formats. VS, MC
- C— Address and Zip/Postal not verified because incompatible formats. VS, MC  
CM info and ship-to info verified. Standard. AMEX
- D, Y, M, X— Address and Zip/Postal Match. VS, MC
- CM Name, Billing Address and Zip/Postal Match. AMEX
- F— Address and Zip/Postal Match. Applies U.K. Only. VS
- G, I— Address info not verified for international transaction. VS, MC  
CM info and ship-to info verified. Fraud Protection Program. AMEX
- K— CM Name matches. AMEX
- L— CM Name and Billing Zip match. AMEX
- N— Neither address nor Zip/Postal matches. VS, MC, AMEX
- O— CM Name and Billing Address match. AMEX
- P— Zip/Postal match, Address not verified because incompatible formats. VS, MC
- R— Retry, system unable to process. VS, MC, AMEX
- S— Not applicable. Currently not supported. VS, MC  
Address verification subscribed valid (not subscribed). AMEX
- U— Retry, system unable to process. VS, MC, AMEX
- W, Z— Zip/Postal Match, Address does not. VS, MC
- **RetryCount**—Number of communication attempts executed to obtain the final transaction result.
- **TransactionStart**—The timestamp that the transaction began at the host.
- **TransactionStop**—The timestamp of the transaction’s completion at the host.
- **XMLRequest**—The XML payload posted to the host.
- **XMLResponse**—The XML payload received from the host.



|                  |              |  |
|------------------|--------------|--|
| vs_Transtype*    | Input        | One of “SALE”;<br>“AUTHONLY”;<br>“VOID”; “CREDIT”;<br>“FORCE”; “CHECK”;<br>“CAPT”; “ENQ”;<br>“REVERSAL”;<br>“VOIDACH” or “PING”  |
| Vd_Price*        | Input        | A Double value. Amount of the transaction.   |
| vs_CardNum*      | Input        | A String value. Credit card number.  |
| vs_CardExpMonth* | Input        | A String value. Credit card expiration month in MM format.   |
| vs_CardExpYear*  | Input        | A String value. Credit card expiration year in YY format.  |
| rs_ActionCode*   | Output       | A String value. Three-character response on transaction from host. See Appendix B “Host Response Codes and Text” on page 35 for a list of action codes and their meanings. |
| rs_AddressMatch* | Output       | A String value. One-character response on credit card address Match from host.   |
| rs_Approval*     | Input/Output | A String value. Six-character approval code from host.   |
| rb_Cleared*      | Output       | A Boolean value. As Boolean Success flag (evaluation of rs_ActionCode = “000”).  |
| rs_ErrMsg*       | Output       | A String value. Error message from gateway.  |
| rs_ResponseText* | Output       | A String value. Text response (interpretation of rs_ActionCode) from host.   |

|                           |              |   |
|---------------------------|--------------|---|
| rs_ResponseTransactionID* | Output       | A String value. Transaction identifier returned from host.  |
| rs_TransactionID*         | Input/Output | A String value. Transaction identifier sent to host.  |
| rs_ZipMatch*              | Output       | A String value. One-character response on credit card Zip Code match from host.   |
| vs_CardName               | Input        | A String value. Name of the customer.   |
| vs_CardType               | Input        | A String value. Type of credit card (“MasterCard”; “Visa”; “American Express”; “Discover”; “Diners Club” or “Carte Blanche”). |
| vs_BillingAddress         | Input        | String value. Billing address of customer.  |
| vs_BillingCity            | Input        | A String value. Billing city of customer.   |
| vs_BillingCountry         | Input        | A String value. Billing country of customer.  |
| vs_BillingPhone           | Input        | String value. Billing phone of customer.  |
| vs_BillingPostalCode      | Input        | A String value. Billing Zip Code of customer.   |
| vs_BillingStateProv       | Input        | A String value. Billing state of Customer.  |
| vs_CVV2                   | Input        | A String value. Three- or four-character credit card CVV2 code.   |
| vs_Email                  | Input        | A String value. Consumer e-mail address.  |
| vs_OrderNumber            | Input        | A String value. Merchant invoice number.  |
| vs_ProxyServer            | Input        | A String value. Proxy server IP address used to access Internet.  |

|                          |       |  |
|--------------------------|-------|--|
| vl_ProxyPort             | Input | A Long value. Proxy server port used to access Internet.   |
| vl_Test                  | Input | A Transaction Test value.  |
| vs_UDField1              | Input | A String value. User (merchant) defined.   |
| vs_UDField2              | Input | A String value. User (merchant) defined.   |
| vs_UDField3              | Input | A String value. User (merchant) defined.   |
| vb_UseProxy              | Input | A Boolean value. Flag indicating need for use of proxy to access Internet. Default Value is False. |
| vs_UserHost              | Input | A String value. Customer Web host.   |
| vs_UserIPAddr            | Input | A String value. Customer IP address.   |
| vl_Timeout               | Input | A Long value. Maximum seconds to allow for transaction. Default Value is 30 Sec.                   |
| vs_ABANumber             | Input | A String value. ABA number for ACH transactions.   |
| vs_CheckingAccountNumber | Input | A String value. DDA number for ACH transactions.   |
| vs_CheckNumber           | Input | A String value. Check number for ACH transactions.   |
| vd_CheckSurcharge        | Input | A Double value. Surcharge included in price for check transactions.                                |
| vs_DispositionType       | Input | A String value. Custom host disposition code.  |
| vd_TaxAmount             | Input | A Double value. Tax amount   |

|                 |        |   |
|-----------------|--------|---|
| vs_IndustryType | Input  | A String value. Industry Type. One of the following nine values: "ECOMMERCE"; "RETAIL"; "MOTO"; "HOTEL"; "RESTAURANT"; "AUTORENTAL"; "AIRLINE"; "PARKING"; "QUASICASH". The default value is "ECOMMERCE". |
| rs_CVV2Response | Output | A String value. One-character response on Card Validation Code from the host.   |

## ExecuteTransactionASP Method Properties

For the **CJetPayCCMTS** method, **ExecuteTransactionASP**, the syntax has these parts (\* indicates a required property; the rest are optional):

| Parameter        | Host Direction | Description   |
|------------------|----------------|---|
| vs_MerchantID*   | Input          | A String value. Transactional Technologies supplied merchant identifier.  |
| vs_Transtype*    | Input          | A String value. One of “SALE”; “AUTHONLY”; “VOID”; “CREDIT”; “FORCE”; “CHECK”; “VOIDACH”; “CAPT”; “ENQ”; “PING”; or “REVERSAL”  |
| vd_Price*        | Input          | A Double value. Amount of transaction.  |
| vs_CardNum*      | Input          | A String value. Credit card number.   |
| vs_CardExpMonth* | Input          | A String value. Credit card expiration month in MM format.  |
| vs_CardExpYear*  | Input          | A String value. Credit card expiration year in YY format.   |
| rv_ActionCode*   | Output         | A Variant value. Three-character response on transaction from host. See Appendix B “Host Response Codes and Text” on page 35 for a list of action codes and their meanings. |
| rv_AddressMatch* | Output         | A Variant value. One-character response on credit card address match from host.   |
| rv_Approval*     | Input/Output   | A Variant value. Six-character approval code from host.   |

|                           |              |   |
|---------------------------|--------------|---|
| rv_Cleared*               | Output       | A Variant value. As Boolean Success flag (evaluation of rs_ActionCode = "000").                                     |
| rv_ErrMsg*                | Output       | A Variant value. Error message from gateway.  |
| rv_ResponseText*          | Output       | A Variant value. Text response (interpretation of rs_ActionCode) from host.   |
| rv_ResponseTransactionID* | Output       | A Variant value. Transaction identifier returned from host.   |
| rv_TransactionID*         | Input/Output | A Variant value. Transaction identifier sent to host.   |
| rv_ZipMatch*              | Output       | A Variant value. One-character response on credit card Zip Code match from host.                                    |
| vs_CardName               | Input        | A String value. Name of customer.   |
| vs_CardType               | Input        | A String value. Type of credit card ("MasterCard"; "Visa"; "American Express"; "Discover"; "Diners Club" or "JCB"). |
| vs_BillingAddress         | Input        | String value. Billing address of customer.  |
| vs_BillingCity            | Input        | A String value. Billing city of customer.   |
| vs_BillingCountry         | Input        | A String value. Billing country of customer.  |
| vs_BillingPhone           | Input        | String value. Billing phone of customer.  |
| vs_BillingPostalCode      | Input        | A String value. Billing Zip Code of customer.   |
| vs_BillingStateProv       | Input        | A String value. Billing state of customer.  |
| vs_CVV2                   | Input        | A String value. Three-or four-character credit card   |

|                          |       |  |
|--------------------------|-------|--|
|                          |       | CVV2 code.   |
| vs_Email                 | Input | A String value. Customer e-mail address.   |
| vs_OrderNumber           | Input | A String value. Merchant invoice number.   |
| vs_ProxyServer           | Input | A String value. Proxy server IP address used to access Internet.                                   |
| vl_ProxyPort             | Input | A Long value. Proxy server port used to access Internet.   |
| vl_Test                  | Input | A Transaction test value.  |
| vs_UDField1              | Input | A String value. User (merchant) defined.   |
| vs_UDField2              | Input | A String value. User (merchant) defined.   |
| vs_UDField3              | Input | A String value. User (merchant) defined.   |
| vb_UseProxy              | Input | A Boolean value. Flag indicating need for use of proxy to access Internet. Default Value is False. |
| vs_UserHost              | Input | A String value. Customer Web host.   |
| vs_UserIPAddr            | Input | A String value. Customer IP address.   |
| vl_Timeout               | Input | A Long value. Maximum seconds to allow for transaction. Default Value is 30 Sec.                   |
| vs_ABANumber             | Input | A String value. ABA number for ACH transactions.   |
| vs_CheckingAccountNumber | Input | A String value. DDA number for ACH transactions.   |
| vs_CheckNumber           | Input | A String value. Check number for ACH transactions.   |
| vd_CheckSurcharge        | Input | A Double value. Surcharge included in  |

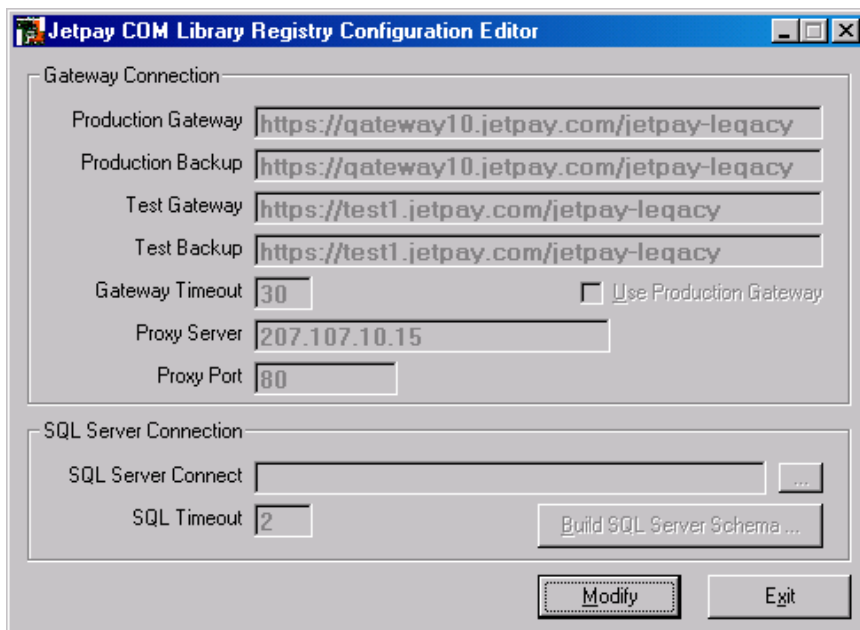
|                    |        |   |
|--------------------|--------|---|
|                    |        | price for check transactions.   |
| vs_DispositionType | Input  | A String value. Custom host disposition code.   |
| rv_CVV2Response    | Output | A String value. One-character response on Card Validation Code from the host.   |
| vd_TaxAmount       | Input  | A Double value. Tax amount  |
| vs_IndustryType    | Input  | A String value. Industry Type. One of the following nine values: "ECOMMERCE"; "RETAIL"; "MOTO"; "HOTEL"; "RESTAURANT"; "AUTORENTAL"; "AIRLINE"; "PARKING"; "QUASICASH". The default value is "ECOMMERCE". |

# JetPay COM Library Configuration Editor

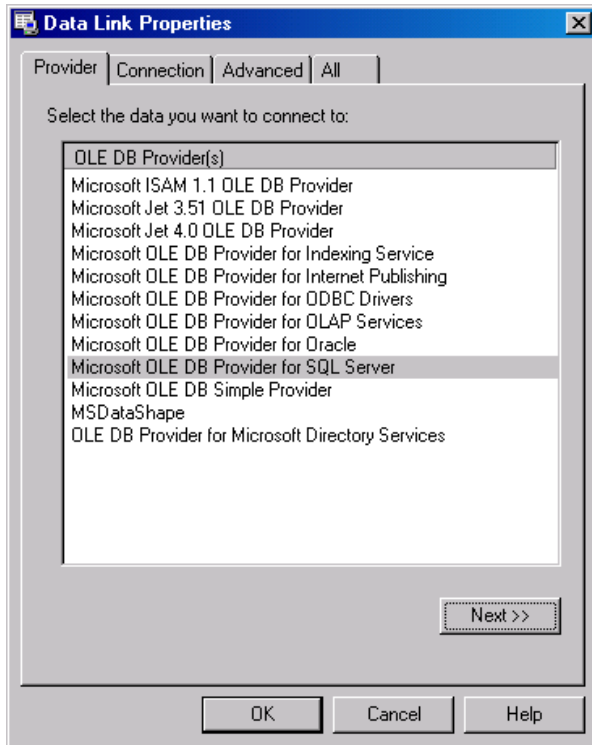
The **JetPay COM Library Configuration Editor** enables you to set up the JetCom engine to log transactions on the merchant side. The computer on which JetCom is installed must have access to a Microsoft SQL Server 7 database. Before running this application you must create the database where the transactions will be logged.

## To select a database for local transaction logging

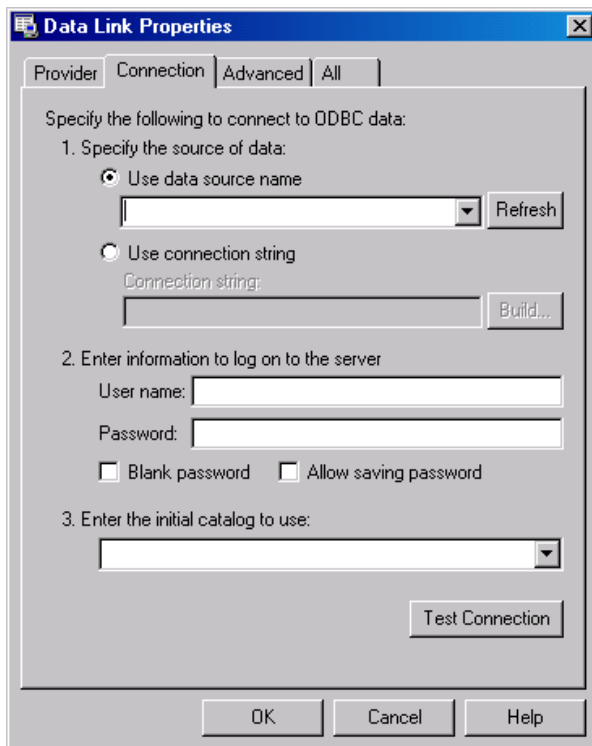
1. Click **Start**, point to **Programs**, point to **JetPay Credit Card Gateway COM Library**, and click **JetPay COM Library Configuration Editor**. The JetPay™ COM Library Configuration Editor window appears.



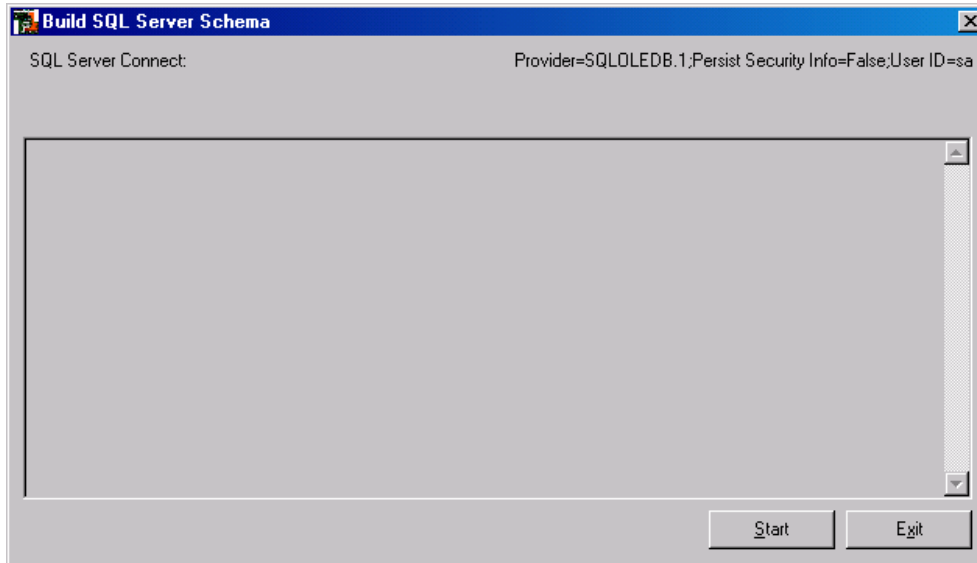
2. Click **Modify** to edit the Gateway Connection fields and the SQL Server Connection fields  
3. Click **...** to select the database information. The Data Link Properties window appears.



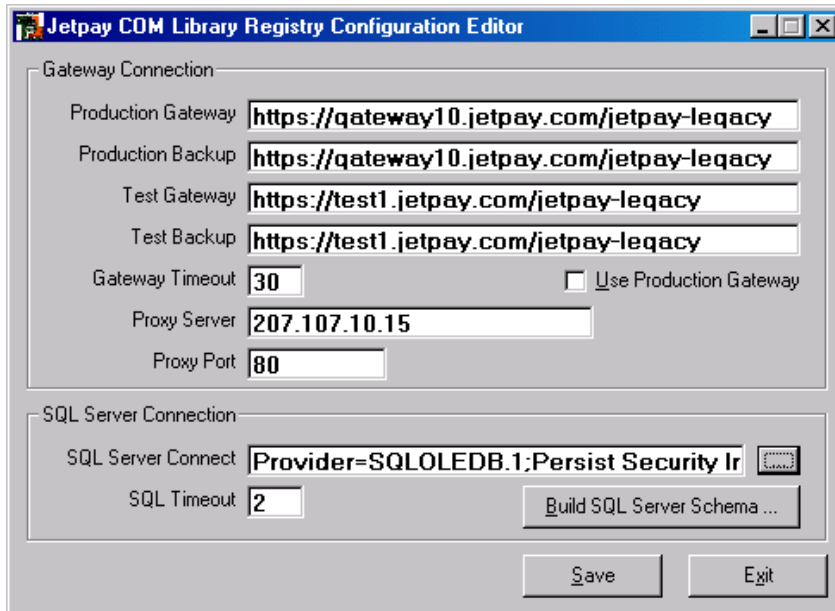
3. On the **Provider** tab, select the OLE database provider that you will be using.
4. Click **Next**. The **Connection** tab appears.



5. Under “1. Specify the source of data”, select either the **Use data source name** or the **Use connection string** option button.
6. If you selected **Use data source name**, select the data source from the list box. If you selected **Use connection string**, click **Build** and select your data source.
7. Under “2. Enter information to log on to the server”, type your database security information in the appropriate boxes.
8. Under “3. Enter the initial catalog to use”, select the database that you have created to store transaction data.
9. Click **Test Connection** to make sure that there are no problems connecting to the database.
10. Click **OK** to return to the **JetPay COM Library Configuration Editor** window.
11. Click **Build SQL Server Schema**. The Build SQL Server Schema window appears.



12. Click **Start** to build the SQL Server Schema.
13. Once the Schema completes successfully, click **Exit** to return to the **JetPay COM Library Configuration Editor** window.
14. Click **Save** to save the SQL Server connect string that you have created.



15. Click **Save** to keep the changes, then press **Exit**. JetCom is now configured to log transaction to the specified database.

# Demo Test Bench Application

Included in the installation is the JetPay Demo Test Bench Application. This Visual Basic application lets you run transactions against the test host. In addition to running transactions, you can also run “load tests” for a single transaction.

---

**Note** Transactional Technologies also distributes the JetPay Demo Test Bench Application source code when you install the package. To view the source code, click **Start**, point to **Programs**, point to **JetPay™ Credit Card Gateway COM Library**, and click **Demo Test App VB Source**.

---

You can invoke the transaction engine through the Test Bench using one of two methods: **CJetPayCC** (which houses a set of properties that describe the inputs, outputs, and configuration elements used in transaction processing against the JetPay transaction processing host), and **CCJetPayCCMTS** (which provides an alternate, single-function call).

## The Basics Tab

The following window appears when you run the JetPay Demo Test Bench Application:

JetPay Test Bench

File

Basics Options Repeat Certify

Credit Card Info

CardType: Visa

Card Number: 4000300020001000 CV2: 1234

Expires: (MM YY): 12 04

Merchant ID: TESTTERMINAL

Approval: [ ] Action Code: [ ] AVS: [ ]

Transaction ID: [ ]

Zip Match: [ ] CV2 Response: [ ] Address Match: [ ]

Response Transaction ID: [ ]

Response Text: [ ]

Error Message: [ ]

Test Mode: Use Gateway In Regis

TransType: SALE

Origin: INTERNET

Industry Type: MOTO

Amount \$: 1.00

Use JetPayCCMTS

Fetch Trans ID

Submit

Component Name: vxJetCom

Component Version: 1.0.209

Once you have filled in the fields under the Credit Card Info group, you must choose a **Test Mode**, a **Trans Type**, the **Origin**, the **Industry Type** and then enter a dollar amount.

If you select **Use Gateway in Registry**, the Jet Pay Test Bench defaults to the setting you chose in the JetPay COM Library Registry Configuration Editor (see page 19 for more information).

If you select **Fetch Trans ID**, the Jet Pay Test Bench will automatically retrieve a unique transaction I.D. from the Jet Pay server.

If you select the **Use JetPay CCMTS** checkbox, you will invoke the transaction engine using CCJetPayCCMTS, which provides an alternate, single-function call.

There are eleven transaction types:

**Sale**—Authorizes and closes a credit card charge in a single transaction •••

**Authonly**—The credit card limit is checked to verify that a certain amount is available and to reserve that amount, but the card is not charged.

**Void**—This removes a credit card charge from the host before it settles the transaction.

**Credit**—A credit or refund to a customer’s credit card account.

**Force**—A credit card charge using an amount equal to or less than the amount of a previous authorized transaction.

**Check**—An ACH check transaction.

**Capture**—To capture an Authonly transaction for settlement.

**Enquire**— Check the status of a transaction.

**Ping**— Verify the connection to the Jet Pay Server.

**Reversal**—This removes a check transaction before settlement.

**VoidACH**— This removes an ACH charge from the host before it settles the transaction.

Click **Submit** to run the transaction. The fields at the bottom of the screen are filled in depending on the success of the transaction.

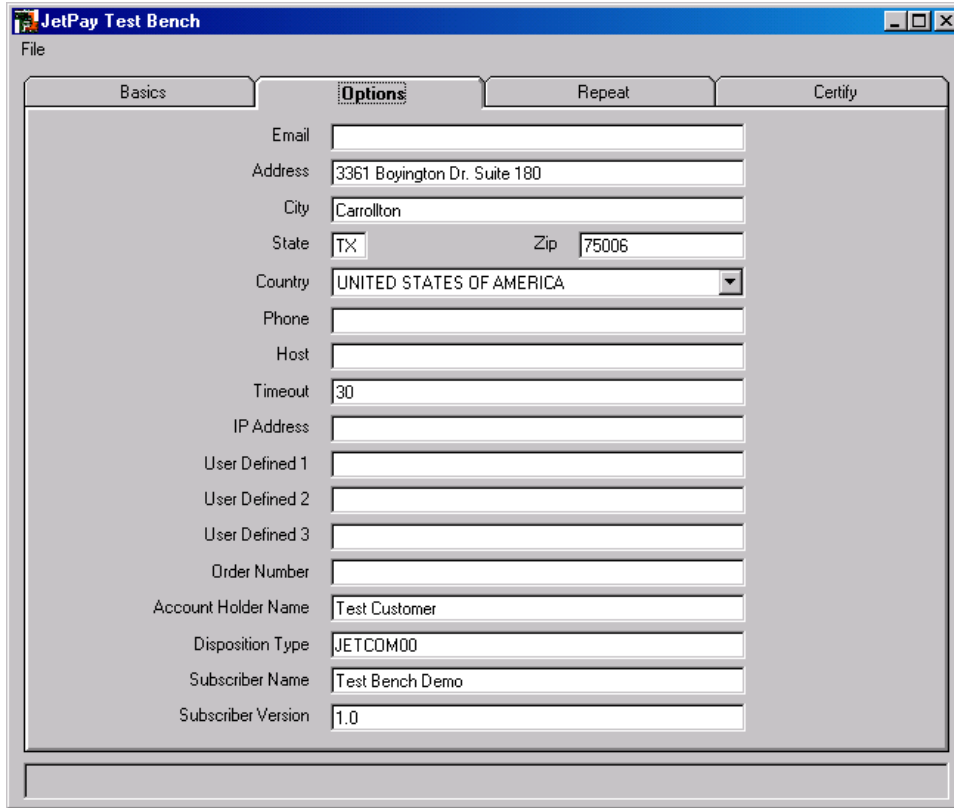
The screenshot shows the 'JetPay Test Bench' application window. The window has a menu bar with 'File' and a toolbar with 'Basics', 'Options', 'Repeat', and 'Certify' tabs. The 'Basics' tab is active. The form contains the following fields and controls:

- Credit Card Info:** CardType (dropdown: Visa), Card Number (4000300020001000), CW2 (1234), Expires: (MM YY) (12 | 04).
- Merchant ID:** TESTTERMINAL
- Approval:** 000338, Action Code (000), AVS (Y)
- Transaction ID:** 0306241457347E1357
- Zip Match:** Y, CW2 Response (P), Address Match (Y)
- Response:** Transaction ID (0306241457347E1357), Response Text (APPROVED), Error Message (empty).
- Test Mode:** Use Gateway In Regis (dropdown)
- TransType:** SALE (dropdown)
- Origin:** INTERNET (dropdown)
- Industry Type:** MOTO (dropdown)
- Amount \$:** 1.00
- Use JetPayCCMTS
- Fetch Trans ID
- Submit** button
- Component Name:** vxJetCom
- Component Version:** 1.0.209

At the bottom of the window, the 'Elapsed time' is 0.870375.

# The Options Tab

The following window appears when you run the JetPay Demo Test Bench Application and click the **Options** tab:



The screenshot shows a window titled "JetPay Test Bench" with a menu bar containing "File". Below the menu bar are four tabs: "Basics", "Options", "Repeat", and "Certify". The "Options" tab is selected and contains the following fields:

|                     |   |
|---------------------|---|
| Email               | <input type="text"/>                                      |
| Address             | <input type="text" value="3361 Boyington Dr. Suite 180"/> |
| City                | <input type="text" value="Carrollton"/>                   |
| State               | <input type="text" value="TX"/>                           |
| Zip                 | <input type="text" value="75006"/>                        |
| Country             | <input type="text" value="UNITED STATES OF AMERICA"/>     |
| Phone               | <input type="text"/>                                      |
| Host                | <input type="text"/>                                      |
| Timeout             | <input type="text" value="30"/>                           |
| IP Address          | <input type="text"/>                                      |
| User Defined 1      | <input type="text"/>                                      |
| User Defined 2      | <input type="text"/>                                      |
| User Defined 3      | <input type="text"/>                                      |
| Order Number        | <input type="text"/>                                      |
| Account Holder Name | <input type="text" value="Test Customer"/>                |
| Disposition Type    | <input type="text" value="JETCOM00"/>                     |
| Subscriber Name     | <input type="text" value="Test Bench Demo"/>              |
| Subscriber Version  | <input type="text" value="1.0"/>                          |

You can enter information into any of the first 17 optional fields on this window. However, in the **Disposition Type** box, you must enter **JETCOM00**. You only need to change the disposition code if Transactional Technologies has requested it of you. The Disposition Type value is used to make the values in the user-defined fields invoke special processing.

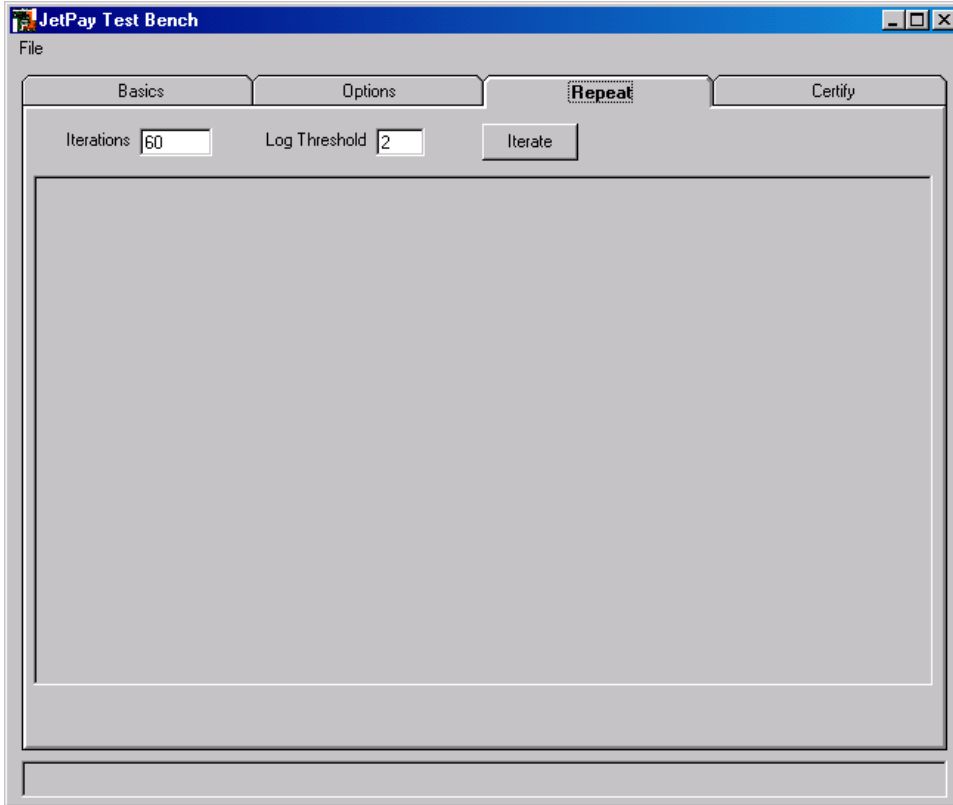
---

**Note** For additional information about string lengths, and so forth, consult the JetPay Credit Card Gateway COM Library Help file (**vxJetCom.chm**). To open this file, click **Start**, point to **Programs**, point to **JetPay Credit Card Gateway COM Library**, and click **Help**.

---

# The Repeat Tab

The following window appears when you run the JetPay Demo Test Bench Application and click the **Repeat** tab:



You can perform “load tests” on a single transaction in this window.

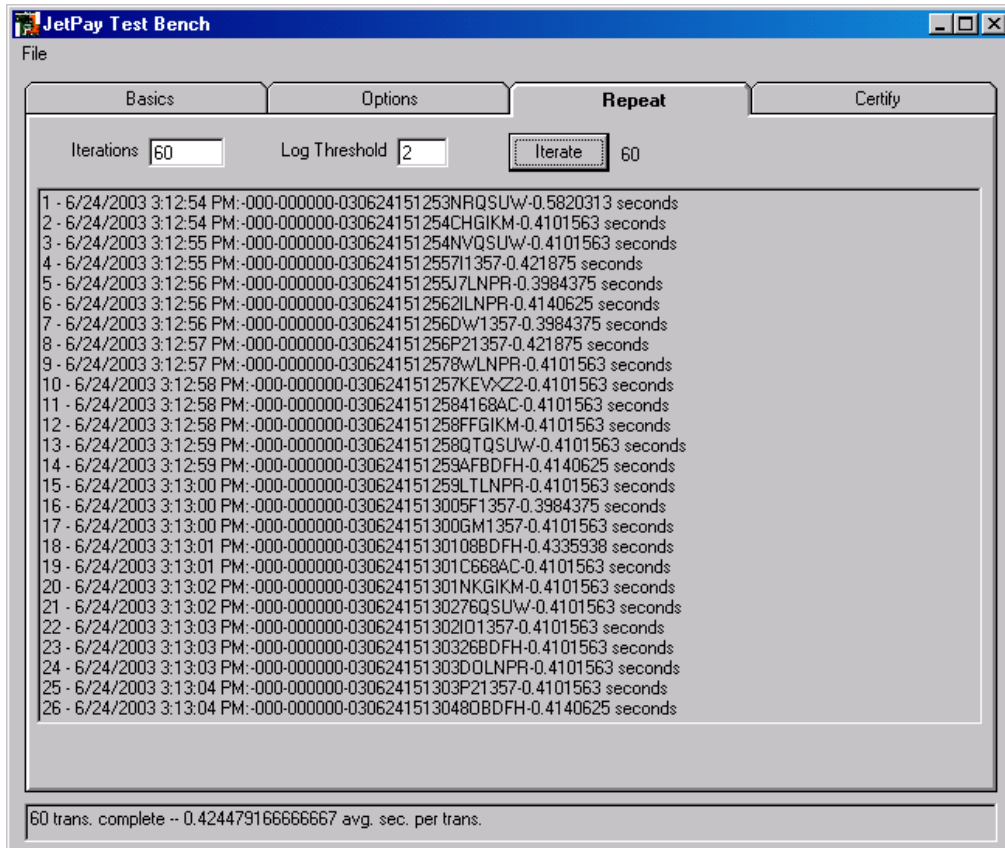
In the **Iterations** box, enter the number of times that you want the transaction to be run. In the **Log Threshold** box, enter the number of seconds that a transaction must take in order for it to be written to the on-screen log.

---

**Note** Any transaction that passes back anything but a 000 code (that is, an error code) is also written to the on-screen log.

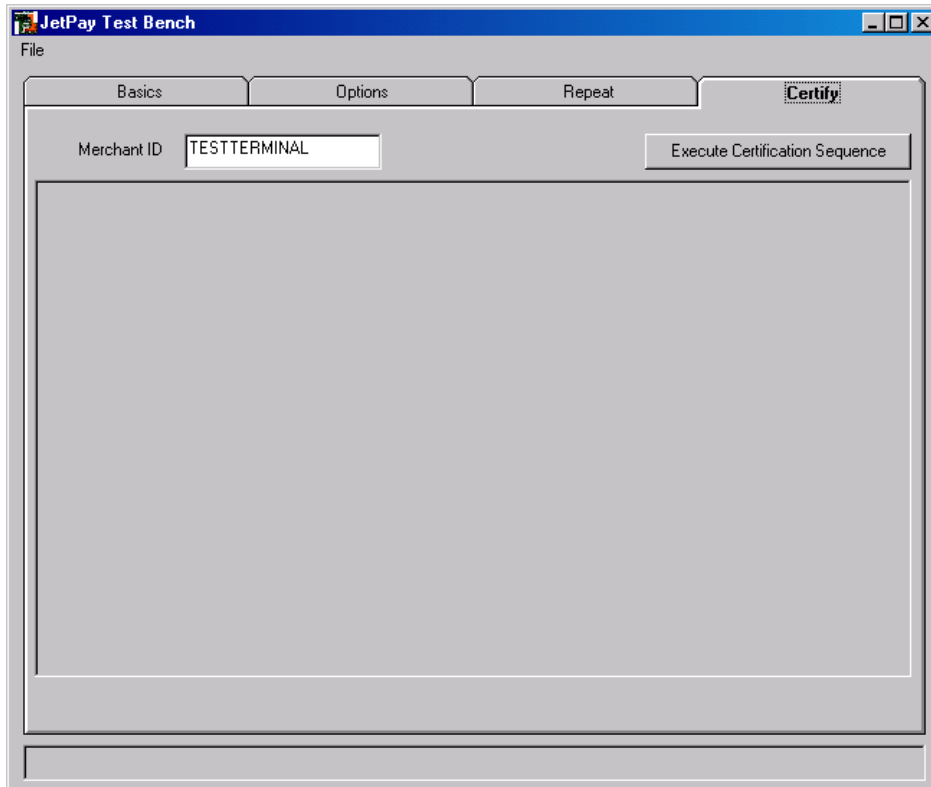
---

The following graphic is an example of how the Repeat tab window will look after repeating a transaction 60 times:



# The Certify Tab

The following window appears when you run the JetPay Demo Test Bench Application and click the **Certify** tab:



When you click **Execute Certification Sequence**, the Test Bench “exercises” the component by running through 14 transactions. These transactions include all seven transactions (sale; void; authonly; force; check; reversal; and credit) as both “simple” and “extended” transactions.

Simple transactions only differ from extended transactions by the amount of fields filled out.

The transactions run in a specific sequence. For example, “force” must always follow “authonly” because a “force” is required to complete the “authonly” transaction. The Certification Sequence runs according to the order of the files in the **sequence.txt** file. You can find this file in the **JetPay Credit Card Gateway COM Library\DemoApp\Cert** folder.

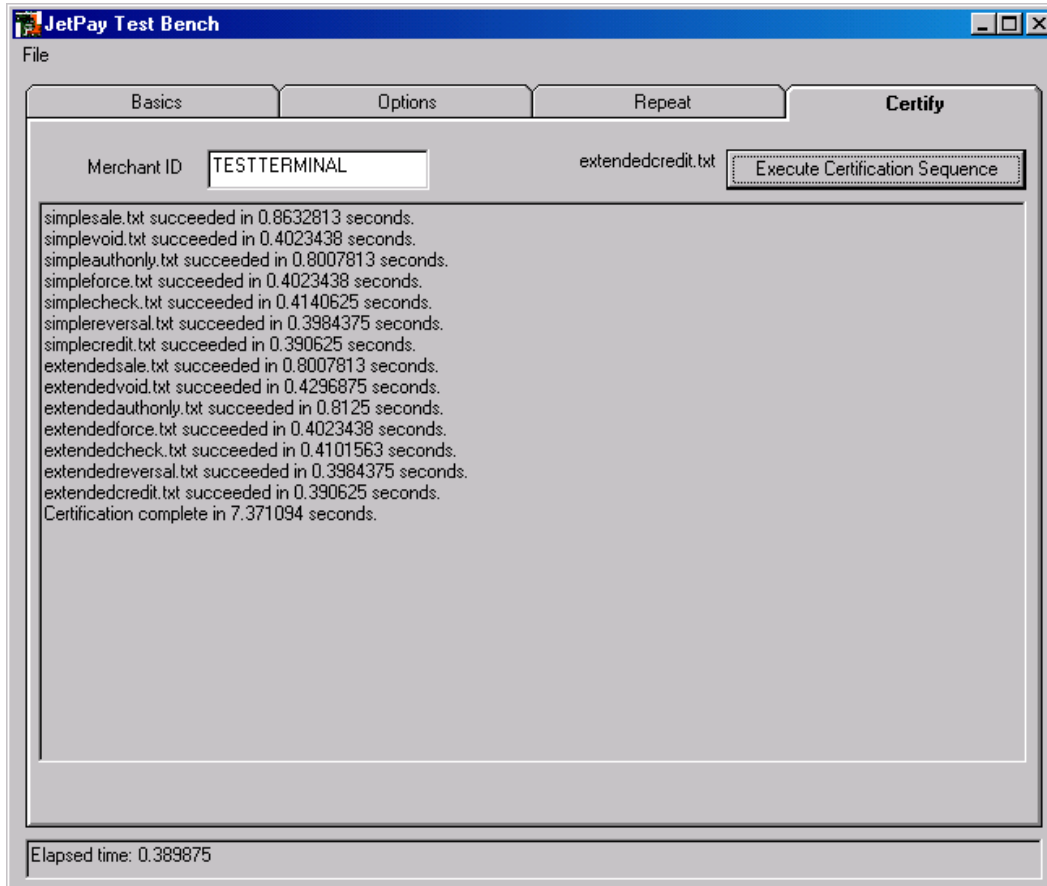
Although you can change the sequence order, you must keep the following files in order:

“sale” then “void” ...

“authonly” then “force”

“check” then “reversal”

The following graphic is an example of how the Certify tab window will look after executing the Certification Sequence:



# Appendix A

## CJetPayCC Properties

| Parameter         | Host Direction              | Description   | Data Type                             |
|-------------------|-----------------------------|---|---------------------------------------|
| ABANumber         | Input                       | Routing number of the bank upon which check is drawn.   | String of nine characters.            |
| ActionCode        | Output                      | Three-character code returned from the transaction-processing engine.   | String of three characters.           |
| SubscriberVersion | Input                       | Identify the Subscriber Version.  | String value.                         |
| SubscriberName    | Input                       | Identify the Subscriber Name.   | String value                          |
| DllVersion        | Output                      | Identify the JetCOM Dll Component Version.  | String value                          |
| DllName           | Output                      | Identify the JetCOM Dll Component Name.   | String value                          |
| AddressMatch      | Output                      | Code indicating address match result for credit card transactions.  | One-character string                  |
| Approval          | Input / Output ECH-assigned | six-character approval code.  | String of six characters.             |
| BillingAddress    | Input Billing               | address of consumer.  | String of up to thirty-two characters |
| BillingCity       | Input                       | Billing city of consumer.   | String of up to thirty-two characters |
| BillingCountry    | Input                       | Billing ISO country code of consumer. Valid values are full alphabetic country name or ISO Country Code. Country names are automatically converted ISO Country Codes. | String value.                         |
| BillingPhone      | Input Billing               | phone number of the consumer.   | String of up to thirty-two characters |
| BillingPostalCode | Input                       | Billing postal or zip code of the consumer.   | String of up to ten characters        |
| BillingStateProv  | Input                       | Billing state of the consumer.  | String of up to two characters        |

|         |       |  |               |
|---------|-------|--|---------------|
| CardExp | Input | Read-only four-digit rollup of CardExpMonth and CardExpYear. | String value. |
|---------|-------|--|---------------|

|                       |        |  |                                      |
|-----------------------|--------|--|--------------------------------------|
| CardExpMonth          | Input  | Two-digit card expiration month  | String of two characters             |
| CardExpYear           | Input  | Two-digit card expiration year   | String of two characters             |
| CardName              | Input  | Name of the consumer.  | String of up to forty characters     |
| CardNum               | Input  | Credit card number to be submitted on the transaction.   | String of up to sixteen characters   |
| CardType              | Input  | Credit card type. Valid values are:<br>"MasterCard"; "MC";<br>"Visa"; "American Express"; "AMEX";<br>"Discover"; "Disc";<br>"Diners Club" or<br>"Carte Blanche".                                     | String value                         |
| CheckingAccountNumber | Input  | Checking account number to be submitted on the transaction.  | String of up to seventeen characters |
| CheckNumber           | Input  | Check number to be submitted on the transaction  | String of up to six characters       |
| CheckSurcharge        | Input  | Amount of check surcharge to be submitted on the transaction. Must be greater than or equal to zero (0) and less than 100000.00.   | Double value                         |
| CheckingAccountType   | Input  | The type of the account to be submitted on the transaction. The value must be equal to one of the following values:<br>"CHECKING";<br>"SAVINGS" or<br>"BUSINESSCK". The default value is "CHECKING". | Double value                         |
| Cleared               | Output | Transaction success indicator.   | Boolean value                        |

|              |             |   |                                 |
|--------------|-------------|---|---------------------------------|
| CardPresent  | Input       | Card Present indicator.   | Boolean value                   |
| CVV2         | Input       | Credit card CVV2 number to be submitted on the transaction. This is made up of three or four additional digits on the credit card. Using these digits provides an additional measure of security for Internet transactions. | String of up to four characters |
| CVV2Response | Output Code | indicating the host response for the CVV2 value submitted by the merchant. One of the following values will be returned: "M"; "N"; "P"; "S" or "U".   | One-character String            |

|                 |        |  |  |
|-----------------|--------|--|--|
| DispositionType | Input  | Code used to indicate a custom extension disposition model on the transaction-processing engine. Do not alter unless instructed to do so.        | Eight character string                 |
| Email           | Input  | Consumer e-mail address.   | String of up to sixty-four characters  |
| ErrMsg          | Output | Error message indicating cause of a process failure.   | String value                           |
| MerchantID      | Input  | Merchant identifier used to route transaction proceeds to the appropriate merchant's account.  | Twelve-character string.               |
| OrderNumber     | Input  | Order number   | String of up to twenty-five characters |
| Price           | Input  | Purchase price to be transacted. It should be formatted with a decimal and no dollar sign. Must be greater than zero (0) and less than 100000.00 | Double value                           |
| ProxyPort       | Input  | Port of proxy server used to access the Internet.  | Long value                             |

|                       |        |  |                           |
|-----------------------|--------|--|---------------------------|
| ProxyServer           | Input  | IP address of proxy server used to access the Internet.  | String value              |
| ResponseText          | Output | Text message returned from the transaction processing engine   | String value              |
| ResponseTransactionID | Output | Transaction identifier returned from host. Should match TransactionID  | Eighteen-character string |
| RetryCount            | Output | Number of communication attempts executed to obtain result   | Long value                |
| Test                  | Input  | Testing mode   |                           |
| Timeout               | Input  | Maximum number of seconds to wait for transaction response. The minimum value is thirty (30)   | Long value.               |
| TerminalType          | Input  | Optional for backward compatibility. One of the following nine values:<br>"ECOMMERCE" "RETAIL" "MOTO"<br>"HOTEL"<br>"RESTAURANT"<br>"AUTORENTAL"<br>"AIRLINE"<br>"PARKING"<br>"QUASICASH". The default value is "ECOMMERCE". | String Value              |
| Track1Data            | Input  | Track 1 Raw Data   | String Value              |
| Track2Data            | Input  | Track 2 Raw Data   | String Value              |

|                  |                           |   |                                       |
|------------------|---------------------------|---|---------------------------------------|
| TransactionID    | Input / Output Identifier | for transaction                                       | Eighteen-character string             |
| TransactionStart | Output                    | Timestamp transaction was started                     | String value                          |
| TransactionStop  | Output                    | Timestamp transaction was completed                   | String value                          |
| Transtype        | Input                     | Type of transaction to post. Default value is "SALE." | String value.                         |
| UDField1         | Input                     | Merchant defined field number one                     | String of up to sixty-four characters |

|              |        |  |                                       |
|--------------|--------|--|---------------------------------------|
| UDField2     | Input  | Merchant defined field number one  | String of up to sixty-four characters |
| UDField3     | Input  | Merchant defined field number one  | String of up to sixty-four characters |
| UseProxy     | Input  | Flag indicating that proxy bypass functionality should be used. This allows transactions to be posted from behind firewalls that are configured to prevent secure transactions.  | Boolean value                         |
| UserHost     | Input  | Consumer host name   | String of up to thirty-two characters |
| UserIPAddr   | Input  | Consumer IP address  | String of up to thirty-two characters |
| XMLRequest   | Output | XML payload posted to host   | String value                          |
| XMLResponse  | Output | XML payload received from host   | String value                          |
| ZipMatch     | Output | Code indicating zip code match result for credit card transactions   | One-character String                  |
| TaxAmount    | Input  | Tax Amount for the transaction.  | Double value                          |
| IndustryType | Input  | One of the following nine values: "ECOMMERCE" "RETAIL" "MOTO" "HOTEL" "RESTAURANT" "AUTORENTAL" "AIRLINE" "PARKING" "QUASICASH". The default value is "ECOMMERCE".   | String Value                          |
| AVS          | Output | A code indicating the Address and Zip Code match result information for a credit card transaction. One of the following values will be returned: "A"; "B"; "C"; "D"; "Y"; "M"; "X"; "F"; "G"; "P"; "K"; "L"; "N"; "O"; "P"; "R"; "S"; "U"; "W"; "Z". | String Value                          |

|                    |        |  |               |
|--------------------|--------|--|---------------|
| UseDialup          | Input  | Indicate the use of Dial Up instead of Ethernet.   | Boolean       |
| PortSettings       | Input  | Settings for the Port during the Dial Up.  | String Value  |
| AuthPhone          | Input  | Authorization Phone during the Dial Up.  | String Value  |
| UseWinInet         | Input  | Indicate the Win Inet uses.  | Boolean Value |
| Cavv               | Output | Transaction Stain from PAREs (28 Character, Base 64).  | String Value  |
| Xid                | Output | Transaction Xid from PAREs.  | String Value  |
| Eci                | Output | E-commerce Indicator (ECI) from PAREs.   | String Value  |
| VerificationResult | Output | Visa Status Result after invoke <b>ExecuteTransaction()</b> .  | String Value  |
| MAPUrl             | Input  | MAP URL address.   | String Value  |
| Version            | Input  | Version identifier, currently "1.1".   | String Value  |
| ProcessorID        | Input  | Acquiring Institution identification code. Typically a 6 digit BIN assigned to the acquirer by Visa. | String Value  |
| Password           | Input  | Acquirer defined merchant password.  | String Value  |
| PurchaseCurrency   | Input  | 3 digit numeric, ISO 4217 currency code for the sale amount.   | String Value  |
| OrderDescription   | Input  | Brief description of items purchased, limited to 125.  | String Value  |
| UserAgent          | Input  | The exact content of the HTTP user-agent header.   | String Value  |
| Recurring          | Input  | Recurring Flag to specify if the Merchant and cardholder have agreed to recurring payments.          | String Value  |
| RecurringFrecuency | Input  | Integer value indicating the minimum number of days between authorizations. A                        | String Value  |

|                       |        |   |              |
|-----------------------|--------|---|--------------|
|                       |        | frequency of monthly is indicated by the value 28.  |              |
| RecurringEnd          | Input  | The date after which no further recurring authorizations should be preformed. Format YYYYMMDD.  | String Value |
| Installment           | Input  | An integer indicating the maximum number of permitted authorizations for installment payments. Must be included if the merchant and cardholder have agreed to installment payments. | String Value |
| Enrolled              | Output | Must be one of the following values: "Y"; "N" or "U"  | String Value |
| ACSUrl                | Output | Contains the fully qualified URL of the Issuer ACS. This will be used by the merchant to redirect the cardholder. Available if Enrolled = Y.  | String Value |
| PAReq                 | Input  | Contains encoded PAReq generated by the MAPS Available if Enrolled = Y  | String Value |
| PARes                 | Output | Transaction status after finish the execution of the redirect to <b>ACSUrl</b> . Y/N/U ("A" future) value Y   | String Value |
| PAResStatus           | Output | Transaction status from PARes. Y/N/U ("A" future) value Y   | String Value |
| SignatureVerification | Output | Signature Verification status. Y/N value.   | String Value |
| ErrorNo               | Output | Contains Error No if any processing error occurred.   | String Value |
| ErrorDesc             | Output | Contains error description corresponding to the ErrorNo.  | String Value |
| FolioNumber           | Input  | Number of Folio.  | String Value |
| CheckInDate           | Input  | Lodging check in date.  | String Value |

|                 |        |  |               |
|-----------------|--------|--|---------------|
| CheckOutDate    | Input  | Lodging check out date.  | String Value  |
| OrigAuthAmount  | Input  | Original authorization amount made during the check in.  | Double Value  |
| TotalAuthAmount | Input  | Final authorization amount made during the check out.  | Double Value  |
| RoomRate        | Input  | Indicates the Room Rate.   | Double Value  |
| PrestigiousProp | Input  | Prestigious Value.   | Boolean Value |
| SpecialCode     | Input  | Special Code.  | String Value  |
| Extra           | Input  | Extra field.   | String Value  |
| Origin          | Input  | Must be equal to one of the following five values: "INTERNET"; "POS"; "RECURRING"; "MAIL ORDER"; "PHONE ORDER". The default value is "INTERNET". | String Value  |
| ReferenceNumber | Output | Reference Number Identifier  | String Value  |
| TicketNumber    | Input  | Number of Ticket.  | String Value  |
| ServerID        | Input  | Waiter/Waitress Server ID.   | String Value  |
| BaseAmount      | Input  | Base Amount for the Transaction.   | Double Value  |
| TipAmount       | Input  | Tip Amount for the Transaction.  | Double Value  |
| TableNumber     | Input  | Table Number.  | Double Value  |

# Appendix B

## Host Response Codes and Text

As part of a transaction, the Transactional Technologies host supplies a three-character response code that is used to populate the ActionCode field. The following table lists the response code, the message text associated with that code, and an explanation of its meaning.

| Code  | Error Message               | Description   |
|---|-----------------------------|---|
| General response codes                          |                             |   |
| 000   | ALL OK                      | The card is approved  |
| 300   | DECLINED                    | Generic decline for fraud prevention. Normally one could verify identity in person, but cannot for e-commerce.  |
| Transactional Technologies host response codes. |                             |   |
| 900   | INVALID MESSAGE TYPE        | The message received is not in a valid format.  |
| 901   | MERCHANT ID NOT FOUND       | The merchant information is not valid.  |
| 902   | INVALID PROCESSING CODE     | The processing code was invalid for the switch or merchant parameters.  |
| 903   | DEBIT NOT SUPPORTED         |   |
| 904   | PRIVATE LABEL NOT SUPPORTED |   |
| 905   | INVALID CARD TYPE           | The card type indicated in the transaction is not valid. Valid values for this field are "0", "1" and "2". Refer to Appendix A for a description of this field and a description of the values. Contact Practice Management Vendor. |
| 906   | UNIT NOT ACTIVE             | Transactional Technologies has deactivated the terminal.  |
| 907   | DUPLICATE ID                | The clerk ID is a duplicate.  |

### Code Error Message Description

|     |                                     |  |
|-----|-------------------------------------|--|
| 908 | MANUAL CARD ENTRY<br>INVALID        | The transaction sent to the RPM Server has indicated that the transaction is for a manually entered credit card, but the card information is not in the format expected for a manual entry.  |
| 909 | INVALID TRACK<br>INFORMATION        | The transaction sent to the RPM Server has indicated that the transaction is for a swiped credit card, but the magnetic stripe information is missing or not properly formatted.   |
| 910 | UNSUPPORTED CARD<br>INFORMATION     | The credit card is not one that is supported by the customer. The RPM System can process VISA, MasterCard, American Express and Discover. Your business office has elected not to accept the type of credit card that was just swiped. |
| 911 | MASTER MERCHANT<br>RECORD NOT FOUND | Invalid Master Merchant record or a system database error has occurred.  |
| 912 | INVALID CARD FORMAT                 | There is a problem with the card format received in the transaction. For swiped card transactions, the sentinel characters are missing; for manually entered card numbers, the format of the data is invalid.                          |
| 913 | INVALID TRAN TYPE                   | The transaction type is not valid. Operator may have selected a function that has not been implemented on the transaction Server.  |
| 914 | INVALID CARD LENGTH                 | The number of digits in the credit card account number is less than 13 characters in length. Re-enter the credit card number and re-try the transaction. If condition persists, contact Practice Management Vendor.                    |

|     |                       |  |
|-----|-----------------------|--|
| 915 | INVALID MONTH         | If an invalid entry was made, re-try the transaction with the corrected date. If problem persists, contact Practice Management Vendor.   |
| 916 | INVALID YEAR          | If an invalid entry was made, re-try the transaction with the corrected date. If problem persists, contact Practice Management Vendor.   |
| 917 | EXPIRED CARD          | If an invalid entry was made, re-try the transaction with the corrected date. If problem persists, contact Practice Management Vendor.   |
| 918 | INVALID TRACK 2 FIELD | One of the data fields on the second track of the magnetic stripe is not valid. Re-try the transaction and if it fails again, manually enter the account number. If this problem persists on other transactions where the credit card is swiped through the magnetic stripe reader, report this problem to the Practice Management Vendor. |
| 919 | INVALID ENTRY MODE    | Entry mode of the card information is invalid.   |
| 920 | INVALID AMOUNT        | The amount is not within proper ranges.  |
| 921 | INVALID MSG FORMAT    | The message has formatting errors.   |
| 923 | INVALID ABA NUMBER    | The ABA number for the ACH transaction is invalid.   |
| 940 | RECORD NOT FOUND      | Can be seen on transaction upload, and edit transactions.  |
| 941 | MERCHANT ID ERROR     | Possible invalid merchant id or id is inactive.  |
| 942 | REFUND NOT ALLOWED    | This merchant is not allowed to do refunds.  |
| 943 | REFUND DENIED         | Generic code. Action code was not acceptable.  |

|     |                                |   |
|-----|--------------------------------|---|
| 950 | REPORT FORMAT NOT SUPPORTED    | A valid report format was selected; however that report format is not available for the report that was selected. Contact Practice Management Vendor for additional assistance.   |
| 951 | NO DETAIL RECORDS FOR MERCHANT |   |
| 952 | GROUP/PROVIDER ID MISSING      | The communication record did not include a GROUP/PROVIDER ID Number. This is a mandatory field for ALL transactions, and the transaction cannot be processed. Contact Practice Management Vendor for additional assistance. |
| 953 | REPORT TYPE NOT AVAILABLE      | A valid report type was selected, however that report type has not been implemented on the RPM Server.  |
| 954 | NO TOTAL RECORDS FOR MERCHANT  |   |
| 955 | INVALID BLOCK REQUESTED        | An invalid block was requested on a report transaction.   |
| 956 | NO STATUS RECORD FOR MERCHANT  |   |
| 957 | INVALID RECEIPT REQUESTED      | The receipt that was requested is not valid. Contact the RPM Help Desk for assistance   |
| 958 | BAD STATUS                     | Record in database has a status showing a processing error – Call customer service.   |
| 982 | BATCH TOTALS MISMATCH          | Totals Mismatch.  |
| 987 | ISSUER UNAVAILABLE             | The card issuer did not respond to the authorization request.   |
| 988 | SWAP DOWN                      | This is a system problem. Try the transaction again.  |

|     |                                 |  |
|-----|---------------------------------|--|
| 989 | DATABASE ERROR                  | If error persists call customer service.   |
| 990 | CLOSE FAILED - COMM ERROR       | The “close batch” portion of the settlement transaction has failed. The settle function needs to be performed again. If error persists, refer to Error Code: 999.  |
| 991 | RELEASE FAILED - COMM ERROR     | The “release batch” portion of the settlement transaction has failed. The settle function needs to be performed again. If error persists, refer to Error Code: 999.  |
| 992 | TRANSACTION TIMEOUT             |  |
| 996 | TERMINAL ID NOT FOUND           |  |
| 997 | INVALID RESPONSE FROM PROCESSOR | The RPM Server received an invalid response from the Processor. Attempt the transaction again. If condition persists, contact the RPM Help Desk for assistance.  |
| 998 | MESSAGE NOT SUPPORTED           |  |
| 999 | COMMUNICATION FAILURE           | A problem was encountered in attempting to connect to the host system. Please check the following items: 1) Check the communications modem to be sure it is powered on; 2) ensure that communications modem is plugged into the telephone jack. The problem may be corrected by power-cycling the communications modem. If failure persists, contact the RPM Help Desk for assistance. |

# Abort Codes and Text

If a transaction processing or formatting error is detected before that transaction is sent to the host, it will be terminated and an abort code will be registered as the ActionCode. Abort codes begin with the letter A followed by two numeric digits. Codes marked A00 – A49 are Gateway Server abort codes. Codes marked A50 – A99 are client abort codes.

| Error Code  | Error Message                                       |
|---|---|
| Gateway abort codes indicate an error before data was transmitted to the host.        |   |
| A00   | GATEWAY ABORT – UNEXPECTED ERROR                    |
| A01   | GATEWAY ABORT – NO INPUT RECEIVED                   |
| A02   | GATEWAY ABORT – UNABLE TO DECODE INPUT              |
| A03   | GATEWAY ABORT – INVALID TRANSACTION REQUEST FORMAT  |
| A04   | GATEWAY ABORT – UNABLE TO PARSE TRANSACTION REQUEST |
| A05   | GATEWAY ABORT – UNABLE TO PARSE PAYLOAD             |
| A06   | GATEWAY ABORT – UNABLE TO CONNECT TO HOST           |
| Client-side abort codes indicate an error before data was transmitted to the gateway. |   |
| A50   | UNEXPECTED ERROR                                    |
| A53   | INVALID TRANSACTION ID                              |
| A80   | INVALID TRANSACTION TYPE                            |
| A81   | MISSING MERCHANT ID                                 |
| A82   | INVALID PRICE                                       |
| A83   | INVALID ABA NUMBER                                  |
| A84   | MISSING CHECKING ACCOUNT NUMBER                     |

|     |                                     |
|-----|-------------------------------------|
| A85 | MISSING CHECK NUMBER                |
| A86 | INVALID CHECK SURCHARGE             |
| A87 | INVALID CREDIT CARD EXPIRATION DATE |
| A88 | MISSING CREDIT CARD NUMBER          |
| A89 | INVALID CREDIT CARD NUMBER          |
| A90 | INVALID CVV2 NUMBER                 |

## Error Codes and Text

Error codes are generated when a transaction may have gone through, but because of an error we cannot verify it. Codes marked E00 – E49 are Gateway Server error codes. Codes marked E50 –E99 are client error codes.

| Error Code  | Error Message                                     |
|---|---|
| Gateway error codes that indicate an error after the data were transmitted to the host. |   |
| E00   | GATEWAY ERROR – UNEXPECTED ERROR                  |
| E01   | GATEWAY ERROR – NO INPUT RECEIVED                 |
| E02   | GATEWAY ERROR – UNABLE TO DECODE INPUT            |
| E03   | GATEWAY ERROR – UNABLE TO CONVERT RESPONSE TO XML |
| E04   | GATEWAY ERROR – ERROR IN XML RESPONSE             |
| E05   | GATEWAY ERROR – NO RESPONSE MESSAGE GENERATED     |
| E06   | GATEWAY ERROR – RECEIVE TIMEOUT                   |
| E07   | GATEWAY ERROR – INVALID RESPONSE FROM HOST        |
| E08   | GATEWAY ERROR – UNABLE TO PARSE RESPONSE          |

|   |   |
|---|---|
| E09   | GATEWAY ERROR – TRANSACTION ID MISMATCH   |
| E10   | GATEWAY ERROR – TRANSACTION ID NOT ON FILE  |
| Client-side error codes that indicate an error after data was transmitted to the gateway. |   |
| E50   | UNEXPECTED ERROR  |
| E51   | NO RESPONSE RECEIVED FROM JETPAY GATEWAY  |
| E52   | INCOMPLETE RESPONSE RECEIVED FROM JETPAY GATEWAY  |
| E53   | INVALID RESPONSE RECEIVED FROM JETPAY GATEWAY   |
| E54   | ACTION CODE NOT RETURNED FROM JETPAY GATEWAY  |
| E55   | TRANSACTION IDENTIFIER RETURNED FROM JETPAY GATEWAY DID NOT MATCH OUTBOUND TRANSACTION IDENTIFIER |

# Appendix C

## Test Host Response Code Generation

Because the test host does not contact card-issuing institutions, another method is required to create responses to populate the **ActionCode** field. The test host refers to the last two digits in the **Price** field to generate response codes. If the last two digits of the price of a transaction are listed in the following table, the test host will return the corresponding response code. All values other than the ones listed will return a response code of “000”, “APPROVED”.

| <b>Price</b> | <b>Response Code</b> | <b>Description</b> |
|--------------|----------------------|--------------------|
| .01          | 001                  | DECLINED           |
| .02          | 002                  | DECLINED           |
| .04          | 004                  | DECLINED           |
| .05          | 005                  | DECLINED           |
| .06          | 006                  | DECLINED           |
| .07          | 900 or 930           | DECLINED           |
| .14          | 014                  | DECLINED           |
| .15          | 015                  | DECLINED           |
| .41          | 041                  | DECLINED           |
| .43          | 043                  | DECLINED           |
| .51          | 051                  | DECLINED           |
| .52          | 930                  | DECLINED           |
| .54          | 054                  | DECLINED           |
| .57          | 057                  | DECLINED           |
| .62          | 062                  | DECLINED           |
| .90          | 992                  | DECLINED/TIMEOUT   |

# Appendix D

## Luhn Check Calculation

This self-checking scheme (referred to as the Luhn Mod-10 Method) is an international standard for validating card account numbers (ISO 2894/ANSI 4.13). Such account numbers, which cannot exceed 19 digits including the check digit, are assigned, embossed, and encoded to include a single check digit in the rightmost position.

The check digit is calculated as follows:

1. Beginning on the right with the digit which immediately precedes the check digit and moving toward the left, double every other digit. After doubling each selected digit, if the result is 10 or greater, add the two digits together to arrive at a single-digit result.
2. All individual resulting digits (plus those skipped above) are then added together.
3. This sum is then subtracted from the lowest multiple of 10 which is equal to or greater than the sum and the single-digit result is the check digit.

For example, in the case of the following 15-digit account number 7951-0287-9015-54?

$$4 \times 2 = 8$$

$$5 \times 1 = 5$$

$$5 \times 2 = 10 : 1 + 0 = 1$$

$$1 \times 1 = 1$$

$$0 \times 2 = 0$$

$$9 \times 1 = 9$$

$$7 \times 2 = 14 : 1 + 4 = 5$$

$$8 \times 1 = 8$$

$$2 \times 2 = 4$$

$$0 \times 1 = 0$$

$$1 \times 2 = 2$$

$$5 \times 1 = 5$$

$$9 \times 2 = 18 : 1 + 8 = 9$$

$$7 \times 1 = 7$$

$$\text{Sum} = 64$$

Subtracted from 70

Resulting check digit: 6

# Appendix E

## Valid Card BIN Ranges

The standards organizations determine and publish the account numbering schemes by passing ranges of numbers to various card issuers to facilitate the identification of the type of card represented by any given Cardholder Account Number. In addition, the issuers themselves follow certain conventions that allow for further edit checks on the validity of a number. Listed below are the currently known BIN ranges for most popular card issuers. For information on identifying any other card types, check with the respective card issuers.

| <b>Card Type</b>              | <b>No. of Digits</b> | <b>Bin Range</b>                            |
|-------------------------------|----------------------|---|
| MasterCard                    | 16                   | 5000 – 5999                                 |
| VISA                          | 13 or 16             | 4000 – 4999                                 |
| American Express              | 15                   | 3400 – 3499 or<br>3700 – 3799               |
| Diners Club<br>/Carte Blanche | 14                   | 3000 – 3059, 3600 – 3699,<br>or 3800 – 3899 |
| Discover                      | 16                   | 6011  |
| JCB                           | 16                   | 3528 – 3589                                 |