PayTrace API

Application Programmer Developer's Guide

XML Version

April 2009

1. PayTrace API Overview

The PayTrace API (Application Programmer Interface) is a powerful software solution that allows any software developer to integrate the power of the PayTrace Payment Gateway into their proprietary software. Software developers may use the PayTrace API to add payment-processing functionality into their software through the seamless integration of XML (eXtensible Markup Language).

Through the power and efficiency of the PayTrace API, your software applications may process electronic payments in real-time and receive payment authorizations within 3 to 6 seconds. The PayTrace API is also built to provide real-time shipping quotes, store customer and transaction profiles, process recurring payments and email receipts.

2. Obtaining and Installing the PayTrace API

Because the PayTrace API XML version uses XML, the PayTrace API does NOT need to be installed or registered on the web server or client computer that is running the software application.

3. Referencing the PayTrace API and Formatting a Request

Once the PayTrace API XML version is declared in your code, request strings may be sent to the PayTrace Payment Gateway, and response may be retrieved and parsed in your code.

3.1 Declaring and Initializing the PayTrace API

The following code example illustrates how the PayTrace API may be referenced in you software's code. All examples are provided in Visual Basic (VB) Script 5.0

```
'Declare the connection tools
Dim objXML, strRequest, strResponse

'Create the XML object
set objXML=createobject("MSXML2.XMLHTTP")

'open the XML object and point it to the PayTrace secure servers
objXML.Open "POST", "https://paytrace.com/api/default.pay",  false

'set the Request Header of the XML object to a URL encoded form
objXML.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"

'-----------------------------…continued to Section 4…----------------- ----------
```

Image 3.1a

3.2 Formatting a PayTrace API Request String

The PayTrace API accepts requests strings that formatted in name / value pairs that are separated with tildes (~) and delimited with pipes (|). For example, the string "name1~value1|name2~value2|" is correctly formatted to be sent through the PayTrace API. However, the PayTrace API only accepts specific names and each value must meet the expected criteria found in section *3.3 PayTrace API Name / Value Pairs Data Definitions*.

Please note that the portion of the request following the "parmList=" designation should be URL encoded. Many programming languages have built in URLEncode functions such as Server.URLEncode() in ASP and URLEncode() in PHP.

3.3 PayTrace API Name / Value Pairs Data Definitions

The PayTrace API accepts request strings that are formatted in name / value pairs. Also, the PayTrace API returns response strings that are formatted in name / value pairs.

3.3.1 PayTrace Request Name / Value Pairs Data Definitions

Name	Description	Format	Length
METHOD	Function that you are requesting PayTrace perform.	alphabetic	1-50
UN	Your PayTrace user name is required to authenticate your request.	alpha-numeric	7-50
PSWD	Your PayTrace password is required to authenticate your request.*	alpha-numeric*	7-50*
CUSTID	Unique identifier for a customer profile. Each customer must have their own unique ID.	alpha-numeric	1-25
CUSTPSWD	Password that customer uses to log into customer profile in shopping cart. Only required if you are using the PayTrace shopping cart.	alpha-numeric	7-25
BNAME	Name that appears of the credit card.	alphabetic	1-50
BADDRESS	Address that the credit card statement is delivered.	alpha-numeric	1-50
BADDRESS2	Second line of the address the credit card statement is delivered.	alpha-numeric	1-50
BCITY	City that the credit card statement is delivered.	alphabetic	1-50
BSTATE	State that the credit card statement is delivered.	alphabetic	2
BZIP	Zip code that the credit card statement is delivered.	numeric	5, 9
SNAME	Name of the person where the product is delivered.	alphabetic	1-50
SADDRESS	Address where the product is delivered.	alpha-numeric	1-50
SADDRESS2	Second line of the address where the product is delivered.	alpha-numeric	1-50
SCITY	City where the product is delivered.	alphabetic	1-50
SCOUNTY	County where the product is delivered.	alphabetic	1-50
SSTATE	State where the product is delivered.	alphabetic	2
SZIP	Zip code where the product is delivered.	numeric	5, 9
EMAIL	Customer's email address where the sales receipt may be sent.	alpha-numeric	7-50
FAX	Customer's fax number (i.e. (555)555-5555, 555-555-5555, or 5555555555).	alpha-numeric	10, 12, 13
PHONE	Customer's phone number (i.e. (555)555-5555, 555-555-5555, or 5555555555).	alpha-numeric	10, 12, 13
TRANXTYPE	The transaction type is the type of transaction you wish to process if the METHOD is set to ProcessTranx. TRANXTYPE must be set to one of the following: Sale, Authorization, Refund, Void, Capture, or Force.	alphabetic	4, 5, 6, 7, 13

SWIPE	The value of the magnetic stripe on the back of the credit card as recorded from a magnetic card reader. PayTrace strongly recommends that software providers avoid storing the swiped value. Because the first value in the magnetic stripe is a (%) symbol, we strongly recommend URL encoding the SWIPE value before posting it to PayTrace. Please note that PayTrace supports encrypted card readers. However, most encrypted card readers include pipe (|) symbols in the magstripe value. It is imperative that you replace these pipe symbols with "***" to ensure that the API is able to parse your request.	alpha-numeric	255+
CC	Customer's credit card number must be a valid credit card number that your PayTrace account is set up to accept.	numeric	15, 16, 19
EXPMNTH	Expiration month must be the two-digit month of the credit cards expiration date.	numeric	2
EXPYR	Expiration year must be the two digit year of the credit cards expiration date.	numeric	2
CSC	CSC is the 3 or 4 digit code found on the signature line of the credit card. CSC is found on the front of Amex cards.	numeric	3-4
INVOICE	Invoice is the identifier for this transaction in your accounting or inventory management system.	alpha-numeric	1-50
CUSTREF	Customer reference ID is only used for transactions that are identified as corporate or purchasing credit cards. The customer reference ID is an identifier that your customer may ask you to provide in order to reference the transaction to their credit card statement.	alpha-numeric	1-17
AMOUNT	Dollar amount of the transaction. Must be a positive number up to two decimal places.	numeric	1-12
TAX	Portion of the original transaction amount that is tax. Must be a number that reports the tax amount of the transaction. Use -1 if the transaction is tax exempt	numeric	1-12
DESCRIPTION	Optional text describing the transaction, products, customers, or other attributes of the transaction.	alpha-numeric	1-255
APPROVAL	Approval code for the forced sale is only required and used if TranxType is set to 'Force'.	alpha-numeric	6
SDATE	Start date is used for export functions to indicate when to start searching for items to export. Must be a valid date formatted as MM/DD/YYYY.	Date	12
EDATE	End date is used for export functions to indicate when to end searching for items to export. Must be a valid date formatted as MM/DD/YYYY.	Date	12
TERMS	Must be to 'Y' in order to process any methods through the PayTrace API. Setting this variable to 'Y' indicates that you agree to the PayTrace terms and conditions found at https://paytrace.com/terms.html	alphabetic	1

USER	User is the user name of the PayTrace user who created or processed the customer or transaction you are trying to export. This variable is a search criterion for the export methods.	alpha-numeric	1-50
SOURCEZIP	Zip code that the package will be sent from.	numeric	5,9
SHIPPERS	Comma separated string of shipping service providers you would like shipping quotes from. String may contain USPS, FEDEX, or DHL in any order or combination as long as the shipping service providers are separated by commas without spaces.	alphabetic	3-19
WEIGHT	Weight of the package that is being shipped. Weight must be provided in pounds and my have up to two decimals. Weight must be less than 70 pounds	numeric	1-5
TEST	The TEST attribute may be used with any of the transaction types (TranxType) of the ProcessTranx method. The value of the TEST attribute may only be a "Y". Transactions processed with a TEST value of "Y" will be processed as test transactions with standardized test responses. Test transactions will not place a hold on the customer's credit card.	alphabetic	1
ORDERID	Developer's identifier for an order that is placed through the PayTrace API Secure Checkout	alpha-numeric	1-50
APPROVEURL	Optional URL that the customer will have the option of selecting if their transaction is approved via the PayTrace API Secure Checkout	alpha-numeric	1-255
DECLINEURL	Optional URL that the customer will have the option of selecting if their transaction is declined via the PayTrace API Secure Checkout	alpha-numeric	1-255
FORCEEMAIL	Setting in the PayTrace API Secure Checkout that may be set to 'Y' if the customer's email address is required.	alphabetic	1
FORCEADDRESS	Setting in the PayTrace API Secure Checkout that may be set to 'Y' if the customer's complete billing address is required.	alphabetic	1
FORCECSC	Setting in the PayTrace API Secure Checkout that may be set to 'Y' if the customer's CSC is required.	alphabetic	1
RECURID	The ID of the Recurring Transaction that is being updated.	numeric	1-50
FREQUENCY	The billing cycle of the recurring transaction must be 1 for annually, 2 for quarterly, 3 for monthly, 4 for bi-weekly, 5 for weekly, or 6 for daily.	numeric	1
TOTALCOUNT	The total number of times the recurring transaction should be processed. Use 999 if the recurring transaction should be processed indefinitely.	numeric	1-3
START	Date the recurring transaction should be processed for the first time.	Date	12
NEXT	Next date the updated recurring transaction should be processed.	Date	12
CUSTRECEIPT	Defaulted to N, the customer receipt must be set to Y if a receipt should be emailed to the customer each time the recurring transaction is processed.	alphabetic	1
SOURCESTATE	State that the package will be sent from.	alphabetic	2
SCOUNTRY	Country that the package will be delivered to.	alphabetic	2

NEWPSWD	Your new PayTrace password when updating user password through the UpdatePassword method.	alpha-numeric	7-50
NEWPSWD2	Confirmation of you new PayTrace password when updating user password through the UpdatePassword method.	alpha-numeric	7-50
NTAX	Portion of the original transaction amount that is national tax. Generally only applicable to orders shipped to countries with a national or value added tax.	Numeric	1-12
MERCHANTTAXID	Merchant's tax identifier used for tax reporting purposes.	alpha-numeric	1-20
CUSTOMERTAXID	Customer's tax identifier used for tax reporting purposes	alpha-numeric	1-13
CCODE	Commodity code that generally applies to each product included in the order. Commodity codes are generally assigned by your merchant service provider.	alpha-numeric	1-4
DISCOUNT	Discount value should represent the amount discounted from the original transaction amount	Numeric	1-12
FREIGHT	Freight value should represent the portion of the transaction amount that was generated from shipping costs.	Numeric	1-12
DUTY	Duty should represent any costs associated with shipping through a country's customs.	Numeric	1-12
ADDTAX	Any tax generated from freight or other services associated with the transaction.	Numeric	1-12
ADDTAXRATE	Rate at which additional tax was assessed.	Numeric	1-4
ADDTAXIND	A flag used to indicate where additional tax was included in this transaction. Set to Y if additional tax was included and N if no additional tax was applied.	alphabetic	1
CCODELI	The complete commodity code unique to the product referenced in this specific line item record. Commodity codes are generally assigned by your merchant service provider	alpha-numeric	1-12
PRODUCTID	Your unique identifier for the product.	alpha-numeric	1-12
QUANTITY	Item count of the product in this order	numeric	1-12
MEASURE	Unit of measure applied to the product and its quantity. For example, LBS/LITERS, OUNCES, etc.	alpha-numeric	1-12
UNITCOST	Product amount per quantity.	numeric	1-12
ADDTAXLI	Additional tax amount applied to the transaction applicable to this line item record.	numeric	1-12
ADDTAXRATELI	Rate at which additional tax was calculated in reference to this specific line item record.	numeric	1-4
DISCOUNTLI	Discount amount applied to the transaction amount in reference to this line item record.	numeric	1-12
AMOUNTLI	Total amount included in the transaction amount generated from this line item record.	numeric	1-12
ADDTAXINDLI	Flag used to indicate whether additional tax was applied to the transaction amount in reference to this specific line item.	alpha-numeric	1-4
DISCOUNTIND	Flag used to indicate whether discount was applied to the transaction amount in reference to this specific line item record.	alphabetic	1
NETGROSSIND	Flag used to indicate whether the line item amount is	alphabetic	1

	net or gross to specify whether the line item amount includes tax.		
DCIND	Flag used to determine whether the line item amount was a debit or a credit to the customer. Generally always a debit or a factor that increased the transaction amount.	alphabetic	1
DISCOUNTRATE	Rate at which discount was applied to the transaction in reference to this specific line item.	alpha-numeric	1-5
TRANXID	A unique identifier for each transaction in the PayTrace system. This value is returned in the TRANSACTIONID parameter of an API response and will consequently be included in requests to email receipts, void transactions, add level 3 data, etc.	Numeric	1-8
CASHADVANCE	When set to Y, this attribute causes a Sale transaction to be processed as a cash advance where cash is given to the customer as opposed to a product or service. Please note that Cash Advances may only be processed on accounts that are set up on the TSYS/Vital network and are configured to process Cash Advances. Also, only swiped/card present Sales may include the CashAdvance parameter.	Alphabetic	1
PHOTOID	Only used when processing Cash Advances. This required field may be the card holder's drivers license number or other form of photo ID.	Alpha-numeric	1-20
IDEXP	Only used when processing Cash Advances. This required field is the expiration date of the card holder's photo ID. MM/DD/YYYY	Date	10
LAST4	Only used when processing Cash Advances. This required field is the last 4 digits of the card number as it appears on the face of the card.	Numeric	4
RETURNID	If set to Y will return the value of ORIGINALID in the response.	Alphabetic	1
ORIGINALID	Unique identifier of the request from your system that will be returned in the response if RETURNID is set to Y	Alpha-numeric	100
BATCHNUMBER	Number of the batch of transactions you wish to export via an ExportBatch request	Numeric	1-3
POSTURL	Optional URL with any request where the response/error generated from the request may be sent in addition to the comptuer that originated the request.	Alpha-Numeric	1-255
DDA	Checking account number for processing check transactions or managing customer profiles.	Numeric	1-20
TR	Transit routing number for processing check transactions or managing customer profiles.	Numeric	9
RECURTYPE	Default value is C which represents credit card number. Alternative is A which represents an ACH/check transaction.	Alphabetic	1
CHECKTYPE	The transaction type is the type of transaction you wish to process if the METHOD is set to ProcessCheck. CHECKTYPE must be set to one of the following: Sale, Hold, Refund, Fund, or Void.	alphabetic	4, 6
CHECKID	A unique identifier for each transaction in the PayTrace system. This value is returned in the	Numeric	1-8

	CHECKIDENTIFIER parameter of an API response and will consequently be included in requests to email receipts, manage checks, etc.		
RETURNBIN	If set to Y, card numbers from ExportTranx and ExportCustomers requests will include the first 6 and last 4 digits of the card number	Alphabetic	1

*Reference section _7. Password Management_

3.3.2 PayTrace Response Name / Value Pairs Data Definitions

Name	Description	Format	Length
RESPONSE	Sentence or two that confirms the method that was requested.	alpha-numeric	1-255
CUSTOMERID	ID assigned by PayTrace to each customer at the time the customer profile is created. CustomerID is returned with a successful call to CreateCustomer or UpdateCustomer.	alpha-numeric	1-255
TRANSACTIONRECORD	Formatted transaction record returned when a successful call to ExportTranx method is requested.	alpha-numeric	1-25
CUSTOMERRECORD	Formatted customer record returned when a successful call to ExportCustomers method is requested.	alpha-numeric	...
TRANSACTIONID	ID assigned by PayTrace to each transaction at the time the transaction is processed. TransactionID is returned with a successful call to ProcessTranx.	alpha-numeric	1-16
APPCODE	Approval code is generated by credit card issuer and returned when a successful call to ProcessTranx is requested.	alpha-numeric	6
APPMSG	Approval message is the textual response from the credit card issuer that is returned when a successful call to ProcessTranx is requested.	alpha-numeric	1-255
AVSRESPONSE	The address verification system response is generated by the credit card issuer when a successful call to ProcessTranx is requested. AVS compares the billing address and billing zip code provided with the ProcessTranx request to the address where the customer's credit card statement is delivered. See Appendix B for possible AVS responses	alpha-numeric	1-255
CSCRESPONSE	The card security code response is generated by the credit card issuer when a successful call to the ProcessTranx is requested. The CSC provided with the ProcessTranx request. is compared to the CSC assigned to the credit card. See Appendix B for possible AVS responses	alpha-numeric	1-255
STATUS	Status is returned with each transaction in the transaction list that is returned from a successful call to the ExportTranx method. If the status contains the letters "GB" then the transaction has been settled, and the batch number will be appended to the status. If the status contains the letters "RB" the transaction was part of a rejected batch, and the batch number will be appended to the status. If the status contains the letters "QD" the transaction was part of a duplicate batch, and the batch number will be appended to the status. If the is "Y" then the transaction will be settled that evening. If the status is "N" then the transaction was voided or declined.	alpha-numeric	1-10
WHEN	Date and Time returned with the TransactionRECORD and the CustomerRECORD formatted a general date (i.e. MM/DD/YYYY HH:MM:SS). When represents the date and time the transaction was first processed or the customer profile was created.	alpha-numeric	19

IP	IP address of the computer that originally requested the customer profile or transaction be created or processed formatted as a standard IP address (I.e. 111.111.111.111). IP address is returned with a successful call to ExportCustomers or ExportTranx.	alpha-numeric	7-15
ERROR	PayTrace validates each name / value pair it receives. If any errors or inconsistencies in this data or the request, PayTrace will return an error. Each request may return multiple errors.	alpha-numeric	1-255
SHIPPINGRECORD	Shipping records are returned when a successful call to CalculateShipping is requested.	alpha-numeric	...
SHIPPINGCOMPANY	The name of the shipping company that quoted the price (i.e. DHL, UPS, USPS, FEDEX).	alphabetic	3-5
SHIPPINGMETHOD	The method of shipment that was quoted (i.e. Next Day, Priority, Ground, etc.).	alpha-numeric	1-255
SHIPPINGRATE	Cost to use the specified shipping service provider and method formatted in U.S. dollars as provided by the shipping service provider.	Numeric	1-5
AUTHKEY	Authorization key is returned with a successful request to validate an order through the PayTrace API Secure Checkout.	alpha-numeric	1-255
TRANXCOUNT	Transaction count is returned with a successful request to settle transactions. This value is the total number of transactions that were included in the batch.	Numeric	1-3
NETAMOUNT	Net amount is returned with a successful request to settle transactions. This value is the net amount (sales minus refunds) of the batch that was initiated.	Numeric	1-15
BATCHNUM	Batch number is returned with a successful request to settle transactions. This value is the sequential number assigned to the batch that was initiated.	Numeric	1-3
VISASALESCOUNT	Total number of Visa sales that were settled in the exported batch. Similar values will be returned for all applicable card types, i.e. MasterCardSalesCount, AmexSalesCount, DiscoverSalesCount, etc.	Numeric	1-3
VISASALESAMOUNT	Total sum of Visa sales that were settled in the exported batch. Similar values will be returned for all applicable card types, i.e. MasterCardSalesAmount, AmexSalesAmount, DiscoverSalesCount, etc.	Numeric	1-15
VISAREFUNDCOUNT	Total number of Visa refunds that were settled in the exported batch. Similar values will be returned for all applicable card types, i.e. MasterCardRefundCount, AmexRefundCount, DiscoverRefundCount, etc.	Numeric	1-3
VISAREFUNDAMOUNT	Total sum of Visa refunds that were settled in the exported batch. Similar values will be returned for all applicable card types, i.e. MasterCardRefundAmount, AmexRefundAmount, DiscoverRefundCount, etc.	Numeric	1-15
CHECKIDENTIFIER	ID assigned by PayTrace to each check at the time the check is processed. CHECKIDENTIFIER is returned with a successful call to ProcessCheck.	alpha-numeric	1-16
ACHCODE	Flag that is returned for checks processed through a real-time check processor. 0/zero indicates that the check was accepted.	Numeric	6
ACHMSG	Message returned from real-time check processor	alpha-numeric	1-255

4. Formatting and Sending PayTrace API Requests

The following example requests may be processed through the Demo PayTrace account. The example responses for these requests may be viewed in section *5 Receiving and Parsing PayTrace API Response*.

4.1 Processing Transactions through the PayTrace API

All of the same credit card transactions that may be processed through a standard credit terminal and the PayTrace Virtual Terminal may be processed through the PayTrace API. Customer billing information may be referenced to an existing customer profile, swiped through a card reader, or key entered through the PayTrace API.

Please note that the PayTrace Secure Checkout page may be used to process Authorizations, Sales, and Forced Sales for those developers who wish to use PayTrace Secure Checkout as a means for their customers to provide their billing information. Please refer to section 6. Using PayTrace's Secure Checkout.

Any transaction may be processed through the PayTrace API as a test transaction by setting the "TEST" attribute to "Y". Test transactions return standardized responses in the same format as live transactions, but authorizations are not obtained or placed on the credit card account.

4.1.1.a Required Name / Value Pairs of a Sale Request

Processing a sale through the PayTrace API may be accomplished by providing a new customer's swiped credit card information, a new customer's key entered credit card information, or the customer ID of an existing customer.

Processing a sale with a new customer's swiped credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, SWIPE

Processing a sale with a new customer's key entered credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CC, EXPMNTH, EXPYR

Processing a sale with an existing customer's customer ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CUSTID

4.1.1.b Optional Name / Value Pairs of a Sale Request

Several optional name / value pairs may be sent with a sale request in order to minimize the risk of the transaction, reduce transaction costs, and enhance the reporting value of the receipt and the transaction. The following name / value pairs may be provided in the sale request:

BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, CSC, INVOICE, DESCRIPTION, TAX, CUSTREF

4.1.1.c Example of a Sale Request

```
'-------------------------------...continued from Image 3.1a...------------------------------

'format the request string to process a sale for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Sale|CC~4012881888818888|EXPMNTH~12|EXPYR~05|"
strRequest = strRequest & "AMOUNT~1.00|CSC~999|BADDRESS~1234|BZIP~83852|INVOICE~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.1.1...------------------------------
```

<div align="center">Image 4.1.1</div>

4.1.2.a Required Name / Value Pairs of an Authorization Request

Processing an authorization through the PayTrace API will request authorization for specified amount. However, the approved funds will not be charged or funded until the transaction is captured and settled.

The required name / value pairs for processing an authorization are the same as processing a sale as referenced in section *4.1.1.a Required Name / Value Pairs of a Sale Request*

4.1.2.b Optional Name / Value Pairs of an Authorization Request

The optional name / value pairs for processing an authorization are the same as processing a sale as referenced in section *4.1.1.b Optional Name / Value Pairs of a Sale Request*

4.1.2.c Example of an Authorization Request

```
'-------------------------------…continued from Image 3.1a…-----------------------------

'format the request string to process an authorization for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Authorization|CC~4012881888818888|EXPMNTH~12|"
strRequest = strRequest & "EXPYR~05|AMOUNT~1.00|CSC~999|BADDRESS~1234|BZIP~83852|"
strRequest = strRequest & "INVOICE~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------…continued to Image 5.1.1…-----------------------------
```

Image 4.1.2

4.1.3.a Required Name / Value Pairs of a Refund Request

Processing a refund through the PayTrace API may be accomplished by providing a new customer's swiped credit card information, providing a new customer's key entered credit card information, providing the customer ID of an existing customer, or providing the transaction ID of the original transaction that should be refunded.

Processing a refund with a new customer's swiped credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, SWIPE

Processing a refund with a new customer's key entered credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CC, EXPMNTH, EXPYR

Processing a refund with an existing customer's customer ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CUSTID

Processing a refund with an existing transaction ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, TRANXID

4.1.3.b Optional Name / Value Pairs of a Refund Request

Several optional name / value pairs may be sent with a refund request in order to enhance the reporting value of the receipt and the transaction. The following name / value pairs may be provided in the refund request:

BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, CSC, INVOICE, DESCRIPTION, TAX, CUSTREF, AMOUNT

4.1.3.c Example of a Refund Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to process a refund for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Refund|CC~4012881888818888|EXPMNTH~12|"
strRequest = strRequest & "EXPYR~05|AMOUNT~1.00|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing


'--------------------------------...continued to Image 5.1.3...----------------------------
```

Image 4.1.3

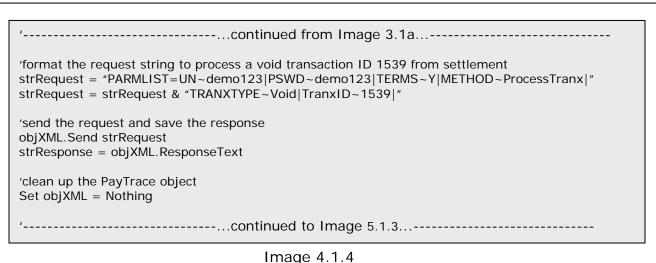## 4.1.4.a Required Name / Value Pairs of a Void Request

Processing a void through the PayTrace API may only be accomplished by providing the transaction ID of the unsettled transaction that should be voided.

UN, PSWD, TERMS, METHOD, TRANXTYPE, TRANXID

4.1.4.b Optional Name / Value Pairs of a Void Request

Since voiding a transaction is just removing the transaction from settlement, there are no optional name / value pairs in for void requests.

4.1.4.c Example of a Void Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to process a void transaction ID 1539 from settlement
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Void|TranxID~1539|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.1.3...-----------------------------
```
Image 4.1.4

4.1.5.a Required Name / Value Pairs of a Forced Sale Request

Processing a forced sale through the PayTrace API may be accomplished by providing a new customer's swiped credit card information, providing a new customer's key entered credit card information, or providing the customer ID of an existing customer. A forced sale is a sale where the approval code for the purchase amount has been obtained outside of the PayTrace Payment Gateway or has been voided from the settlement record.

Processing a force with a new customer's key entered credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CC, EXPMNTH, EXPYR, APPROVAL

Processing a force with a new customer's swiped credit card number requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, SWIPE, APPROVAL

Processing a force with an existing customer's customer ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CUSTID, APPROVAL

4.1.5.b Optional Name / Value Pairs of a Forced Sale Request

Several optional name / value pairs may be sent with a forced sales request in order to minimize the risk of the transaction, reduce transaction costs, and enhance the reporting value of the receipt and the transaction. The following name / value pairs may be provided in the force request:

BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, CSC, INVOICE, DESCRIPTION, TAX, CUSTREF

4.1.5.c Example of a Force Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to process a forced sale for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Force|CC~4012881888818888|EXPMNTH~12|"
strRequest = strRequest & "EXPYR~05|AMOUNT~1.00|APPROVAL~123456|INVOICE~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.1.3...----------------------------
```

Image 4.1.5

4.1.6.a Required Name / Value Pairs of a Capture Request

Capturing a transaction updates an approved authorization to a pending settlement status that will initiate a transfer of funds. Processing a capture through the PayTrace API may only be accomplished by providing the transaction ID of the unsettled transaction that should be settled.

UN, PSWD, TERMS, METHOD, TRANXTYPE, TRANXID

4.1.6.b Optional Name / Value Pairs of a Capture Request

The only optional name value pair when capturing a transaction is AMOUNT which must be less than or equal to the original authorization amount.

4.1.6.c Example of a Capture Request

```
'-------------------------------...continued from Image 3.1a...----------------------------

'format the request string to capture transaction ID 1539 for settlement
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|"
strRequest = strRequest & "TRANXTYPE~Capture|TRANXID~1539|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.1.3...-----------------------------
```

Image 4.1.6

4.1.7.a Required Name / Value Pairs of a CashAdvance Request

Processing a Cash Advance transaction is similar to processing a Sale; however, Cash Advances are special transactions that result in cash disbursements to the card holder. Consequently, additional information is required to process Cash Advances. Cash Advances should always be swiped unless your card reader is not able to reader the card's magnetic stripe. Additionally, your PayTrace account must be specially configured to process this type of transaction.

Please note that Cash Advances may also be processed as forced transactions by setting the TranxType to FORCE and including a valid APPROVAL value, all other fields remain the same. Forced Cash Advance transactions should be also be swiped unless your card reader is not able to read the card's magnetic stripe.

Processing a cash advance with a swiped credit card number requires the following name / value pairs.

UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, SWIPE, CASHADVANCE, PHOTOID, IDEXP, LAST4, BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP

Processing a cash advance with a key entered credit card number requires the following name / value pairs.
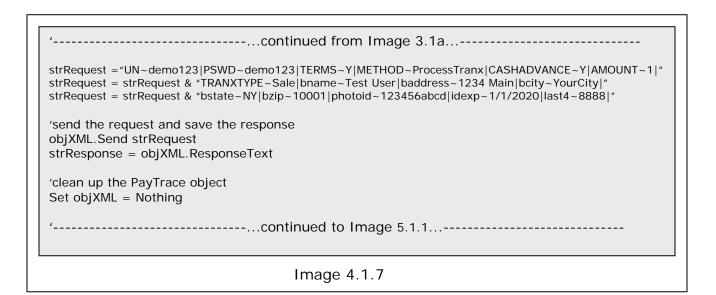UN, PSWD, TERMS, METHOD, TRANXTYPE, AMOUNT, CC, EXPMNTH, EXPYR, CASHADVANCE, PHOTOID, IDEXP, LAST4, BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP

4.1.7.b Optional Name / Value Pairs of a CashAdvance Request

Several optional name / value pairs may be sent with a Cash Advance request in order to minimize the risk of the transaction, reduce transaction costs, and enhance the reporting value of the receipt and the transaction. The following name / value pairs may be provided in the sale request.
SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, CSC, INVOICE, DESCRIPTION, TAX, CUSTREF

4.1.7.c Example of a CashAdvance Request

```
'---------------------------...continued from Image 3.1a...----------------------------

strRequest ="UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|CASHADVANCE~Y|AMOUNT~1|"
strRequest = strRequest & "TRANXTYPE~Sale|bname~Test User|baddress~1234 Main|bcity~YourCity|"
strRequest = strRequest & "bstate~NY|bzip~10001|photoid~123456abcd|idexp~1/1/2020|last4~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'---------------------------...continued to Image 5.1.1...----------------------------
```

Image 4.1.7

4.2 Managing Customer Profiles through the PayTrace API

The PayTrace API provides an interface for customer profiles to be created, updated, and deleted. Since the customer billing information is saved on the PayTrace secured servers and accessed by authenticated users at any time, your software does not need to store sensitive information on your server or the client computer.

4.2.1.a Required Name / Value Pairs for Creating a Customer

Processing a successful request to create a customer through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CUSTID, BNAME, CC, EXPMNTH, EXPYR

4.2.1.b Optional Name / Value Pairs of a CreateCustomer Request

Several optional name / value pairs may be sent with requests to create a customer profile in order to enhance the reporting and stored value of the customer profile for later use. The following name / value pairs may be provided in the create customer request:

BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, PHONE, FAX, CUSTPSWD, DDA, TR

4.2.1.c Example of a CreateCustomer Request

```
'-------------------------------...continued from Image 3.1a...------------------------------

'format the request string to create customer ID johndoe
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~CreateCustomer|"
strRequest = strRequest & "BNAME~John Doe|BADDRESS~1234 Main|BCITY~Spokane|BSTATE~WA|"
strRequest = strRequest & "BZIP~99201|SNAME~John Doe|SADDRESS~5678 First|SCITY~Portland|"
strRequest = strRequest & "SSTATE~OR|SZIP~97201|CUSTID~johndoe|CC~4012881888818888|"
strRequest = strRequest & "EXPMNTH~12|EXPYR~05|EMAIL~support@paytrace.com|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.2...------------------------------
```

Image 4.2.1

4.2.2.a Required Name / Value Pairs for Updating a Customer

Processing a successful request to update an existing customer through the
PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CUSTID

4.2.2.b Optional Name / Value Pairs of an UpdateCustomer Request

Several optional name / value pairs may be sent with requests to update a
customer profile in order to enhance the reporting and stored value of the customer
profile for later use. The following name / value pairs may be provided in the
update customer request:

BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2,
SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, PHONE, FAX, CC, EXPMNTH, EXPYR,
CUSTPSWD, DDA, TR

![PayTrace Payment Gateway logo]

4.2.2.c Example of an UpdateCustomer Request

```
'-------------------------------...continued from Image 3.1a...------------------------------

'format the request string to update customer ID johndoe
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~UpdateCustomer|"
strRequest = strRequest & "BNAME~John Doe|BADDRESS~1234 Main|BCITY~Spokane|BSTATE~WA|"
strRequest = strRequest & "BZIP~99201|SNAME~John Doe|SADDRESS~5678 First|SCITY~Portland|"
strRequest = strRequest & "SSTATE~OR|SZIP~97201|CUSTID~johndoe|CC~4012881888818888|"
strRequest = strRequest & "EXPMNTH~12|EXPYR~05|EMAIL~support@paytrace.com|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'------------------------------...continued to Image 5.2...------------------------------
```

Image 4.2.2

4.2.3.a Required Name / Value Pairs for Deleting a Customer

Processing a successful request to delete an existing customer through the
PayTrace API requires the following name / value pairs:

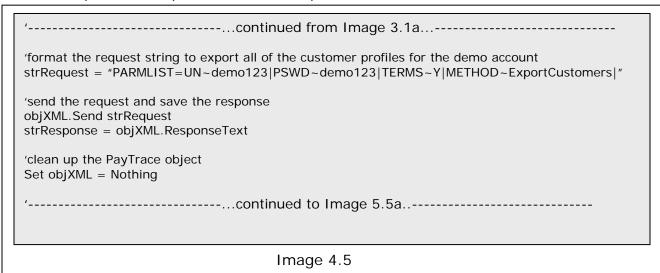UN, PSWD, TERMS, METHOD, CUSTID

4.2.3.b Optional Name / Value Pairs of a DeleteCustomer Request

Since the customer profile is being deleted, no optional name / value pairs are
applicable for delete customer requests.

4.2.3.c Example of a DeleteCustomer Request

```
'-------------------------------…continued from Image 3.1a…----------------------------

'format the request string to delete customer ID johndoe
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~DeleteCustomer|"
strRequest = strRequest & "CUSTID~johndoe|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------…continued to Image 5.2…-----------------------------
```

Image 4.2.3

4.3 Emailing Receipts

Through the PayTrace API, requests may be made to have transaction receipts emailed to a specific email address for any transaction processed through the PayTrace Payment Gateway.

4.3.a Required Name / Value Pairs for Emailing a Receipt

Processing a successful request to email a transaction receipt through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXID, EMAIL

Processing a successful request to email a check receipt through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CHECKID, EMAIL

4.3.b Optional Name / Value Pairs of an EmailReceipt Request

No optional name / value pairs are applicable for requests to email receipts.

4.3.c Example of an EmailReceipt Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to email a receipt  for transaction ID 1498 to support@paytrace.com
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~EmailReceipt|"
strRequest = strRequest & "EMAIL~support@paytrace.com|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.3...-----------------------------
```

Image 4.3

4.4 Exporting Transaction Information

Transaction information may be exported through the PayTrace API at any time allowing transaction records to be viewed without being stored on the client computer.

4.4.a Required Name / Value Pairs for Exporting Transactions

Processing a successful request to export transactions through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, SDATE, EDATE, METHOD

4.4.b Optional Name / Value Pairs of an ExportTranx Request

In order to reduce the number of exported transactions and provide more detailed searching, the PayTrace API will allow the following optional name / value pairs for export transaction requests:

TRANXTYPE, CUSTID, USER, RETURNBIN

(Please note the TRANXTYPE name may also include the values "SETTLED", "PENDING", and "DECLINED" in addition to the values in section 3.3.1)

4.4.c Example of an ExportTranx Request

```
'--------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to export all of the transactions for the demo account processed in the first
'week of December 2004
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ExportTranx|"
strRequest = strRequest & "SDATE~12/01/2004|EDATE~12/07/2004|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.4a..-----------------------------
```
Image 4.4

4.5 Exporting Customer Profiles

Customer profile information may be exported through the PayTrace API at any time allowing customer information to be viewed without being stored on the client computer.

4.5.a Required Name / Value Pairs for Exporting Customers

Processing a successful request to export customers through the PayTrace API requires the following name / value pairs:
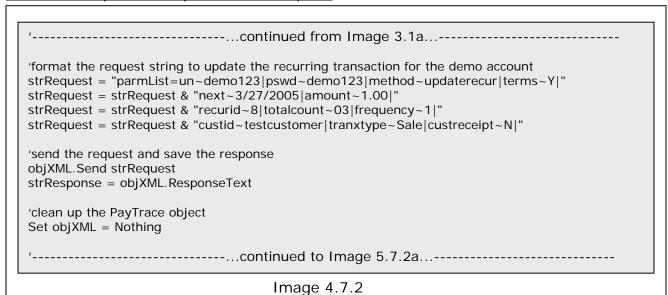
UN, PSWD, TERMS, METHOD
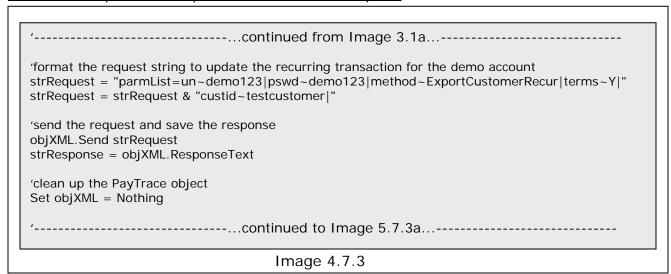
4.5.b Optional Name / Value Pairs of an ExportCustomers Request

In order to reduce the number of exported transactions and provide more detailed searching, the PayTrace API will allow the following optional name / value pairs for export transaction requests:

CUSTID, EMAIL, USER, RETURNBIN

4.5.c Example of an ExportCustomers Request

```
'-------------------------------...continued from Image 3.1a...----------------------------

'format the request string to export all of the customer profiles for the demo account
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ExportCustomers|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.5a..----------------------------
```

Image 4.5

4.6 Calculate Shipping Rates

Real-time shipping rates from DHL, United States Postal Service, and FedEx may be obtained through the PayTrace API at any time allowing your software to provide accurate real-time shipping rates.

4.6.a Required Name / Value Pairs for Calculating Shipping Rates

Processing a successful request to calculate shipping rates through the PayTrace
API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, SOURCEZIP, SOURCESTATE, SZIP, WEIGHT,
SHIPPERS, SSTATE

4.6.b Optional Name / Value Pairs of a CalculateShipping Request

SCOUNTRY

4.6.c Example of a CalculateShipping Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to export all of the customer profiles for the demo account
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~CalculateShipping|"
strRequest = strRequest & "SOURCEZIP~99201|SZIP~97201|WEIGHT~5.5|SOURCESTATE~WA|"
strRequest = strRequest & "SHIPPERS~DHL,USPS,UPS,FEDEX|SSTATE~OR|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.6a...----------------------------
```

<div align="center">Image 4.6</div>

4.7 Managing Recurring Transactions

Recurring transactions may be created and updated through the PayTrace API. All
recurring transactions must be referenced to an existing customer profile, and they
will be processed per the specified frequency until the transaction has been
processed the same number of times as the specified total count.

4.7.1.a Required Name / Value Pairs for Creating a Recurring Transaction

Processing a successful request to create a recurring transaction through the

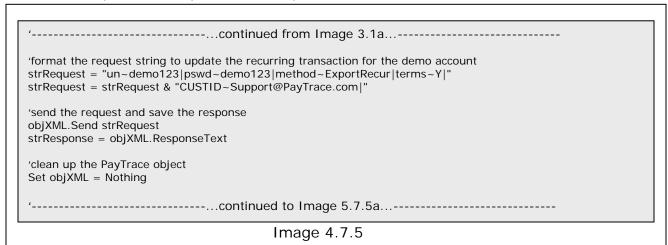PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CUSTID, FREQUENCY, START, TOTALCOUNT,

AMOUNT, TRANXTYPE

4.7.1.b Optional Name / Value Pairs of a CreateRecur Request

The PayTrace API will allow the following optional name / value pairs for create

recurring transaction requests:

DESCRIPTION, CUSTRECEIPT, RECURTYPE

4.7.1.c Example of a CreateRecur Request

```
'--------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to create a recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~createrecur|terms~Y|"
strRequest = Request & "start~3/26/2005|amount~1.00|totalcount~03|frequency~1|"
strRequest = strRequest & "custid~testcustomer|tranxtype~SALE|custreceipt~N|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.7.1a...-----------------------------
```

Image 4.7.1

4.7.2.a Required Name / Value Pairs for Updating a Recurring Transaction

Processing a successful request to update a recurring transaction through the

PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, RECURID

![PayTrace Payment Gateway logo]

4.7.2.b Optional Name / Value Pairs of an UpdateRecur Request

The PayTrace API will allow the following optional name / value pairs for create recurring transaction requests:

CUSTID, FREQUENCY, NEXT, TOTALCOUNT, AMOUNT, TRANXTYPE, DESCRIPTION, CUSTRECEIPT, RECURTYPE

4.7.2.c Example of an UpdateRecur Request

```
'-------------------------------…continued from Image 3.1a…-----------------------------

'format the request string to update the recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~updaterecur|terms~Y|"
strRequest = strRequest & "next~3/27/2005|amount~1.00|"
strRequest = strRequest & "recurid~8|totalcount~03|frequency~1|"
strRequest = strRequest & "custid~testcustomer|tranxtype~Sale|custreceipt~N|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------…continued to Image 5.7.2a…-----------------------------
```
Image 4.7.2

4.7.3.a Required Name / Value Pairs for Exporting a Recurring Transaction

This method may be used to return the date, amount, and approval code of the most recent approved recurring payment processed on the customer profile provided. Processing a successful request to export a recurring transaction through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CUSTID

4.7.3.b Optional Name / Value Pairs of an ExportCustomerRecur Request

The PayTrace API does not allow any optional name / value pairs for exporting recurring transaction requests.

4.7.3.c Example of an ExportCustomerRecur Request

```
'-------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to update the recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~ExportCustomerRecur|terms~Y|"
strRequest = strRequest & "custid~testcustomer|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.7.3a...-----------------------------
```

Image 4.7.3

4.7.4.a Required Name / Value Pairs for Deleting a Recurring Transaction

This method may be used to permanently delete a recurring payment. Processing a successful request to delete a recurring transaction through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, RECURID

Or

UN, PSWD, TERMS, METHOD, CUSTID

4.7.4.b Optional Name / Value Pairs of a DeleteRecur Request

The PayTrace API does not allow any optional name / value pairs for deleting a recurring transaction request.

4.7.4.c Example of a DeleteRecur Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to update the recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~DeleteRecur|terms~Y|"
strRequest = strRequest & "RecurID~1333|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.7.4a...----------------------------
```

Image 4.7.4

4.7.5.a Required Name / Value Pairs for Exporting Recurring Transactions

This method may be used to export details of a single recurring payment or all recurring payments for a specific customer. Processing a successful request to export recurring payments through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, RECURID

Or

UN, PSWD, TERMS, METHOD, CUSTID

4.7.5.b Optional Name / Value Pairs of an ExportRecur Request

The PayTrace API does not allow any optional name / value pairs for exporting recurring payments request.

![PayTrace Payment Gateway logo]

4.7.5.c Example of an ExportRecur Request

```
'--------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to update the recurring transaction for the demo account
strRequest = "un~demo123|pswd~demo123|method~ExportRecur|terms~Y|"
strRequest = strRequest & "CUSTID~Support@PayTrace.com|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.7.5a...-----------------------------
```
<div align="center">Image 4.7.5</div>

4.8 Updating a User Password

User passwords may be updated through the PayTrace API. User passwords must be changed at least once every 60 days. For specific information about PayTrace passwords, please reference section **7. Password Management**

4.8.a Required Name / Value Pairs for Updating a User Password
Processing a successful request to create a recurring transaction through the PayTrace API requires the following name / value pairs:
UN, PSWD, TERMS, METHOD, NEWPSWD, NEWPSWD2

4.8.b Optional Name / Value Pairs of an UpdatePassword Request
The PayTrace API does not allow optional name / value pairs for updating user password requests.

4.8.c Example of an UpdatePassword Request

```
'-------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to create a recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~updatepassword|terms~Y|"
strRequest = strRequest & "newpswd~ demo12345|newpswd2~ demo12345| "

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.8.1a...-----------------------------
```

Image 4.8.1

4.9 Adding Level 3 Data to a Transaction

Level 3 data is additional information that may be applied to enrich a transaction's reporting value to both the merchant and the customers. Generally, merchant service providers offer reduced or qualified pricing for transactions that are processed with Level 3 data.

Level 3 data may be added to any Visa or MasterCard sale that is approved and pending settlement. Some level 3 data, specifically enhanced data such as Invoice and Customer Reference ID, may overlap with data provided with the base transaction. Enhanced data, when applied, will always overwrite such data that may already be stored with the transaction.

Level 3 data consists of enhanced data and 1 or more line item records. This information is intended to describe the details of the transaction and the products

or services rendered. However, defaults may be applied in the event that some data is missing or unknown. So, all required fields must be present, even if their values are empty. Empty values will be overwritten with PayTrace defaults.

Please note that Visa and MasterCard each have their own requirements for level 3 data, so your application should be able to determine if the transaction being updated in a Visa or a MasterCard before formatting and sending the request. All Visa account numbers begin with "4" and contain 16 digits. All MasterCard account numbers begin with "5" and also contain 16 digits.

4.9.1.a Required Name / Value Pairs for Adding Level 3 Data to a Visa Sale
Processing a successful request to add level 3 data to a Visa sale through the PayTrace API requires the following name / value pairs:
UN, PSWD, TERMS, METHOD, TRANXID

4.9.1.b Optional Name / Value Pairs of a Level3VISA Request
Processing a successful request to add level 3 data to a Visa sale through the PayTrace API accepts the following optional name / value pairs:

INVOICE, CUSTREF, TAX, NTAX, MERCHANTTAXID, CUSTOMERTAXID, CCODE, DISCOUNT, FREIGHT, DUTY, SOURCEZIP, SZIP, SCOUNTRY, ADDTAX, ADDTAXRATE

With one or more Line Item records. Each Line Item record may contain the following name/value pairs:
CCODELI, PRODUCTID, DESCRIPTION, QUANTITY, MEASURE, UNITCOST, ADDTAXLI, ADDTAXRATELI, DISCOUNTLI, AMOUNTLI

4.9.1.c Example of a Level3VISA Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to email a receipt  for transaction ID 1498 to support@paytrace.com
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~Level3Visa|"
strRequest = strRequest & "TRANXID~1234|INVOICE~12345|CUSTREF~1234|TAX~-1|NTAX~0|"
strRequest = strRequest & "MERCHANTTAXID~123456789|CUSTOMERTAXID~987654321|"
strRequest = strRequest & "SOURCEZIP~97201|SZIP~99201|SCOUNTRY~US|"
strRequest = strRequest & "CCODE~1234|DISCOUNT~0|FREIGHT~0|DUTY~0|ADDTAX~0|"
strRequest = strRequest & "ADDTAXRATE~0|LINEITEM~CCODELI=12345678+"
strRequest = strRequest & "PRODUCTID=TESTPRODUCT+DESCRIPTION=TEST DESCRIPTION+"
strRequest = strRequest & "QUANTITY=1+MEASURE=LBS+UNITCOST=1+ADDTAXLI=0+"
strRequest = strRequest & "ADDTAXRATELI=0+DISCOUNTLI=0+AMOUNTLI=1+|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.9.1...----------------------------
```

Image 4.9.1

Please note that each name/value pair is separated by the traditional ~ and followed by a |. However, name/value pairs included in the LINEITEM parameter are separated by the = symbol and followed by a + symbol. So, no values in a Level3Visa request should contain a ~, |, +, or = symbols. The example request above contains 1 Line Item record.

4.9.2.a Required Name / Value Pairs for Adding Level 3 Data to a MasterCard Sale
Processing a successful request to add level 3 data to a MasterCard sale through the PayTrace API requires the following name / value pairs:
UN, PSWD, TERMS, METHOD, TRANXID

4.9.2.b Optional Name / Value Pairs of an Level3MCRD Request

Processing a successful request to add level 3 data to a MasterCard sale through the PayTrace API accepts the following optional name / value pairs:

INVOICE, CUSTREF, TAX, NTAX, FREIGHT, DUTY, SOURCEZIP, SZIP, SCOUNTRY, ADDTAX, ADDTAXIND

With one or more Line Item records. Each Line Item record may contain the following name/value pairs:

PRODUCTID, DESCRIPTION, QUANTITY, MEASURE, MERCHANTTAXID, UNITCOST, ADDTAXRATELI, ADDTAXINDLI, ADDTAXLI, AMOUNTLI, DISCOUNTIND, NETGROSSIND, DCIND, DISCOUNTLI, DICOUNTRATE

4.9.2.c Example of a Level3MCRD Request

```
'--------------------------------…continued from Image 3.1a…-----------------------------

'format the request string to email a receipt  for transaction ID 1498 to support@paytrace.com
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~Level3MCRD|"
strRequest = strRequest & "TRANXID~1234|INVOICE~12345|CUSTREF~1234|TAX~-1|NTAX~0|"
strRequest = strRequest & "SOURCEZIP~97201|SZIP~99201|SCOUNTRY~US|"
strRequest = strRequest & "FREIGHT~0|DUTY~0|ADDTAX~0| ADDTAXIND~N|"
strRequest = strRequest & "LINEITEM~CCODELI=12345678+"
strRequest = strRequest & "PRODUCTID=TESTPRODUCT+DESCRIPTION=TEST DESCRIPTION+"
strRequest = strRequest & "QUANTITY=1+MEASURE=LBS+MERCHANTTAXID=123456789+"
strRequest = strRequest & "DISCOUNTIND=N+NETGROSSIND=N+DCIND=D+DISCOUNTLI=0+"
strRequest = strRequest & "ADDTAXRATELI=0+ADDTAXINDLI=0+ADDTAXLI=0+AMOUNTLI=1+|"
strRequest = strRequest & "LINEITEM~CCODELI=12345679+"
strRequest = strRequest & "PRODUCTID=TESTPRODUCT2+DESCRIPTION=TEST DESCRIPTION2+"
strRequest = strRequest & "QUANTITY=1+MEASURE=LITER+MERCHANTTAXID=123456782+"
strRequest = strRequest & "DISCOUNTIND=N+NETGROSSIND=N+DCIND=D+DISCOUNTLI=0+"
strRequest = strRequest & "ADDTAXRATELI=0+ADDTAXINDLI=0+ADDTAXLI=0+AMOUNTLI=1+|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------…continued to Image 5.9.2…-----------------------------
```

Image 4.9.2

Please note that each name/value pair is separated by the traditional ~ and followed by a |. However, name/value pairs included in the LINEITEM parameter are separated by the = symbol and followed by a + symbol. So, no values in a Level3MCRD request should contain ~, |, +, or = symbols. The example request above contains 2 Line Item records.

4.10 Settling Transactions Through the PayTrace API

Transactions processed through merchant accounts that are set up on the
TSYS/Vital network or other terminal-based networks may initiate the settlement of
batches through the PayTrace API.

4.10.a Required Name / Value Pairs for Settling Transactions
Processing a successful request to settle transactions through the PayTrace API
requires the following name / value pairs:
UN, PSWD, TERMS, METHOD

4.10.b Optional Name / Value Pairs of a SettleTranx Request
The PayTrace API does not allow optional name / value pairs for settling transaction
requests.

4.10.c Example of a SettleTranx Request

```
'--------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to create a recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~settletranx|terms~Y|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.10.1a...-----------------------------
```

Image 4.10.1

4.11 Adjusting Transaction Amounts Through the PayTrace API

Transactions processed through merchant accounts that are set up on the TSYS/Vital network or other terminal-based networks may adjust transaction amounts to any amount that is less than or equal to the original transaction amount and greater than zero. Amounts may be adjusted for the following transaction conditions:

- Approved Sale that is not yet settled
- Forced Sale that is not yet settled
- Authorization that is approved and not yet settled
- Refund that is not yet settled

Please note that amounts for cash advance transaction may also not be adjusted.

4.11.a Required Name / Value Pairs for Adjusting Amounts

Processing a successful request to adjust an amount through the PayTrace API requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, TRANXID, AMOUNT

4.11.b Optional Name / Value Pairs of an AdjustAmount Request

The PayTrace API does not allow optional name / value pairs for adjust transaction amount requests.

4.11.c Example of an AdjustAmount Request

```
'--------------------------------…continued from Image 3.1a…----------------------------

'format the request string to create a recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~adjustamount|terms~Y|"
strRequest = strRequest & "TRANXID~1234|amount~0.75|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing
'--------------------------------…continued to Image 5.11.1a…----------------------------
```

Image 4.11.1

4.12 Exporting Batch Information Through the PayTrace API

Verifying batch details is sometimes necessary for your application to be able to determine deposit and transaction sums. The ExportBatch method is useful for extracting a summary of a specific batch or currently pending settlement break-down by card and transaction type.

4.12.a Required Name / Value Pairs for Exporting Batch details
Processing a successful request to export batch details through the PayTrace API requires the following name / value pairs:
UN, PSWD, TERMS, METHOD

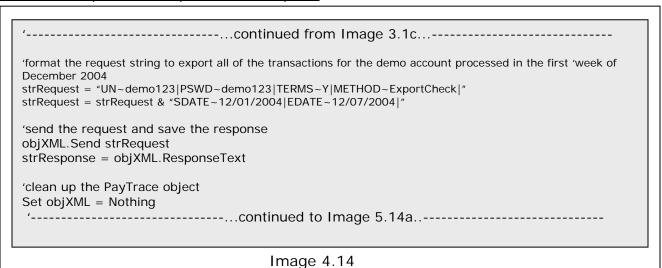### 4.12.b Optional Name / Value Pairs of an ExportBatch Request
The following parameters may be included when processing a request to ExportBatch:
SDATE, BATCHNUMBER

4.12.c Example of an ExportBatch Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to create a recurring transaction for the demo account
strRequest = "parmList=un~demo123|pswd~demo123|method~exportbatch|terms~Y|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.12.1a...----------------------------
```

Image 4.12.1

4.13 Processing Checks through the PayTrace API

Check or ACH (Automated Clearing House) transactions may be processed through the PayTrace API. Customer billing information may be referenced to an existing customer profile or key entered through the PayTrace API.

Please note that the PayTrace Secure Checkout page may be used to process check sales and holds/authorizations for those developers who wish to use PayTrace Secure Checkout as a means for their customers to provide their billing information. Please refer to section 6. Using PayTrace's Secure Checkout.

Any check may be processed through the PayTrace API as a test by setting the "TEST" attribute to "Y". Test checks return standardized responses in the same format as live checks, but funds will not actually be transferred.

4.13.1.a Required Name / Value Pairs of a Sale Request

Processing a sale through the PayTrace API may be accomplished by providing a new customer's key entered checking account information or the customer ID of an existing customer.

Processing a sale with a new customer's key entered checking account requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CHECKTYPE, AMOUNT, DDA, TR

Processing a sale with an existing customer's customer ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CHECKTYPE, AMOUNT, CUSTID

4.13.1.b Optional Name / Value Pairs of a Sale Request

Several optional name / value pairs may be sent with a sale request. The following name / value pairs may be provided in the sale request:

BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, INVOICE, DESCRIPTION, TAX, CUSTREF

4.13.1.c Example of a Sale Request

```
'-------------------------------…continued from Image 3.1a…-----------------------------

'format the request string to process a sale for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessCheck|"
strRequest = strRequest & "CHECKTYPE~Sale|DDA~123456|TR~999999999|"
strRequest = strRequest & "AMOUNT~1.00| BADDRESS~1234|BZIP~83852|INVOICE~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------…continued to Image 5.13.1…-----------------------------
```
<div align="center">Image 4.3.1</div>

4.13.2.a Required Name / Value Pairs of a Hold Request

Processing a hold through the PayTrace API will create a check transaction for the specified amount. However, the check amount will not be funded until the check is captured. Please note that some check processors, such as GETI, do not support Hold requests.

The required name / value pairs for processing a hold are the same as processing a sale as referenced in section *4.13.1.a Required Name / Value Pairs of a Sale Request*

4.13.2.b Optional Name / Value Pairs of an Hold Request

The optional name / value pairs for processing a hold are the same as processing a sale as referenced in section *4.13.1.b Optional Name / Value Pairs of a Sale Request*

4.13.2.c Example of an Hold Request

```
'-------------------------------...continued from Image 3.1a...----------------------------

'format the request string to process an authorization for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessCheck|"
strRequest = strRequest & "CHECKTYPE~Hold|DDA~123456|TR~999999999|"
strRequest = strRequest & "AMOUNT~1.00|BADDRESS~1234|BZIP~83852|"
strRequest = strRequest & "INVOICE~8888|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------...continued to Image 5.13.1...----------------------------
```

Image 4.13.2

4.13.3.a Required Name / Value Pairs of a Refund Request

Processing a refunded check through the PayTrace API may be accomplished by
providing a new customer's key entered checking account information, providing
the customer ID of an existing customer, or providing the transaction ID of the
original check that should be refunded. Please note that some check processors,
such as GETI, do not support Refund requests.

Processing a refund with a new customer's key entered checking account requires
the following name / value pairs:
UN, PSWD, TERMS, METHOD, CHECKTYPE, AMOUNT, DDA, TR

Processing a refund with an existing customer's customer ID requires the following
name / value pairs:
UN, PSWD, TERMS, METHOD, CHECKTYPE, AMOUNT, CUSTID

Processing a refund with an existing check ID requires the following name / value pairs:

UN, PSWD, TERMS, METHOD, CHECKTYPE, CHECKID

4.13.3.b Optional Name / Value Pairs of a Refund Request

Several optional name / value pairs may be sent with a refund request in order to enhance the reporting value of the receipt and the check. The following name / value pairs may be provided in the refund request:

BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, SNAME, SADDRESS, SADDRESS2, SCITY, SCOUNTY, SSTATE, SZIP, EMAIL, INVOICE, DESCRIPTION, TAX, CUSTREF, AMOUNT

4.13.3.c Example of a Refund Request

```
'--------------------------------...continued from Image 3.1a...-----------------------------

'format the request string to process a refund for $1.00
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessCheck|"
strRequest = strRequest & "CHECKTYPE~Refund|DDA~123456|TR~999999999|"
strRequest = strRequest & "AMOUNT~1.00|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.13.3...-----------------------------
```

Image 4.13.3

4.13.4.a Required Name / Value Pairs of a Manage Check Request

Managing checks through the PayTrace API may only be accomplished by providing the check ID of the unsettled check and the type or status that you'd like to assign it. CheckType (status) may be set to Void, Hold, or Fund (Capture). Please note

that some check processors, such as GETI, do not support Void, Hold, and Fund requests.

UN, PSWD, TERMS, METHOD, CHECKTYPE, CHECKID

4.13.4.b Optional Name / Value Pairs of a Manage Check Request

The check amount may be revised when sending a request to manage a check that's not processed by a real-time check processor.

AMOUNT

4.13.4.c Example of a ManageCheck Request

```
'--------------------------------...continued from Image 3.1a...----------------------------

'format the request string to process a void transaction ID 1539 from settlement
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ManageCheck|"
strRequest = strRequest & "CHECKTYPE~Void|CheckID~1539|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'--------------------------------...continued to Image 5.13.3...----------------------------
```

Image 4.13.4

4.14 Exporting Check Information

Check information may be exported through the PayTrace API at any time allowing check records to be viewed without being stored on the client computer.

4.14.a Required Name / Value Pairs for Exporting Check

Processing a successful request to export checks through the PayTrace API requires the following name / value pairs.

UN, PSWD, TERMS, SDATE, EDATE, METHOD

4.14.b Optional Name / Value Pairs of an ExportCheck Request

In order to reduce the number of exported checks and provide more detailed searching, the PayTrace API will allow the following optional name / value pairs for export check requests.

CHECKTYPE, CUSTID, USER

(Please note the CHECKTYPE name may also include the values "SETTLED" and "PENDING" in addition to the values in section 3.3.1)

4.14.c Example of an ExportCheck Request

```
'-------------------------------...continued from Image 3.1c...-----------------------------

'format the request string to export all of the transactions for the demo account processed in the first 'week of
December 2004
strRequest = "UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ExportCheck|"
strRequest = strRequest & "SDATE~12/01/2004|EDATE~12/07/2004|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing
 '-------------------------------...continued to Image 5.14a..-----------------------------
```

Image 4.14

5. Receiving and Parsing PayTrace API Response

The following sections illustrate how to parse the response strings that are returned from requests passed through the PayTrace API. All requests that are sent to the correct fully qualified domain name should receive a response.

Responses returned by the PayTrace API are formatted in the same name / value pair format as the requests. However, the names in the responses are often different from the names in the requests.

5.1 Parsing Transaction Responses from the PayTrace API

Each transaction request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the responses to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user. The code samples in this document loop through responses one character at a time. However, you may find it more convenient to use the Split() function in VB Script or the Explode() function in PHP to accomplish the same results.

5.1.1.a Returned Name / Value Pairs of a Sale Response

Responses elicited from a ProcessTranx request and a TranxType of sale will always return either one or more error messages or a set of responses from the credit card issuer. Successful responses from the credit card issuer will always include:
RESPONSE, TRANSACTIONID, APPCODE, APPMSG, AVSRESPONSE, CSCRESPONSE

5.1.1.b Example of Parsing a Sale Response

```
'--------------------------------…continued from Image 4.1.1…-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strTransactionID As String
Dim strAppCode As String
Dim strAppMsg As String
Dim strAVSResponse As String
Dim strCSCResponse As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

            'store the name/value pair character in the curPair string and keep looping.
            curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

            If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                      StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                      strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "TRANSACTIONID" Then

                      strTransactionID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "APPCODE" Then

                      strAppCode = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "APPMSG" Then

                      strAppMsg = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "AVSRESPONSE" Then

                      strAVSResponse = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CSCRESPONSE" Then

                      strCSCResponse = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

            StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.1.1

5.1.2.a Returned Name / Value Pairs of an Authorization Response

Responses elicited from a ProcessTranx request and a TranxType of authorization will always return either one or more error messages or a set of responses from the credit card issuer. Successful responses from the credit card issuer will always include:

RESPONSE, TRANSACTIONID, APPCODE, APPMSG, AVSRESPONSE, CSCRESPONSE

5.1.2.b Example of Parsing an Authorization Response

Parsing a response that was returned from an authorization request is the exact same process as parsing a sale response. Please refer to Image 5.1.1 for a VB Script code sample.

5.1.3.a Returned Name / Value Pairs of a Refund Response

Responses elicited from a ProcessTranx request and a TranxType of refund will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, TRANSACTIONID

5.1.3.b Example of Parsing a Refund Response

```
'------------------------------...continued from Image 4.1.3...------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strTransactionID As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

            'store the name/value pair character in the curPair string and keep looping.
            curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "TRANSACTIONID" Then

                strTransactionID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.1.3

5.1.4.a Returned Name / Value Pairs of a Void Response

Responses elicited from a ProcessTranx request and a TranxType of void will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, TRANSACTIONID

5.1.4.b Example of Parsing a Void Response

Parsing a response that was returned from a void request is the exact same process as parsing a refund response. Please refer to Image 5.1.3 for a VB Script code sample.

5.1.5.a Returned Name / Value Pairs of a Forced Sale Response

Responses elicited from a ProcessTranx request and a TranxType of force will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, TRANSACTIONID

5.1.5.b Example of Parsing a Force Response

Parsing a response that was returned from a force request is the exact same process as parsing a refund response. Please refer to Image 5.1.3 for a VB Script code sample.

5.1.6.a Returned Name / Value Pairs of a Captured Response

Responses elicited from a ProcessTranx request and a TranxType of capture will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, TRANSACTIONID

5.1.6.b Example of Parsing a Capture Response

Parsing a response that was returned from a capture request is the exact same process as parsing a refund response. Please refer to Image 5.1.3 for a VB Script code sample.

5.1.7.a Returned Name / Value Pairs of a CashAdvance Response

Responses elicited from a ProcessTranx request with TranxType set Sale and CashAdvance set to Y will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, TRANSACTIONID, APPCODE, APPMSG, AVSRESPONSE, CSCRESPONSE

5.1.7.b Example of Parsing a CashAdvance Response

Parsing a response that was returned from a CashAdvance request is the exact same process as parsing a Sale response. Please refer to Image 5.1.1 for a VB Script code sample.

![PayTrace Payment Gateway logo]

5.2 Parsing Customer Profile Responses from the PayTrace API

Each customer request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not null. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.2.1.a Returned Name / Value Pairs of a CreateCustomer Response

Responses elicited from a CreateCustomer request will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, CUSTOMERID

5.2.1.b Example of Parsing a CreateCustomer Response

```
'------------------------------...continued from Image 4.2...------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strCustomerID As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CUSTOMERID" Then

                strCustomerID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.2

5.2.2.a Returned Name / Value Pairs of an UpdateCustomer Response

Responses elicited from a UpdateCustomer request will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CUSTOMERID

5.2.2.b Example of Parsing an UpdateCustomer Response

Parsing a response that was returned from an UpdateCusomter request is the exact same process as parsing a CreateCustomer response. Please refer to Image 5.2 for a VB Script code sample.

5.2.3.a Returned Name / Value Pairs of a DeleteCustomer Response

Responses elicited from a DeleteCustomer request will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CUSTID

5.2.3.b Example of Parsing a DeleteCustomer Response

Parsing a response that was returned from an DeleteCustomer request is the exact same process as parsing a CreateCustomer response. Please refer to Image 5.2 for a VB Script code sample.

5.3 Parsing Email Receipt Responses from the PayTrace API

Through the PayTrace API, receipts of processed transactions may always be emailed to any address specified in the request. Successfully emailed receipts will return a confirmation response.

5.3.a Returned Name / Value Pairs of an EmailReceipt Response

Responses elicited from an *EmailReceipt* request will always return either one or more error messages or a response message.

5.3.b Example of Parsing an EmailReceipt Response

```
'------------------------------...continued from Image 4.3...------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.3

5.4 Parsing Export Transactions Responses from the PayTrace API

Through the PayTrace API, transaction information may always be exported for any transaction that has been processed through the PayTrace Payment Gateway. In order to minimize the number of transaction records may be searched by the following criterion:

SDATE, EDATE, TRANXTYPE, CUSTID, USER

5.4.a Returned Name / Value Pairs of an ExportTranx Response

Responses elicited from an ExportTranx request will always return either one or more error messages or one or more transaction records.

![PayTrace Payment Gateway logo]

5.4.b Example of Parsing an ExportTranx Response

```
'-------------------------------...continued from Image 4.4...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare tools to loop through the current transaction record
Dim subCurChar as Integer
Dim subCurPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strTRANXID As String
Dim strCC As String
Dim strEXPMNTH As String
Dim strEXPYR As String
Dim strTRANXTYPE As String
Dim strDESCRIPTION As String
Dim strAMOUNT As String
Dim strINVOICE As String
Dim strSNAME As String
Dim strSADDRESS As String
Dim strSADDRESS2 As String
Dim strSCITY As String
Dim strSCOUNTY As String
Dim strSSTATE As String
Dim strSZIP As String
Dim strBNAME As String
Dim strBADDRESS As String
Dim strBADDRESS2 As String
Dim strBCITY As String
Dim strBSTATE As String
Dim strBZIP As String
Dim strEMAIL As String
Dim strTAX As String
Dim strCUSTREF As String
Dim strAPPROVAL As String
Dim strAPPMSG As String
Dim strAVSRESPONSE As String
Dim strCSCRESPONSE As String
Dim strSTATUS As String
Dim strWHEN As String
Dim strUSER As String
Dim strIP As String

'-------------------------------...continued to Image 5.4b-------------------------------
```

Image 5.4a

```
'--------------------------------...continued from Image 5.4a...------------------------------
'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While curChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a transaction record
If Mid(strResponse, curChar, 1) <> "|" Then

    'store the transaction record character in the curPair string and keep looping.
    curPair = curPair & Mid(strResponse, curChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the transaction record

    'determine the name of the name / value pair
    If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

        strError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

    ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "TRANSACTIONRECORD" Then

        'set the sub looping tools to their starting points
subCurChar = 0
subCurPair = ""

'loop through the current transaction record, one character at a time
While subCurChar <= Len(curPair)

'check if the subCurChar is not a "+" signifying the end of a sub name / value pair
If Mid(curPair, subCurChar, 1) <> "+" Then

'store the name/value pair character in the SubCurPair string and keep looping.
        subCurPair = subCurPair & Mid(Response, subCurChar, 1)

Else 'the SubCurChar is a "+" indicating we are at the end of a name / value pair

        'determine the name of the name / value pair
        If UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TRANXID" Then

            strTRANXID = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CC" Then

            strCC = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EXPMNTH" Then

            strEXPMNTH = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    '-------------------------------...continued to Image 5.4c---------------------------
```

Image 5.4b

```
'--------------------------------...continued from Image 5.4b...--------------------------
    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EXPYR" Then

        strEXPYR = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TRANXTYPE" Then

        strTRANXTYPE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "DESCRIPTION" Then

        strDESCRIPTION = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "AMOUNT" Then

        strAMOUNT = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "INVOICE" Then

        strINVOICE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SNAME" Then

        strSNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS" Then

        strSADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS2" Then

        strSADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCITY" Then

        strSCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCOUNTY" Then

        strSCOUNTY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SSTATE" Then

        strSSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SZIP" Then

        strSZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BNAME" Then

        strBNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))
'--------------------------------...continued to Image 5.4d-------------------------
```

Image 5.4c

```
'-------------------------------…continued from Image 5.4c…-------------------------------
      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS" Then

          strBADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS2" Then

          strBADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BCITY" Then

          strBCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BSTATE" Then

          strBSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BZIP" Then

          strBZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EMAIL" Then

          strEMAIL = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TAX" Then

          strTAX = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CUSTREF" Then

          strCUSTREF = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "APPROVAL" Then

          APPROVAL = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "APPMSG" Then

          strAPPMSG = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "AVSRESPONSE" Then

          strAVSRESPONSE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CSCRESPONSE" Then

          strCSCRESPONSE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "STATUS" Then

          strSTATUS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))
'-------------------------------…continued to Image 5.4e-------------------------
```

Image 5.4d

```
'-------------------------------...continued from Image 5.4d...-------------------------------
ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "WHEN" Then

        strWHEN = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "USER" Then

        strUSER = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "IP" Then

        strIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    End If 'end if the Name of the current sub Pair is TRANXID

SubCurPair = ""

End If 'end if the SubCurChar is not a "+"

'increment the sub current character and look at the next character
subCurChar = subCurChar + 1

Wend 'end of the while loop that increments through the subPair string


        'SAVE OR DISPLAY THE TRANSACTION RECORD NAME / VALUE PAIRS TO THE USER


    End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
curChar = curChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

    strError = "The response from the PayTrace API was null."

End If 'End if the response was not null
```

Image 5.4e

![PayTrace Payment Gateway logo]

5.5 Parsing Export Customers Responses from the PayTrace API

Through the PayTrace API, stored customer profiles may always be exported at any time. In order to minimize the number of returned customer records, they may be searched by the following criterion:

CUSTID, EMAIL, USER

5.5.a Returned Name / Value Pairs of an ExportCustomers Response

Responses elicited from an ExportCustomers request will always return either one or more error messages or one or more customer records.

5.5.b Example of Parsing an ExportCustomers Response

```
'-------------------------------...continued from Image 4.5...--------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare tools to loop through the current customer record
Dim subCurChar as Integer
Dim subCurPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strCUSTID As String
Dim strCC As String
Dim strEXPMNTH As String
Dim strEXPYR As String
Dim strSNAME As String
Dim strSADDRESS As String
Dim strSADDRESS2 As String
Dim strSCITY As String
Dim strSCOUNTY As String
Dim strSSTATE As String
Dim strSZIP As String
Dim strBNAME As String
Dim strBADDRESS As String
Dim strBADDRESS2 As String
Dim strBCITY As String
Dim strBSTATE As String
Dim strBZIP As String
Dim strEMAIL As String
Dim strWHEN As String
Dim strUSER As String
Dim strIP As String

'------------------------------...continued to Image 5.5b-------------------------
```

Image 5.5a

```
'------------------------------...continued from Image 5.5a...------------------------------
'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While curChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a customer record
If Mid(strResponse, curChar, 1) <> "|" Then

    'store the customer record character in the curPair string and keep looping.
    curPair = curPair & Mid(strResponse, curChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the customer record

    'determine the name of the name / value pair
    If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

        strError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

    ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CUSTOMERRECORD" Then

        'set the sub looping tools to their starting points
subCurChar = 0
subCurPair = ""

'loop through the current customer record, one character at a time
While subCurChar <= Len(curPair)

'check if the subCurChar is not a "+" signifying the end of a sub name / value pair
If Mid(curPair, subCurChar, 1) <> "+" Then

'store the name/value pair character in the SubCurPair string and keep looping.
        subCurPair = subCurPair & Mid(Response, subCurChar, 1)

Else 'the SubCurChar is a "+" indicating we are at the end of a name / value pair

        'determine the name of the name / value pair
        If UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CUSTID" Then

            strCUSTID = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CC" Then

            strCC = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EXPMNTH" Then

            strEXPMNTH = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    '------------------------------...continued to Image 5.5c------------------------------
```

Image 5.5b

```
'--------------------------------...continued from Image 5.5b...------------------------------
    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EXPYR" Then

        strEXPYR = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SNAME" Then

        strSNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS" Then

        strSADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS2" Then

        strSADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCITY" Then

        strSCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCOUNTY" Then

        strSCOUNTY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SSTATE" Then

        strSSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SZIP" Then

        strSZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BNAME" Then

        strBNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS" Then

        strBADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS2" Then

        strBADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BCITY" Then

        strBCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BSTATE" Then

        strBSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))
'-------------------------------...continued to Image 5.5d-------------------------
```

Image 5.5c

```
'--------------------------------...continued from Image 5.5c...--------------------------------
      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BZIP" Then

          strBZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EMAIL" Then

          strEMAIL = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "PHONE" Then

          strPHONE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "FAX" Then

          strFAX = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "WHEN" Then

          strWHEN = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "USER" Then

          strUSER = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "IP" Then

          strIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

       End If 'end if the Name of the current sub Pair is TRANXID

SubCurPair = ""

End If 'end if the SubCurChar is not a "+"

'increment the sub current character and look at the next character
subCurChar = subCurChar + 1

Wend 'end of the while loop that increments through the subPair string

          'SAVE OR DISPLAY THE CUSTOMER RECORD NAME / VALUE PAIRS TO THE USER

    End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
curChar = curChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

    strError = "The response from the PayTrace API was null."

End If 'End if the response was not null
```

Image 5.5d

PayTrace
Payment Gateway

5.6 Parsing Calculate Shipping Responses from the PayTrace API

Through the PayTrace API, real time shipping quotes may be obtained from DHL, USPS, and FEDEX. These shipping quotes are returned in shipping records, and each call to the CalculateShipping method will return a response with one or more error messages or one or more shipping records.

5.6.a Returned Name / Value Pairs of a CalculateShipping Response

Responses elicited from a CalculateShipping request will always return either one or more error messages or one or more shipping records.

5.6.b Example of Parsing a CalculateShipping Response

```
'--------------------------------...continued from Image 4.6...--------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare tools to loop through the current customer record
Dim subCurChar as Integer
Dim subCurPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strShippingCompany As String
Dim strShippingCompany As String
Dim strShippingMethod As String
Dim strShippingRate As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While curChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a shipping record
If Mid(strResponse, curChar, 1) <> "|" Then

    'store the shipping record character in the curPair string and keep looping.
    curPair = curPair & Mid(strResponse, curChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the shipping record

    'determine the name of the name / value pair
    If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

        strError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

    ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "SHIPPINGRECORD" Then

'--------------------------------...continued to Image 5.6b--------------------------
```

Image 5.6a

```
'-------------------------------...continued from Image 5.6b...------------------------------
'set the sub looping tools to their starting points
subCurChar = 0
subCurPair = ""

'loop through the current customer record, one character at a time
While subCurChar <= Len(curPair)

'check if the subCurChar is not a "+" signifying the end of a sub name / value pair
If Mid(curPair, subCurChar, 1) <> "+" Then

'store the name/value pair character in the SubCurPair string and keep looping.
      subCurPair = subCurPair & Mid(Response, subCurChar, 1)

Else 'the SubCurChar is a "+" indicating we are at the end of a name / value pair

        'determine the name of the name / value pair
        If UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SHIPPINGCOMPANY" Then

          strSHIIPINGCOMPANY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SHIPPINGMETHOD" Then

        strSHIPPINGMETHOD = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SHIPPINGRATE" Then

        strSHIPPINGRATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

      End If 'end if the Name of the current sub Pair is SHIPPINGCOMPANY

SubCurPair = ""

End If 'end if the SubCurChar is not a "+"

'increment the sub current character and look at the next character
subCurChar = subCurChar + 1

Wend 'end of the while loop that increments through the subPair string

          'SAVE OR DISPLAY THE SHIPPING RECORD NAME / VALUE PAIRS TO THE USER

    End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
curChar = curChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

   strError = "The response from the PayTrace API was null."

End If 'End if the response was not null
```

Image 5.6b

5.7 Parsing Recurring Transaction Responses from the PayTrace API

Each recurring transaction request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not null. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.7.1.a Returned Name / Value Pairs of a CreateRecur Response

Responses elicited from a CreateRecur request will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, RECURID

5.7.1.b Example of Parsing a CreateRecur Response

```
'-------------------------------...continued from Image 4.7.1...------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strRecurID As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RECURID" Then

                strRecurID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.7.1

5.7.2.a Returned Name / Value Pairs of an UpdateRecur Response

Responses elicited from a UpdateRecur request will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, RECURID

5.7.2.b Example of Parsing an UpdateRecur Response

Parsing a response that was returned from an UpdateRecur request is the exact same process as parsing a CreateRecur response. Please refer to Image 5.7 for a VB Script code sample.

5.7.3.a Returned Name / Value Pairs of an ExportCustomerRecur Response

Responses elicited from a ExportCustomerRecur request will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, WHEN, AMOUNT, APPROVAL, NEXT, TOTALCOUNT, CURRENTCOUNT, DESCRIPTION, RECURID

5.7.3.b Example of Parsing an ExportCustomerRecur Response

```
'-------------------------...continued from Image 4.7.3...-----------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strWhen As String
Dim strAmount As String
Dim strApproval As String


'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

            'store the name/value pair character in the curPair string and keep looping.
            curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

            If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                    StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                    strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "WHEN" Then

                    strWhen = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "AMOUNT" Then

                    strAmount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "APPROVAL" Then

                    strApproval = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

            StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.7.3

5.7.4.a Returned Name / Value Pairs of a DeleteRecur Response

Responses elicited from a DeleteRecur request will always return either one or more error messages or a set of responses. Successful responses will always include:
RESPONSE, RECURID
Or
RESPONSE, CUSTID

5.7.4.b Example of Parsing a DeleteRecur Response

Parsing a response that was returned from a DeleteRecur request is the exact same process as parsing a CreateRecur response. Please refer to Image 5.7 for a VB Script code sample.

5.7.5.a Returned Name / Value Pairs of an ExportRecur Response

Responses elicited from an ExportRecur request will always return either one or more error messages or one or more recurring payment records.

5.7.5.b Example of Parsing an ExportRecur Response

```
'-------------------------------...continued from Image 4.7.5...-------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare tools to loop through the current customer record
Dim subCurChar as Integer
Dim subCurPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strRecurID As String
Dim strAmount As String
Dim strCustID As String
Dim strNext As String
Dim strTotalCount As String
Dim strCurrentCount As String
Dim strDescription As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While curChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a payment record
If Mid(strResponse, curChar, 1) <> "|" Then

    'store the recurring payment record character in the curPair string and keep looping.
    curPair = curPair & Mid(strResponse, curChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the shipping record

    'determine the name of the name / value pair
    If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

        strError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

    ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RECURRINGPAYMENT" Then

'set the sub looping tools to their starting points
subCurChar = 0
subCurPair = ""

'loop through the current customer record, one character at a time
While subCurChar <= Len(curPair)

'check if the subCurChar is not a "+" signifying the end of a sub name / value pair
If Mid(curPair, subCurChar, 1) <> "+" Then

'store the name/value pair character in the SubCurPair string and keep looping.
    subCurPair = subCurPair & Mid(Response, subCurChar, 1)


'-------------------------------...continued to Image 5.7.5b-------------------------
```

Image 5.7.5a

```
'-------------------------------...continued from Image 5.6b...-------------------------------

Else 'the SubCurChar is a "+" indicating we are at the end of a name / value pair

        'determine the name of the name / value pair
        If UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "RECURID" Then

            strRecurID = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "AMOUNT" Then

            strAmount = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CUSTID" Then

            strCustID = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "NEXT" Then

            strNext = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TOTALCOUNT" Then

            strTotalCount = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CURRENTCOUNT" Then

            strCurrentCount = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "DESCRIPTION" Then

            strDescription = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        End If 'end if the Name of the current sub Pair is SHIPPINGCOMPANY

SubCurPair = ""

End If 'end if the SubCurChar is not a "+"

'increment the sub current character and look at the next character
subCurChar = subCurChar + 1

Wend 'end of the while loop that increments through the subPair string

            'SAVE OR DISPLAY THE SHIPPING RECORD NAME / VALUE PAIRS TO THE USER

    End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
curChar = curChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

    strError = "The response from the PayTrace API was null."

End If 'End if the response was not null
```

Image 5.7.5b

5.8 Parsing Update User Password Responses from the PayTrace API

Each update password request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.8.a Returned Name / Value Pairs of an UpdatePassword Response

Responses elicited from an UpdatePassword request will always return either one or more error messages or a response.

5.8.b Example of Parsing an UpdatePassword Response

```
'-------------------------------...continued from Image 4.8...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.8

5.9 Parsing Level 3 Responses from the PayTrace API

Each request to add level 3 data sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.9.1.a Returned Name / Value Pairs of a Level3VISA Response

Responses elicited from a Level3VISA request will always return either one or more error messages or a response.

5.9.1.b Example of Parsing a Level3VISA Response

```
'-------------------------------...continued from Image 4.9.1...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.9.1

5.9.2.a Returned Name / Value Pairs of a Level3MCRD Response

Responses elicited from a Level3MCRD request will always return either one or more error messages or a response.

5.9.2.b Example of Parsing a Level3MCRD Response

```
'---------------------------------...continued from Image 4.9.2...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.9.2

![PayTrace Payment Gateway logo]

5.10 Parsing Settle Transaction Responses from the PayTrace API

Each settle transaction request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.10.a Returned Name / Value Pairs of a SettleTranx Response

Responses elicited from a SettleTranx request will always return either one or more error messages or a response. Successful responses will always include:
RESPONSE, TRANXCOUNT, NETAMOUNT, BATCHNUM

5.10.b Example of Parsing a SettleTranx Response

```
'-------------------------------…continued from Image 4.10…-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String
Dim strTranxCount As String
Dim strNetAmount as String
Dim strBatchNum as String


'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

            'store the name/value pair character in the curPair string and keep looping.
            curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

            If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                        StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                        strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "TRANXCOUNT" Then

                        strTranxCount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "NETAMOUNT" Then

                        strNetAmount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "BATCHNUM" Then

                        strBatchNum = Right(curPair, Len(curPair) - InStr(curPair, "~"))

            End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

            StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.10

5.11 Parsing Adjust Amount Responses from the PayTrace API

Each adjust transaction amount request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.11.a Returned Name / Value Pairs of an AdjustAmount Response

Responses elicited from an AdjustAmount request will always return either one or more error messages or a response. Successful responses will always include:
RESPONSE

5.11.b Example of Parsing an AdjustAmount Response

```
'--------------------------------...continued from Image 4.11...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMessage As String


'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.11

5.12 Parsing Export Batch Details Responses from the PayTrace API

Each request to export batch details sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not empty. Your application will also need to parse the response to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.12.a Returned Name / Value Pairs of an ExportBatch Response

Responses elicited from an AdjustAmount request will always return either one or more error messages or a response. Successful responses will always include: BATCHNUM, WHEN, VISASALESCOUNT, VISASALESAMOUNT, VISAREFUNDCOUNT, VISAREFUNDAMOUNT, MASTERCARDSALESCOUNT, MASTERCARDSALESAMOUNT, MASTERCARDREFUNDCOUNT, MASTERCARDREFUNDAMOUNT, AMEXSALESCOUNT, AMEXSALESAMOUNT, AMEXREFUNDCOUNT, AMEXREFUNDAMOUNT, DISCOVERSALESCOUNT, DISCOVERSALESAMOUNT, DISCOVERREFUNDCOUNT, DISCOVERREFUNDAMOUNT, DINERSSALESCOUNT, DINERSSALESAMOUNT, DINERSREFUNDCOUNT, DINERSREFUNDAMOUNT, JCBSALESCOUNT, JCBSALESAMOUNT, JCBREFUNDCOUNT, JCBREFUNDAMOUNT, PRIVATESALESCOUNT, PRIVATESALESAMOUNT, PRIVATEREFUNDCOUNT, PRIVATEREFUNDAMOUNT

5.12.b Example of Parsing an ExportBatch Response

```
'---------------------------------...continued from Image 4.11...--------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strBatchNum As String
Dim strWhen As String
Dim strVisaSalesCount As String
Dim strVisaSalesAmount As String
Dim strVisaRefundCount As String
Dim strVisaRefundAmount As String
'declare additional variables for other card types here

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

                'store the name/value pair character in the curPair string and keep looping.
                curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

                If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                            StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "BATCHNUM" Then

                            strBatchNum = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "WHEN" Then

                            strWhen = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = " VISASALESCOUNT" Then

                            strVisaSalesCount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = " VISASALESAMOUNT " Then

                            strVisaSalesAmount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = " VISAREFUNDCOUNT" Then

                            strVisaRefundCount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "VISAREFUNDAMOUNT" Then

                            strVisaRefundAmount = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                'check for other card types here

                End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

                StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.12

PayTrace
Payment Gateway

5.13 Parsing Check Responses From the PayTrace API

Each check request sent to the PayTrace API should elicit a response. However, your application should validate the response to ensure it is not null. Your application will also need to parse the responses to determine if errors occurred. Your application will certainly also need to display any errors or successful responses to the user.

5.13.1.a Returned Name / Value Pairs of a Sale Response

Responses elicited from a ProcessCheck request and a CheckType of sale will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CHECKIDENTIFIER

Please note that following values will be returned when a check is processed through a real-time check processor:

RESPONSE, CHECKIDENTIFIER, ACHCODE, ACHMSG

5.13.1.b Example of Parsing a Sale Response

```
'---------------------------...continued from Image 4.1.1...----------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMsg As String
Dim strCheckID As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

                'store the name/value pair character in the curPair string and keep looping.
                curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

                If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                        StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                        strResponseMsg = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CHECKIDENTIFIER" Then

                        strCheckID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

                End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.13.1

5.13.2.a Returned Name / Value Pairs of a Hold Response

Responses elicited from a ProcessCheck request and a CheckType of hold will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CHECKIDENTIFIER

5.13.2.b Example of Parsing a Hold Response

Parsing a response that was returned from a hold request is the exact same process as parsing a sale response. Please refer to Image 5.14.1 for a Visual Basic 6.0 code sample.

5.13.3.a Returned Name / Value Pairs of a Refund Response

Responses elicited from a ProcessCheck request and a CheckType of refund will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CHECKIDENTIFIER

5.13.3.b Example of Parsing a Refund Response

```
'-------------------------------...continued from Image 4.1.3...-----------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strResponseMsg As String
Dim strCheckID As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(Response)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMsg = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CHECKIDENTIFIER" Then

                strTransactionID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 5.13.3

5.13.4.a Returned Name / Value Pairs of a Manage Check Response

Responses elicited from a ProcessCheck request of ManageCheck and a CheckType of void, hold, or fund will always return either one or more error messages or a set of responses. Successful responses will always include:

RESPONSE, CHECKIDENTIFIER

5.13.4.b Example of Parsing a Manage Check Response

Parsing a response that was returned from a mange check request is the exact same process as parsing a refund response. Please refer to Image 5.14.3 for a Visual Basic 6.0 code sample.

5.14 Parsing Export Transactions Responses from the PayTrace API

Through the PayTrace API, check information may always be exported for any check that has been processed through the PayTrace Payment Gateway.

5.14.a Returned Name / Value Pairs of an ExportCheck Response

Responses elicited from an ExportCheck request will always return either one or more error messages or one or more check records.

5.14.b Example of Parsing an ExportTranx Response

```
'-------------------------------...continued from Image 4.4...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare tools to loop through the current transaction record
Dim subCurChar as Integer
Dim subCurPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strCheckID As String
Dim strDDA As String
Dim strTR As String
Dim strCheckTYPE As String
Dim strDESCRIPTION As String
Dim strAMOUNT As String
Dim strINVOICE As String
Dim strSNAME As String
Dim strSADDRESS As String
Dim strSADDRESS2 As String
Dim strSCITY As String
Dim strSCOUNTY As String
Dim strSSTATE As String
Dim strSZIP As String
Dim strBNAME As String
Dim strBADDRESS As String
Dim strBADDRESS2 As String
Dim strBCITY As String
Dim strBSTATE As String
Dim strBZIP As String
Dim strEMAIL As String
Dim strTAX As String
Dim strCUSTREF As String
Dim strSTATUS As String
Dim strWHEN As String
Dim strUSER As String
Dim strIP As String

'-------------------------------...continued to Image 5.14b-------------------------------
```

Image 5.14a

```
'---------------------------------...continued from Image 5.14a...------------------------------
'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'set the looping tools to their starting points
curChar = 0
curPair = ""

'loop through the response, one character at a time
While curChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a transaction record
If Mid(strResponse, curChar, 1) <> "|" Then

    'store the transaction record character in the curPair string and keep looping.
    curPair = curPair & Mid(strResponse, curChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the transaction record

    'determine the name of the name / value pair
    If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

        strError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

    ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "CHECKRECORD" Then

        'set the sub looping tools to their starting points
subCurChar = 0
subCurPair = ""

'loop through the current transaction record, one character at a time
While subCurChar <= Len(curPair)

'check if the subCurChar is not a "+" signifying the end of a sub name / value pair
If Mid(curPair, subCurChar, 1) <> "+" Then

'store the name/value pair character in the SubCurPair string and keep looping.
        subCurPair = subCurPair & Mid(Response, subCurChar, 1)

Else 'the SubCurChar is a "+" indicating we are at the end of a name / value pair

        'determine the name of the name / value pair
        If UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CHECKID" Then

            strCheckID = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "DDA" Then

            strDDA = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

        ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TR" Then

            strTR = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    '---------------------------------...continued to Image 5.14c---------------------------
```

Image 5.14b

```
'--------------------------------...continued from Image 5.14b...------------------------------

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CHECKTYPE" Then

        strCheckTYPE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "DESCRIPTION" Then

        strDESCRIPTION = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "AMOUNT" Then

        strAMOUNT = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "INVOICE" Then

        strINVOICE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SNAME" Then

        strSNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS" Then

        strSADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SADDRESS2" Then

        strSADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCITY" Then

        strSCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SCOUNTY" Then

        strSCOUNTY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SSTATE" Then

        strSSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "SZIP" Then

        strSZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BNAME" Then

        strBNAME = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

'--------------------------------...continued to Image 5.14d------------------------
```

Image 5.4c

```
'--------------------------------...continued from Image 5.14c...------------------------------
    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS" Then

        strBADDRESS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BADDRESS2" Then

        strBADDRESS2 = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BCITY" Then

        strBCITY = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BSTATE" Then

        strBSTATE = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "BZIP" Then

        strBZIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "EMAIL" Then

        strEMAIL = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "TAX" Then

        strTAX = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "CUSTREF" Then

        strCUSTREF = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "STATUS" Then

        strSTATUS = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))
'--------------------------------...continued to Image 5.14e-------------------------
```

Image 5.14d

```
'------------------------------…continued from Image 5.14d…------------------------------
ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "WHEN" Then

        strWHEN = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "USER" Then

        strUSER = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    ElseIf UCase(Left(subCurPair, InStr(subCurPair, "=") - 1)) = "IP" Then

        strIP = Right(subCurPair, Len(subCurPair) - InStr(subCurPair, "="))

    End If 'end if the Name of the current sub Pair is CHECKID

SubCurPair = ""

End If 'end if the SubCurChar is not a "+"

'increment the sub current character and look at the next character
subCurChar = subCurChar + 1

Wend 'end of the while loop that increments through the subPair  string


        'SAVE OR DISPLAY THE CHECK RECORD NAME / VALUE PAIRS TO THE USER


    End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
curChar = curChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

    strError = "The response from the PayTrace API was null."

End If 'End if the response was not null
```

Image 5.14e

6. Using PayTrace's Secure Checkout

The PayTrace Secure Checkout page is a fully customizable secure web page that may be interface to a web site that does not have a SSL certificate. Developers may send silent post messages to the PayTrace Secure Checkout page to specify the attributes of the order, and then redirect their customers to the secure page to enter their billing information. Once the transaction has been completed, a silent post is sent back the developer's web page with the standard PayTrace API transaction responses. Sales, Authorizations, and Forced Sales may be processed through the PayTrace API Secure Checkout.

6.1 Manage the PayTrace API Secure Checkout Design

Just like the PayTrace Shopping Cart, the PayTrace Secure Checkout page may be customized to look like your web site. With the click of a button the colors, fonts, and logos may be changed to mirror your existing website.

To navigate to the PayTrace API Secure Checkout design page, Select "Manage API Design" from the Programmer's Menu.

= Programmer's Menu
- PayTrace API
- API User Guide DLL
- API User Guide XML
- Manage API Design
- Manage API Checkout
Image 6.1a

Image 6.1b

6.1.1 Uploading a Business Logo to the PayTrace API Secure Checkout Page

You may also upload your company's logo into the banner of the checkout page by clicking on the "Would you like to upload your logo for display in the header?" Clicking this link will cause a new window to appear.

Welcome to PayTrace's File Upload Utility.

Please note that you may only upload logos that are less than 100 kilobytes and are saved as .jpg or .gif files. Also, please only upload images that are less than 100 pixels in height and 750 pixels wide.

Click browse to select a file to be uploaded: [   ] Browse...

Upload Logo

Image 6.1.1a

In the image upload window, you may click on the Browse button and select the logo that you wish to upload and display in the header of your shopping cart.

Image 6.1.1b

Once you've found the image you wish to upload, click on Open and click on "Upload Logo" from the image upload window.

Welcome to PayTrace's File Upload Utility.

Your file, PT(1).gif, was successfully saved. It may be referenced by visiting https://paytrace.com/cart/Images/PT(1).gif

Would you like to upload another logo or close this window?

Image 6.1.1c

Your logo should be displayed to you, and you may now close the image upload window. Now that you have uploaded your logo, refresh your PayTrace API Secure Checkout page to see your logo displayed in your secure checkout page header.

Please note that the logo must be less than 750 pixels wide and 100 pixels tall.

Image 6.1.1d

6.1.2 Changing The PayTrace API Secure Checkout Footer Information

If you decide to change the web address, email, address, or business name that appears in the footer of the PayTrace API Secure Checkout page, please contact your PayTrace reseller or Support@PayTrace.com

Image 6.1.2

6.2 Manage the PayTrace API Secure Checkout Settings

The PayTrace Secure Checkout page may be customized to send information and customers to designated pages on your website. The PayTrace Secure Checkout page may also be configured to require appropriate billing information at the time of checkout.

To navigate to the PayTrace API Secure Checkout settings page, Select "Manage API Checkout" from the Programmer's Menu.

= Programmer's Menu
- PayTrace API
- API User Guide DLL
- API User Guide XML
- Manage API Design
- Manage API Checkout

Image 6.2a

The PayTrace API Secure Checkout page may be configured with the following options.

Checkout Page Management	
Silent Post:	https://paytrace.com/api/1.pay
Terms Link:	https://paytrace.com/terms.html
Approval Link:	https://paytrace.com/app.pay
Decline Link:	https://paytrace.com/decline.pay
Require Email Address:	Yes
Require Billing Address:	Yes
Require CSC:	Yes

Image 6.2b

Silent Post URL must be a valid web address where you want the PayTrace API Secure Checkout page to send the transaction responses. Both approved and declined transactions will trigger a silent post message to be sent to the Silent Post URL. Transaction responses are formatted exactly like the transaction responses outlined in section *5.1 Parsing Transaction Responses from the PayTrace API.* The Silent Post URL may be overwritten at the time of order validation.

Terms Link must be a valid web address were your business' terms and conditions may be referenced. Each customer processing a transaction through the PayTrace API Secure Checkout will be forced to accept your business' terms and conditions.

Approval Link is an optional web address that will be displayed to your customers whose transactions were approved. If this URL is left blank, then the URL of your web site's home page will be displayed. The Approval Link may be overwritten at the time of order validation.

Decline Link is an optional web address that will be displayed to your customers whose transactions were declined. If this URL is left blank, then the URL of your web site's home page will be displayed. The Decline Link may be overwritten at the time of order validation.

Require Email Address may be set to Yes if you wish PayTrace to force you customer to provide their email address. The Require Email Address setting may be overwritten at the time of order validation.

Require Billing Address may be set to Yes if you wish PayTrace to force you customer to provide their complete billing address. The Require Billing Address setting may be overwritten at the time of order validation.

Require CSC may be set to Yes if you wish PayTrace to force you customer to provide their CSC. The Require CSC setting may be overwritten at the time of order validation.

6.3 Validating an Order through the PayTrace API Secure Checkout

Before a transaction may be processed through the PayTrace API Secure Checkout page, a corresponding order must be created and validated.

Validated order requests must contain the following attributes:

UN, PSWD, AMOUNT, TRANXTYPE, ORDERID, TERMS

Please note that TRANXTYPE may also be set to "CreateCustomer" or "UpdateCustomer" to provide an interface to manage customers with out first processing a transaction. ORDERID will be handled as the CUSTID. Please note that some check processors, such as GETI, do not support Authorization, Refund, Forced Sale,and Void requests.

Validated order request may contain the following attributes:
RETURNURL, APPROVEURL, DECLINEURL, FORCEEMAIL, FORCEADDRESS, FORCECSC

6.3.1 Formating and Sending a Request to Validate an Order

The PayTrace API Secure Checkout code samples are written in Visual Basic Script

```
'Declare the connection tools
Dim, objXML, Request, Response

'Create the XML object
set objXML=createobject("MSXML2.XMLHTTP")

'open the XML object and point it to the PayTrace secure servers
objXML.Open "POST", "https://paytrace.com/api/validate.pay",  false

'set the Request Header of the XML object to a URL encoded form
objXML.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"

'format the request string to validate order ID 1234
strRequest = "PARMLIST=UN~demo123|PSWD~demo123|TERMS~Y|ORDERID~1234|"
strRequest = strRequest & "AMOUNT~1.00|TRANXTYPE~SALE|"

'send the request and save the response
objXML.Send strRequest
strResponse = objXML.ResponseText

'clean up the PayTrace object
Set objXML = Nothing

'-------------------------------…continued to Image 6.3.2..-----------------------------
```

Image 6.3.1

6.3.2 Parsing a Response to Validate an Order

```
'-------------------------------...continued from Image 6.3.1...-------------------------------
'declare tools to loop through the response and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strError As String
Dim strOrderID As String
Dim strAuthKey As String

'check to make sure the response was not null
if strResponse <> "" and inStr(strResponse,"|") = true and inStr(strResponse,"~") = true then

'loop through the response, one character at a time
While CurChar <= Len(strResponse)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strResponse, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ORDERID" Then

                strOrderID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "AUTHKEY" Then

                strAuthKey = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 6.3.2

6.4 Redirecting a Customer to the PayTrace API Secure Checkout

After an order has been validated and supplied an Authorization Key, then the customer may be redirected to the secure checkout page to enter their billing information and complete the transaction.

6.4.1 Displaying a Hyperlink to the PayTrace API Secure Checkout

```
<a
href="https://paytrace.com/api/checkout.pay?parmList=orderID~1234|AuthKey~
20|">Checkout through PayTrace</a>
```

Image 6.4.1

6.4.2 Displaying a Command Button to the PayTrace API Secure Checkout

```
<form action = "https://paytrace.com/api/checkout.pay" method = post>
<input type="hidden" name="parmList" value="orderID~1234|AuthKey~21|">
<input type="submit" value="Checkout through PayTrace">
</form>
```

Image 6.4.2

Please note that redirection requests may include additional name/value pairs. BNAME, BADDRESS, BADDRESS2, BCITY, BSTATE, BZIP, BCOUNTRY, EMAIL, PHONE, INVOICE, and DESCRIPTION may all be defaulted to information that you may have already collected from the customer.

DISABLELOGIN may be set to 'Y' to prevent customers from being able to log into their account.

DISABLEOPTIONAL may be set to 'Y' to hide optional data fields.

HIDEDESCRIPTION may be set to 'Y' to hide the description value on the checkout page and receipt.

HIDEINVOICE may be set to 'Y' to hide the invoice value on the receipt.

HIDEPASSWORD may be set to 'Y' to prevent customers from being able to log into their account. This will also prevent customers from being prompted to provide a password.

RETURNPARIS may be set to 'Y' to have additional data values including BNAME, CARDTYPE, EXPMNTH, and EXPYR in the silent post response.

ENABLEREDIRECT may be set to 'Y' to force customers to be redirected to your approval/decline URL once the payment is complete.

ENABLESWIPE may be set to 'Y' to allow cardholders to swipe their cards into the checkout page.

TEST may be set to 'Y' to treat the transaction as a test.

DISPLAYTRUSTLOGO may be set to 'Y' to display a security trust logo on the checkout page.

DISABLETERMS may be set to 'Y' to hide the payment terms link and checkbox.

As this list of parameters is constantly growing, please check with Support@PayTrace.com for a current list of options.

6.5 Handling a Silent Post Transaction Response from the PayTrace API Secure Checkout

Keep in mind that the PayTrace API Secure Checkout will send the silent post transaction responses to the URL that is specified in the "Manage API Checkout" section of your PayTrace account or the Return URL that is provided when the order

is validated. The Return URL should be pointed to a page that has the code from Image 6.5. The code in Image 6.5 is written in Visual Basic Script.

Please note that Requests where TRANXTYPE is set to "CreateCustomer" or "UpdateCustomer" will only return the OrderID and CustomerID. If the customer profile was created, the CustomerID value will be the same as OrderID. However, updated customer profiles will return the existing CustomerID as the CustomerID.

```
'declare tools to loop through the strParmList and store the current name / value pair
Dim curChar as Integer
Dim curPair as String

'declare the tools to store the values of the appropriate responses
Dim strParmList As String
Dim strError As String
Dim strResponseMessage As String
Dim strTransactionID As String
Dim strAppCode As String
Dim strAppMsg As String
Dim strAVSResponse As String
Dim strCSCResponse As String

strParmList = Request.Form("ParmList")

'check to make sure the strParmList was not null
if strParmList <> "" and inStr(strParmList,"|") = true and inStr(strParmList,"~") = true then

'loop through the strParmList, one character at a time
While CurChar <= Len(strParmList)

'check if the curChar is not a "|" signifying the end of a name / value pair
If Mid(strResponse, CurChar, 1) <> "|" Then

        'store the name/value pair character in the curPair string and keep looping.
        curPair = curPair & Mid(strParmList, CurChar, 1)

Else 'the curChar is a "|" indicating we are at the end of the pair

        If UCase(Left(curPair, InStr(curPair, "~") - 1)) = "ERROR" Then

                StrError = strError & " , " & Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "RESPONSE" Then

                strResponseMessage = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "TRANSACTIONID" Then

                strTransactionID = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "APPCODE" Then

                strAppCode = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "APPMSG" Then

                strAppMsg = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = "AVSRESPONSE" Then

                strAVSResponse = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        ElseIf UCase(Left(curPair, InStr(curPair, "~") - 1)) = " CSCRESPONSE " Then

                strCSCResponse = Right(curPair, Len(curPair) - InStr(curPair, "~"))

        End If 'end if the Name of the current Pair is ERROR

curPair = ""

End If 'end if the curChar is not a "|"

'increment the current character and look at the next character in the response
CurChar = CurChar + 1

Wend 'end of the while loop that increments through the response string

Else 'the response was null

        StrError = "The response from the PayTrace API was null."

End if 'End if the response was not null
```

Image 6.5

7. Password Management

PayTrace requires that all passwords be changed at least once every 90 days. This applies to passwords used through the Virtual Terminal as well as passwords used through the API. Passwords must be changed periodically to prevent unauthorized access to your PayTrace account.

Additionally, PayTrace user accounts, including user accounts used with the PayTrace API, become disabled after 4 sequential unsuccessful log in attempts. If your user account becomes disabled, contact PayTrace to have your account enabled or go to https://paytrace.com/enableaccount.pay

PayTrace also requires that passwords are unique to the previous four passwords used for a specific user account, and passwords must contact 7 alpha-numeric digits with at least one letter and one number. PayTrace encourages our users to use special characters such as &, *, !, and $ in their passwords.

Requests sent to the PayTrace API with invalid user names, passwords, or disabled accounts will elicit an error message "ERROR~998. Log in failed.|". Your application should check for this specific error message and provide your end-users with instructions to contact you or your client to resolve the error.

Appendix A – Code Samples

All of the previous code samples provided in this user manual have been written Visual Basic Script. The following code samples have been submitted by PayTrace users to help illustrate how the PayTrace API XML Version may be implemented in other programming languages.

A.1 Using PHP and cURL() to Process a Sale

```php
<?php

//format the parameter string to process a transaction through PayTrace
$parmlist = "parmlist=UN~demo123|PSWD~demo123|TERMS~Y|";
$parmlist .= "METHOD~ProcessTranx|TRANXTYPE~Sale|";
$parmlist .= "CC~4012881888818888|EXPMNTH~01|EXPYR~09|";
$parmlist .= "AMOUNT~1.00|CSC~999|";
$parmlist .= "BADDRESS~1234 Main|BZIP~10001|";

$header = array("MIME-Version: 1.0","Content-type: application/x-www-form-urlencoded","Contenttransfer-encoding: text");

//point the cUrl to PayTrace's servers
$url = "https://paytrace.com/api/default.pay";

$ch = curl_init();

// set URL and other appropriate options
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_VERBOSE, 1);
curl_setopt ($ch, CURLOPT_PROXYTYPE, CURLPROXY_HTTP);

//Depending on your PHP Host, you may need to specify their proxy server
//curl_setopt ($ch, CURLOPT_PROXY, "http://proxyaddress:port");

curl_setopt($ch, CURLOPT_HTTPHEADER, $header);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $parmlist);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt ($ch, CURLOPT_TIMEOUT, 10);

// grab URL and pass it to the browser
$response = curl_exec($ch);

// close curl resource, and free up system resources
curl_close($ch);

//parse through the response.
$responseArr = explode('|', $response);
foreach ($responseArr as $pair ){
$tmp = explode('~',$pair);
$vars[$tmp[0]] = $tmp[1];
}

$approved = False;

//search through the name/value pairs for the APCODE
foreach($vars as $key => $value){

if ( $key == "APPCODE" ) {
    if ( $value != "" ) {
       $approved = True;
    }
}
elseif ( $key == "ERROR" ) {
        $ErrorMessage .= $value;
}
} // end for loop

if ( $ErrorMessage != "" ) {
   echo "Your transaction was not successful per this response, " . $ErrorMessage . "<br>";
   //Not approved because an error caught by PayTrace (i.e. invalid card number, amount, etc.)
}
else {
        if ( $approved == True ) {
           echo "Your transaction was successfully approved.<br>";
        }
        else {
                echo "Your transaction was not successful was not approved.<br>";
                //Not approved by issuing bank.
        } //end if transaction was approved
} //end if error message
?>
```

Image A.1

A.2 Using C++ and libCurl/cURL() to Process a Sale

```cpp
//include necessary libraries
#include <string>
#include <iostream>
#include "curl/curl.h"

using namespace std;

// Write any C++ generated errors in here
static char errorBuffer[CURL_ERROR_SIZE];

// Write all returned data in here
static string buffer;

//Simple function to create cURL instance, send request, and display response
void main(void)
{

    // Our curl objects
    CURL *curl;
    CURLcode result;

    // Create our curl handle
    curl = curl_easy_init();

    if (curl)
    {
      // Now set up all of the curl options
      curl_easy_setopt(curl, CURLOPT_ERRORBUFFER, errorBuffer);
            //point object to API directory on PayTrace network
      curl_easy_setopt(curl, CURLOPT_URL, "https://paytrace.com/api/default.pay");
      curl_easy_setopt(curl, CURLOPT_HEADER, 0);
            //trust SSL certificate w/o prompting for acceptance
      curl_easy_setopt(curl, CURLOPT_SSL_VERIFYPEER, TRUE);
            //all requests will be "posted"
      curl_easy_setopt(curl, CURLOPT_POST, true);
            //pass name value pairs of request
            //note the posted attribute is parmList and the request
            //string is formated with (~) characters delimiting the
            //name/value pairs and (|) characters delimiting each pair.
            //the last character in the string must be a (|).
      curl_easy_setopt(curl, CURLOPT_POSTFIELDS,
"parmlist=UN~demo123|PSWD~demo123|TERMS~Y|METHOD~ProcessTranx|TRANXTYPE~Sale|CC~4012881888818888|EXPMNTH~01|EXPYR~09|AMOUNT~1.00
|CSC~999|BADDRESS~1234 Main|BZIP~10001|");


        // Post the request and catch the response
        result = curl_easy_perform(curl);

        // cleanup CURL object
        curl_easy_cleanup(curl);

        // Did we get a response?
        if (result == CURLE_OK)
        {
                //output the raw response...
                        //parse response/error to determine if transaction was approved
                        //all responses will contain one or more errors or the response
                        //package specified in the user guide for the request method.
                        //responses contain the same name/value pair formatting as the request.
          cout << buffer << "\n";
          exit(0);
        }
        else
                //response was not received
        {
          cout << "Error: [" << result << "] - " << errorBuffer;
          exit(-1);
        } //end if (result == CURLE_OK)
    } //end if (curl)

} //end int main()
```

Image A.2

Appendix B – How to Determine if a Transaction Has Been Approved and Should Be Settled

Any approved transaction (transaction whose approval code is not equal to the empty string) may be settled. However, an approved transaction does not guarantee that the customer who provided the billing information is truly the card holder. Preventing fraud is paramount in the payment processing industry for multiple reasons, primarily minimizing merchant exposure to chargebacks and strengthening customer confidence in electronic payments.

Depending on your style of business and potential for chargeback exposure, PayTrace encourages you to validate each transaction's fraud indicators, in addition to approval response, to verify if a transaction is legitimate and should be settled.

AVS (Address Verification System) and CSC (Card Security Code) Responses are excellent indicators that may be used to help verify that your customer is the true card holder. Therefore, PayTrace encourages you and your application to validate the AVS and CSC Responses against the following potential responses to determine if the appropriate fraud prevention features have been met before settling a transaction.

AVS Responses

- Full Exact Match
- Address Match Only
- Zip Match Only
- No Match
- Address Unavailable
- Non-US Issuer does not participate
- Issuer System Unavailable
- Not a Mail/Phone Order

- Service Not Supported

CSC Responses*

- Match
- No Match
- Not Processed
- Not Present
- Issuer Does Not Support CSC

*If you are processing an American Express transaction (15 digit card number starts with 37) and you provide a CSC value, the CSC Response will be empty. Note that American Express does not approve transactions whose CSC value is invalid. Visa, MasterCard, Discover, and Diner's Club will return a CSC Response. JCB does not support CSC.

PayTrace strongly encourages you to evaluate each transaction's AVS and CSC Responses to determine if the transaction should be settled. If you ever have a question about a transaction or are unsure if possible fraud is taking place, **please contact PayTrace or your merchant service provider immediately.**