MerchantPartners

Online Commerce Suite™
Integration Guide

Release 7.5
March 2013

Merchant Partners
11121 Willows Rd NE
Suite 120
Redmond, WA 98052

Table of Contents

Table of Contents

What's in This Guide

The Online Commerce Suite Integration Guide is specialized to meet the needs of your operational, technical, and accounting staff. This guide will help your users to quickly familiarize themselves with the application to make full use of its many powerful tools and to maximize the profitability of your e-commerce operation.

How to Comment on This Guide

Every effort has been made to produce an accurate and easy to understand the Integration Guide.

Contact Information

For more information about Online Commerce Suite, refer to the following:

Customer Service

If you have problems with this document, or find that the instructions are incorrect, incomplete, or inadequate, please let us know.

Send your comments to support@merchantpartners.com

Phone: (866) 242-9933

Additional Documentation

The full suite of documentation is available at

https://www.onlinemerchantcenter.com/mpartners/html/user_manuals.html

Certified Networks

For a complete list of certified networks please visit

https://www.onlinemerchantcenter.com/mpartners/html/networks.html

Compatible Hardware

For a complete list of compatible hardware please visit

https://www.onlinemerchantcenter.com/mpartners/html/equipment.html

Introduction

Welcome to the Online Commerce Suite system. Online Commerce Suite is a Web-based payment gateway that allows you to process secure credit card and electronic check payments for goods and services over the Internet. Using the Online Merchant Center™ web-based administrative user interface, you can configure your Online Commerce Suite account, add users, and manage your e-business. Online Commerce Suite provides a comprehensive set of online and downloadable transaction management and accounting reports.

Setting Up an Online Commerce Suite Account

The first step in setting up your Online Commerce Suite account is to contact Customer Service to complete your registration by telephone. When your account is confirmed and set up, you will receive a five character Online Commerce Suite Account ID (Acct ID). This ID identifies your account in the Online Commerce Suite system and allows the system to authenticate transactions originating from you. Be sure to include your Online Commerce Suite Acct ID in all correspondence with Customer Service.

See the companion *Getting Started Guide* for more information about setting up your account.

Note: Some accounts have the Merchant PIN transaction security option enabled by default, in which case you will not be able to process transactions using the API unless a valid Merchant PIN is included in your transactions. To view the status of the Merchant PIN option on your account, login to the Online Merchant Center and select the FRISK™ "Configure Options" menu item.

Integrating your System with Online Commerce Suite

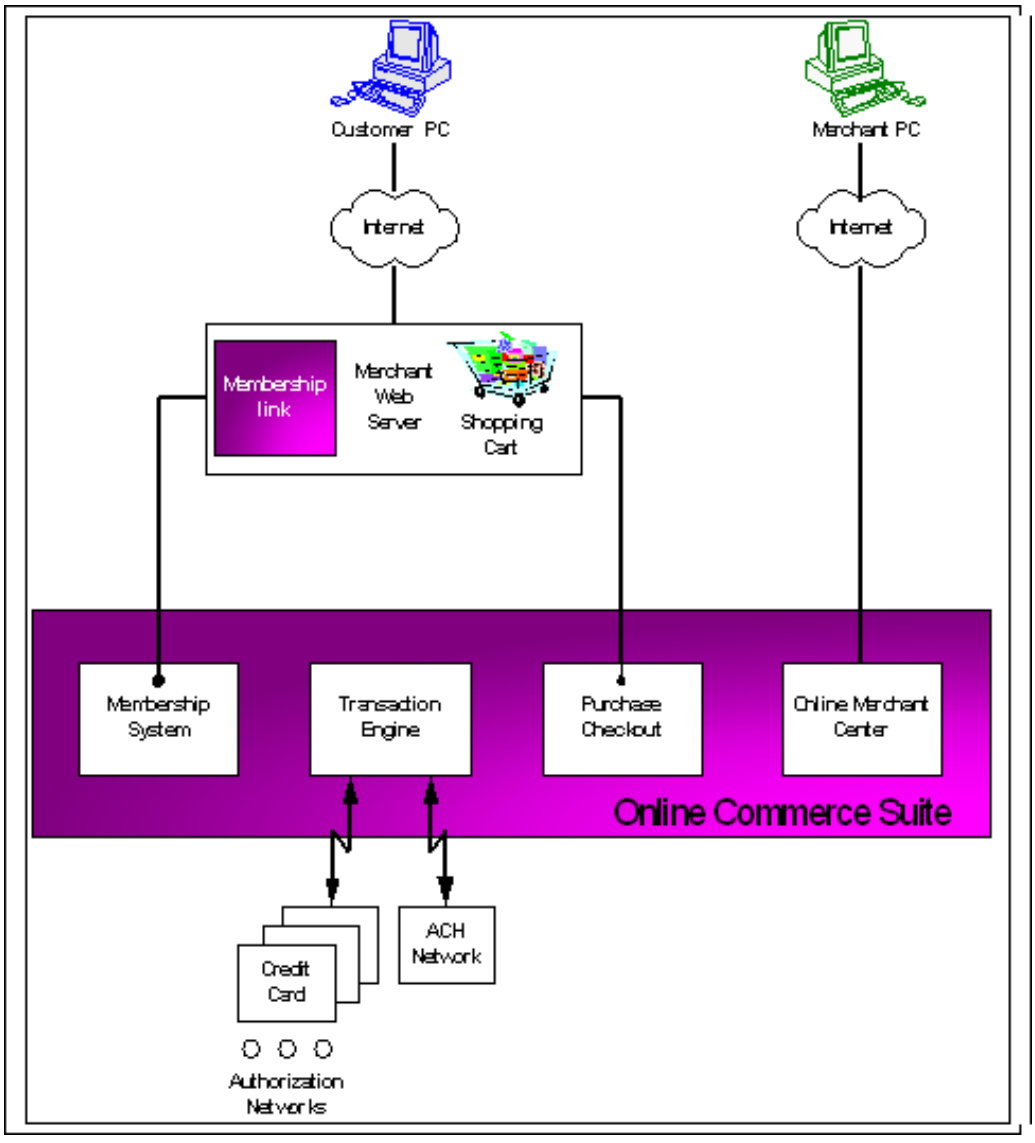
Your e-commerce goals determine the transaction processing method you will use with Online Commerce Suite. Various factors make a difference how you integrate Online Commerce Suite with your e-commerce business, whether you offer products or subscriptions or both. Factors include:

- Do you use Shopping Cart software?
- Did you develop your own Web site?
- Is your Web site hosted on a secure server?
- Do you want Online Commerce Suite to calculate shipping and tax?
- Do you have your own database to track your inventory and business?

The answers to these questions determine the complexity of your integration with Online Commerce Suite. Depending on your requirements, integration can be very straightforward or may require a sophisticated understanding of HTML, CGI, ASP or other Web technologies.

Integration Overview

The Online Commerce Suite system processes secure on-line credit cards and electronic check payments. The authorization network secures credit card payment authorizations while the Automated Clearing House (ACH) network processes electronic checks. Online Commerce Suite automatically selects the appropriate payment network based upon the requested transaction type (credit card or ACH) and the authorizing network or Federal Reserve Bank region supported by the bank where you have a merchant account. The following image is a conceptual schematic of the Online Commerce Suite system.



Consumer PC

An online customer accesses your Merchant Web Server through the Internet to initiate the shopping experience. Typically, items are selected for purchase and placed in a virtual Shopping Cart. When the customer's purchase decisions are completed and confirmed, the Web Link module is activated. Web Link uses preloaded parameters like price, appropriate tax rate, item weight, and available shipping options to calculate the total amount due for the transaction. This calculated charge is passed back to the on-line customer as an invoice for their approval. The approved invoice activates the Transaction Engine to secure payment authorization through the appropriate authorization network.

Merchant Web Server

The Merchant Web Server supports the customer's shopping and purchasing experiences. It may host its own Shopping Cart and link to the Web Link and Transaction Engine modules for payment processing.

Shopping Cart

Shopping Cart programs are used to collect customer purchase and product pricing information. The Shopping Cart program resides on the Merchant's web server and is typically provided by a third party or ISP. Online Commerce Suite also provides a hosted shopping cart solution (the Web Link) for those merchants who do not have a shopping cart or do not want to host their own cart. The Transaction Engine uses the information collected by the Shopping Cart to determine the amount for account debit authorizations.

Merchant PC

The Merchant connects to the Online Merchant Center module through the Internet. Using this interface, the Merchant may collect transaction, accounting, and customer information reports.

Online Commerce Suite System Modules

Online Commerce Suite modules accomplish the following functions:

Web Link completes the customer purchase experience by collecting payment information from the shopping cart, calculating the amount to be charged, and passing this information to the Transaction Engine for processing.

Transaction Engine processes online, batch, recurring, and membership subscription payment authorization requests. Transactions are processed immediately in real time or in batch. Within seconds, consumers receive an acceptance or decline notification. Funds from accepted credit card transactions are deposited into your merchant bank account, typically within 24 hours. Funds from accepted electronic check (ACH) transactions are deposited into your checking account within six business days.

Online Merchant Center provides merchant account management functions, virtual terminal transaction processing, and reports.

Transaction Security

Payment processing requires mechanisms using encryption to protect customer payment information. A customer's private information, especially credit card numbers and bank account numbers, are securely encrypted as they are transmitted over the Internet.

The three most common data encryption methods are:

- SSL (Secure Sockets Layer)
- Triple DES (Data Encryption Standard).
- PGP (Pretty Good Privacy)

SSL Encryption

Secure Socket Layer (SSL) uses a public key to provide encryption between the host server and client browser and is the most secure encryption method. Your site must be hosted on a secure server running SSL.

When Internet transmissions are made via SSL, the protocol for the Uniform Resource Locator (URL) address must include HTTPS, rather than HTTP to direct the transmission to the secure SSL port. The SSL public key encryption

system works this way. The receiving computer discloses its public key and any other computer can use that public key to encrypt data that it sends to the receiving computer.

While the public key empowers anyone to encrypt a message, decryption is **not**possible on the basis of the public key. Only the receiving computer has the ability to decrypt, therefore, there is no need to distribute or store private keys, which may fall into the wrong hands.

Triple DES Encryption

Triple DES (3DES) utilizes a private key generated by the gateway and used by the merchant to encrypt account information. Both the transmitting computer and the receiving computer must possess the same private key to encrypt and then decrypt the message.

To enable the 3DES option on your account, you must generate a key using the FRISK Configuration menu in the Online Merchant Center, and then use that key to encrypt your data before transmitting it to the gateway. Sample C++ source code is available for the recommended implementation of the Triple DES algorithm.

Pretty Good Privacy

For secure transmission of batch transactions, you can submit files encrypted using PGP (Pretty Good Privacy), a registered trademark of Network Associates, Inc. PGP is a widely accepted and trusted method of data encryption that utilizes public key encryption technology.

Public key encryption uses two complimentary keys: one public and one private, to implement secure communications. The public key can be distributed freely to other users for encrypting files but cannot be used for decryption. The private key is kept by the user who is to receive the files, and must be used to decrypt files secured with the public key.

For more information about sending and receiving PGP encrypted transaction files, refer to the *Batch Processing Guide*.

Merchant PIN

In addition to encryption, a unique Merchant PIN (or password) can be required for each merchant account. When the Merchant PIN option is enabled, transactions submitted that do not have a valid matching Merchant PIN will be rejected.

To enable the Merchant PIN option on your account, login to the Online Merchant Center and select the Merchant PIN option under the FRISK "Configuration Options" menu. A 32 character alphanumeric key will be displayed, which will need to be passed in the `merchantpin` field in each transaction.

Note: Some accounts have this option enabled by default, in which case you will not be able to process transactions unless a valid Merchant PIN is included in your transactions. To view the status of the Merchant PIN option on your account, login to the Online Merchant Center and select the FRISK "Configure Options" menu.

Online Transaction Processing

The Online Commerce Suite system allows you to use one of the following transaction processing methods:

- Quicksale Method
  - Web form
  - Socket Connection
  - SecurePost
- Batch Payment Submission
  - via FTP Upload
  - via Email
- Membership

Each method is a distinct way to connect your system and your customer's Web browser to the Online Commerce Suite system and the banking system. You must choose a method when you sign up. We recommend you read the entire guide and examine the following table before you decide which method to use.

Method	OCS URL visible?	Transaction Security	Difficulty	Interface
Quicksale via Web form	Yes	You must provide a secure server.	Moderate	HTML
Quicksale via Socket Connection	Hidden	You must provide a secure server.	Difficult	HTTPS
Quicksale via SecurePost	Hidden	Inherent in method	Difficult	COM object
Batch Submission via FTP	Hidden	Need to use PGP	Moderate	FTP

Alternate Integration Options

For merchants who do not have integration programming expertise, the following hosted services are available:

Web Link

Web Link is a simple hosted shopping cart checkout service that provides a basic, easy to use way for merchants to sell a few items from their web site. It is intended to be used when your customers will be purchasing only one or two products at a time, and has a limited number of configurable options. For detailed instructions, refer to the *Web Link Guide* which can be downloaded from the documentation site.

Web Cart

Web Cart is a full featured shopping cart and order processing system that's powerful yet simple to use. There is no software to buy or install. All you need to do is add a few "ADD TO CART" buttons that link your web site to our secure servers, and we take care of the rest. No additional changes need to be made to the server where your website is hosted. For more information regarding Web Cart, download the *Web Cart Guide* from the documentation site.

Membership

Membership is a turnkey solution for managing password protected subscription or Web membership sites. Consumers join to access your Member Only areas. Online Commerce Suite allows you to set up charges on a recurring basis. Technical Support remotely installs necessary software on you server and your programmers provide a few links to appropriate pages on your Web site. Please refer to the *Membership Guide* for more information, or contact Customer Service.

Batch Processing

Batch processing is oriented toward the non-interactive approach to data processing. Your system accumulates a number of transaction requests (a batch), submits them all for processing, and then gets a return batch of transaction results. This is not a good approach if your customer is waiting online, but it is an excellent way to process a large number of recurring billings at the end of the month. Please refer to the *Batch Processing Guide* for more information.

Quicksale Method

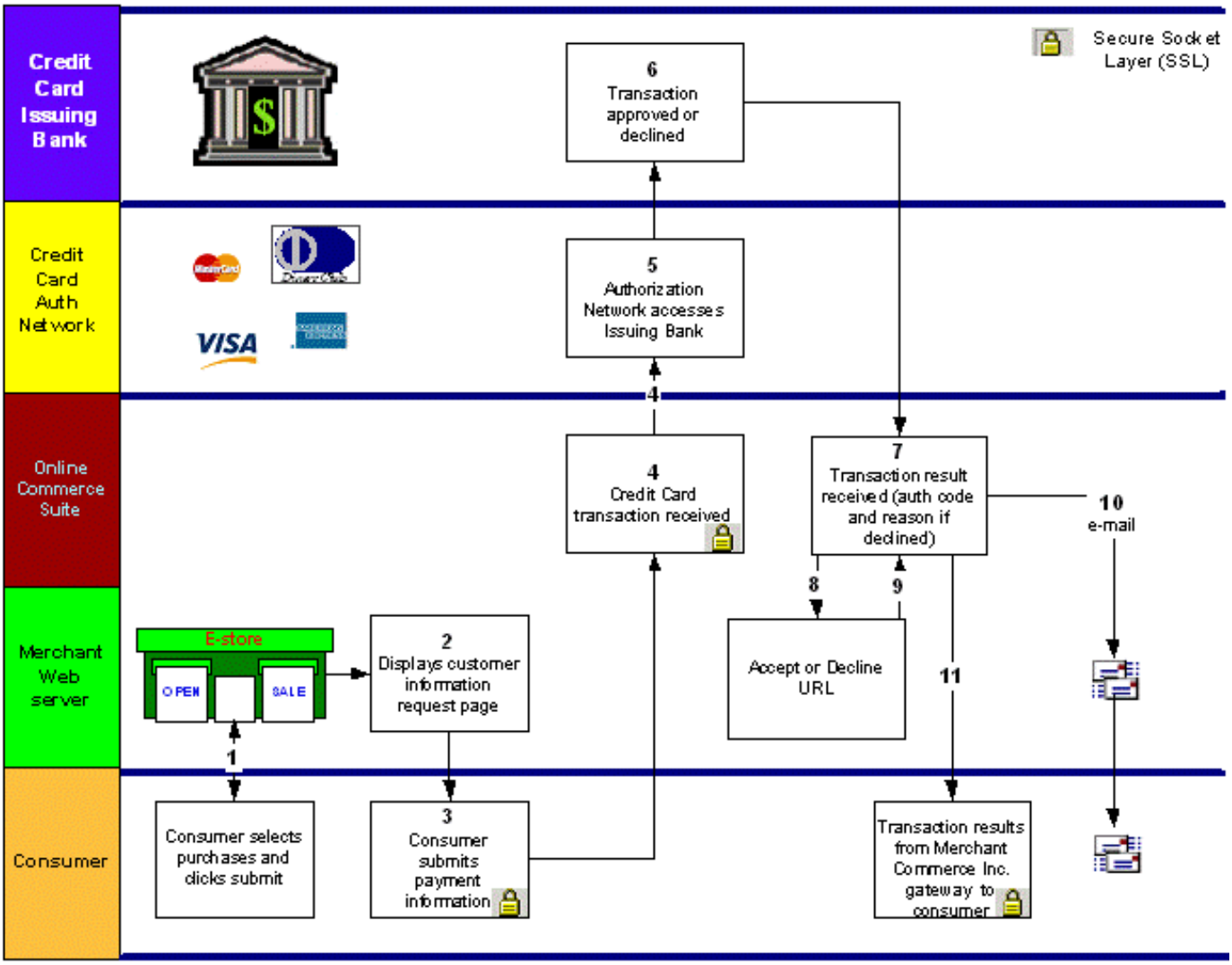
The Quicksale Method gives you almost complete control over what your customers see on their browsers. Your programmers can customize your transaction pages with CGI scripts, Active Server Pages, Cold Fusion applications, and so on. This method requires you to be on a secure server (with SSL installed).

Your system gathers customer information and calculates shipping and tax in a manner controlled by your programmers. Online Commerce Suite performs the final step: processing the transaction and telling your system the result of the transaction.

How It Works

1. Your system displays the transaction request page, which asks your customer to supply either a credit card number or bank account number, to your customer's browser.
2. After your customer submits the account information, the data is securely transferred to Online Commerce Suite, which processes the transaction request.
3. If the transaction is accepted, Online Commerce Suite generates a receipt for the transaction and e-mails it to you and your customer. You can also configure the system to **not** send these receipt emails.
4. Online Commerce Suite securely sends the transaction result data to your system. At minimum, the data contains the accept/decline flag, the authorization code if accepted and the reason if declined.
5. Your programmers can take advantage of additional secure transaction data that Online Commerce Suite can provide in this step by including the `usepost=1` parameter in the transaction request.
6. Your system sends a transaction result page from either your acceptur l or your declineurl to Online Commerce Suite. Your programmers can use CGI or other methods to dynamically create this page.
7. Online Commerce Suite passes the transaction result page to your customer's browser.

The following figure illustrates the Quicksale payment process.



Security

The most sensitive transaction data like the credit card number or bank account number is transmitted directly from the customer's browser to the Online Commerce Suite server via SSL. Customer data received by your system prior to asking the customer for a credit card number or bank account number (Step 1), which might be used to create the transaction request page, is also transmitted securely because you have installed SSL on your server.

You can also incorporate the Merchant PIN and/or 3DES option to further enhance security.

Getting Information About Your Customers

There are four ways to receive information about your customer's order:

1. Data your system obtains from your customer prior to your system asking the customer for a credit card number or bank account number (Step 1). Remember, this data can only be securely obtained if you have a secure server.
2. E-mail automatically generated and sent to you at the time of the transaction. Credit card and bank account numbers are omitted from email receipts. Transaction receipts can be configured to be sent to the consumer, the merchant, or both. To configure transaction receipt options, login to the Online Merchant Center and select the "Account Profile" menu.
3. Data included at the time of the transaction, which your system can use to update your database. This data is only sent if your programmers set the **usepost** flag. Sensitive information such as consumer credit card numbers and account numbers are not sent back.
4. Logging into the Online Merchant Center and using the Transaction menu to generate detailed transaction reports.

What Your Programmers Do

When you use the Quicksale Method, your programmers create all of the pages leading up to and including the final transaction request page, updating your local database in the process with customer information. This transaction request page is transferred from your system to your customer's browser, where your customer supplies the credit card or bank account number. The transaction request is then submitted from your custom designed transaction request page in your customer's browser directly to Online Commerce Suite. Results are reported to your system, and your programmers provide a page for the final transaction results to display.

To create both the transaction request page and the transaction results pages, your programmers can use Web technologies, including CGI scripts, Active Server Pages, Cold Fusion applications, and so on.

Transaction Types

The following table defines the transaction types supported by the Quicksale Method.

Action	Description
NS_QUICKSALE_CC	Process a credit card transaction. The default transaction is a sale, but options can be included to process a pre-auth or post-auth.
NS_QUICKSALE_CHECK	Process an ACH (electronic check) transaction.
NS_CREDIT	Refund credit card or ACH funds back to the consumer. A previous approved sale transaction is required, unless stand-alone credits have been enabled on the AcctID. To enable stand-alone credits, the IP address of the server submitting the credits needs to be registered with the gateway using the FRISK menu option in the Online Merchant Center.
NS_VOID	Void an existing credit card or ACH transaction. Voids are not available on all authnet platforms, and can only be performed on transactions that have not yet settled. If a transaction has settled, a Credit must be issued.
EXTACH_SALE	Process an Extended ACH sale transaction via a 3rd party check processing service provider.
EXTACH_CONSUMERDISBURSEMENT	Process an Extended ACH consumer payment transaction.
EXTACH_REFUND	Refund a prior Extended ACH sale.
EXTACH_VOID	Void a prior Extended ACH sale.

EXTACH_CK21SALE	Process a Check21 sale.
EXTACH_CK21VOID	Void a prior Check21 sale.
EXTACH_CK21REFUND	Refund a prior Check21 sale.
RECUR_ADD	Add/import a recurring billing record.
RECUR_UPDATE	Update a consumer's billing information and recurring parameters for an existing recurring record.
RECUR_CANCEL	Cancel a recurring billing record.

Credit Card Transactions

The following tables define the required and optional fields related to credit card transactions.

Credit Card Sale/Auth

Field	Required	Description
action	•	`ns_quicksale_cc`
acctid	•	5-character alphanumeric Account ID assigned to the merchant. Use `TEST0` for testing if you do not have an Account ID. Change to your Account ID for live transaction processing.
subid		Merchant Sub ID.
merchantpin		Merchant PIN defined in the Online Merchant Center FRISK Options. If the Merchant PIN option has been enabled for the merchant, transactions without a valid `merchantpin` will be declined with a decline response of `DECLINED:1101150001:DECLINED`.
amount	•	Transaction amount in `0.00` format. Do not include currency symbols or commas.
ccname	•	Consumer name as it appears on the credit card.
ccnum	•	Consumer credit card number. Do not include spaces or dashes. Use `5454545454545454` or `4111111111111111` for testing. If the 3DES encryption option has been enabled in the Online Merchant Center FRISK Options, this value must be encrypted using the 3DES key assigned to the AcctID.
expmon	•	Expiration month (1 - 12) of the consumer credit card.
expyear	•	Expiration year of the consumer credit card in yyyy format.
cvv2		Credit card cvv2/cvc2 code
authonly		A value of `1` pre-authorizes the credit card. A pre-auth will "reserve" the amount specified in the amount field, it will not actually bill the consumer's credit card. This process is used for Book and Ship sales transactions, where a Merchant gets an order and at a later date, completes the transfer of funds.
accepturl		The URL to be displayed to the consumer if the transaction is processed and approved. Example: `http://example.com/sale/accepted.html` The URLs specified in `accepturl` and `declineurl` will always be called by a connection originating from trans.merchantpartners.com. Do **not** set the return URL to https:
declineurl		

		The URL to be displayed to the consumer if the transaction is declined. Example: `http://example.com/sale/declined.html` Do **not** set the return URL to https:
usepost		A value of `1` will return order form information to your `accepturl` or `declineurl`.
merchantordernumber		A unique alphanumeric identifier of your own by which you can reference the order.
ci_companyname		Consumer company name
ci_billaddr1		Consumer billing address
ci_billaddr2		Second line of the consumer billing address
ci_billcity		Consumer city
ci_billstate		Consumer state or province
ci_billzip		Consumer Zip code or Postal code
ci_billcountry		Consumer country (refer to Appendix D)
ci_shipaddr1		Consumer shipping address
ci_shipaddr2		Second line of consumer shipping address
ci_shipcity		Consumer shipping city
ci_shipstate		Consumer shipping state or province
ci_shipzip		Consumer shipping Zip Code or Postal Code
ci_shipcountry		Consumer shipping country (refer to Appendix D)
ci_phone		Consumer phone number
ci_email		Consumer email address
ci_memo		Miscellaneous information field
ci_dlnum		Consumer driver license number
ci_ssnum		Consumer Social Security Number
ci_ipaddress		Consumer's IP Address
currencycode		Three digit currency code (refer to Appendix D)
dynamicdescriptor		This field will be used on the consumer's credit card statement. This is only available for TSYS.
Custom Receipt Email		
emailto		Email address to send the consumer email receipt. Default is the `ci_email` value.
emailfrom		Return email address on consumer's email receipt. Default email address is `null@atsbank.com`.
emailsubject		Subject line on consumer's receipt email. Default subject is *Payment Receipt #xyz*.
emailtext		Consumer's receipt email body text. Default is a generic receipt message.
FSA		
healthcareflag		A value of `1` indicates an FSA purchase.
rxamount		For an FSA purchase, this value specifies the qualified prescription amount.
Purchase Card Level II		
pocustomerrefid		A purchase order number or invoice number.

taxamount		Total tax charged for the transaction.
taxexempt		`1` = a tax exempt purchase, `0` = a taxable purchase.
AMEX Commercial Card		
supplierrefnum		A reference number that helps the merchant identify the charge in case of an inquiry by the credit card company.
cardholderrefnum		The card holder reference number. Similar to PO #.
salestax		Sales tax amount
chargedescriptor		Free form field describing the charge.
Visa/MC Commercial Card		
optionalamountid		A code describing the `optionalamount` value. `0` = Not used `1` = Local Sales Tax Amount `2` = Tax Exempt
optionalamount		An optional amount as described by the `optionalamountid` value.
Merchant Custom Fields		
custom1		Merchant's custom field.
custom2		Merchant's custom field.
custom3		Merchant's custom field.
custom4		Merchant's custom field.
custom5		Merchant's custom field.
custom6		Merchant's custom field.
Recurring		
recur_create		A value of `1` creates a recurring billing record for the consumer.
recur_billingcycle		`0` = No Recurring Billing Cycle `1` = Weekly Recurring Billing Cycle `2` = Monthly Recurring Billing Cycle `3` = Quarterly Recurring Billing Cycle `4` = Semi-Annual Recurring Billing Cycle `5` = Annual Recurring Billing Cycle `6` = Bi-Weekly Recurring Billing Cycle `7` = Bi-Annual Recurring Billing Cycle `8` = Quad Weekly (28 day) Recurring Billing Cycle `9` = One Time Recurring Billing Cycle `10` = Daily Recurring Billing Cycle `11` = Bi-Monthly Recurring Billing Cycle
recur_billingmax		Maximum number of times a consumer's account is debited through recurring billing. For example, `recur_billingmax=6` bills the consumer 6 times. -1 = Unlimited number of times 0 = No Recurring Billing
recur_start		Number of days after an initial payment where the consumer is debited on a recurring cycle.
recur_amount		Amount the consumer is to be re-debited on the recurring cycle. **Do not include a dollar sign.**

Credit Card Post/Capture

Field	Required	Description
action	•	`ns_quicksale_cc`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN
amount	•	Transaction amount in the format 0.00.
postonly	•	RefCode or HistoryID of original pre-auth transaction. This value is returned in the response fields of an approved pre-auth transaction.

Credit Card Refund

Field	Required	Description
action	•	`ns_credit`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN
amount	•	Transaction amount in the format 0.00.
orderkeyid	•	Order ID of the original transaction
historykeyid	•	History ID of the original transaction

Note: The ability to process credits is disabled by default. To enable this option on your account, use the FRISK Configuration menu in the Online Merchant Center.

Credit Card Void

Field	Required	Description
action	•	`ns_void`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN
amount	•	Transaction amount in the format 0.00.
orderkeyid	•	Order ID of the original transaction
historykeyid	•	History ID of the original transaction

ACH Transactions

The following tables define the required and optional fields related to ACH transactions.

ACH Sale

Field	Required	Description
action	•	`ns_quicksale_check`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN defined in the Online Merchant Center FRISK Options.
amount	•	Transaction amount in the format 0.00. Do not include currency symbol or commas.
ckaba	•	Nine-digit numeric value without spaces for checking account routing or ABA number.
ckacct	•	Variable length numeric value without spaces for checking account number. If the 3DES encryption option has been enabled on the account, this value must be encrypted using the 3DES key assigned to the AcctID. Use the FRISK "Configuration Options" menu in the Online Merchant Center to view and configure the 3DES encryption option.
ckno	•	Check number
ckaccttype		"1" - Checking, "2" - Savings
cktype		SEC Code: "POP" or "ARC" or "TEL" or "PPD" or "ICL" or "RCK" or "BOC".
accepturl		The URL to be displayed to the consumer if the transaction is processed and approved. Example: `http://example.com/sale/accepted.html` The URLs specified in `accepturl` and `declineurl` will always be called by a connection originating from trans.merchantpartners.com. Do **not** set the return URL to https:
declineurl		The URL to be displayed to the consumer if the transaction is declined. Example: `http://example.com/sale/declined.html` Do **not** set the return URL to https:
usepost		A value of `1` will return order form information to your `accepturl` or `declineurl`.
ci_companyname		Consumer company name
ci_billaddr1		Consumer billing address
ci_billaddr2		Second line of the consumer billing address
ci_billcity		Consumer city
ci_billstate		Consumer state or province
ci_billzip		Consumer Zip code or Postal code
ci_billcountry		Consumer country (refer to Appendix D)
ci_shipaddr1		Consumer shipping address
ci_shipaddr2		Consumer second line of shipping address
ci_shipcity		Consumer shipping city
ci_shipstate		Consumer shipping state or province
ci_shipzip		Consumer shipping Zip Code or Postal Code
ci_shipcountry		Consumer shipping country (refer to Appendix D)

ci_phone		Consumer phone number
ci_email		Consumer email address
ci_memo		Miscellaneous information field
ci_dlnum		Consumer driver license number
ci_ssnum		Consumer Social Security Number
emailto		Email address to send the consumer email receipt. Default is the `ci_email` value.
emailfrom		Return email address on consumer's email receipt. Default email address is `null@atsbank.com`.
emailsubject		Subject line on consumer's receipt email. Default subject is *Payment Receipt #xyz*.
emailtext		Consumer's receipt email body text. Default is a generic receipt message.
ci_ipaddress		Consumer's IP Address
merchantordernumber		A unique alpha-numeric identifier of your own by which you can reference the order.
custom1		Merchant's custom field.
custom2		Merchant's custom field.
custom3		Merchant's custom field.
custom4		Merchant's custom field.
custom5		Merchant's custom field.
custom6		Merchant's custom field.
recur_create		A value of `1` creates a recurring billing record for the consumer.
recur_billingcycle		`0` = No Recurring Billing Cycle `1` = Weekly Recurring Billing Cycle `2` = Monthly Recurring Billing Cycle `3` = Quarterly Recurring Billing Cycle `4` = Semi-Annual Recurring Billing Cycle `5` = Annual Recurring Billing Cycle `6` = Bi-Weekly Recurring Billing Cycle `7` = Bi-Annual Recurring Billing Cycle `8` = Quad Weekly (28 day) Recurring Billing Cycle `9` = One Time Recurring Billing Cycle `10` = Daily Recurring Billing Cycle `11` = Bi-Monthly Recurring Billing Cycle
recur_billingmax		Maximum number of times a consumer's account is debited through recurring billing. For example, `recur_billingmax=6` bills the consumer 6 times. -1 = Unlimited number of times 0 = No Recurring Billing
recur_start		Number of days after an initial payment where the consumer is debited on a recurring cycle.
recur_amount		Amount the consumer is to be re-debited on the recurring cycle. **Do not include a dollar sign.**

ACH Refund

Field	Required	Description
action	•	`ns_credit`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN
amount	•	Transaction amount in the format 0.00.
orderkeyid	•	Order ID of the original transaction
historykeyid	•	History ID of the original transaction

ACH Void

Field	Required	Description
action	•	`ns_void`
acctid	•	5-character alphanumeric Account ID assigned to the merchant
subid		Merchant Sub ID
merchantpin		Merchant PIN
amount	•	Transaction amount in the format 0.00.
orderkeyid	•	Order ID of the original transaction
historykeyid	•	History ID of the original transaction

ExtACH Transactions

The following tables define the required and optional fields related to ExtACH transactions, also known as 3rd Party Check Processing Service Providers.

ExtACH Sale

Field	Required	Description
action	•	`EXTACH_SALE`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
amount	•	Transaction amount in the format 0.00.
acctname	•	Consumer name as it appears on the checking account.
ckaba	•	Nine-digit numeric value without spaces for checking account routing or ABA number.
ckacct	•	Variable length numeric value without spaces for checking account number. If the 3DES encryption option has been enabled on the account, this value must be

		encrypted using the 3DES key assigned to the AcctID. Use the FRISK "Configuration Options" menu in the Online Merchant Center to view and configure the 3DES encryption option.
accounttypedesc	•	ACH account type description. One of: Personal Checking Personal Saving Business Checking Business Saving
achtransactiontype	•	SEC code for transaction (WEB, POP, ARC, PPD, ICL, RCK, BOC, TEL)
ckno	•	Check number (Required for TEL, RCK, BOC, ARC)
acctdata3		ACH account type specification. 1 = Checking (default if not provided) 2 = Savings
accepturl		The consumer is transferred to this URL after the transaction is approved Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/accepted.html
declineurl		The consumer is transferred to this URL after the transaction is declined. Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/declined.html
ci_companyname		Your company name
ci_billaddr1		Consumer billing address
ci_billaddr2		Second line of the consumer billing address
ci_billcity		Consumer city
ci_billstate		Consumer state or province
ci_billzip		Consumer zip code or postal code
ci_billcountry		Consumer country
ci_shipaddr1		Consumer shipping address
ci_shipaddr2		Second line of consumer shipping address
ci_shipcity		Consumer shipping city
ci_shipstate		Consumer shipping state or province
ci_shipzip		Consumer shipping zip or postal code
ci_shipcountry		Consumer shipping country
ci_phone		Consumer phone number
ci_email		Consumer's e-mail address
ci_memo		Miscellaneous information field
ci_dlnum		Consumer driver license number
ci_ssnum		Consumer Social Security number
emailto		E-mail address where to send consumer's e-mail receipt. Default is ci_email
emailfrom		Return address on consumer's e-mail receipt. Default is null@atsbank.com.
emailsubject		Subject line on the consumer's receipt e-mail. Default is *Payment Receipt #xzy*.
emailtext		Text for consumer's e-mail receipt. Default is a generic receipt message.

ci_ipaddress		Consumer's IP Address
merchantordernumber		Customer's unique alpha-numeric number
custom1		Custom field for information to be included with the transaction.
custom2		Custom field for information to be included with the transaction.
custom3		Custom field for information to be included with the transaction.
custom4		Custom field for information to be included with the transaction.
custom5		Custom field for information to be included with the transaction.
custom6		Custom field for information to be included with the transaction.
currencycode		Three digit currency code 'USD' for US$ (Refer to Appendix: D for complete list)

ExtACH Verification

Field	Required	Description
action	•	EXTACH_SALE
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
VerificationFlag	•	A value of "1" is required to activate Verification.
acctname	•	Consumer name as it appears on the checking account.
ckaba	•	Nine-digit numeric value without spaces for checking account routing or ABA number.
ckacct	•	Variable length numeric value without spaces for checking account number. If the 3DES encryption option has been enabled on the account, this value must be encrypted using the 3DES key assigned to the AcctID. Use the FRISK "Configuration Options" menu in the Online Merchant Center to view and configure the 3DES encryption option.
ckno	•	Check number
ci_phone	•	Consumer phone number (10 numeric character)
ci_dlnum	•	Consumer driver license number (The first two characters of this field should be the State Code of the DL, followed by a dash (ASCII 45), then the ID Data.
ci_ssnum	•	Consumer Social Security number (9 numeric characters)

ExtACH Refund

Field	Required	Description
action	•	EXTACH_REFUND
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
amount	•	Transaction dollar amount in US dollars in the form of 0.00. If not supplied defaults to the full amount of the original transaction

orderkeyid	•	Order ID of original transaction
historykeyid	•	History ID of original transaction

ExtACH Void

Field	Required	Description
action	•	`EXTACH_VOID`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
amount	•	Transaction dollar amount in US dollars in the form of 0.00.
orderkeyid	•	Order ID of original transaction
historykeyid	•	History ID of original transaction

Check21 Transactions

The following tables define the required and optional fields related to Check21 transactions.

Check21 Sale

Field	Required	Description
action	•	`EXTACH_CK21SALE`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
amount	•	Transaction amount in the format 0.00.
ckname	•	Name on account - IF ckaccttype is "P" Should be the exact wording as the consumer's bank knows it. For example, “Jonathan Doe”, “John and Jane Doe”, “Mr. and Mrs. Doe.” If field is omitted, then the person specified in the first name and last name will be used.
ci_firstname	•	Signatory First Name. The first name of an authorized signatory (one person only).
ci_lastname	•	Signatory Last Name. The last name of a authorized signatory (one person only).
acctname	•	Consumer name as it appears on the checking account.
ckaba	•	Nine-digit numeric value without spaces for checking account routing or ABA number.
ckno	•	Check number
ckacct	•	Variable length numeric value without spaces for checking account number.
ckaccttype	•	"P" - Personal, "B" - Business
ci_companyname	•	Company name if `ckaccttype` is `B`

ci_billhousenumber	•	Payer's house number
ci_billstreet	•	Payer's Street
accepturl		The consumer is transferred to this URL after the transaction is approved Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/accepted.html
declineurl		The consumer is transferred to this URL after the transaction is declined. Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/declined.html
ci_billaddr1	•	Consumer billing address
ci_billaddr2		Second line of the consumer billing address
ci_billcity	•	Consumer city
ci_billstate	•	Consumer state or province
ci_billzip	•	Consumer zip code or postal code
ci_billcountry	•	Consumer country
ci_phone	•	Consumer phone number
ci_email	•	Consumer's e-mail address
ci_billzip4		Payers zip code. USA ZIP+4 format
currencycode		Three digit currency code 'USD' for US$ (Refer to Appendix: D for complete list)
riskmodifier		Extended risk check of the check routing number and account number can be modified by this field 0 disables risk checks 1 forces risk checks. If not specified, the contract default will be used, according to a threshold. *This feature might not be available.
authenticationmodifier		Authentication of the payer name and address can be modified by this field 0 disables payer authentication checks 1 forces payer authentication checks. If not specified, the contract default will be used, according to a threshold. *This feature might not be available.
insurancemodifier		Insurance that the amount will clear can be modified by this field 0 disables insurance 1 forces insurance. If not specified, the contract default will be used, according to a threshold. *This feature might not be available.
accepturl		The consumer is transferred to this URL after the transaction is approved Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/accepted.html
declineurl		The consumer is transferred to this URL after the transaction is declined. Point to a CGI script or html page like http://trans.merchantpartners.com/~ats/declined.html
ci_shipaddr1		Consumer shipping address
ci_shipaddr2		Second line of consumer shipping address
ci_shipcity		Consumer shipping city
ci_shipstate		Consumer shipping state or province
ci_shipzip		Consumer shipping zip or postal code
ci_shipcountry		Consumer shipping country

ci_memo		Miscellaneous information field
ci_dlnum		Consumer driver license number
ci_ssnum		Consumer Social Security number
emailto		E-mail address where to send consumer's e-mail receipt. Default is ci_email
emailfrom		Return address on consumer's e-mail receipt. Default is null@atsbank.com.
emailsubject		Subject line on the consumer's receipt e-mail. Default is *Payment Receipt #xzy*.
emailtext		Text for consumer's e-mail receipt. Default is a generic receipt message.
ci_ipaddress		Consumer's IP Address
merchantordernumber		Customer's unique alpha-numeric number
custom1		Custom field for information to be included with the transaction.
custom2		Custom field for information to be included with the transaction.
custom3		Custom field for information to be included with the transaction.
custom4		Custom field for information to be included with the transaction.
custom5		Custom field for information to be included with the transaction.
custom6		Custom field for information to be included with the transaction.

Check21 Refund

The Check 21 environment does not natively support reversals, where the funds are refunded to the consumer's account a physical check is mailed to the customer. By submitting a refund, a check is printed and mailed to the customer to reverse the payment. To post refunds you must be setup to process bill payments.

Field	Required	Description
action	•	`EXTACH_CK21REFUND`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
amount	•	Amount to be refunded. This amount cannot be greater than the original transaction amount.
orderkeyid	•	Order ID of original transaction
historykeyid	•	History ID of original transaction
senddate	•	The date on which the refund check will be printed. Format: yyyy-MM-dd. This value must not refer to a past date.

Check21 Void

Field	Required	Description
action	•	`EXTACH_CK21VOID`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.

merchantpin		The 32 character Merchant PIN.
amount	•	Amount of the original transaction.
orderkeyid	•	Order ID of original transaction
historykeyid	•	History ID of original transaction

Recurring Billing Operations

Recurring Add

Field	Required	Description
action	•	`recur_add`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
acctname	•	Name on account.
accttype	•	"1": credit card; "2": ACH (electronic check)
acctdata1	•	If AcctType = "2", then ACH account number If AcctType = "1", then credit card number
acctdata2	•	If AcctType = "2", then ACH Routing number If AcctType = "1", then credit card expiration date in MM/YYYY format
recur_amount	•	Amount to debit consumer in recurring cycle (decimal format 123.45)
recur_billingcycle	•	0 = No Recurring Billing Cycle; 1 = Weekly Recurring Cycle; 2 = Monthly Recurring Cycle; 3 = Quarterly Recurring Cycle; 4 = Semi-Annual Recurring Cycle; 5 = Annual Recurring Cycle; 6 = Bi-Weekly Recurring Cycle; 7 = Bi-Annual Recurring Cycle; 8 = Quad Weekly (28 day) Recurring Cycle
recur_nextbillingdate	•	Next scheduled date to bill recurring consumer (MM/DD/YYYY)
recur_billingmax	•	-1 = Unlimited number of times; 0 = No Recurring Billing; > 0 = Maximum number of times a consumerÕs account is re-debited through recurring billing
ci_companyname		Consumer's company name.
ci_firstname		Consumer's first name
ci_lastname		Consumer's last name
ci_billaddr1		Consumer's address line 1.
ci_billcity		Consumer's city.
ci_billstate		Consumer's state.

ci_billzip		Consumer's zipcode.
ci_country		Consumer's country.
ci_phone		Consumer's phone number.
ci_email		Consumer's email address.
ci_memo		Merchant-supplied information.
ci_dlnum		Consumer's driver's license number
ci_ssnum		Consumer's social security number
merchantordernumber		Customer's unique alpha-numeric number
custom1		Custom field 1 for information to be included with the transaction.
custom2		Custom field 2 for information to be included with the transaction.
custom3		Custom field 3 for information to be included with the transaction.
custom4		Custom field 4 for information to be included with the transaction.
custom5		Custom field 5 for information to be included with the transaction.
custom6		Custom field 6 for information to be included with the transaction.

Recurring Update

Allows a merchant to update the consumer's billing information and recurring parameters for an existing recurring record.

Field	Required	Description
action	•	`recur_update`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
orderkeyid	•	Order ID of the original transaction.
ci_companyname		Consumer's company name.
ccname		Consumer name as it appears on the credit card.
ccnum		Consumer's credit card number.
expmon		Consumer's credit card expiration month.
expyear		Consumer's credit card expiration year.
ci_billaddr1		Consumer's address.
ci_billcity		Consumer's city.
ci_billstate		Consumer's state.
ci_billzip		Consumer's zipcode.
ci_country		Consumer's country.
ci_phone		Consumer's phone number.
ci_email		Consumer's email address.

ci_memo		Merchant-supplied information.
recur_amount		Amount to debit consumer in recurring cycle (format 123.45).
recur_billingcycle		0 = No Recurring Billing Cycle 1 = Weekly Recurring Cycle 2 = Monthly Recurring Cycle 3 = Quarterly Recurring Cycle 4 = Semi-Annual Recurring Cycle 5 = Annual Recurring Cycle 6 = Bi-Weekly Recurring Cycle 7 = Bi-Annual Recurring Cycle 8 = Quad Weekly (28 day) Recurring Cycle
recur_nextbillingdate		Next scheduled date to bill recurring consumer (MM/DD/YYYY).
recur_billingmax		-1 = Unlimited number of times. 0 = No Recurring Billing. > 0 = Maximum number of times a consumer's account is re-debited through recurring billing.

Recurring Cancel

Allows a merchant to cancel the consumer's billing information and recurring parameters for an existing recurring record.

Field	Required	Description
action	•	`recur_cancel`
acctid	•	Five character alphanumeric Account ID assigned to the merchant.
subid		Merchant Sub ID.
merchantpin		The 32 character Merchant PIN.
orderkeyid	•	Order ID of the original transaction.
canceltype	0 = Immediately, 1 = Next Billing, 2 = Cancel immediately, and add to negative database (scrub).	

Transaction Response

Response Format (Without Accepturl/Declineurl):

Transaction Accepted (Without Accepturl):

```
Accepted=SALE:TEST:::46031495:::
historyid=46031495
orderid=36665845
Accepted=SALE:TEST:::46031495:::
ACCOUNTNUMBER=**********5454
authcode=TEST
AuthNo=SALE:TEST:::46031495:::
historyid=46031495
orderid=36665845
recurid=0
refcode=46031495-TEST
result=1
```

```
Status=Accepted
transid=0
```

Transaction is Declined (Without Declineurl):

```
Declined=DECLINED:1101440001:Invalid Expiration Date
historyid=46031833
orderid=36666162
ACCOUNTNUMBER=************5454
Declined=DECLINED:1101440001:Invalid Expiration Date
historyid=46031833
orderid=36666162
rcode=1101440001
Reason=DECLINED:1101440001:Invalid Expiration Date
recurid=0
result=0
Status=Declined
transid=0
```

Response Format (With AcceptURL/DeclineURL):

When a transaction is processed, Online Commerce Suite retrieves the accepted or declined URL directly from the Online Commerce Suite Server, then relays it to the customer's browser. The customer's browser is connected to the Online Commerce Suite secure system during the entire transaction process event. The mechanics of this process are:

1. After a consumer's order is processed, Online Commerce Suite initiates a TCP/IP connection between the Online Commerce Suite server and the Merchant system.
2. An HTTP GET command retrieves the accepted or declined URL.
3. If the Merchant's e-commerce application includes a query string as part of the accepted or declined URL, Online Commerce Suite appends its own responses to the end of the query string. This allows the Merchant to pass as much additional information in the query string as desired.
4. HTTP GET command output is sent to the customer's browser. If the `accepturl` or `declineurl` references a static HTML page, it is displayed. If the `accepturl` or `declineurl` references a CGI script, the script output is displayed.

Online Commerce Suite uses the following formats to return the query string response to the consumer's browser:

Transaction Accepted (Without UsePost)

If the transaction is accepted, the following format is used to GET your AcceptURL. Your script can then parse the response (querystring) and display a custom accepted page for the customer.

```
http://www.myserver.com/cgibin/mycgi?Accepted=SALE%3ATEST%3A%3A%3A46031570%3A%3A%3A
&ACCOUNTNUMBER=************5454&authcode=TEST
&AuthNo=SALE%3ATEST%3A%3A%3A46031570%3A%3A%3A&historyid=46031570
&orderid=36665918&recurid=0&refcode=46031570-TEST&result=1
&Status=Accepted&transid=0
```

Transaction Accepted (With UsePost)

If the **UsePost** flag is enabled (`usepost=1`), a POST request is made your AcceptURL with all the name/value pairs (including custom name/value pairs from your system). Your script can then parse the POST data (`name1=value1&name2=value2&`), and display an accepted page to your customer.

Transaction Declined (Without Usepost)

If the transaction is declined, the following format is used:

```
http://www.myserver.com/cgibin/mycgi?
```

```
Declined=DECLINED:1101440001:Invalid+Expiration+Date
&ACCOUNTNUMBER=************5454&historyid=46032097
&orderid=36666304&rcode=1101440001
&Reason=DECLINED:1101440001:Invalid+Expiration+Date
&recurid=0&result=0&Status=Declined&transid=0
```

Transaction is Declined (With UsePost)

If the **UsePost** flag is enabled (`usepost=1`), a POST request is made your DeclineURL with all the name/value pairs (including custom name/value pairs from your system). Your script can then parse the POST data (`name1=value1&name2=value2&`), and display a decline page to your customer.

Examples

Using an HTML Form

Sample HTML to Submit a Credit Card Transaction:

```
<form method="post" action="https://trans.merchantpartners.com/cgi-bin/process.cgi">
<input type="hidden" name="action" value="ns_quicksale_cc">
<input type="hidden" name="acctid" value="TEST0">
<input type="hidden" name="amount" value="1.00">
<input type="hidden" name="ccname" value="Joe Customer">
<input type="hidden" name="ccnum" value="5454545454545454">
<input type="hidden" name="expmon" value="01">
<input type="hidden" name="expyear" value="2008">
<input type="submit">
</form>
```

Sample HTML to Submit a Check Transaction:

```
<form method="post" action="https://trans.merchantpartners.com/cgi-bin/process.cgi">
<input type="hidden" name="action" value="ns_quicksale_check">
<input type="hidden" name="acctid" value="TEST0">
<input type="hidden" name="amount" value="1.00">
<input type="hidden" name="ckname" value="Joe Customer">
<input type="hidden" name="ckaba" value="123456789">
<input type="hidden" name="ckacct" value="123456789012345">
<input type="submit">
</form>
```

Using a Socket Connection

Credit Card Transaction Example:

```
POST /cgi-bin/process.cgi HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 112
action=ns_quicksale_cc&atsid=TEST0&amount=1.00 &ccname="John"%20Doe&ccnum=4111111111111111
&expmon=01&expyear=2009
```

Check Transaction Example:

```
POST /cgi-bin/process.cgi HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: 102
action=ns_quicksale_check&atsid=TEST0&amount=1.00
&ckname="John"%20Doe&ckaba=999999999&ckacct=999999999
```

Transaction Accepted (Credit Card):

```
Accepted=SALE:VITAL5:::46031495:::
historyid=46031495
orderid=36665845
Accepted=SALE:VITAL5:::46031495:::
ACCOUNTNUMBER=**********5454
authcode=VITAL5
AuthNo=SALE:VITAL5:::46031495:::
historyid=46031495
orderid=36665845
recurid=0
refcode=46031495-VITAL5
result=1
Status=Accepted
transid=0
```

Transaction Declined (Credit Card):

```
Declined=DECLINED:1101440001:Invalid Expiration Date
historyid=46031833
orderid=36666162
ACCOUNTNUMBER=**********5454
Declined=DECLINED:1101440001:Invalid Expiration Date
historyid=46031833
orderid=36666162
rcode=1101440001
Reason=DECLINED:1101440001:Invalid Expiration Date
recurid=0
result=0
Status=Declined
transid=0
```

Transaction Accepted (Check):

```
Accepted=CHECKAUTH:TEST:::50078389:::
historyid=50078389
orderid=39809622
Accepted=CHECKAUTH:TEST:::50078389:::
ACCOUNTNUMBER=*****9999
authcode=TEST
AuthNo=CHECKAUTH:TEST:::50078389:::
historyid=50078389
orderid=39809622
recurid=0
refcode=50078389-TEST
result=1
Status=Accepted
transid=0
```

Transaction Declined (Check):

```
Declined=DECLINED:1101640001:Invalid Acct Number
historyid=50078338
orderid=39809571
ACCOUNTNUMBER=***9999
Declined=DECLINED:1101640001:Invalid Acct Number
historyid=50078338
orderid=39809571
rcode=1101640001
Reason=DECLINED:1101640001:Invalid Acct Number
recurid=0
result=0
Status=Declined
transid=0
```

Sample Java Servlet code snippet

The following Java servlet code snippet gives an example of how the Quicksale Method can be used within a Java application to submit transactions to the gateway.

Note: Processing transactions using a module has many advantages. It allows the customer to remain on your secure server while the transaction is being processed giving you control of the entire session.

```
// libraries to declare
Import java.net.*;
Import java.io.*;

// Store required field into an object before passing it through processing.
URL url = new URL("https://trans.merchantpartners.com/cgi-bin/process.cgi");
URLConnection connection = url.openConnection();
connection.setDoOutput(true);
PrintWriter out1 = new PrintWriter(connection.getOutputStream());
out1.print("action=ns_quicksale_cc");
out1.print("&acctid="+obj.getAcctid());
out1.print("&subid="+obj.getSubid());
out1.print("&amount="+obj.getAmount());
out1.print("&ccname="+obj.getCcName());
out1.print("&ccnum="+obj.getCardNumber());
out1.print("&expyear="+obj.getExpYear());
out1.print("&expmon="+obj.getExpMon());
out1.print("&ci_email="+obj.getEmail());
out1.print("&ci_billaddr1="+obj.getAddress1());
out1.print("&ci_billaddr2="+obj.getAddress2());
out1.print("&ci_billstate="+obj.getState());
out1.print("&ci_billzip="+obj.getZip());
out1.print("&ci_billcountry="+obj.getCountry());
out1.print("&ci_billcity="+obj.getCity());

System.out.println(out1.toString());
out1.close();
BufferedReader in = new BufferedReader(
        new InputStreamReader(
                connection.getInputStream()));

String inputLine="";
// store result in saveLine
saveLine="";
while ((inputLine = in.readLine()) != null)
        saveLine+="&"+inputLine;

in.close();
```

Sample perl script

The following Perl program snippet gives an example of how the Quicksale Method can be used. This example allows you to submit either a credit card auth or sale.

Note: Processing transactions using a module has many advantages. It allows the customer to remain on your secure server while the transaction is being processed giving you control of the entire session.

```
#!/usr/bin/perl

use strict;
use vars qw($VERSION $AUTOLOAD @ISA);
use Net::SSLeay qw(get_https post_https post_http sslcat make_headers make_form);

my $self = shift;
```

```
my $type = ref($self);

($self->{_page}, $self->{_response}, %{$self->{_reply_headers}})
        = post_https("trans.merchantpartners.com", '443', "/cgi-bin/process.cgi", "",
                make_form(
                        action => 'ns_quicksale_cc',
                        acctid => 'TEST0',
                        #**** set authonly => 1 for pre-auth ***#
                        authonly => '0',
                        amount => '1.00',
                        ccname => 'FirstName LastName',
                        ccnum => '5454545454545454',
                        expmon => '06',
                        expyear => '2006',
                        ci_billaddr1 => '123 address1',
                        ci_billaddr2 => '456 address2',
                        ci_billcity => 'Los Angeles',
                        ci_billstate => 'CA',
                        ci_billzip => '90031',
                        ci_phone => '310-123-4567',
                        merchantordernumber => 'inv-1234',
                ));

my($key, $value);

foreach (split(/\n/, $self->{_page})) {
   //g; s/\
        ($key, $value) = split(/\=/);
        $key =~ tr/ /+/;
        $value =~ tr/ /+/;
        $self->{_response_codes}->{$key} = $value;

        print "$self->{_response_codes}->{$key}\n";
}
exit;
```

Sample PHP script

The following PHP program snippet gives an example of how the Quicksale Method can be used. This example allows you to submit either a credit card auth or sale.

Note: Processing transactions using a module has many advantages. It allows the customer to remain on your secure server while the transaction is being processed giving you control of the entire session.

```
<?php

$url = 'https://trans.merchantpartners.com/cgi-bin/process.cgi';
$params = "action=ns_quicksale_cc" . "&" .
        "acctid=TEST0" . "&" .
        "amount=1.00" . "&" .
        "ccname="Tony" Test" . "&" .
        "ccnum=5454545454545454" . "&" .
        "expmon=09" . "&" .
        "expyear=2008";

$ch = curl_init();
curl_setopt($ch, CURLOPT_POST,1);
curl_setopt($ch, CURLOPT_POSTFIELDS,$params);
curl_setopt($ch, CURLOPT_URL,$url);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST,  2);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);

$result=curl_exec ($ch);
```

```
curl_close($ch);

if ($result == "")
{
        echo("No Response\n");
        exit;
}
else
{
        echo("$result");
        exit;
}
?>
```

Using the SecurePost Object (Windows platform)

Overview

The ATS SecurePost object provides an easy way to integrate credit card or electronic check (ACH) payment processing into your custom applications or Web sites. The SecurePost object is implemented as a COM object capable of running on Windows 95, Windows 98, or Windows NT 4.0 or later. It can be used with any programming language capable of calling a COM object, such as Visual C++, Visual Basic, Active Server Pages (ASP), Microsoft Office applications, and others. The transaction data is transferred over a Secure HTTP (HTTPS) connection using the installed Windows Sockets (WINSOCK) stack and Internet Extensions (WININET). If these components are not already present on your system, the easiest way to get them is to install the latest version of Microsoft Internet Explorer.

Installation

To install the ATS SecurePost object, copy the ATSSecurePost.dll file to the location where you want the object to reside (typically the same location where your application resides, or in the Windows SYSTEM or SYSTEM32 directory), and register it in the Windows Registry by running:

```
REGSVR32 ATSSecurePost.dll
```

After executing REGSVR32 from a command-prompt, you should see the following dialog:

If you get an error message during the registration process, your system contains an old version of the Microsoft Internet Extensions (WININET) libraries. In this case, install the latest version of Microsoft Internet Explorer or its updates.

Uninstallation

To uninstall the ATS SecurePost object, unregister the object by executing the following command, and then delete the ATSSecurePost.dll file:

```
REGSVR32 /u ATSSecurePost.dll
```

If the command is successful, you will see the following dialog:

Interfaces

The SecurePost object exposes a single COM (Component Object Model) interface through a class named *SecurePost Class*. This interface provides several methods for processing transactions, as well as a large number of properties that can be manipulated. There are also several methods that can be used for debugging and testing during the development cycle.

Properties

Name	Type & Size	Default	Required	Purpose, and Online Merchant Center tag
ATSID	String(5)	None	Always	ATS Merchant ID, always 5 characters long. (ATSID)
ATSSubID	String(5)	None		ATS Merchant Sub ID, 5 characters long or empty. (SubID)
CCName	String(32)	None	*[1]	Credit Card Account Name. (CCName)
CCNumber	String(32)	None	*	Credit Card Number. (CCNum)
ExpMonth	String(2)	None	*	Credit Card Expiration Month. (ExpMon)
ExpYear	String(4)	None	*	Credit Card Expiration Year. (ExpYear)
CVV2	String(3)	None		Credit Card CVV2 Code (CVV2)
CKName	String(32)	None	†[2]	Checking Account Name. (CKName)
CKRoutingNumber	String(9)	None	†	Checking Account Routing (ABA) Number, always 9 digits long. (CKABA)
CKAccountNumber	String(18)	None	†	Checking Account Number. (CKAcct)
RecurBillingCycle	Short Integer	0	•[3]	Billing cycle code for recurring billing. (Recur_Billingcycle) 0 = No Recurring Billing Cycle 1 = Weekly Recurring Billing Cycle 2 = Monthly Recurring Billing Cycle 3 = Quarterly Recurring Billing Cycle 4 = Semi-Annual Recurring Billing Cycle 5 = Annual Recurring Billing Cycle 6 = Bi-Weekly Recurring Billing Cycle 7 = Bi-Annual Recurring Billing Cycle 8 = Quad Weekly (28 day) Recurring Billing Cycle 9 = One Time Recurring Billing Cycle 10 = Daily Recurring Billing Cycle 11 = Bi-Monthly Recurring Billing Cycle
RecurBillingMax	Short Integer	-1	•	Maximum number of billing cycles before the recurring transaction will automatically be deleted. A value of -1 indicates that there is no maximum, i.e. billing continues indefinitely. (Recur_Billingmax)
RecurStart	Short Integer	-1	•	Number of days between the initial transaction, and the start of the billing cycle. (Recur_Start)
RecurAmount	Short Integer >= 100	0	•	The amount that will be charged during each recurring billing in cents, i.e. $14.95 is specified as 1495. The minimum amount is $1.00. (Recur_Amount)
Recurringflag	Short Integer	0		

				Internal recurring billing indicator (0 = none, 1 = active)
Amount	Short Integer > 0	0	Always	The amount to be charged to the credit card in cents. (Amount)
Currencyode	String(3)	None		Three digit currency code 'USD' for US$ (Refer to Appendix: D for complete list)
Custom1	String(32)	None		Custom field for information to be included with the transaction..
Custom2	String(32)	None		Custom field for information to be included with the transaction..
Custom3	String(32)	None		Custom field for information to be included with the transaction..
Custom4	String(32)	None		Custom field for information to be included with the transaction..
Custom5	String(32)	None		Custom field for information to be included with the transaction..
Custom6	String(32)	None		Custom field for information to be included with the transaction..
CI_CompanyName	String(64)	None		The consumer's company name. (CI_CompanyName)
CI_BillAddr1	String(64)	None	AVS[4]	The consumer's billing address. (CI_BillAddr1)
CI_BillAddr2	String(64)	None	AVS	The consumer's billing address. (CI_BillAddr2)
CI_BillAddr3	String(64)	None	AVS	The consumer's billing address. (CI_BillAddr3)
CI_BillCity	String(32)	None	AVS	The consumer's billing city. (CI_BillCity)
CI_BillState	String(32)	None	AVS	The consumer's billing state. (CI_BillState)
CI_BillZip	String(16)	None	AVS	The consumer's billing zip code. (CI_BillZip)
CI_BillCountry	String(32)	None	AVS	The consumer's billing country. (CI_BillCountry)
CI_ShipAddr1	String(64)	None		The consumer's shipping address. (CI_ShipAddr1)
CI_ShipAddr2	String(64)	None		The consumer's shipping address. (CI_ShipAddr2)
CI_ShipAddr3	String(64)	None		The consumer's shipping address. (CI_ShipAddr3)
CI_ShipCity	String(32)	None		The consumer's shipping city. (CI_ShipCity)
CI_ShipState	String(32)	None		The consumer's shipping state. (CI_ShipState)
CI_ShipZip	String(16)	None		The consumer's shipping zip code. (CI_ShipZip)
CI_ShipCountry	String(32)	None		The consumer's shipping country. (CI_ShipCountry)
CI_Phone	String(16)	None		The consumer's phone number. (CI_Phone)
CI_Email	String(64)	None		The consumer's e-mail address. (CI_Email)
CI_Memo	String(4096)	None		The consumer's memo. (CI_Memo)
CI_DLNum	String(32)	None		The consumer's Drivers License number. (CI_DLNum)
CI_SSNum	String(32)	None		The consumer's Social Security number. (CI_SSNum)
CI_IP Address	String(16)	None		The consumer's IP address. (CI_IP Address)

CI_BillHouseNumber	String(32)	None	Always	Payers House Number
CI_BillStreet	String(32)	None	Always	Payers Street
CI_BillZip4	String(16)	None	Always	Payers zip code. USA ZIP+4 format
CI_FirstName	String(32)	None	Always	Signatory First Name
CI_LastName	String(32)	None	Always	Signatory Last Name
AccounttypeDesc	String(18)	None	Always	ACH account type description
AcctName	String(32)	None	Always	Consumer name as it appears on the checking account.
ACHTransactionType	String(3)	None	Always	SEC code for transaction
AuthenticationModifier	Short Integer	None		Authentication of the payer name and address can be modified by this field.
CkAcctType	String(1)	None	Always	"P" - Personal, "B" - Business
InsuranceModifier	Short Integer	None		Insurance that the amount will clear can be modified by this field.
RiskModifier	Short Integer	None		Extended risk check of the check routing number and account number can be modified by this field.
SendDate	String(10)	None	Always	The time at which the refund check will be printed.
EmailSubject	String(256)	None		The e-mail message subject of the transaction acknowledgement sent to the consumer. (EmailSubject)
EmailText	String(4096)	None		The e-mail message text of the transaction acknowledgement sent to the consumer. (EmailText)
MerchantReferrerInfo	String(64)	None		The Merchant Referrer Information. (MerchantReferrerInfo)
MerchantOrderNumber	String(64)	None		The Merchant Order Number. If non-empty, this must be a value that is unique for the *ATSID*/*ATSSubID* combination. (MerchantOrderNumber)
MerchantOrderType	String(64)	None		The Merchant Order Type. (MerchantOrderType)
SwipeData	String(64)	None		The swipe data from the magnetic strip on the back of a credit or debit card. Used for card-present transactions.
SupplierRefNum	String(9)	None	◊[5]	A mandatory reference number that helps the merchant identify the charge in case of an inquiry by the credit card company. (SupplierRefNum)
CardholderRefNum	String(17)	None	◊	The cardholder reference number. (CardholderRefNum)
SalesTax	Short Integer >= 0	0	◊	The sales tax amount in cents. (SalesTax)
ChargeDescriptor	String(40)	None	◊	A mandatory free-form field describing the charge. (ChargeDescriptor)
OptionalAmountID	String(1)	None	◊	A code describing the Optional Amount value. (OptionalAmountID) 0 = Not used

				1 = Local Sales Tax Amount 2 = Tax Exempt
OptionalAmount	Short Integer >= 0	None	◊	An optional amount as described by the OptionalAmountID field. (OptionalAmount)
POCustomerRefID	String(17)	None	◊	A purchase Order Number. (POCustomerRefID)
ResultAccepted	Read-Only Boolean			TRUE if the transaction was approved, FALSE if it was declined.
ResultRefCode	Read-Only String			The reference code of the transaction.
ResultOrderID	Read-Only String			The ATS Order ID assigned to the transaction.
ResultTransID	Read-Only String			The ATS Transaction ID assigned to the transaction.
ResultAuthCode	Read-Only String			The authorization code of the transaction.
DevMode	Boolean	False		Determines whether the SecurePost object runs in development mode.
AVSOverride	Boolean	False		Overrides the Address Verification System (AVS) fraud protection, allowing transactions that would ordinarily be declined to be processed anyway.
ResultErrorFlag	Read-Only Short Integer			A non-zero value indicates an error occurred during processing.
LastError	Read-Only String			A string describing the error that occurred.
TransactionTimeout	Short Integer >= 5 & <= 600	300		The maximum number of seconds that the SecurePost object will wait for a response from the authorization network. NOTE: If a reply is not received within the timeout period, the SecurePost object will return an error. However, the authorizing network may continue to process the transaction, and eventually approve or decline it. You can retrieve the result later using the *GetTransactionResult* method.

1. An asterisk (*) indicates that this property must be set for all credit card transactions except ProcessPost.
2. A dagger (†) indicates that this property must be set for all checking account transactions.
3. A bullet (•) indicates that this property must be set for all recurring transactions.
4. "AVS" indicates that this property may be required if the Address Verification System (AVS) option has been enabled for the merchant account.
5. A diamond (◊) indicates a field that is required for Purchase Card Level 2 (PCL2) transactions.

Methods

ProcessSale

Processes a sale using the properties currently set. After the transaction has been processed, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction.

NOTE: Whether the transaction is a credit card or electronic check (ACH) transaction depends on the properties that are set. Setting any of the credit card properties will clear all electronic check properties, and vice versa, so the last property that was set controls the transaction type.

ProcessAuth

Processes an authorization using the properties currently set. After the transaction has been processed, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction.

ProcessPost

Processes a post for a previous auth-only transaction. The only properties that are used and must be set are the *ATSID*, *ATSSubID, CCNumber*, *ExpMonth*, and *ExpYear*. In addition, the ProcessPost method requires as an argument the *ResultRefCode* that was returned by the corresponding *ProcessAuth* call. After the transaction has been processed, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction.

ProcessCredit

Processes a credit for an account. The only properties that are used and must be set are the *ATSID* and *ATSSubID*. In addition, the ProcessCredit method requires as arguments the *ResultTransID* and *ResultOrderID* of the transaction to be credited. After the transaction has been processed, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction.

ProcessVoid

Processes a void of a previous transaction. The only properties that are used and must be set are the *ATSID*, *ATSSubID,* and *Amount.* In addition, the ProcessVoid method requires as an argument the *ResultTransID* that was returned by the corresponding *ProcessSale*, *ProcessAuth*, or *ProcessPost* call. After the transaction has been processed, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction.

ProcessQuasiCash

Processes a quasi-cash transaction.

OnStartPage

For ASP support - called automatically by the Web server when a page is opened. This method should not be called directly.

OnEndPage

For ASP support - called automatically by the Web server when a page is closed. This method should not be called directly.

ClearAll

Clears the values of all properties, and resets them to their default values.

GenerateRuntimeError(short ErrorCode)

Generates a runtime error for the purpose of debugging the error handling code in your application. The supported ErrorCode values are as follows:

ErrorCode	Error	Description
0	None	No error is generated
1	E_UNEXPECTED	Catastrophic failure
2	E_NOTIMPL	Not implemented
3	E_OUTOFMEMORY	Ran out of memory
4	E_INVALIDARG	One or more arguments are invalid
5	E_NOINTERFACE	No such interface supported
6	E_POINTER	Invalid pointer
7	E_HANDLE	Invalid handle
8	E_ABORT	Operation aborted
9	E_FAIL	Unspecified error
10	E_ACCESSDENIED	General access denied error
11	E_PENDING	The data necessary to complete this operation is not yet available

A full explanation of these common COM error codes and their meaning is beyond the scope of this document. In addition, the error message generated in your programming environment in the absence of your own error handler may vary from that shown in the Description column above.

Finally, not all of these errors can result from interacting with the SecurePost object. The only errors that should ever be generated are E_UNEXPECTED, E_NOTIMPL, and E_INVALIDARG. Support for other errors is included solely for the convenience of the developer writing the error handler.

ForceAccept(BSTR ApprovalCode)

Forces an approval of the next call to ProcessSale or ProcessAuth with the specified approval code. The data properties are not actually transmitted for processing, and the values of the *ResultRefCode*, *ResultOrderID*, and *ResultTransID* properties are not meaningful following the call. This method should be used for testing only.

ForceDecline(BSTR Reason)

Forces a decline of the next call to ProcessSale or ProcessAuth for the specified reason. The data properties are not actually transmitted for processing. This method should be used for testing only.

GetTransactionResult

Retrieves the result of a previously processed transaction given the current *ATSID*, *ATSSubID*, and *MerchantOrderNumber* properties. After the results have been retrieved, you can examine the *ResultErrorFlag*, *ResultAccepted*, *ResultRefCode*, *ResultOrderID*, *ResultTransID*, and *ResultAuthCode* properties to get detailed information about the transaction. If the transaction engine responds to the *GetTransactionResult* request, but the transaction cannot be found, both the *ResultErrorFlag* and *ResultAccepted* properties will be FALSE, and the *ResultAuthCode* property will be blank.

NOTE: In order to uniquely identify the transaction for which the results are to be retrieved, you must have assigned a unique *MerchantOrderNumber* property when the transaction was originally processed. If the *MerchantOrderNumber* property was left blank or wasn't unique, there is no way to retrieve the results.

Error Handling

The SecurePost object validates all properties to ensure that their values are within the permissible range. For example, all ATS Merchant IDs (the *ATSID* property) are always exactly five characters long, so any attempt to set the property to a value that is shorter or longer than five characters will generate an error of type E_INVALIDARG.

How errors are handled depends entirely on the environment in which the SecurePost object is being used: in C++, you will have to provide an exception handler using a *try* / *catch* block, and in Visual Basic, you will need an error handler using *On Error Goto*. For details, see the examples in the next section.

Examples

All of the examples below send a transaction for a $19.95 credit card sale using a fictitious MasterCard credit card to the credit card processor.

Visual C++

The example below assumes that a wrapper class called ISecurePost was created in the Class Wizard of Visual Studio from the type library or DLL.

```
HRESULT hr;
IUnknown* pIUnknown;
IDispatch* pIDispatch;

const CLSID CLSID_ATSSecurePost = {0x94A1A587,0x1CF1,0x11D3,{0x99,0x3D,0x00,0xE0,0x29,0x1F,0x9A,0x9C}};

try
{
    hr = CoCreateInstance(CLSID_ATSSecurePost, NULL, CLSCTX_INPROC_SERVER, IID_IUnknown,
        (LPVOID *) &pIUnknown);

    if (SUCCEEDED(hr))
    {
                hr = pIUnknown->QueryInterface(IID_IDispatch, (LPVOID *) &pIDispatch);

                if (SUCCEEDED(hr))
                {
                        ISecurePost MyObject(pIDispatch);

                        try
                        {
                                MyObject.SetATSID("TEST0");
                                MyObject.SetAmount(1995);
                                MyObject.SetCCName("John Doe");
                                MyObject.SetCCNumber("5454545454545454");
                                MyObject.SetExpMonth("10");
                                MyObject.SetExpYear("2002");

                                MyObject.ProcessSale();

                                if (MyObject.GetResultAccepted())
                                        MessageBox(MyObject.GetResultAuthCode(),"Accepted", MB_OK);

                                else if (MyObject.GetResultErrorFlag())
                                        MessageBox(MyObject.GetLastError(),"Error", MB_OK | MB_ICONERROR);

                                else
                                        MessageBox(MyObject.GetResultAuthCode(),"Declined", MB_OK);
```

```
                        }

                        catch (...)
                        {
                                MessageBox(MyObject.GetLastError(),"Error", MB_OK | MB_ICONERROR);
                        }

                }

        pIUnknown->Release();

        }

}

catch (...)
{

}
```

Visual Basic

Visual Basic supports both early and late binding of objects. The example shown here uses the late binding by instantiating the object based on the class name of the ATS SecurePost object.

```
Dim MyObject As ATSSECUREPOSTLib.SecurePost

On Error GoTo ErrHandler

Set MyObject = CreateObject("ATS.SecurePost")

MyObject.ATSID = "TEST0"
MyObject.Amount = 1995
MyObject.CCName = "John Doe"
MyObject.CCNumber = "5454545454545454"
MyObject.ExpMonth = "10"
MyObject.ExpYear = "2002"

MyObject.ProcessSale

If MyObject.ResultAccepted Then
        MsgBox "Accepted: " + MyObject.ResultAuthCode
Else
        If MyObject.ResultErrorFlag Then
                MsgBox "Error: " + MyObject.LastError
        Else
                MsgBox "Declined: " + MyObject.ResultAuthCode
    End If
End If

ErrHandler:

Set MyObject = Nothing
```

Active Server Pages

The Active Server Pages example shown here processes a sale, and displays the result of the transaction in HTML:

```
Set MyObject = CreateObject("ATS.SecurePost")

MyObject.ATSID = "TEST0"
MyObject.Amount = 1995
MyObject.CCName = "John Doe"
MyObject.CCNumber = "5454545454545454"
MyObject.ExpMonth = "10"
```

```
MyObject.ExpYear = "2002"
MyObject.ProcessSale
```

Microsoft Office Applications

The sample shown below can be used from within any Microsoft Office application, such as Microsoft Word or Microsoft Excel. The version of Basic available for scripting Microsoft Office applications is a subset of the full Visual Basic language, and does not support early binding. The Microsoft Office Applications example differs only slightly from the Visual Basic version - in this example, the SecurePost object is declared as a generic object, rather than tying it to a specific type library.

```
Function Microsoft_Office_Example()

Dim MyObject As Object

On Error GoTo ErrHandler

Set MyObject = CreateObject("ATS.SecurePost")

MyObject.ATSID = "TEST0"
MyObject.Amount = 1995
MyObject.CCName = "John Doe"
MyObject.CCNumber = "5454545454545454"
MyObject.ExpMonth = "10"
MyObject.ExpYear = "2002"

MyObject.ProcessSale



If MyObject.ResultAccepted Then
        MsgBox "Accepted: " + MyObject.ResultAuthCode
Else
        If MyObject.ResultErrorFlag Then
                MsgBox "Error: " + MyObject.LastError
    Else
                MsgBox "Declined: " + MyObject.ResultAuthCode
    End If
End If

ErrHandler:

Set MyObject = Nothing

End Function
```

The example shown above is included in this document, and can be executed by pressing Alt-F11 to access Visual Basic, selecting the *Microsoft_Office_Example* function from the *ThisDocument* scope, and pressing F8 to step through the code one line at a time.

Copyright Notice

Appendix A: Transaction Authorization Specification.

Credit Card Approval response format

The transaction approval authorization response message consists of a string of eight fields delimited by the colon ":" character. Here is an example of the format of the complete approval message:

```
AVSSALE:123456:1234567890123:9:12345678:Y:AUTHNETSPECIFIC:M:PARTIAL
```

The following table describes each of the fields returned in the approval response message.

Field	Description	Value
Transaction Type	Type of transaction submitted	SALE AVSSALE AUTH AVSAUTH POST AVSPOST VOICEPOST VOID CREDIT
Authorization Code	The six digit authorization or approval code provided by the authorizing network	Varies
Reference Number	Additional reference information provided by the authorizing network	Varies
Batch Number	Batch settlement number in which this transaction is included	Number
Transaction ID	Unique number assigned by the Online Commerce Suite to this transaction.	Number
AVS Result Code	Result code generated by the Address Verification System.	See Appendix B: AVS response codes
Auth Net Specific	Miscellaneous auth net message	
CVV2/CVC2 Result Code	One character result code generated by the CVV2/CVC2 system	See Appendix C: CVV2/CVC2 Response Codes
PARTIAL AUTH	Contains the ":PARTIAL" string if it's a Partial Auth	(Auth Net Specific). Will also return the approved partial auth "amount" and the "ACTIONCODE", and "PARTIAL_APPROVAL=1" value.

Credit Card Decline response format

The transaction decline authorization response message consists of the string "DECLINE" followed by two fields delimited by the colon ":" character.

Here is an example of the format of the complete approval message:

```
DECLINED:1234567890:TEXT RESPONSE
```

The following table describes each of the fields returned in the decline response message.

Field	Description	Value
Transaction Result	Result of the transaction	DECLINE
Decline Code	10 digit decline code.	**First Digit:** 0 = Authorizing Network declined the transaction. 1 = Gateway declined the transaction. 2 = Authorizing Network returned an error, forcing a decline. 3 = Gateway returned an error, forcing a decline. **Digits 2-10:** Internal decline number.
Text Response	Text message indicating the reason for the decline.	Varies

Appendix B: AVS Response Codes

The following table defines AVS response codes returned from the Address Verification System.

Response Code	Definition
A	Street addresses matches, but the ZIP code does not. The first five numerical characters contained in the address match. However, the ZIP code does not match.
E	Ineligible transaction. The card issuing institution is not supporting AVS on the card in question.
N	Neither address nor ZIP matches. The first five numerical characters contained in the address do not match, and the ZIP code does not match.
R	Retry (system unavailable or timed out).
S	Card type not supported. The card type for this transaction is not supported by AVS. AVS can verify addresses for Visa cards, MasterCard, proprietary cards, and private label transactions.
U	Address information unavailable. The address information was not available at the issuer.
W	9 digit ZIP code match, address does not. The nine digit ZIP code matches that stored at the issuer. However, the first five numerical characters contained in the address do not match.
X	Exact match (9 digit zip and address) Both the nine digit postal ZIP code as well as the first five numerical characters contained in the address match.
Y	Address and 5 digit zip match. Both the five digit postal ZIP code as well as the first five numerical characters contained in the address match.
Z	5 digit ZIP matches, but the address does not. The five digit postal ZIP code matches that stored at the VIC or card issuer's center. However, the first five numerical characters contained in the address do not match.
FOREIGN CODES	
B	Street address matches for international transaction. Postal Code not verified due to incompatible formats.
C	Street address and Postal Code not verified for international transaction due to incompatible format.
D	Street address and Postal Code match for international transaction.
P	Postal Code match for international transaction. Street address not verified due to incompatible formats.

Appendix C: CVV2/CVC2 Response Codes

The following table defines CVV2/CVC2 response codes returned from the credit card authorizing network.

Response Code	Definition
Space	CVV2 processing not requested
M	CVV2/CVC2 Match
N	CVV2/CVC2 not matched
P	Not processed
S	CVV2 should be printed on the card, but it was indicated that the value was not present
U	Issuer does not support CVV2
X	Service provider did not respond

Appendix D: Country and Currency Code

You must first verify that your credit card merchant account processor and the gateway support the currency code submitted prior to attempting any transactions other than those in **"U.S."** dollars.

The following table defines the country, currency code, and the requirement of decimals in amount fields. **"NONE"** indicates that the decimal is not required when setting the amount.

Country	Currency Code	Decimal
Argentina	ARS	
Australia	AUD	
Christmas Island	AUD	
Cocos (Keeling) Islands	AUD	
Heard and McDonald Islands	AUD	
Kiribati	AUD	
Nauru	AUD	
Norfolk Island	AUD	
Tuvalu	AUD	
Aruba	AWG	
Azerbaijan	AZN	
Bulgaria	BGN	
Bermuda	BMD	
Singapore	BND	
Bolivia	BOB	
Bolivia	BOV	
Brazil	BRL	
Bahamas	BSD	
Bhutan	BTN	
Botswana	BWP	
Belarus	BYR	NONE
Democratic Republic of Congo	CDF	
Switzerland	CHF	
Liechtenstein	CHF	
Chile	CLP	NONE
China	CNY	
Colombia	COP	
Colombia	COU	
Costa Rica	CRC	

Country	Currency Code	Decimal
Cuba	CUP	
Cape Verde	CVE	
Czech Republic	CZK	
Djibouti	DJF	NONE
Denmark	DKK	
Greenland	DKK	
Algeria	DZD	
Estonia	EEK	
Egypt	EGP	
Eritrea	ERN	
Ethiopia	ETB	
Andorra	EUR	
Kosovo	EUR	
Monaco	EUR	
Montenegro	EUR	
San Marino	EUR	
Vatican	EUR	
Belgium	EUR	
Cyprus	EUR	
Finland	EUR	
France	EUR	
Germany	EUR	
Ireland	EUR	
Italy	EUR	
Luxembourg	EUR	
Malta	EUR	
Portugal	EUR	
Slovenia	EUR	
Spain	EUR	
Fiji	FJD	
Falkland Islands	FKP	
Isle of Man	GBP	
Georgia	GEL	
Ghana	GHS	
Gibraltar	GIP	

Country	Currency Code	Decimal
Gambia	GMD	
Guinea	GNF	NONE
Guatemala	GTQ	
Guyana	GYD	
Croatia	HRK	
Haiti	HTG	
Hungary	HUF	
Indonesia	IDR	
Israel	ILS	
Bhutan	INR	
India	INR	
Iraq	IQD	
Iran	IRR	
Iceland	ISK	NONE
Jamaica	JMD	
Jordan	JOD	
Japan	JPY	NONE
Kenya	KES	
Kyrgyzstan	KGS	
Cambodia	KHR	
Comoros	KMF	NONE
North Korea	KPW	
South Korea	KRW	NONE
Kuwait	KWD	
Cayman Islands	KYD	
Kazakhstan	KZT	
Laos	LAK	
Lebanon	LBP	
Sri Lanka	LKR	
Liberia	LRD	
Libya	LYD	
Moldova	MDL	
Nicaragua	NIO	
Philippines	PHP	
Romania	RON	

Country	Currency Code	Decimal
Serbia	RSD	
Russia	RUB	
Slovakia	SKK	
Somalia	SOS	
Central African Republic	XAF	
Gabon	XAF	
Anguilla	XCD	
Antigua and Barbuda	XCD	
Dominica	XCD	
Grenada	XCD	
Montserrat	XCD	
Benin	XOF	
Burkina Faso	XOF	
Yemen	YER	
South Africa	ZAR	
Zambia	ZMK	
Zimbabwe	ZWD	
Canada	CAD	
Bosnia and Herzegovina	BAM	
Barbados	BBD	
Bangladesh	BDT	
Bahrain	BHD	
Burundi	BIF	NONE
Brunei	BND	
Faroe Islands	DKK	
British Indian Ocean Territory	GBP	
Hong Kong	HKD	
Honduras	HNL	
Lesotho	LSL	
Morocco	MAD	
Western Sahara	MAD	
Madagascar	MGA	
The former Yugoslav Republic of Macedonia	MKD	
Myanmar	MMK	
Mongolia	MNT	

Country	Currency Code	Decimal
Macau	MOP	
Mauritania	MRO	
Mauritius	MUR	
Maldives	MVR	
Malawi	MWK	
Malaysia	MYR	
Mozambique	MZN	
Namibia	NAD	
Nigeria	NGN	
Norway	NOK	
Nepal	NPR	
Cook Islands	NZD	
New Zealand	NZD	
Niue	NZD	
Pitcairn	NZD	
Tokelau	NZD	
Oman	OMR	
Panama	PAB	
Peru	PEN	
Papua New Guinea	PGK	
Pakistan	PKR	
Poland	PLN	
Paraguay	PYG	NONE
Qatar	QAR	
Rwanda	RWF	NONE
Saudi Arabia	SAR	
Solomon Islands	SBD	
Seychelles	SCR	
Sudan	SDG	
Sweden	SEK	
Singapore	SGD	
Brunei	SGD	
Saint Helena	SHP	
Sierra Leone	SLL	
Suriname	SRD	

Country	Currency Code	Decimal
São Tomé and Príncipe	STD	
Syria	SYP	
Swaziland	SZL	
Thailand	THB	
Tajikistan	TJS	
Turkmenistan	TMM	
Tunisia	TND	
Tonga	TOP	
Turkey	TRY	
Cyprus	TRY	
Trinidad and Tobago	TTD	
Taiwan	TWD	
Tanzania	TZS	
Ukraine	UAH	
Uganda	UGX	
American Samoa	USD	
British Indian Ocean Territory	USD	
Ecuador	USD	
El Salvador	USD	
Guam	USD	
Haiti	USD	
Marshall Islands	USD	
Micronesia	USD	
Northern Mariana Islands	USD	
Palau	USD	
Panama	USD	
Puerto Rico	USD	
East Timor	USD	
Turks and Caicos Islands	USD	
United States	USD	
United States Virgin Islands	USD	
Bermuda	USD	
Uruguay	UYU	
Uzbekistan	UZS	
Venezuela	VEF	

Country	Currency Code	Decimal
Vietnam	VND	
Vanuatu	VUV	NONE
Samoa	WST	
Cameroon	XAF	
Congo	XAF	
Chad	XAF	
Equatorial Guinea	XAF	NONE
Saint Kitts and Nevis	XCD	
Saint Lucia	XCD	
Saint Vincent and the Grenadines	XCD	
Côte d'Ivoire	XOF	
Guinea-Bissau	XOF	
Mali	XOF	
Niger	XOF	
Senegal	XOF	
Togo	XOF	
French Polynesia	XPF	NONE
New Caledonia	XPF	NONE
Wallis and Futuna	XPF	NONE
United Kingdom	GBP	
Belize	BZD	
Dominican Republic	DOP	
Austria	EUR	
Greece	EUR	
Netherlands	EUR	
United Arab Emirates	AED	
Afghanistan	AFN	
Albania	ALL	
Armenia	AMD	
Netherlands Antilles	ANG	
Angola	AOA	
Latvia	LVL	
South Georgia and the South Sandwich Islands	GBP	
Lithuania	LTL	
Mexico	MXN	