# Online Commerce Suite™
# Card Present Transactions Guide

Release 2.5
March 2013

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

# Table of Contents

# Table of Contents

# Overview

The Online Commerce Suite Card Present Transactions Guide is specialized to meet the needs of merchants processing credit card, debit card and check conversion transactions in a retail POS environment. This guide will help your users to quickly familiarize themselves with the application programming interface (API) used to send and receive transactions through the Online Commerce Suite payment gateway.

## What's in this Guide

Welcome to the Online Commerce Suite system! The 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.

## How to Comment on This Guide

Every effort has been made to produce an accurate and easy to understand the Card Present Transactions 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

### Documentation

Web Site URL: https://www.onlinemerchantcenter.com/mpartners/html/user_manuals.html

### Certified Networks

Web Site URL: https://www.onlinemerchantcenter.com/mpartners/html/networks.html

### Hardware Compatible List

Web Site URL: https://www.onlinemerchantcenter.com/mpartners/html/equipment.html

### Online Commerce Suite FTP site:

Web Site URL: ftp.atsbank.com

### Online Commerce Suite Payment Gateway:

Web Site URL: https://trans.merchantpartners.com/cgi-bin/process.cgi

## Setting Up an Online Commerce Suite Account

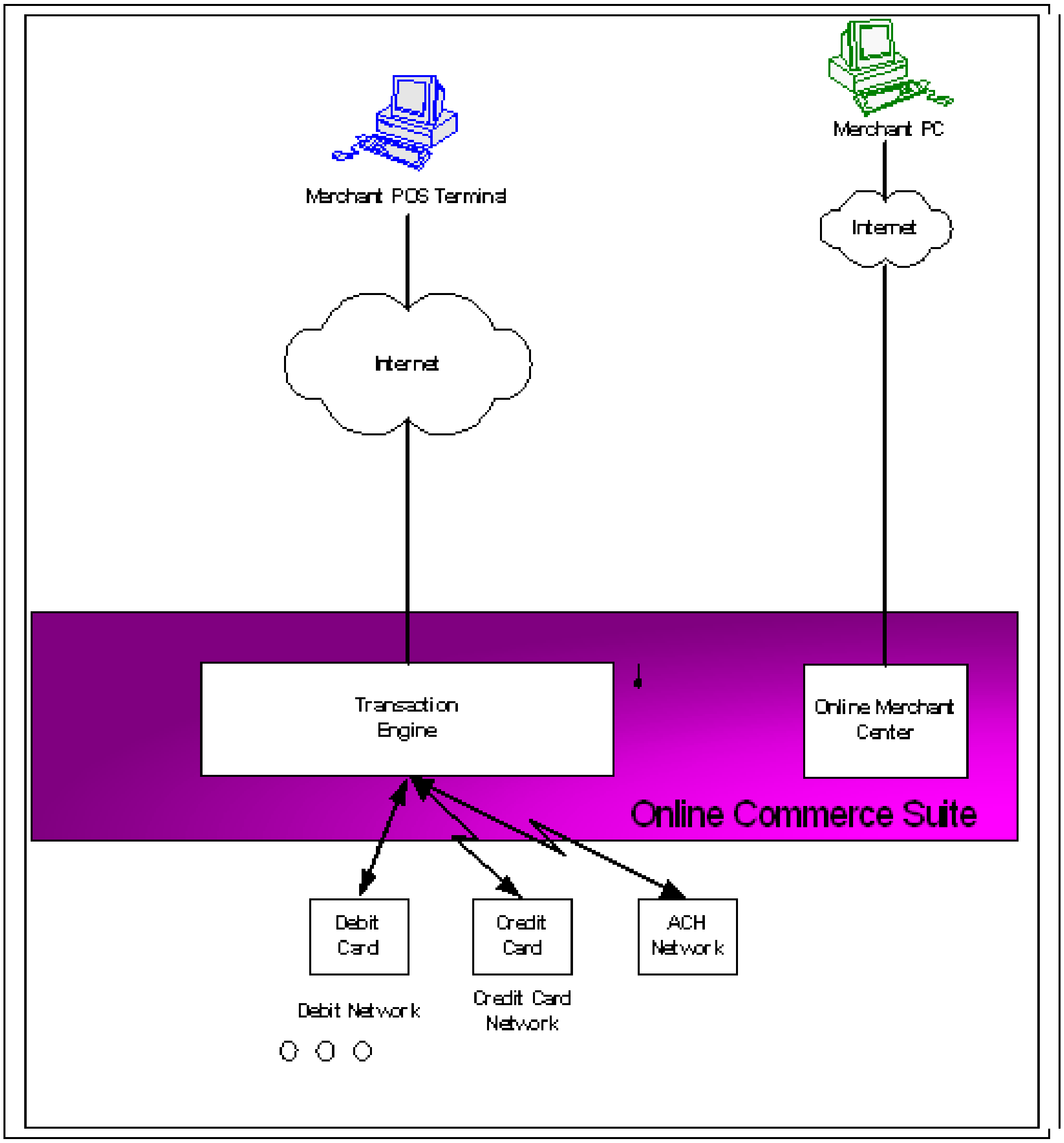
The first step in setting up your Online Commerce Suite account is to contact Customer Service to complete your registration by telephone. When your account is confirmed and set up, you will receive your five-digit Online Commerce Suite Account ID number (Acct ID) This important number identifies your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

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

# Integration Overview

Payment transactions consist of a two part conversation between the merchant's POS unit and the Online Commerce Suite (OCS) gateway:

1. An authorization request is sent to the gateway.
2. The gateway returns an authorization response to the POS unit.



## Typical credit card transaction scenario

1. The customer swipes their credit card via a swipe terminal attached to your POS system.
2. The customer's data along with the transaction amount is then securely transmitted to the Online Commerce Suite gateway for processing.
3. The Transaction Engine then requests authorization through the credit card authorizing network and returns an Accept or Decline result code and an authorization code.
4. Your POS system logs the transaction and prints out a transaction which is signed by the customer.

## Typical debit transaction scenario

1. The customer swipes their debit card and enters their PIN number via a Swipe / PIN pad terminal attached to your POS system.
2. The customer's data along with the transaction amount and optional cash back amount is then securely transmitted to the Online Commerce Suite gateway for processing.
3. The Transaction Engine then requests authorization through the debit card network and returns an Accept or Decline along with a unique confirmation message.
4. Your POS system logs the transaction and issues the cash back amount (if applicable).

## Typical check conversion scenario (Retail POP):

1. The Merchant provides notice to the consumer that paper checks will be used to make an electronic conversion. Notice can take the form of a sign posted at the register or a written notice that the consumer is asked to sign.
2. The Consumer presents paper check for payment. The check can be completely filled out, partially filled out, or blank.
3. Using a MICR scanner, the Merchant scans MICR data from the check capturing bank ABA number, personal account number, and serial number (check number).
4. The Merchant adds the payment amount and electronically submits the transaction.
5. The gateway processes the ACH transaction and returns an Approve or Decline along with a unique confirmation message.
6. If the transaction was approved, the paper check is stamped "VOID" and returned to the consumer (along with a copy of the receipt).

## Typical check conversion scenario (Accounts Receivable ARC):

1. The Merchant notifies all consumers that their paper checks will be converted. The Consumers should receive the notice prior to sending in a check.
2. The Consumer writes a check to the merchant for the amount of the payment.
3. Using a MICR scanner, the Merchant scans MICR data from the check capturing bank ABA number, personal account number, and serial number (check number).
4. The Merchant makes a copy of the check (front required, back optional). The copy needs to be retained for 2 years. Online Commerce Suite can automatically capture an electronic image of the check and maintain that copy online for 2 years.
5. The Merchant adds the payment amount and electronically submits the transaction.
6. The gateway processes the ACH transaction and returns an Approve or Decline along with a unique confirmation message.
7. If the transaction was approved, the Merchant destroys the paper check within 14 calendar days.

## Online Commerce Suite System Modules

**Online Commerce Suite** modules accomplish the following functions:

- **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 scramble data to protect customer payment information. Customer's private information (credit card numbers and bank account numbers) are securely encrypted as they are transmitted over the Internet.

## 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. Many browsers including Microsoft Internet Explorer and Netscape Navigator support SSL encryption, but you will need to host your site 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.

# Online Transaction Processing

The Online Commerce Suite system allows you to use one of the following integration methods to process card present transactions:

- HTTPS Post
- Socket Connection
- SecurePost COM Object

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 Table 1 before you decide which method to use.

| Method | URL Visible? | Transaction Security | Interface |
|---|---|---|---|
| **QuickSale**<br>via HTTPS Post | Depends on the implementation | You must provide a secure server | HTTPS |
| **QuickSale**<br>via Socket Connection | Hidden | You must provide a secure server | HTTPS |
| **Quicksale**<br>Via SecurePost | Hidden | Inherent in method | COM object |

# Quicksale Method with an HTTPS Post

Quicksale Method with an HTTPS Post allows you to create an application to collect data from your consumer, and then post the transaction data to the gateway via secure HTTPS post. 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).

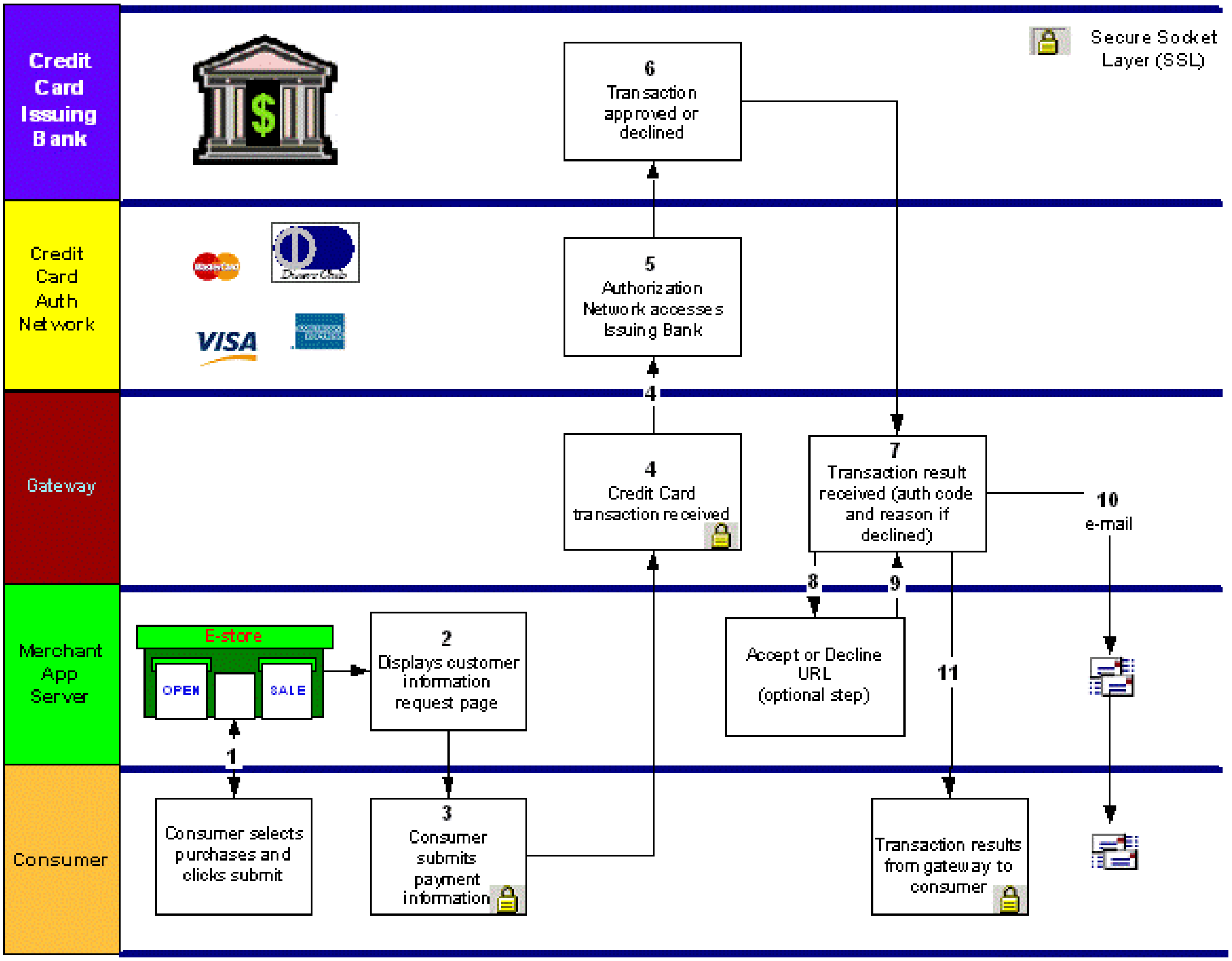
## How It Works

1. Your system displays the transaction request page, which asks your customer to supply transaction information either by keying it into a form or swiping a card. For example, data could be entered into an HTML web page or it could come from a Point-of-Sale application.

2. After your customer submits the account information, the data is securely transferred to gateway via an HTTPS post action. The gateway then processes the transaction request.

3. If the transaction is accepted, the gateway generates a receipt for the transaction and e-mails it to you and to your customer. You can configure the system to **not** e-mail either you or your customer, or both.

4. The gateway 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.


Credit Card Issuing Bank
Credit Card Auth Network
Gateway
Merchant App Server
Consumer
Secure Socket Layer (SSL)
6 Transaction approved or declined
5 Authorization Network accesses Issuing Bank
4
4 Credit Card transaction received
7 Transaction result received (auth code and reason if declined)
10 e-mail
8
9
E-store
OPEN
SALE
2 Displays customer information request page
Accept or Decline URL (optional step)
11
1
Consumer selects purchases and clicks submit
3 Consumer submits payment information
Transaction results from gateway to consumer
VISA

## Security

The most sensitive transaction data like credit card number or bank account number is transmitted directly from merchant application to the gateway via SSL. If you are using a web form to have the consumer enter payment information, you need to have SSL installed on your web server.

## Transaction Types (Actions)

The following card present transaction types are available:

| Action | Description |
|---|---|
| ns_quicksale_cc | Process a credit card order (sale, auth, post/capture, voice post |
| ns_void | Process a void of an existing credit card order |
| ns_credit | Process a credit of an existing credit card order |
| debit | Process a debit card sale |

| | |
|---|---|
| debit_return | Process a debit card return |
| quasi_cash | Process a credit card quasi cash sale |
| ns_quicksale_check | Process a check conversion transaction (ARC or POP conversion of a paper check into an electronic ACH transaction) |
| Authreverse | Process an authorization reversal to reduce the amount of a previous preauth. (Hotel/Lodging only) |
| Incrementalauth | Increase the amount of a previous preauth. (Hotel/Lodging only, and only available for Visa cards). |

## Data Elements

The following tables describe required and optional data elements that can be **sent** to the Online Commerce Suite payment gateway to complete a transaction.

### Credit Card Sale/Auth (required fields)

| Field | Description |
|---|---|
| action | The "ns_quicksale_cc" action instructs Online Commerce Suite to process a credit card transaction. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. |
| swipedata | Card swipe data (must include either track1 or track2 data) |
| CardPresent | A value of 1 indicates that the card was present |
| CardReaderPresent | A value of 1 indicates that a card reader was present |
| ccname | Consumer name as it appears on the debit card. |
| **Additional required fields for Hotel/Lodging** | |
| IndustryCode | One character value: "H": Hotel |
| ChargeTypeAmx | Code indicating whether the primary charge is Lodging, Restaurant, or Gift Shop. One character value: "1" Hotel, "2" Restaurant, "3" Gift Shop |
| Roomrateamt | Room rate dollar amount in US dollars in the form of 0.00. |
| Checkindate | The date (yymmdd) that the guest checked in to the hotel |
| Checkoutdate | The date (yymmdd) that the guest checked out of the hotel |
| PurchaseID | Hotel Folio number (25 characters alphanumeric) |
| Pproperty | Prestigious property indicatory. Used by participants in Visa's Prestigious Lodging Program. A transaction amount of $1 should be entered in the amount field if the merchant wishes the transaction to participate in the program. Number value: 500, 1000, or 1500 |
| Extracharges | 6 1-digit codes, each a partial or complete explanation of why charged amount differs from receipt cardholder received at checkout. Digit values: 0: no extra, 2: Restaurant, 3: Gift Shop, 4: Mini-bar, 5: Telephone, 6: Other, 7: Laundry. Example: "240000" indicates restaurant and mini-bar charges. |

## Credit Card Sale/Auth (optional fields)

| Field | Description |
|---|---|
| subid | Merchant Sub ID.  If unsure whether you have one, leave blank. |
| authonly | A value of 1 pre-authorizes the credit card.  A pre-authorization 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. |
| postonly | The Refcode returned from a previous Authonly=1 (pre-authorization transaction). This is the post-authorization step of a Book and Ship sales transaction.  See Authonly above. |
| voiceauth | 6 character approval code obtained from the cardholder's issuing bank. |
| track1 | Track 1 card swipe data |
| track2 | Track 2 card swipe data |
| ccnum | Credit Card number keyed in (when a swipe reader is unable to detect the card number) |
| expmon | Expiration month keyed in in (when a swipe reader is unable to detect the card number) |
| expyear | Expiration year keyed in in (when a swipe reader is unable to detect the card number) |
| usepost | To return order form information to a database, specify usepost=1. |
| 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 | 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 |
| ci_phone | Consumer phone number |
| ci_email | Consumer email address |
| ci_memo | Miscellaneous information field |
| ci_dlnum | Consumer driver's license number |
| ci_ssnum | Consumer Social Security Number |
| dynamicdescriptor | This field will be passed all the way to the Consumers Credit Card statement. This is only available for TSYS. |
| emailto | E-mail address to send the consumer e-mail receipt. Default is ci_email address. |
| emailfrom | Return address on consumer's e-mail receipt.  Default is *null@atsbank.com*. |

| | |
|---|---|
| emailsubject | Subject line on consumer's receipt email. Default message is *Payment Receipt #xzy*. |
| emailtext | Consumer's e-mail receipt body text. Default is a generic receipt message. |
| merchantordernumber | Customer's unique alpha-numeric number |

## Credit Card Void (required fields)

| Field | Description |
|---|---|
| action | The "ns_void" action instructs Online Commerce Suite to issue a void. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| Subid | Required only If transaction was submitted under a subid. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. |
| orderkeyid | Orderkeyid of the original transaction |
| historykeyid | Historykeyid of the original transaction |

## Credit Card Credit/Refund (required fields)

| Field | Description |
|---|---|
| action | The "ns_credit" action instructs Online Commerce Suite to issue a credit. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| Subid | Required only If transaction was submitted under a subid. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. |
| orderkeyid | Orderkeyid of the original transaction |
| historykeyid | Historykeyid of the original transaction |

## Debit transaction (required fields)

| Field | Description |
|---|---|
| action | The debit action instructs Online Commerce Suite to process a debit transaction. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. |
| cashbackamount | Cash back amount in US dollars in the form of 0.00 (total amount transferred is amount + cashbackamount) |
| swipedata | Card swipe data (must include track 2 data) |
| customerID | 32 byte Encrypted PIN pad data. Includes 6 byte key set followed by 10 byte PIN Pad Serial number and 16 byte PIN data. |
| cardpresent | A value of 1 indicates that the card was present |

| | |
|---|---|
| cardreaderpresent | A value of 1 indicates that a card reader was present |

## Debit transaction (optional fields)

| Field | Description |
|---|---|
| subid | Merchant Sub ID.  If unsure whether you have one, leave blank. |
| Ccname | Consumer name |
| usepost | To return order form information to a database, specify usepost=1. |
| 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 | 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 |
| ci_phone | Consumer phone number |
| ci_email | Consumer email address |
| ci_memo | Miscellaneous information field |
| ci_dlnum | Consumer driver's license number |
| ci_ssnum | Consumer Social Security Number |
| emailto | E-mail address to send the consumer e-mail receipt. Default is ci_email address. |
| emailfrom | Return address on consumer's e-mail receipt.  Default is *null@atsbank.com*. |
| emailsubject | Subject line on consumer's receipt email.  Default message is *Payment Receipt #xyz*. |
| emailtext | Consumer's e-mail receipt body text.  Default is a generic receipt message. |
| merchantordernumber | Customer's unique alpha-numeric number |

## Auth Reversal (required fields, Hotel only, Visa only)

| Field | Description |
|---|---|
| action | The "authreversal" action instructs Online Commerce Suite to issue an authorization reversal for the dollar amount specified.  This will reduce the amount being held on a cardholders account. |

| | |
|---|---|
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| Subid | Required only If transaction was submitted under a subid. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. This is the amount by which the original authorization will be reversed. Amount cannot exceed amount of original preauth prior auth reversals). |
| orderkeyid | Orderkeyid of the original transaction |
| historykeyid | Historykeyid of the original pre-auth transaction |

## Incremental Auth (required fields, Hotel only, Visa only)

| Field | Description |
|---|---|
| action | The "incrementalauth" action instructs Online Commerce Suite to issue an incremental authorization for the dollar amount specified. This will increase the hold amount on a cardholders account. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| Subid | Required only If transaction was submitted under a subid. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. This is the amount by which the original authorization will be increased. |
| orderkeyid | Orderkeyid of the original transaction |
| historykeyid | Historykeyid of the original pre-auth transaction |
| swipedata | Card swipe data (must include either track1 or track2 data) |
| CardPresent | A value of 1 indicates that the card was present |
| CardReaderPresent | A value of 1 indicates that a card reader was present |

## Debit Return (required fields)

| Field | Description |
|---|---|
| action | ns_debit_return instructs Online Commerce Suite to issue a credit. |
| acctid | Use TEST0 for testing if you do not have an Account ID. Change to your Account ID for live transaction processing. |
| subid | Required only If transaction was submitted under a subid. |
| amount | Transaction dollar amount in US dollars in the form of 1.00 |
| swipedata | Card swipe data (must include Track 2 data) |
| customerid | 32 byte Encrypted PIN pad data. Includes 6 byte key set followed by 10 byte PIN Pad Serial number and 16 byte PIN data. |
| orderkeyid | Orderkeyid of the original transaction |
| historykeyid | Historykeyid of the original transaction |

## QuasiCash (required fields)

| Field | Description |
|---|---|
| action | quasi_cash instructs Online Commerce Suite to process a credit card quasi cash transaction. |
| acctid | Use TEST0 for testing if you do not have an Account ID.  Change to your Account ID for live transaction processing. |
| amount | Transaction dollar amount in US dollars in the form of 0.00. |
| swipedata | Card swipe data (must include either Track 1 or 2 data) |
| CardPresent | A value of 1 indicates that the card was present |
| CardReaderPresent | A value of 1 indicates that a card reader was present |
| ccname | Consumer name as it appears on the card. |

## QuasiCash (optional fields)

| Field | Description |
|---|---|
| subid | Merchant Sub ID.  If unsure whether you have one, leave blank. |
| track1 | Track 1 card swipe data |
| track2 | Track 2 card swipe data |
| ccnum | Credit Card number keyed in (when a swipe reader is unable to detect the card number) |
| expmon | Expiration month keyed in (when a swipe reader is unable to detect the card number) |
| expyear | Expiration year keyed in (when a swipe reader is unable to detect the card number) |
| usepost | To return order form information to a database, specify usepost=1. |
| 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 | 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 |
| ci_phone | Consumer phone number |
| ci_email | Consumer email address |

| | |
|---|---|
| ci_memo | Miscellaneous information field |
| ci_dlnum | Consumer driver's license number |
| ci_ssnum | Consumer Social Security Number |
| emailto | E-mail address to send the consumer e-mail receipt. Default is ci_email address. |
| emailfrom | Return address on consumer's e-mail receipt.  Default is *null@atsbank.com*. |
| emailsubject | Subject line on consumer's receipt email.  Default message is *Payment Receipt #xzy*. |
| emailtext | Consumer's e-mail receipt body text.  Default is a generic receipt message. |
| merchantordernumber | Customer's unique alpha-numeric number |

## Check Conversion (required fields)

| Field | Description |
|---|---|
| action | ns_quicksale_check instructs Online Commerce Suite |
| acctid | Use TEST0 for testing if you do not have an Account ID.  Change to your Account ID for live transaction processing. |
| 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` |
| amount | Transaction dollar amount in US dollars in the form of 1.00 |
| 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. |
| cktype | "POP" or "ARC" |
| ckno | Check Number (serial number) |
| ckmicrdata | MICR data read from the MICR line |
| ckterminalcity | 4 character abbreviated name for the city where the POS terminal is located. Required for POP transactions |
| ckterminalstate | 2 character state abbreviation of the state where the POS terminal is located. Required for POP transactions |
| ckimagefront | Front image of the check.  Maximum of 20K.  Only required for ARC transactions |
| ckimagetype | "TIF", "JPG", or "GIF".  Only required if ckimagefront or ckimageback is used. |

## Check Conversion (optional fields)

| Field | Description |
|---|---|
| subid | Merchant Sub ID.  If unsure if you have one, leave blank. |
| ckname | Consumer name as it appears on the checking account. |
| 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 IpAddress |
| merchantordernumber | Customer's unique alpha-numeric number |
| ckimageback | Back image of the check. Maximum of 20K. |

## Web Form Submission Data Formats

To send the transaction to the Online Commerce Suite payment gateway server via HTTPS, post the fields to the following URL:

```
https://trans.merchantpartners.com/cgi-bin/process.cgi
```

The **accepturl** and **declineurl** are required when using the web form method. They can reference either static HTML pages or CGI scripts.

### Request Format

The following example shows how to use HTML to transmit a transaction using HTTPS.

```
<form method="post" action="https://trans.merchantpartners.com/cgibin/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="swipedata"
        value="%B4012888888881^LASTNAME/FIRSTNAME^05121015432112345678?">
<input type="hidden" name="ccnum" value="5454545454545454">
<input type="hidden" name="expmon" value="01">
<input type="hidden" name="expyear" value="2004">
<input type="hidden" name="accepturl"
        value="http://trans.merchantpartners.com/~ats/accepted.html">
<input type="hidden" name="declineurl"
        value="http://trans.merchantpartners.com/~ats/declined.html">
<input type="submit">
</form>
```

## Response Format

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 accepted or declined URL references a static HTML page, it is displayed. If the accepted or declined URL 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 is Accepted

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

```
http://www.myserver.com/cgibin/mycgi?Status=Accepted&AuthNo=AUTHCODE
```

The authorization code is the alphanumeric value returned by the credit card processing network, representing a receipt of each individual transaction.

### Transaction is Accepted and Consumer Info is Returned

If the **usepost** flag is enabled, the POST sent to the **accepturl** is printed with the name/value pairs below:

```
Version=1
Status=Accepted
AuthNo=######
PostedVars=BEGIN
action=ns_quicksale_cc
acctid=TEST0
amount=1.00
ccname=Joe+Customer
usepost=1
accepturl=http://trans.merchantpartners.com/cgi-bin/test.cgi
declineurl=http://trans.merchantpartners.com/cgi-bin/test.cgi
PostedVars=END
```

The POST body is encoded as above. All name/value pairs submitted to the OCS will be posted back to the accept URL (except for consumer financial information).

### Transaction is Declined

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

```
http://www.myserver.com/cgibin/mycgi?Status=Declined&Reason=Reason
```

The reasons for a declined transaction are numerous and depend on the processing network through which your account is routed.

### Transaction is Declined and Consumer Info is Returned

If the usepost flag is enabled, the POST sent to the declineurl is transmitted in the following format:

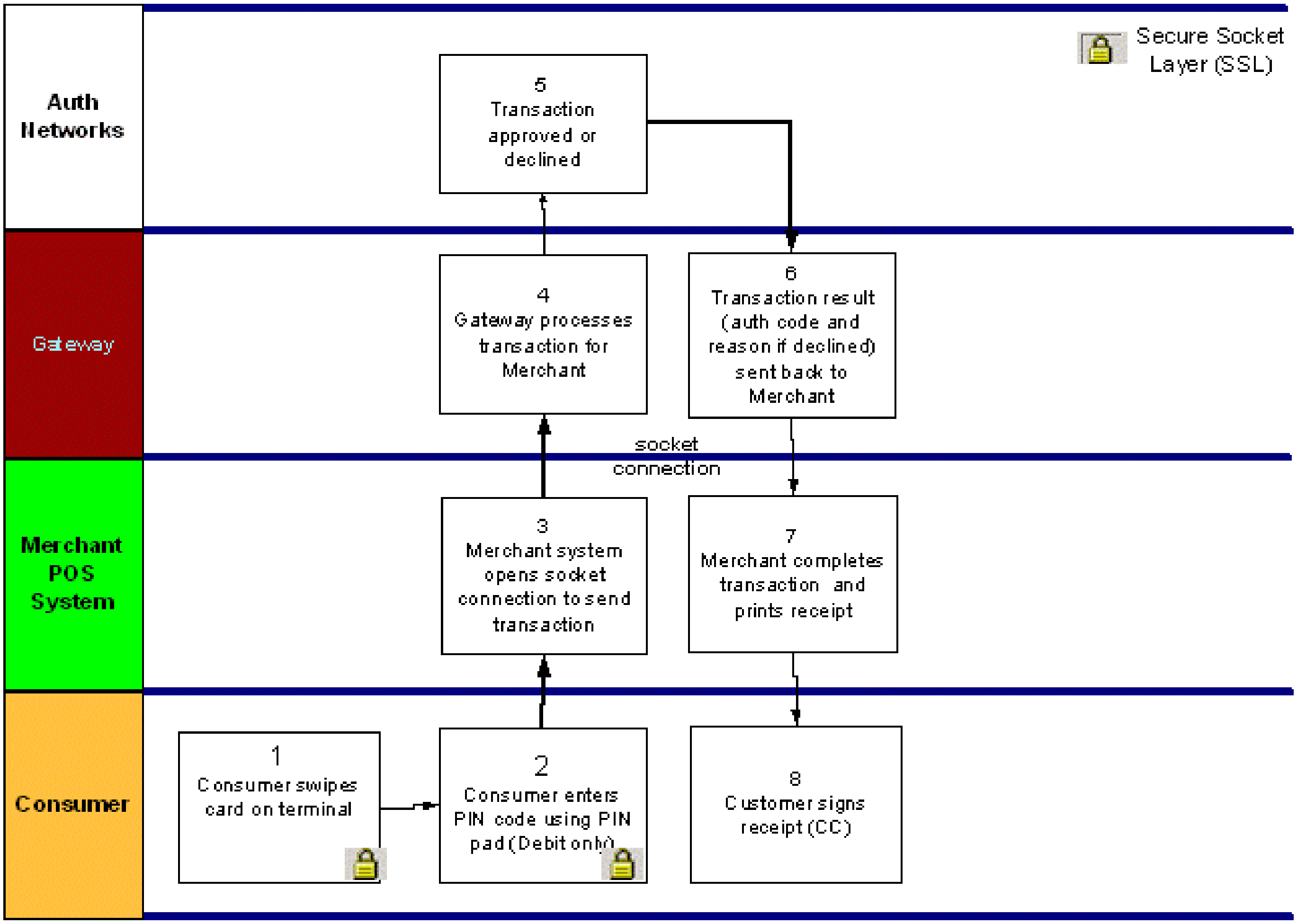
```
Version=1
Status=Decline
Reason= (decline reason goes here)
PostedVars=BEGIN
action=ns_quicksale_ccc
acctid=TEST0
amount=1.00
ccname=Joe+Customer
usepost=1
accepturl=http://trans.merchantpartners.com/cgi-bin/test.cgi
declineurl=http://trans.merchantpartners.com/cgi-bin/test.cgi
PostedVars=END
```

# Quicksale Method with a Socket Connection

This method requires advanced programming skills since your programmers create all of the transaction pages. Your system must be a secure server. Communication between your system and the Online Commerce Suite system is done in single transactions via a direct socket, using HTTP. A socket connection supports interactivity with your customer. Your system submits a single transaction request on behalf of your customer, and gets an immediate transaction result back.

## How It Works

1. Your POS application requests your customer to supply card information via card swipe and PIN pad entry (for debit cards).
2. Your application sends the transaction request data to the Online Commerce Suite server via the direct socket connection. The Online Commerce Suite server then processes the transaction request.
3. The Online Commerce Suite server sends the transaction result data back to your system via the socket connection.
4. Your system processes the result data and displays the appropriate response to the customer.


Auth Networks
Gateway
Merchant POS System
Consumer
Secure Socket Layer (SSL)
5 Transaction approved or declined
4 Gateway processes transaction for Merchant
6 Transaction result (auth code and reason if declined) sent back to Merchant
socket connection
3 Merchant system opens socket connection to send transaction
7 Merchant completes transaction and prints receipt
1 Consumer swipes card on terminal
2 Consumer enters PIN code using PIN pad (Debit only)
8 Customer signs receipt (CC)

## Security

All aspects of transaction security are dependent on the Online Commerce Suite Server. If you choose the socket connection method, your POS system must be secure with SSL to protect your transmissions. You must then decide upon a method to secure transaction request data over the socket connection to the Online Commerce Suite server.

## Requirements

You can receive information about your customer's order in one of three ways:

- Certified credit card swipe reader or DUKPT PIN Pad (for Debit transactions).
- POS application compatible with your hardware.
- POS host capable of transmitting https: over an Internet connection to the Online Commerce Suite payment gateway.

## What Your Programmers Do

Your programmers have more responsibility than with the other transaction processing methods. They create the entire interface to your customer's browsers and must follow specific protocols in communicating with the Online Commerce Suite server. Your programmers must have advanced skills and proficient in HTTP.

## Transaction Types (Actions)

The actions for the socket connection are the same as for the HTTPS post connection method.

## Data Elements

The data elements for the socket connection are the same as for the HTTPS post connection method.

## Socket Connection Data Formats

To prepare to transmit a transaction via the direct socket post method, you must first initiate a socket connection to the Online Commerce Suite Payment Gateway URL.  Use **Port 443** for https (Secure Socket Layer).

### Request Format

When you establish a socket connection to the Online Commerce Suite server, you are ready to transmit the transaction information.

Transactions consist of a series of name/value pairs.  Refer to the HTTPS web form post connection method section of this document for a complete list of values.

**Sample HTTP Request to Submit a Credit Card Transaction**

```
action=ns_quicksale_cc&acctid=TEST0&amount=1.00&ccname=John%20Doe
&swipedata=asdf1234asdf1234&merchantordernumber=1001
```

To use the direct socket post method to transmit transactions, you must be fully proficient with the HTTP protocol. If you are uncertain as to how to correctly use the direct socket post method, use the HTML (Web Form) Submission Method.

### Response Format

After a transaction is submitted via the direct socket submission method, the Online Commerce Suite server responds with the results of the transaction within a few seconds. It is up to your program to interpret the results of the transaction and route the consumer accordingly. Here are sample responses for accepted and declined transactions.

**Transaction is Accepted**

```
Accepted=SALE:VITAL2:414716001201:673:37258251:N::
historyid=37258251
orderid=29999892
```

**Transaction is Declined**

```
Declined=DECLINED:0700540009:INV TRAN TYPE
historyid=37258292
orderid=
```

## Sample Perl program format

The following Perl program snippet gives an example of how the SecurePost with Socket Connection method can be used within an application to submit transactions to the gateway.  This example describes code to submit either a credit card auth or sale.

```
#!/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/trans.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',
                        ci_merchantordernumber => 'ams_account_number',
                        ));

my ($key, $value);

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

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

exit;
```

## Sample Java servlet code snippet

The following Java servlet code snippet gives an example of how the SecurePost with Socket Connection method can be used within a Java application to submit transactions to the gateway.

```
// 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
String saveLine="";

while ((inputLine = in.readLine()) != null)
        saveLine+="&"+inputLine;

in.close();
```

# Quicksale Method with SecurePost Object

## Overview

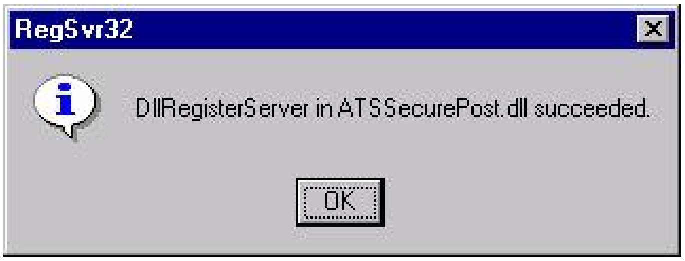
The Online Commerce Suite SecurePost object provides an easy way to integrate credit card or electronic check (ACH) payment processing into your custom applications or Web sites. The SecurePost object is implemented as a COM object capable of running on Windows 95, Windows 98, or Windows NT 4.0 or later. 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 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:



RegSvr32

DllRegisterServer in ATSSecurePost.dll succeeded.

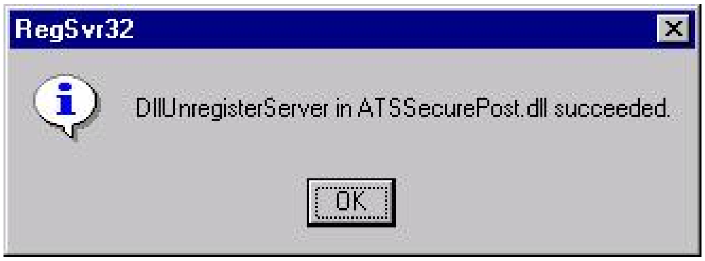
If you get an error message during the registration process, your system contains an old version of the Microsoft Internet Extensions (WININET) libraries. In this case, install the latest version of Microsoft Internet Explorer or its updates.

## Uninstalling

To uninstall the 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:



RegSvr32

DllUnregisterServer in ATSSecurePost.dll succeeded.

## 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.

## Complete Properties

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | All | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| ATSSubID | String(5) | None | | Online Commerce Suite Sub ID, 5 characters long or empty. (SubID) |
| CCName | String(32) | None | All | Debit Card Account Name. (CCName) |
| CCNumber | String(32) | None | CC | Credit Card Number. (CCNum) |
| ExpMonth | String(2) | None | CC | Credit Card Expiration Month. (ExpMon) |
| ExpYear | String(4) | None | CC | Credit Card Expiration Year. (ExpYear) |
| CVV2 | String(3) | None | | Credit Card CVV2 Code (CVV2) |
| CKName | String(32) | None | ACH | Checking Account Name. (CKName) |
| CKRoutingNumber | String(9) | None | ACH | Checking Account Routing (ABA) Number, always 9 digits long. (CKABA) |
| CKAccountNumber | String(18) | None | ACH | Checking Account Number. (CKAcct) |
| Amount | Integer > 0 | 0 | All | The total amount to be authorized. |
| CashBackAmount | Integer > 0 | 0 | | The amount to be issued to the consumer as cash. |
| CardPresent | Short Integer | 0 | | 0 if not present, 1 if present. |
| Swipedata | String(256) | None | DB | Track 1 and/or Track 2 swipe data |
| CardReaderPresent | Short Integer | 0 | | 0 if not present, 1 if present. |
| CustomerID | String(32) | None | DB | Encrypted PIN Block Data |
| VoiceAuth | String(6) | None | | 6 character approval code obtained from the cardholder's issuing bank. |
| CI_CompanyName | String(64) | None | | The consumer's company name. (CI_CompanyName) |
| CI_BillAddr1 | String(64) | None | | The consumer's billing address. (CI_BillAddr1) |
| CI_BillAddr2 | String(64) | None | | The consumer's billing address. (CI_BillAddr2) |
| CI_BillCity | String(32) | None | | The consumer's billing city. (CI_BillCity) |
| CI_BillState | String(32) | None | | The consumer's billing state. (CI_BillState) |
| CI_BillZip | String(16) | None | | The consumer's billing zip code. (CI_BillZip) |
| CI_BillCountry | String(32) | None | | 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_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_IPAddress | String(16) | None | | The consumer's IP address. (CI_IPAddress) |
| 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) |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| ResultHostMessageID | Read-Only String | | | Unique host message identifier assigned by the authorizing system |
| ResultSystemTraceNumber | Read-Only String | | | Six character numeric unique host message assigned by the authorizing system |
| DevMode | Boolean | False | | Determines whether the SecurePost object runs in development mode. |
| 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 | | 300 | | |

| | | | | |
|---|---|---|---|---|
| | Short Integer<br>>= 5 & <= 600 | | | The maximum number of seconds that the SecurePost object will wait for a response from the authorization network.<br>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. |

# Methods

## ProcessSale

Processes a card present credit card 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.

### Minimum Required Properties for ProcessSale for Credit Card

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |
| **Additional required fields for Hotel/Lodging** | | | | |
| IndustryCode | String(1) | None | | One character value: "H": Hotel |
| ChargeType | String(1) | None | | Code indicating whether the primary charge is Lodging, Restaurant, or Gift Shop. One character value: "1" Hotel, "2" |

| | | | | |
|---|---|---|---|---|
| | | | | Restaurant, "3" Gift Shop |
| Roomrate | Short Integer > 0 | 0 | | Room rate dollar amount in US dollars in the form of 150 (one dollar fifty cents) |
| Checkindate | Date yymmdd | None | | The date (yymmdd) that the guest checked in to the hotel |
| Checkoutdate | Date yymmdd | None | | The date (yymmdd) that the guest checked out of the hotel |
| PurchaseID | String(25) | None | | Hotel Folio number (25 characters alphanumeric) |
| Pproperty | Short Integer | None | | Prestigious property indicatory. Used by participants in Visa's Prestigious Lodging Program. A transaction amount of $1 should be entered in the amount field if the merchant wishes the transaction to participate in the program. Number value: 500, 1000, or 1500 |
| Extracharges | String(6) | None | | 6 1-digit codes, each a partial or complete explanation of why charged amount differs from receipt cardholder received at checkout. Digit values: 0: no extra, 2: Restaurant, 3: Gift Shop, 4: Mini-bar, 5: Telephone, 6: Other, 7: Laundry. Example: "240000" indicates restaurant and mini-bar charges. |

## ProcessSale with Voice Authorization

A Voice Authorization transaction is a sale performed after obtaining an approval code from the cardholder's issuing bank. Use the VoiceAuth property to set the 6 character voice authorization code obtained from the issuing bank, then invoke the ProcessSale method.

**Minimum Required Properties for ProcessSale with Voice Authorization**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| VoiceAuth | String(6) | None | • | 6 character approval code obtained from the cardholder's issuing bank. |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessSale for Check Conversion

Processes a check conversion 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.

### Minimum Required Properties for ProcessSale for Check Conversion

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |
| CKType | String(3) | None | • | "POP" or "ARC" |
| CKNo | String(15) | None | • | Check Number (serial number) |
| CKRoutingNumber | String(9) | None | • | Checking Account Routing (ABA) Number, always 9 digits long. (CKABA) |
| CKAccountNumber | Strung(18) | None | • | Checking Account Number. (CKAcct) |
| CKMicrdata | String(64) | None | • | MICR data read from the MICR line |
| CKTerminalcity | String(4) | None | • | 4 character abbreviated name for the city where the POS terminal is located. Required for POP transactions |
| CKTerminalstate | String(2) | None | • | 2 character state abbreviation of the state where the POS terminal is located. Required for POP transactions |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessSale for Check Conversion with image capture

Processes a check conversion 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.

### Minimum Required Properties for ProcessSale for Check Conversion with image capture

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |

| | | | | |
|---|---|---|---|---|
| CKType | String(3) | None | • | "POP" or "ARC" |
| CKNo | String(15) | None | • | Check Number (serial number) |
| CKRoutingNumber | String(9) | None | • | Checking Account Routing (ABA) Number, always 9 digits long. (CKABA) |
| CKAccountNumber | Strung(18) | None | • | Checking Account Number. (CKAcct) |
| CKMicrdata | String(64) | None | • | MICR data read from the MICR line |
| CKTerminalcity | String(4) | None | • | 4 character abbreviated name for the city where the POS terminal is located. Required for POP transactions |
| CKTerminalstate | String(2) | None | • | 2 character state abbreviation of the state where the POS terminal is located. Required for POP transactions |
| CkImagefront | String(128) | None | • | File path and file name where the front check image file is located (i.e. c:\28294058.tif) |
| CKImageback | String(128) | None | • | File path and file name where the back check image file is located (i.e. c:\28294058.tif) |
| CKImagetype | String(3) | None | • | "TIF", "GIF", or "JPG" |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessAuth

Processes a card present credit card 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.

### Minimum Required Properties for ProcessAuth

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. 150 (one dollar fifty cents) |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |
| **Additional required fields for Hotel/Lodging** | | | | |
| IndustryCode | String(1) | None | | One character value: "H": Hotel |
| ChargeType | String(1) | None | | Code indicating whether the primary charge is Lodging, Restaurant, or Gift Shop. One character value: "1" Hotel, "2" Restaurant, "3" Gift Shop |
| Roomrateamount | Short Integer > 0 | 0 | | Room rate dollar amount in US dollars in the form of 150 (one dollar fifty cents) |
| Checkindate | Date yymmdd | None | | The date (yymmdd) that the guest checked in to the hotel |
| Checkoutdate | Date yymmdd | None | | The date (yymmdd) that the guest checked out of the hotel |
| PurchaseID | String(25) | None | | Hotel Folio number (25 characters alphanumeric) |
| Pproperty | Short Integer | None | | Prestigious property indicatory. Used by participants in Visa's Prestigious Lodging Program. A transaction amount of $1 should be entered in the amount field if the merchant wishes the transaction to participate in the program. Number value: 500, 1000, or 1500 |
| Extracharges | String(6) | None | | 6 1-digit codes, each a partial or complete explanation of why charged amount differs from receipt cardholder received at checkout. Digit values: 0: no extra, 2: Restaurant, 3: Gift Shop, 4: Mini-bar, 5: Telephone, 6: Other, 7: Laundry. Example: "240000" indicates restaurant and mini-bar charges. |

## ProcessPost (BSTR RefCode)

Processes a post for a previous credit card auth-only transaction. The only properties that are used and must be set are the *ATSID* and *ATSSubID*. 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.

### Minimum Required Properties for ProcessPost

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| 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. This value is passed as an argument to the ProcessPost method |
| ResultOrderID | Read-Only String | | | The Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessCredit (BSTR TransID, BSTR OrderID)

Processes a card present 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.

**Minimum Required Properties for ProcessCredit**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| 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 Order ID assigned to the transaction. Passed as an argument. |
| ResultTransID | Read-Only String | | • | The Transaction ID assigned to the transaction. Passed as an argument |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessVoid (BSTR TransID)

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.

**Minimum Required Properties for ProcessVoid**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | • | The Transaction ID assigned to the transaction. Passed as an argument. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessQuasiCash

Processes a quasicash 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.

**Minimum Required Properties for ProcessQuasiCash**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessDebit

Processes a debit 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. Track 2 swipe data is required for all Debit transactions (no manual entry option is available).

### Minimum Required Properties

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| CCName | String(32) | None | | Debit Card Account Name. (CCName) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |
| CashBackAmount | Short Integer > 0 | 0 | | The amount to be issued to the consumer as cash. |
| Swipedata | String(256) | None | • | Full Track 2 swipe data |
| CustomerID | String(32) | None | • | Encrypted PIN Block Data. Includes 16 byte PIN data followed by 6 byte key set and 10 byte PIN Pad Serial number |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## ProcessDebitReturn (BSTR TransID, BSTR OrderID)

Processes a credit for a debit account. The only properties that are used and must be set are the *ATSID* and *ATSSubID*. In addition, the ProcessDebitReturn 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. Track 2 swipe data is required for all Debit Return transactions (no manual entry option is available).

### Minimum Required Properties

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| CCName | String(32) | None | | Debit Card Account Name. (CCName) |
| Amount | Short Integer > 0 | 0 | • | The total amount to be authorized. |

| | | | | |
|---|---|---|---|---|
| CashBackAmount | Short Integer > 0 | 0 | | The amount to be issued to the consumer as cash. |
| Swipedata | String(256) | None | • | Full Track 2 swipe data |
| CustomerID | String(32) | None | • | Encrypted PIN Block Data. Includes 16 byte PIN data followed by 6 byte key set and 10 byte PIN Pad Serial number |
| 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 Order ID assigned to the transaction. |
| ResultTransID | Read-Only String | | | The Transaction ID assigned to the transaction. |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## AuthReverse (BSTR TransID, BSTR OrderID)

For hotel/lodging transactions only. Visa transactions only. Reduces the amount of an existing authorization by the amount specified. The only properties that are used and must be set are the *ATSID* and *ATSSubID, Amount, and Swipedata*. In addition, the ProcessCredit method requires as arguments the *ResultTransID* and *ResultOrderID* of the original pre-auth transaction. 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.

**Minimum Required Properties for AuthReverse**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | This is the amount by which the original authorization will be reversed. Amount cannot exceed amount of original preauth prior auth reversals). |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| 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 Order ID assigned to the transaction. Passed as an argument. |
| ResultTransID | Read-Only String | | • | The Transaction ID assigned to the transaction. Passed as an argument |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |

| | | | | |
|---|---|---|---|---|
| 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. |

## IncrementalAuth (BSTR TransID, BSTR OrderID)

For hotel/lodging transactions only. Visa transactions only. Increases the amount of an existing authorization by the amount specified. The only properties that are used and must be set are the *ATSID* and *ATSSubID, Amount, and Swipedata*. In addition, the ProcessCredit method requires as arguments the *ResultTransID* and *ResultOrderID* of the original pre-auth transaction. 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.

**Minimum Required Properties for IncrementalAuth**

| Name | Type & Size | Default | Required | Purpose, and Online Commerce Suite tag |
|---|---|---|---|---|
| ATSID | String(5) | None | • | Online Commerce Suite Account ID, always 5 characters long. (ATSID) |
| Amount | Short Integer > 0 | 0 | • | This is the amount by which the original authorization will be increased. |
| Swipedata | String(256) | None | • | Credit Card swipe data (must include either track 1 or track 2 data) |
| 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 Order ID assigned to the transaction. Passed as an argument. |
| ResultTransID | Read-Only String | | • | The Transaction ID assigned to the transaction. Passed as an argument |
| ResultAuthCode | Read-Only String | | | The authorization code of the transaction. |
| 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. |

## 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, ProcessAuth, or ProcessDebitAuth 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, ProcessAuth, or ProcessDebitAuth 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 Online Commerce Suite Account 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 ficticious 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 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
```

```
%>
<HTML>
<BODY>
<%

If MyObject.ResultAccepted Then
%>
    Accepted: <%=MyObject.ResultAuthCode%>
<%
Else
        If MyObject.ResultErrorFlag Then
%>
                Error: <%=MyObject.LastError %>
<%
        Else
%>
                Declined: <%=MyObject.ResultAuthCode%>
<%
        End If
End If

%>
</BODY>
</HTML>
<%

Set MyObject = Nothing

%>
```

## 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 Response 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
```

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

### Transaction Approval Authorization Response Format

| Field | Description | Value |
|---|---|---|
| Transaction Type | Type of transaction submitted | SALE<br>AVSSALE<br>AUTH<br>AVSAUTH<br>POST<br>AVSPOST<br>VOICEPOST<br>VOID<br>CREDIT<br>QUASICASH |
| 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 |

## Credit Card Decline Response Format

The transaction decline authorization response message consists of the string "DECLINED" 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 approval response message.

## Transaction Decline Authorization Reponse

| Field | Description | Value |
|---|---|---|
| Transaction Result | Result of the transaction | DECLINED |
| Decline Code | 10 digit decline code. | *First Digit:*<br>0 Authorizing network declined the transaction.<br>1 Gateway declined the transaction.<br>2 Authorizing network returned an error, forcing a decline.<br>3 Gateway returned an error, forcing a decline.<br>*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 | |
| Cuba | CUP | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |
| Gambia | GMD | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |
| Serbia | RSD | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |
| Macau | MOP | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |
| São Tomé and Príncipe | STD | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |
| Vietnam | VND | |

| Country | Currency Code | Decimal |
|---|---|---|
| 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 | |