Technical Specifications v1.2.0
November 2014

NeSecure Payments Gateway API

Transactions and Reporting

Table of Contents

Transaction API Overview 3

Server Connections and Sample Credentials 3

List of Transaction Request Fields 5

XSD Schema 11

Sample Code 11

Basic Transaction Fields 11

Running a SALE transaction 12

Running an Authorization 13

Capturing an Authorization 15

Refunding a Payment 15

Partial Refunds 16

Issuing a Saleless Refund 17

Voiding a Payment 17

Voiding a Specific Transaction 18

Partial Approvals 18

Cheque Transactions 19

Voiding a Cheque 20

Transaction Reporting API 21

Understanding Orders, Payments, and Transactions 21

Request Format 23

Important Notes for the Reporting API: 29

Merchant Administration API 31

Adding a Merchant 31

Updating a Merchant 32

Activating a Merchant 33

Deactivating a Merchant 33

Adding Users to a merchant 34

Updating Users 36

Activating and Deactivating Users 36

Merchant Reporting API 37

API Definition 37

Searching for a Merchant: 39
Finding a Merchant by User: 40
Checking for differences against a merchant record: 40

Transaction API Overview

Server Connections and Sample Credentials

Transactions are issued to the NetSecure gateway service via XML requests issued via standard HTTP POST messages.

The API may be tested against the Sandbox URL by posting requests in the formats provided below.

Note: The sandbox environment is a public testing environment. Do not use production data or real credit cards!

Sandbox URL:

https://sandbox.netsecuretechnologies.com/Gateway/GatewayHandler.ashx

Sandbox Account:

Username: **demo@netsecure.ca**

Password: **demo**

Sample Credit Card:

Card Type: VISA

Card Number: 4012888888881881

Card Expiry Date: 10/16

(CVV and Cardholder Name can be chosen at the user's discretion).

Transactions are device-agnostic and can be submitted from any host, provided the request format specified in this document is adhered. The API supports a variety of entry modes, including signature and PIN debit, credit EMV, and NFC, and Gift/Loyalty cards. In addition, the API supports a variety of mobile payment hardware dongles, which can be used to provide card-present transactions.

The transaction API supports the following entry methods:

Supported Entry Methods:

ENTRY MODE	DEBIT	CREDIT	GIFT/LOYALTY
Card Present SWIPED	Yes	Yes	if card supported
Card Present KEYED	Partial*	Yes	if card supported
Card Present DIPPED	Yes	Yes	if card supported
Card Present TAPPED	Yes	Yes	if card supported
Card Not Present KEYED	Partial*	Yes	if card supported

*Signature debit only.

Supported Hardware Devices:

HARDWARE	DEBIT	CREDIT	GIFT/LOYALTY
AnywhereCommerce Rover	PIN or Signature	Yes	if card supported
AnywhereCommerce Rambler	Signature	Yes	if card supported
AnywhereCommerce Nomad	PIN or Signature	Yes	if card supported
AnywhereCommerce Walker	Signature	Yes	If card supported
NetSecure SmartSwipe	Signature	Yes	if card supported

List of Transaction Request Fields

All fields available and their corresponding descriptions are provided in the table below. Sample requests and responses are provided below.

Field	Required	Description
<Request>		
<AccountNumber>9788100663</AccountNumber>	Required for CHEQUE transactions	The account number written on the cheque
<Address>123 Main Street</Address>	Required for KEYED transactions that use AVS	The address of the cardholer. Used for AVS to validate the authenticity of the card.
<CVV2>999</CVV2>	Required for KEYED transactions that use CVV	The 3 or 4 digit security code on the back of the card. Used for keyed-in transactions to validate the authenticity of the card.
<CardExpMonth>10</CardExpMonth>	Required for KEYED transactions	The 2-digit card expiry month
<CardExpYear>16</CardExpYear>	Required for Keyed transactions	The 2-digit card expiry year
<CardName>CARDHOLDER/JOE</CardName>	Optional	The cardholder's name
<CardNum>4012888888881881</CardNum>	Required for KEYED transactions	The account number printed on the front of the card (PAN).
<CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>	Optional	A token representing the card. Provided by the gateway when tokenization is enabled. Can be used in place of the card fields.
<ChequeNumber>1060</ChequeNumber>	May be Required for CHEQUE	The cheque number

	transactions (Depends on processor)	
<ChequeImageFront>[base64data]</ChequeImageFront>	May be Required for CHEQUE transactions (Depends on processor)	A base-64 encoded image of the front of the cheque. Must be in PNG, JPEG, or TIFF format
<ChequeImageBack>[base64data]</ChequeImageBack>	May be Required for CHEQUE transactions (Depends on processor)	A base-64 encoded image of the back of the cheque. Must be in PNG, JPEG, or TIFF format
<CustomerEmail>customer@email.com</CustomerEmail>	Optional	The email address of the customer. Receipts are emailed to this address.
<CustomerDetails />	Optional	A structure containing information about the customer. See XSD schema for options.
<EncryptedSwipe />	Optional	A structure containing information resulting from the swipe of a card through an external dongle. See XSD Schema for options.
<EntryMode>SWIPED</EntryMode>	Required for CREDIT and DEBIT transactions	The entry mode of the card. Can be SWIPED, KEYED, TAPPED, DIPPED, or SCANNED.
<Fee>0.00</Fee>	Optional	A user-defined fee for the transaction.
<Host>iPhone 4S:iOS5.1:en-CA</Host>	Recommended	The hardware and software type, language, and locale of the device or installation that the transaction Is being run from. Recommended for analytics and

		debugging.
<InstallationID>723b0b3b-042e-4a70-aeb1-bf3618d2dd9c</InstallationID>	Recommended	A unique identifier for the installation instance that this transaction is being run from. This should remain unchanged for the lifetime of the installation. Must be GUID type.
<InvoiceNumber>INV123456<InvoiceNumber>	Optional	A user-defined invoice number.
<InvoiceData>[user-supplied content]</InvoiceData>	Optional	Freeform data field that can be used to store invoice information. Can be base64-encoded binary or string
<Latitude>10.424220975</Latitude>	Optional	Used to geo-locate the transaction
<Longitude>-10.527407625</Longitude>	Optional	Used to geo-locate the transaction
<MICR>T490000018T1060U9788100663U</MICR>	May be Required for CHEQUE transactions (Depends on processor)	The MICR number on the bottom of the cheque.
<OrderID>123456</OrderID>	Optional	User-defined field used to group related payments.
<Password>pass</Password>	Required	The password of the user running the transaction.

<PaymentID>20130505121242560323</PaymentID>	Required for CAPT, REVERSEAUTH, VOID (unless RefTransactionID is provided), or REFUND (unless performing a saleless refund)	A unique identifier for the payment object created by the server during SALE or AUTHONLY transactions. Used to group related transactions (such as VOID, CAPT, REFUND, etc..). Used to associate this transaction with the specified payment.
<PaymentType>DEBIT</PaymentType>	Recommended	Specifies what type of payment medium is being used. Valid options are CREDIT, DEBIT, CHECK and GIFT_CARD
<PostalCode>90210</PostalCode>	Required for KEYED transactions that use AVS	The postal code corresponding to the billing address of the card.
<RefTransactionID>20130505121242560323</RefTransactionID>	Required for VOID (unless PaymentID is provided)	The ID of the transaction being referenced by this transaction. For example, to void a specific transaction, put the TransactionID of the transaction to be voided in this field.
<RequestID>7ac12423-b628-4709-964c-07984c74ab7d</RequestID>	Recommended	A unique identifier for the request to prevent duplicate transactions and replays. Must be GUID type.
<Signature>[Base64 Encoded Image Data]</Signature>	Optional	A base64 encoded image of the cardholder's signature capture.
<Routing>490000018</Routing>	May be Required for CHEQUE transactions (Depends on	The routing number on the cheque.

	processor)	
<Subtotal>10.00</Subtotal>	Optional	The subtotal of the transaction.
<Tax1>0.50</Tax1>	Optional	A tax applied to this transaction.
<Tax1Rate>5.00%</Tax1Rate>	Optional	The rate of tax used to calculate Tax1
<Tax2>0.00</Tax2>	Optional	A second tax applied to this transaction
<Tax2Rate>5.00%</Tax2Rate>	Optional	The tax rate used to calculate Tax2
<Tip>5.00</Tip>	Optional	The tip amount for this transaction.
<TipAuthAmount>15.50</TipAuthAmount>	Optional	Used only for AUTHONLY transactions. If present, this value will be added to the TotalAmount field when sent to the payment processor for authorization. This can be used to ensure sufficient funds are available on the card to cover a tip.
<TotalAmount>15.50</TotalAmount>	Required for AUTHONLY, SALE, CAPT, REFUND, REVERSEAUTH	The total requested amount for the transaction. This amount is sent through to the payment processor (regardless of what Subtotal, Tax, Tip, and Fee field values are). The exception is for AUTHONLY transactions: if TipAuthAmount is provided, it will be added to this amount when sent to the processor to ensure the payment method has sufficient funds

		to cover the tip.
<TransactionID>20130505121242560323</TransactionID>	**Strongly Recommended**	A unique identifier for the current transaction. If not provided, it will be generated by the server and returned in the Response. It is strongly recommended that the TransactionID is generated on the client side to assist in state management, particularly in volatile environments like mobility where a stable internet connection is not always guaranteed. Must be in format yyyyMMddHHssmmSSS and ideally should be generated from the system time at the moment the transaction is created.
<TransactionType>AUTHONLY</TransactionType>	Required	The type of transaction. Must be one of AUTHONLY, SALE, CAPT, REFUND, REVERSEAUTH, VOID, ADMIN
<Username>user@domain.com</Username>	Required	The user running the transaction.
<Version>AppName v1.2.3</Version>	Recommended	The application name and version of the application running the request. Recommended for

		analytics and debugging.
</Request>		

XSD Schema

An XSD schema is available upon request. Please contact your integration specialist for access.

Sample Code

Sample code in Objective C, Java, and C# is available upon request. Please contact your integration specialist for access.

Basic Transaction Fields

Every transaction starts with a header, which includes the TransactionType, Username, and Password as mandatory fields.

Minimum Required fields:

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
</Request>
```

In addition, for support and tracking purposes, we strongly recommend adding the following **Recommended** fields: InstallationID, RequestID, Host, and Version. Details of each field and the meaning are available in the table above:

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <InstallationID>52A471C6-A6BA-474E-B8AF-8FE292FC0ABF</InstallationID>
  <RequestID>F19A2946-AEBA-4AAF-8F30-9AAD255C74D6</RequestID>
  <Host>iPhone6:iOS8.1:AppName-v1.0.1:en-US</Host>
  <Version>2</Version>
</Request>
```

For Keyed –in Transactions, it is strongly recommended to include the cardholder's address and ZIP code for AVS (Address Verification Service). This will ensure the transaction qualifies for the best rates.

Running a SALE transaction

Below is an example of running a swiped PIN debit SALE transaction, using the ACRover debit hardware. To run a credit card or cheque transaction, substitute the EncryptedSwipe for the appropriate cheque or credit card details.

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentType>DEBIT</PaymentType>
  <EntryMode>SWIPED</EntryMode>
  <EncryptedSwipe>
    <ReaderUsed>ACRover</ReaderUsed>
    <Track1>D9558F8588D65D8E65D032C(etc..)</Track1>
    <Track2>122C0C0D5F75D55C81F46E8(etc..)</Track2>
    <Track3>23048084AA8908BC72AA32F(etc..)</Track3>
    <TimeStamp>A53E83891117</TimeStamp>
    <EPB>CC4A9CBCBA76BB30</EPB>
    <EPBKsn>FFFF9876543210E00011</EPBKsn>
    <Ksn>FFFF0987654321E00001</Ksn>
  </EncryptedSwipe>
  <CustomerEmail>customer@email.com</CustomerEmail>
  <Subtotal>10.00</Subtotal>
  <Tax1>0.50</Tax1>
  <Tip>5.00</Tip>
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>
```

```
<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST18</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <ApprovedAmount>15.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

Below is an example of running a keyed-in SALE transaction.

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentType>CREDIT</PaymentType>
  <EntryMode>KEYED</EntryMode>
  <CardName>JOE CARDHOLDER</CardName>
```

```
  <CardNum>4012888888881881</CardNum>
  <CardExpMonth>10</CardExpMonth>
  <CardExpYear>16</CardExpYear>
  <Address>123 Main Street</Address>
  <PostalCode>90210</PostalCode>
  <CVV2>999</CVV2>
  <CustomerEmail>customer@email.com</CustomerEmail>
  <Subtotal>10.00</Subtotal>
  <Tax1>0.50</Tax1>
  <Tip>5.00</Tip>
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>
```

If AVS/CVV is used, the response will contain the AVS and CVV information as well.

```
<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST18</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560323</TransactionID>
  <AVS>Y</AVS>
  <CVV>M</CVV>
  <PaymentID>20130505121242560323</PaymentID>
  <ApprovedAmount>15.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

Running an Authorization

Below is an example of running a swiped credit AUTHONLY transaction, using the ACRambler hardware.

```
<Request>
  <TransactionType>AUTHONLY</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentType>CREDIT</PaymentType>
  <EntryMode>SWIPED</EntryMode>
  <EncryptedSwipe>
    <ReaderUsed>ACRambler</ReaderUsed>
    <Track1>D9558F8588D65D8E65D032C(etc..)</Track1>>
    <Track1Length>80</Track1Length>
    <Ksn>FFFF0987654321E00001</Ksn>
  </EncryptedSwipe>
  <CustomerEmail>customer@email.com</CustomerEmail>
  <Subtotal>10.00</Subtotal>
  <Tax1>0.50</Tax1>
  <Tip>5.00</Tip>
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
```

```
</Request>


<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST18</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <ApprovedAmount>15.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

Below is an example of running a keyed-in AUTHONLY transaction. In this case, the TipAuthAmount is provided to ensure the card has available funds to cover the total and a tip.

```
<Request>
  <TransactionType>AUTHONLY</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentType>CREDIT</PaymentType>
  <EntryMode>KEYED</EntryMode>
  <CardName>JOE CARDHOLDER</CardName>
  <CardNum>4012888888881881</CardNum>
  <CardExpMonth>10</CardExpMonth>
  <CardExpYear>16</CardExpYear>
  <Address>123 Main Street</Address>
  <PostalCode>90210</PostalCode>
  <CVV2>999</CVV2>
  <CustomerEmail>customer@email.com</CustomerEmail>
  <Subtotal>10.00</Subtotal>
  <Tax1>0.50</Tax1>
  <TipAuthAmount>5.00</TipAuthAmount>
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>
```

Note that the ApprovedAmount in the response is equal to TotalAmount + TipAuthAmount.

```
<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST18</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560323</TransactionID>
  <AVS>Y</AVS>
  <CVV>M</CVV>
  <PaymentID>20130505121242560323</PaymentID>
  <ApprovedAmount>20.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

Note that if partial approvals are supported by the payment processor, ApprovedAmount may be less than the TotalAmount, in which case the transaction has been partially approved. Please see the Partial Approvals section for further details.

Capturing an Authorization

An authorization can be captured by providing the PaymentID of the authorization in the request, and ensuring the amount is set. In this example, we will capture the Authorized transaction in the example above.

```
<!-- CAPT transaction-->
<Request>
  <TransactionType>CAPT</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560999</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <Tip>2.00</Tip>
  <TotalAmount>17.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>


<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST21</ApprovalCode>
  <ResponseText>CAPTURED</ResponseText>
  <TransactionID>20130505121242560999</TransactionID>
</Response>
```

Note that the authorization's ApprovedAmount was for $20.50, but the final transaction was $17.50. The outstanding difference between the authorized amount and the captured amount will be reversed by the processor.

Refunding a Payment

Refunds are issued using the CREDIT transaction type. Specify the amount of the refund in the TotalAmount field, and the payment to refund using the PaymentID field. Note that refunds cannot exceed the value of the original payment.

```
<!-- REFUND transaction-->
<Request>
  <TransactionType>CREDIT</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560777</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
```

```
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>


<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST23</ApprovalCode>
  <ResponseText>REFUNDED</ResponseText>
  <TransactionID>20130505121242560777</TransactionID>
</Response>
```

Partial Refunds

Refunds can be run for less than the full value of the transaction. For example, if an order contains three items, and the customer wants to return one of them, the value of that item can be refunded from the payment.

To specify a partial refund, simply specify a TotalAmount that is lower than the payment's CurrentAmount field. If successful, the Payment's CurrentAmount field will be reduced by the amount of the refund.

Any number of partial refunds can be run against a payment, as long as the payment's CurrentAmount does not reach zero. Once it reaches zero, the payment is considered fully refunded and can no longer be refunded.

```
<!-- PARTIAL REFUND transaction-->
<Request>
  <TransactionType>CREDIT</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560777</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <TotalAmount>5.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>


<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST23</ApprovalCode>
  <ResponseText>PARTIALLY_REFUNDED</ResponseText>
  <TransactionID>20130505121242560777</TransactionID>
</Response>
```

Issuing a Saleless Refund

Refunds can be created without a corresponding payment object. They are standalone transactions and will not be grouped into any payment. Saleless refunds must contain card details and resemble SALE transactions. In a sense, a saleless refund can be considered a negative sale.

```
<!-- SALELESS REFUND transaction-->
<Request>
  <TransactionType>CREDIT</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560555</TransactionID>
  <PaymentType>CREDIT</PaymentType>
  <EntryMode>KEYED</EntryMode>
  <CardName>JOE CARDHOLDER</CardName>
  <CardNum>4012888888881881</CardNum>
  <CardExpMonth>10</CardExpMonth>
  <CardExpYear>16</CardExpYear>
  <Address>123 Main Street</Address>
  <PostalCode>90210</PostalCode>
  <CVV2>999</CVV2>
  <CustomerEmail>customer@email.com</CustomerEmail>
  <TotalAmount>15.50</TotalAmount>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>


<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST23</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560555</TransactionID>
  <ApprovedAmount>15.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

Voiding a Payment

To void a payment, use the VOID command:

```
  <!-- VOID a payment-->
  <Request>
    <TransactionType>VOID</TransactionType>
    <Username>user@domain.com</Username>
    <Password>pass</Password>
    <TransactionID>20130505121242560999</TransactionID>
    <PaymentID>20130505121242560323</PaymentID>
    <Host>iPhone 4S:iOS5.1:en-CA</Host>
    <Version>AppName v1.2.3</Version>
  </Request>


  <Response>
    <Approved>true</Approved>
    <ResponseText>VOIDED</ResponseText>
```

```
  <TransactionID>20130505121242560999</TransactionID>
</Response>
```

Voiding a Specific Transaction

In some cases, you may not want to void the whole payment, but rather a single transaction. For example, if a SALE transaction was refunded erroneously, you may be able to void the REFUND transaction, leaving the original sale intact.

To void a specific transaction, use the RefTransactionID field instead of the PaymentID field, and populate it with the TransactionID of the specific transaction to be voided.

```
<!-- VOID a specific transaction-->
<Request>
  <TransactionType>VOID</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560999</TransactionID>
  <RefTransactionID>20130505121242560777</RefTransactionID>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>


<Response>
  <Approved>true</Approved>
  <ResponseText>VOIDED</ResponseText>
  <TransactionID>20130505121242560999</TransactionID>
</Response>
```

Partial Approvals

In some circumstances, especially when using prepaid or gift cards with a limited available balance, a SALE or AUTHONLY transaction may be approved for less than the TotalAmount. In these cases, this is referred to as a *Partial Approval*. The amount approved is provided in the ApprovedAmount field of the response. If this field is present, it should always be checked to see if it is less than the TotalAmount. If so, the user should be notified to run another transaction for the difference between the requested amount and the approved amount.

Cheque Transactions

In addition to credit and debit transactions, the API also supports Cheque transactions.

Cheque transactions follow the same format as Credit Card transactions, except one should substitute the credit card details for cheque details.

Optionally, an image of the check in JPEG or PNG format can be provided in the ChequeImageFront and ChequeImageBack fields. The data must be presented as a base-64 encoded string.

Note: Only SALE and VOID transactions are supported for cheques. It is not possible to run a REFUND transaction on a cheque.

Sample check request:

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20141010234323123</TransactionID>
  <TotalAmount>100.00</TotalAmount>
  <PaymentType>CHEQUE</PaymentType>
  <MICR>T490000018T1060U9788100663U</MICR>
  <AccountNumber>9788100663</AccountNumber>
  <Routing>490000018</Routing>
  <ChequeNumber>1060</ChequeNumber>
  <ChequeImageFront>[base64-encoded check data]</ChequeImageFront>
  <ChequeImageBack>[base64-encoded check data]</ChequeImageBack>
</Request>
```

The response follows the same format as credit card transactions:

```
<Response>
  <Approved>true</Approved>
  <ApprovalCode>TEST18</ApprovalCode>
  <ResponseText>APPROVED</ResponseText>
  <TransactionID>20130505121242560323</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <ApprovedAmount>15.50</ApprovedAmount>
  <CardToken>KXKKWKXKVVKWWKXKKVV</CardToken>
</Response>
```

If your check service provider requires ID verification, you can include the relevant identification in the CustomerDetails structure.

```
<Request>
  <TransactionType>SALE</TransactionType>
  <Username>testmerchant@kudospayments.com</Username>
  <Password>testpwd</Password>
  <TransactionID>20141010234323123</TransactionID>
```

```
  <TotalAmount>100</TotalAmount>
  <PaymentType>CHEQUE</PaymentType>
  <MICR>T490000018T1060U9788100663U</MICR>
  <AccountNumber>9788100663</AccountNumber>
  <Routing>490000018</Routing>
  <ChequeNumber>1060</ChequeNumber>
  <ChequeImageFront>[base64-encoded check data]</ChequeImageFront>
  <ChequeImageBack>[base64-encoded check data]</ChequeImageBack>
  <CustomerDetails>
    <DOB>1978-12-10</DOB>
    <DriversLicense>102010333</DriversLicense>
    <DriversLicenseState>NY</DriversLicenseState>
  </CustomerDetails>
</Request>
```

Voiding a Cheque

Voiding a cheque happens in the same manner as any other transaction. Simply provide the PaymentID of the cheque sale to be voided. It is not necessary to provide any other information.

```
<Request>
  <TransactionType>VOID</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <TransactionID>20130505121242560999</TransactionID>
  <PaymentID>20130505121242560323</PaymentID>
  <Host>iPhone 4S:iOS5.1:en-CA</Host>
  <Version>AppName v1.2.3</Version>
</Request>
```

```
<Response>
  <Approved>true</Approved>
  <ResponseText>VOIDED</ResponseText>
  <TransactionID>20130505121242560999</TransactionID>
</Response>
```

Transaction Reporting API

The reporting API provides a powerful yet extremely simple interface to query for individual data points, create simple views, or complex reporting. Every query is based off the same simple API commands. This section discussed reporting of payment-related transactions.

There are three default views – the order view, the payment view, and the transaction view. Each view has its own advantages:

> **Orders** are useful to view everything that would be on a customer receipt, including the individual items, taxes, totals, and payments.
>
> **Payments** are useful to view the current state of all sales. This will be the most useful view for reporting to end-users, as it provides a very simple and intuitive way to group all the transactions related to a particular purchase.
>
> **Transactions** are useful to provide a granular, detailed view of all activity of a merchant. This view is useful for billing purposes and/or reconciliation of a merchant's activity.

The reporting API provides a powerful yet extremely simple interface. You can report on all orders, payments, and transactions independently. Or, you can provide a hierarchical representation, depending on your need.

Understanding Orders, Payments, and Transactions

Orders represent a grouping of one or more items purchased from a merchant. Orders can have zero or more payments.

Payments are groupings of related transactions that represent a money transfer transaction (i.e. a Sale). A new payment is created when a SALE transaction is run, and maintains the aggregate state of all transactions related to that sale. For example, if a merchant subsequently refunds the sale, the payment is updated to reflect that the sale has been refunded. Payments must have one or more transactions.

Transactions represent a single, atomic operation, like an individual SALE, VOID, or REFUND transaction. Transactions are immutable and maintain no state; once a transaction is complete, it can no longer be updated or altered. Consider it a record of an action, like an entry in a log book.

The following example illustrates the relationship between orders, payments, and transactions:

1. A customer enters a merchant's store and buys a sweater for $100.

   (At this step, an Order of $100 is created)

2. The customer has a $25 gift card to this merchant, and wishes to use it. A payment is made for $25 using the gift card, leaving $75 outstanding on the order.

(Sale transaction of $25 created. A new Payment of $25 is also created)

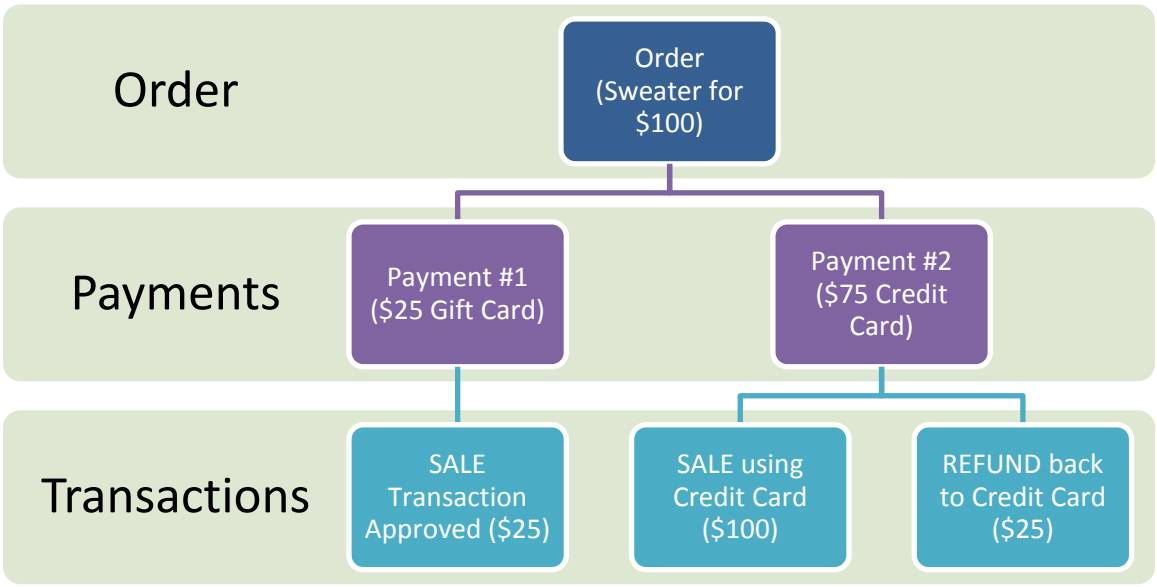
3. The customer uses their credit card for the remaining $75 balance. Unfortunately, the clerk makes an error and accidentally charges the credit card for the full order amount of $100.

   (Sale transaction of $100 created. A new payment of $100 is also created)

4. The clerk corrects the overpayment by refunding the customer's credit card $25.

   (Refund transaction of $25 created. The $100 payment is updated to reflect the $25 refund, making the payment balance now $75).

The representation of this transaction chain in the reporting system would be:



Each line represents a usable view that can be searched and reported on, depending on your needs.

Note: Orders are optional and may not be present for some merchant types or implementations. While transactions will always have a corresponding payment view, a payment can exist without an order.

Request Format

All reporting queries follow the same consistent format. The *Action* field in the request determines the type of records that are queried.

SAMPLE REQUEST	FIELD DESCRIPTION
<Request>	
<TransactionType>ADMIN</TransactionType>	Required. The type of transaction
<Username>user@domain.com</Username>	Required. The operator of the transaction
<Password>pass</Password>	Required. Used to authenticate the operator
<Action>INQPMT</Action>	Required. The action to be performed
<SearchTerms>JOE CARDHOLDER</SearchTerms>	Optional. The terms to search for. SearchTerms will match any records containing the phrase provided (i.e. JOE CARDHOLDERS would be matched as it contans the term, but JOEY CARDHOLDER would not). Multiple terms can be provided by using the pipe ('|') character. When using multiple terms, all terms provided must be matched, but they need not be in the same order or even the same fields. For example, JOE|CARDHOLDER would match JOEY CARDHOLDER, as both terms are contained. It would also match a record where the terms were spread out in different fields. (i.e. FirstName='JOE' and LastName = 'CARDHOLDER' would be matched, as both terms were found, even if in separate fields of the record).
<SearchFilters>	Optional. Allows you to filter the results of the search based on specific field values. Any number of filters may be provided. The Pipe ('|') character may be used to provide OR functionality within an individual filter.
<Filter Field="PaymentStatus">COMPLETED|REFUNDED</Filter>	*Example: Will only return records whose PaymentStatus field is either COMPLETED or REFUNDED.*
<Filter Field="OriginalAmount">20.00</Filter>	*Example: Will only return records whose OriginalAmount field is exactly 20.00*
</SearchFilters>	
<SearchDateFrom>2013-01-01 12:00AM</SearchDateFrom>	Optional. Only returns results created after the specified date and time.
<SearchDateTo>2013-05-05 11:59PM</SearchDateTo>	Optional. Only returns results created before the specified date and time.
<Page>1</Page>	Optional. The page of results to return. Multiplied by ResultPerPage to determine which records are returned. *For example, Page=1 and ResultsPerPage = 25 will return records 1-25. Page=2 and ResultsPerPage=25 will return records* 26-50.
<ResultsPerPage>25</ResultsPerPage>	Optional. The maximum number of records to return. If none is specified, default values are provided.
</Request>	

The request above will search for all payments containing JOE CARDHOLDER in any field. It will then filter the results, returning the records whose PaymentStatus is either COMPLETED *or* REFUNDED, *and* whose OriginalAmount is 20.00. It will then further refine the results to payments only within the date range provided in the SearchDateFrom and SearchDateTo fields. Finally, it will return the first page of results, up to 25 records.

A successful query will generate the following response (edited for brevity):

```
<Response>
```

```
  <Approved>true</Approved>
  <ResponseText>Records 1-25 of 100</ResponseText>
  <Count>100</Count>
  <Payments>
    <Payment>
      <!-- details of the first payment will be here. -->
    </Payment>
    <Payment>
      <!-- details of the second payment will be here. -->
    </Payment>
    <!-- etc.. up to 25 records will be returned, based on ResultsPerPage -->
  </Payments>
</Response>
```

Reporting on Transactions

Transactions represent a single, atomic operation (i.e. exactly one request/response pair). Transactions are immutable and maintain no state; once a transaction is complete, it can no longer be updated or altered. Consider it a record of an action, like an entry in a log book.

There are two ways to report on transactions, *INQTRX*, which provides the full detail of one or more transactions, and *INQTRXSUM*, which provides a summary of the transaction, which is useful for large queries or to display in list format.

The **INQTRX** action is used to provide details of an individual transaction or groups of transactions.

To search for a single transaction, provide the transaction's ID in the RefTransactionID field.

```
<!– Searching for an individual transaction by ID.
Returns exactly one transaction -->

<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQTRX</Action>
  <RefTransactionID>1234567654321</RefTransactionID>
</Request>

<Response>
  <Approved>true</Approved>
  <Transaction>
    <!-- Details of the transaction -->
  </Transaction>
</Response>
```

To search for one or more transactions, you can use the standard search format. In this example, we are searching for all transactions whose value is $10.00:

```
<!--Searching for transactions using the search API.
Returns zero or more matching transactions. -->
```

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQTRX</Action>
  <SearchTerms>10.00</SearchTerms>
</Request>

<Response>
  <Approved>true</Approved>
  <ResponseText>Records 1-10 of 20</ResponseText>
  <Count>20</Count>
  <Transactions>
    <Transaction>
      <!-- Details of transaction 1 will be here -->
    </Transaction>
    <Transaction>
      <!-- Details of transaction 2 will be here -->
    </Transaction>
    <!-- etc.. -->
  </Transactions>
</Response>
```

The **INQTRXSUM** action returns a summary of an individual transaction or groups of transactions, returning only the most relevant fields. It is most useful to provide a line item summary of transactions for display in a list.

The request format is identical to the INQTRX command. The only difference is that the response will contain TransactionSummary objects instead of Transactions.

```
<!-- Searching for an individual transaction by ID.
Returns exactly one transaction -->
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQTRXSUM</Action>
  <RefTransactionID>1234567654321</RefTransactionID>
</Request>

<Response>
  <Approved>true</Approved>
  <TransactionSummary>
    <!-- A summary of the transaction -->
  </TransactionSummary>
</Response>

<!-- Searching for transactions using the search API.
Returns zero or more matching transactions. -->
<Request>
  <TransactionType>ADMIN</TransactionType>
```

```
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQTRXSUM</Action>
  <SearchTerms>10.00</SearchTerms>
</Request>

<Response>
  <Approved>true</Approved>
  <ResponseText>Records 1-20 of 20</ResponseText>
  <Count>20</Count>
  <TransactionSummaries>
    <TransactionSummary>
      <!-- Details of transaction 1 will be here -->
    </TransactionSummary>
    <TransactionSummary>
      <!-- Details of transaction 2 will be here -->
    </TransactionSummary>
    <!-- etc.. -->
  </TransactionSummaries>
</Response>
```

Reporting on Payments

Payments are groupings of related transactions that represent a money transfer transaction (i.e. a Sale). A new payment is created when a SALE transaction is run, and maintains the aggregate state of all transactions related to that sale. For example, if a merchant subsequently refunds the sale, the payment is updated to reflect that the sale has been refunded. Payments must have one or more transactions.

There are two ways to report on payments, *INQPMT*, which provides the full detail of one or more payments, and *INQPMTSUM*, which provides a summary of the payment, which is useful for large queries or to display in list format.

The **INQPMT** action is used to provide details of an individual payment or group of payments

To search for a single payment, provide the payment's ID in the PaymentID field.

```
<!-- Searching for an individual payment by ID.
Returns exactly one payment -->
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQPMT</Action>
  <PaymentID>1234567654321</PaymentID>
</Request>

<Response>
  <Approved>true</Approved>
  <Payment>
    <!-- Details of the Payment -->
  </Payment>
```

```
</Response>
```

To search for one or more payments, you can use the standard search format. In this example, we are searching for all payments whose value is $10.00:

```
<!-- Searching for payments using the search API.
Returns zero or more matching payments. -->
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQPMT</Action>
  <SearchTerms>10.00</SearchTerms>
</Request>

<Response>
  <Approved>true</Approved>
  <ResponseText>Records 1-20 of 20</ResponseText>
  <Count>20</Count>
  <Payments>
    <Payment>
      <!-- Details of Payment 1 will be here -->
    </Payment>
    <Payment>
      <!-- Details of Payment 2 will be here -->
    </Payment>
    <!-- etc.. -->
  </Payments>
</Response>
```

INQPMTSUM returns a summary of an individual payment or groups of payments, returning only the most relevant fields. It is most useful to provide a line item summary of payments for display in a list.

The request format is identical to the INQPMT command. The only difference is that the response will contain PaymentSummary objects instead of Payments.

```
<!-- Searching for an individual payment by ID.  Returns exactly one payment -
->
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQPMTSUM</Action>
  <PaymentID>1234567654321</PaymentID>
</Request>

<Response>
  <Approved>true</Approved>
```

```
  <PaymentSummary>
    <!-- A summary of the payment -->
  </PaymentSummary>
</Response>

<!--
 Searching for transactions using the search API.  Returns zero or more matchi
ng transactions. -->
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQPMTSUM</Action>
  <SearchTerms>10.00</SearchTerms>
</Request>

<Response>
  <Approved>true</Approved>
  <ResponseText>Records 1-20 of 20</ResponseText>
  <Count>20</Count>
  <PaymentSummaries>
    <PaymentSummary>
      <!-- Details of Payment 1 will be here -->
    </PaymentSummary>
    <PaymentSummary>
      <!-- Details of Payment 2 will be here -->
    </PaymentSummary>
    <!-- etc.. -->
  </PaymentSummaries>
</Response>
```

Note: To get a list of all transactions associated with the payment(s), add the DETAIL flag to the request. This applies to both the INQPMT and INQPMTSUM requests. For example:

```
  <Request>
    <TransactionType>ADMIN</TransactionType>
    <Username>user@domain.com</Username>
    <Password>pass</Password>
    <Action>INQPMT</Action>
    <PaymentID>1234567654321</PaymentID>
    <Flags>DETAIL</Flags>
  </Request>

  <Response>
    <Approved>true</Approved>
    <Payment>
        <!-- Details of the Payment will be here -->
      <Transactions>
        <Transaction>
           <!-- Details of the transaction #1 will be here -->
        </Transaction>
        <Transaction>
           <!-- Details of the transaction #2 will be here -->
```

```
      </Transaction>
    </Transactions>
  </Payment>
</Response>
```

Important Notes for the Reporting API:

- For maximum performance, use the SearchFilters instead of SearchTerms if you know explicitly what field will contain the data you are requesting.

- When using SearchFilters, you can use the **Mode** attribute of the Filter field to determine whether to do an exact search, or a keyword/substring search. If Mode is omitted, it defaults to EXACT.

  For example, the following request will search for all transactions where the CardName field is *exactly* JOHN DOE and the CardNum field *contains* the string 4321:

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Action>INQTRXSUM</Action>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <SearchFilters>
    <Filter Field="CardName" Mode="EXACT">JOHN DOE</Filter>
    <Filter Field="CardNum" Mode="KEYWORD">4321</Filter>
  </SearchFilters>
</Request>
```

- By default, reporting queries do not return large binary information, such as the cardholder's signature capture. To obtain the signature capture, add the **DETAIL** flag to the request:

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>INQTRXSUM</Action>
  <SearchTerms>10.00</SearchTerms>
  <Flags>DETAIL</Flags>
</Request>
```

  It is strongly recommended that you only use the DETAIL flag for individual record queries.

- For security reasons, password and binary fields are **not searched**.

- For maximum accuracy, **omit formatting** from currency values when searching for transaction amounts (e.g. use 10.00 instead of $10.00, or 8.75 instead of 8.75%).

- **Query results are filtered based on the access level of the user running the query**. For example, an individual user may only be able to see his or her own transactions, whereas a merchant administrator can see all transactions from all users. A portfolio administrator with sufficient privileges will see all transactions within its portfolio.

- All query results are paged. If you omit the Page and/or ResultsPerPage fields, default values are used

- All SearchTerms must be present in the record, though any terms separated by a Pipe character need not be in the same field within the record.

  It is possible to do an either/or match using the following syntax: **[***val1***^***val2***]**.

  `<SearchTerms>`[Mike^Michael]`</SearchTerms>` ← will search for (Mike OR Michael).

  Either/or matches may be combined with other terms using the Pipe character.

  `<SearchTerms>`[Mike^Michael]|Smith`</SearchTerms>` ← will search for (Mike OR Michael) AND Smith.

Merchant Administration API

Merchants can be created, updated, activated, and deactivated through the merchant administration API.

Note: This is only a small subset of the administration functionality available in the API. If you have specific needs, please contact your integration specialist.

Adding a Merchant

Merchants are added using the ADDMCH command. Please see the schema for more details on which fields can be stored.

One or more users can be created simultaneously with the merchant. At least one user must be provided which must be set to MERCHANT_ADMIN.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>ADDMCH</Action>
  <MerchantProfile>
    <MerchantID>123456</MerchantID>
    <BusinessName>Widgets Inc</BusinessName>
    <DBA>Widgets Inc</DBA>
    <Location>
      <Street>123 Main Street</Street>
      <City>Alpharetta</City>
      <State>GA</State>
      <PostalCode>30004</PostalCode>
      <Country>USA</Country>
      <Phone>8005551234</Phone>
    </Location>
    <Processor Type="TSYS">
      <Setting Name="MerchantKey" Value="1234" />
      <Setting Name="TerminalID" Value="54321" />
      <Setting Name="MID" Value="TEST" />
    </Processor>
    <CashierProfiles>
      <CashierProfile>
        <Username>widgetsadmin@widgetsinc.com</Username>
        <FirstName>Joe</FirstName>
        <LastName>Tester</LastName>
        <Role>MERCHANT_ADMIN</Role>
        <Address>
          <Street>123 Main Street</Street>
          <City>Alpharetta</City>
          <State>GA</State>
          <PostalCode>30004</PostalCode>
          <Country>USA</Country>
          <Phone>8005551234</Phone>
        </Address>
      </CashierProfile>
      <CashierProfile>
```

```
        <Username>cashier@widgetsinc.com</Username>
        <FirstName>Sam</FirstName>
        <LastName>Smith</LastName>
        <Role>CASHIER</Role>
        <Address>
          <Street>123 Main Street</Street>
          <City>Alpharetta</City>
          <State>GA</State>
          <PostalCode>30004</PostalCode>
          <Country>USA</Country>
          <Phone>8005551234</Phone>
        </Address>
      </CashierProfile>
    </CashierProfiles>
    <PaymentMethods>VISA MC AMEX DISC</PaymentMethods>
    <BrandOwner>YourPortfolio</BrandOwner>
    <HelpEmail>help@wigetsinc.com</HelpEmail>
    <HelpPhone>8881235555</HelpPhone>
    <SwiperType>ACRambler</SwiperType>
    <TimeZone>Eastern Standard Time</TimeZone>
  </MerchantProfile>
</Request>
```

If the request is successful, the unique ID of the merchant will be provided in the UID field of the response.

```
<Response>
  <Approved>true</Approved>
  <UID>B4022549-5A21-4114-BECE-4F7FFC032EF7</UID>
</Response>
```

Updating a Merchant

Merchant records can be updated using the UPDMCH command. The command will only update provided fields, leaving all unspecified fields intact. To clear a field, enter an empty tag.

A user must have a minimum of MERCHANT_ADMIN privileges to update their own merchant profile.

The following request clears the HelpPhone field and updates the HelpEmail and PaymentMethods field of the merchant associated with the username provided in the request. For example, if user@domain.com is a MERCHANT_ADMIN of Widgets, Inc, then Widgets Inc will be updated.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>UPDMCH</Action>
  <MerchantProfile ID="B4022549-5A21-4114-BECE-4F7FFC032EF7">
    <PaymentMethods>VISA MC AMEX DISC</PaymentMethods>
    <HelpEmail>support@wigetsinc.com</HelpEmail>
    <HelpPhone />
  </MerchantProfile>
```

```
</Request>
```

```
<Response>
  <Approved>true</Approved>
</Response>
```

To update a different merchant profile, specify the ID of the profile in the ID attribute of the MerchantProfile tag. Note that only users with PORTFOLIO_ADMIN privileges or greater can update another merchant's profile.

The following request updates the HelpPhone of the merchant specified in the ID tag:

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>UPDMCH</Action>
  <MerchantProfile ID="B4022549-5A21-4114-BECE-4F7FFC032EF7">
    <HelpPhone>8881235555</HelpPhone>
  </MerchantProfile>
</Request>
```

```
<Response>
  <Approved>true</Approved>
</Response>
```

Activating a Merchant

Merchants are activated by using the ACTIVATE command. Note that merchants are activated automatically when they are created. A user must have PORTFOLIO_ADMIN privileges to activate/deactivate merchants.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>ACTIVATE</Action>
  <MerchantProfile ID="B4022549-5A21-4114-BECE-4F7FFC032EF7" />
</Request>
```

```
<Response>
  <Approved>true</Approved>
</Response>
```

Deactivating a Merchant

Merchants are deactivated by calling the DEACTIVATE command . A user must have PORTFOLIO_ADMIN privileges to activate/deactivate merchants.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>DEACTIVATE</Action>
  <MerchantProfile ID="B4022549-5A21-4114-BECE-4F7FFC032EF7" />
</Request>

<Response>
  <Approved>true</Approved>
</Response>
```

Adding Users to a merchant

Users can be added to a merchant profile by specifying the ADDUSR command. A user must have a minimum of MERCHANT_ADMIN privileges to add users to their own merchant account, or a minimum of PORTFOLIO_ADMIN privileges to add users to another merchant account.

The following sample request adds two users to the merchant profile associated with user@domain.com:

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>ADDUSR</Action>
  <CashierProfiles>
    <CashierProfile>
      <Username>widgetsadmin@widgetsinc.com</Username>
      <FirstName>Joe</FirstName>
      <LastName>Tester</LastName>
      <Role>MERCHANT_ADMIN</Role>
      <Address>
        <Street>123 Main Street</Street>
        <City>Alpharetta</City>
        <State>GA</State>
        <PostalCode>30004</PostalCode>
        <Country>USA</Country>
        <Phone>8005551234</Phone>
      </Address>
    </CashierProfile>
    <CashierProfile>
      <Username>cashier@widgetsinc.com</Username>
      <FirstName>Sam</FirstName>
      <LastName>Smith</LastName>
      <Role>CASHIER</Role>
      <Address>
        <Street>123 Main Street</Street>
        <City>Alpharetta</City>
        <State>GA</State>
        <PostalCode>30004</PostalCode>
        <Country>USA</Country>
        <Phone>8005551234</Phone>
```

```
      </Address>
    </CashierProfile>
  </CashierProfiles>
</Request>


<Response>
  <Approved>true</Approved>
</Response>
```

To add users to another merchant's account, specify the merchant profile's ID in the tag. The user running the request must have PORTFOLIO_ADMIN privileges:

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>UPDMCH</Action>
  <MerchantProfile ID="B4022549-5A21-4114-BECE-4F7FFC032EF7">
   <CashierProfiles>
    <CashierProfile>
      <Username>widgetsadmin@widgetsinc.com</Username>
      <FirstName>Joe</FirstName>
      <LastName>Tester</LastName>
      <Role>MERCHANT_ADMIN</Role>
      <Address>
        <Street>123 Main Street</Street>
        <City>Alpharetta</City>
        <State>GA</State>
        <PostalCode>30004</PostalCode>
        <Country>USA</Country>
        <Phone>8005551234</Phone>
      </Address>
    </CashierProfile>
    <CashierProfile>
      <Username>cashier@widgetsinc.com</Username>
      <FirstName>Sam</FirstName>
      <LastName>Smith</LastName>
      <Role>CASHIER</Role>
      <Address>
        <Street>123 Main Street</Street>
        <City>Alpharetta</City>
        <State>GA</State>
        <PostalCode>30004</PostalCode>
        <Country>USA</Country>
        <Phone>8005551234</Phone>
      </Address>
    </CashierProfile>
   </CashierProfiles>
  </MerchantProfile>
</Request>
```

```
<Response>
  <Approved>true</Approved>
</Response>
```

Updating Users

User profiles can be updated by using the UPDUSR command. Only the specified fields are updated. To clear a field, enter an empty tag.

The following example updates a user's last name and username and clears the users HomePhone field, leaving all other fields intact.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@pass.com</Username>
  <Password>pass</Password>
  <Action>UPDUSR</Action>
  <CashierProfile ID ="89B45F92-3EB8-4B2A-9B5D-27B2502FDE4D">
    <LastName>JOHNSON</LastName>
    <Username>shelly.johnson@domain.com</Username>
    <HomePhone />
  </CashierProfile>
</Request>

<Response>
  <Approved>true</Approved>
</Response>
```

Activating and Deactivating Users

Users can be activated or deactivated by using the ACTIVATE command. Similarly, the same syntax is used for the DEACTIVATE command.. For example, the following request activates the user. *(Note: Deactivation does not delete the record – it simply disables the user's login).*

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>pass</Password>
  <Action>DEACTIVATE</Action>
  <CashierProfile ID ="89B45F92-3EB8-4B2A-9B5D-27B2502FDE4D" />
</Request>

<Response>
  <Approved>true</Approved>
</Response>
```

Merchant Reporting API

The merchant reporting API follows the same consistent syntax as the Transaction reporting API. All commands and fields behave the same. Only the dataset changes.

API Definition

SAMPLE REQUEST	FIELD DESCRIPTION
<Request>	
<TransactionType>ADMIN</TransactionType>	Required. The type of transaction
<Username>user@domain.com</Username>	Required. The operator of the transaction
<Password>pass</Password>	Required. Used to authenticate the operator
<Action>INQMCH</Action>	Required. The action to be performed
<SearchTerms>JOE MERCHANT</SearchTerms>	Optional. The terms to search for. SearchTerms will match any records containing the phrase provided (i.e. JOE MERCHANTS would be matched as it contans the term, but JOEY MERCHANTS would not). Multiple terms can be provided by using the pipe ('|') character. When using multiple terms, all terms provided must be matched, but they need not be in the same order or even the same fields. For example, JOE|MERCHANT would match JOEY MERCHANT, as both terms are contained. It would also match a record where the terms were spread out in different fields. (i.e. FirstName='JOE' and LastName = 'MERCHANT' would be matched, as both terms were found, even if in separate fields of the record).

<SearchFilters>	Optional. Allows you to filter the results of the search based on specific field values. Any number of filters may be provided. The Pipe ('|') character may be used to provide OR functionality within an individual filter.
<Filter Field="BrandOwner">MyCompany|MySubCompany</Filter>	*Example: Will only return records whose BrandOwner field is either MyCompany or MySubCompany.*
<Filter Field="Inactive">false</Filter>	*Example: Will only return records whose Inactive field is false*
</SearchFilters>	
<SearchDateFrom>2013-01-01 12:00AM</SearchDateFrom>	Optional. Only returns results created after the specified date and time.
<SearchDateTo>2013-05-05 11:59PM</SearchDateTo>	Optional. Only returns results created before the specified date and time.
<Page>1</Page>	Optional. The page of results to return. Multiplied by ResultPerPage to determine which records are returned. *For example, Page=1 and ResultsPerPage = 25 will return records 1-25. Page=2 and ResultsPerPage=25 will return records* 26-50.
<ResultsPerPage>25</ResultsPerPage>	Optional. The maximum number of records to return. If none is specified, default values are provided.
</Request>	

Note: The SearchTerms field will search on all fields in the merchant AND its users. This allows one to easily find a merchant profile given a particular user's details, such as their username.

Searching for a Merchant:

To search for an individual merchant, provide its identifier in the MerchantProfile's MerchantID field.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>Pass</Password>
  <Action>INQMCH</Action>
  <MerchantProfile>
    <MerchantID>491648C2-A3D2-4380-B0B7-136BFF8E4926</MerchantID>
  </MerchantProfile>
</Request>
```

```
<Response>
  <Approved>true</Approved>
  <MerchantProfile ID="491648c2-a3d2-4380-b0b7-136bff8e4926">
    <MerchantID>491648C2-A3D2-4380-B0B7-136BFF8E4926</MerchantID>
    <BusinessName>NetSecure Technologies Ltd</BusinessName>
    <DBA>NetSecure Technologies Ltd</DBA>
    <!-- etc... -->
  </MerchantProfile>
</Response>>
```

To search for multiple merchants, use the standard Search API:

```
<Request>
    <TransactionType>ADMIN</TransactionType>
    <Username>user@domain.com</Username>
    <Password>pass</Password>
    <Action>INQMCH</Action>
    <SearchTerms>NetSecure</SearchTerms>
    <Page>1</Page>
    <ResultsPerPage>25</ResultsPerPage>
</Request>
```

```
<Response>
  <Approved>true</Approved>
  <Count>2</Count>
  <ResponseText>Results 1-2 of 2</ResponseText>
  <MerchantProfiles>
    <MerchantProfile ID="491648c2-a3d2-4380-b0b7-136bff8e4926">
      <MerchantID>491648C2-A3D2-4380-B0B7-136BFF8E4926</MerchantID>
      <BusinessName>NetSecure Technologies Ltd</BusinessName>
      <DBA>NetSecure Technologies Ltd</DBA>
      <!-- etc... -->
    </MerchantProfile>
    <MerchantProfile ID="CC1643c2-2131-AAC0-BC7A-2899983CCFB6">
      <MerchantID>1234321</MerchantID>
      <BusinessName>NetSecure Payments Inc</BusinessName>
```

```
      <DBA>NetSecure Payments Inc</DBA>
      <!-- etc... -->
    </MerchantProfile>
  </MerchantProfiles>
</Response>
```

Finding a Merchant by User:

The SearchTerms field will search on all fields in the merchant AND its users. This allows one to easily find a merchant profile given a particular user's details, such as their username:

```
<Request>
    <TransactionType>ADMIN</TransactionType>
    <Username>user@domain.com</Username>
    <Password>pass</Password>
    <Action>INQMCH</Action>
    <SearchTerms>user@netsecurepayments.com</SearchTerms>
    <Page>1</Page>
    <ResultsPerPage>25</ResultsPerPage>
</Request>

<Response>
  <Approved>true</Approved>
  <Count>1</Count>
  <ResponseText>Results 1-1 of 1</ResponseText>
  <MerchantProfiles>
    <MerchantProfile ID="CC1643c2-2131-AAC0-BC7A-2899983CCFB6">
      <MerchantID>1234321</MerchantID>
      <BusinessName>NetSecure Payments Inc</BusinessName>
      <DBA>NetSecure Payments Inc</DBA>
      <!-- etc... -->
    </MerchantProfile>
  </MerchantProfiles>
</Response>
```

Checking for differences against a merchant record:

The DIFF command can be used to query for differences between a provided record and the record in the system. This is useful for ensuring synchronization between systems.

The DIFF command checks both merchant and user records, if provided.

A ResposneCode of 0 means that the records are identical. ResponseCode of 1 means that differences exist in the record. If the VERBOSE flag is sent with the request, the ResponseText field will contain a list of all differences.

In the following example, the database record for the provided ID has both a DBA and BusinessName of "NetSecure Technologies Ltd". Since the BusinessName records match, they are not returned in the list

of differences. However, the DIFF command has the DBA set to NetSecure Technology Ltd, which does not match the database record, so it is returned in the list of differences.

```
<Request>
  <TransactionType>ADMIN</TransactionType>
  <Username>user@domain.com</Username>
  <Password>Pass</Password>
  <Action>DIFF</Action>
  <MerchantProfile>
    <MerchantID>491648C2-A3D2-4380-B0B7-136BFF8E4926</MerchantID>
    <BusinessName>NetSecure Technologies Ltd</BusinessName>
    <DBA>NetSecure Technology Ltd</DBA>
  </MerchantProfile>
  <Flags>VERBOSE</Flags>
</Request>
```

```
<?xml version="1.0" encoding="utf-8"?><Response>
  <ResponseCode>1</ResponseCode>
  <Approved>true</Approved>
  <ResponseText>
    DBA mismatch (NetSecure Technologies Ltd [vs] NetSecure Technology Ltd),
  </ResponseText>
</Response>
```