

# Direct Mode 3.1 Protocol Documentation

## 1.0 Introduction to the Direct Mode 3.1 Protocol

Fast Transact, a leader in high-performance digital payment processing solutions, provides multiple interfaces to its transaction processing system. These interfaces include:

- Interactive Payment Forms
- Virtual Terminal
- Emulation Modes for backward compatibility with competitor's interfaces
- Interactive Batch Upload
- Direct Mode
- [Direct Mode Batch](#)

This documentation describes version 3.1 of the Direct Mode interface.

### 1.1 Direct Mode Overview

Direct Mode is designed for Merchants who need to have complete control over their transaction processing. Using Direct Mode, the card holder's browser is never directed to our site. In fact, Direct Mode can be used for Automatic Billing, Telephone and Retail orders as well as Internet orders.

Transaction data is sent to our system from the back-end of the Merchant's server by software written or installed by the merchant. This gives the Merchant the ability to completely customize and integrate real time payment processing into his own application. To implement a Direct Mode client, familiarity with sockets and network programming is required.

### 1.2 Direct Mode Clients

If you are using Java, you can take advantage of our [Java Client](#) for Direct Mode 3.1, which handles all the protocol encapsulation, encoding and decoding for you.

### 1.3 Principles of Direct Mode Operation

To make client implementation as simple as possible, the Direct Mode 3.1 protocol is based on the standard HTTP protocol and SSL (HTTPS) encryption.

Most modern languages have built in libraries that can be used to implement HTTP clients with very little effort.

A transaction is initiated by sending an HTTP POST to our host. The host will respond momentarily with the transaction result on the same socket. All data is [URL encoded](#), just like a standard <form> POST from a browser.

## 1.4 Host addresses and ports

The following host addresses and ports are used for Direct Mode 3.1:

Host	Port	Path	Description
secure.fasttransact.com	1401 (http+) 1402 (https)	/gw/sas/getid3.1	ID generator
		/gw/sas/direct3.1	Transaction interface
		/gw/sas/settle3.1	Batch settlement interface
	1403	/	Verisign™ 3 compatibility (not documented here)

Table 1.4.1

+ = *Unencrypted HTTP should NOT be used except when testing.*

## 1.5 A quick example

In this example, we wish to send the following parameters in a request to the system:

Parameter	Value
pay_type	C
tran_type	A
account_id	110006559149
card_number	4444333322221186
card_expire	0909
amount	5.00

Table 1.5.1

We can expect to receive the following parameters back in the response:

Parameter	Value
status_code	?
trans_id	?
auth_code	?
auth_date	?
auth_msg	?
avs_code	?

cvv2_code	?
ticket_code	?

Table 1.5.2

See the [Protocol Response Reference](#) section for more information. First, the URL encoded HTTP POST request is assembled and sent to secure.fasttransact.com port 1401:

```
POST /gw/sas/direct3.1 HTTP/1.0
Host: secure.fasttransact.com:1401
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
```

```
pay_type=C&tran_type=A&account_id=110006559149&card_number=4444333322221
186&card_expire=0909&amount=5.00
```

The system will respond with:

```
HTTP/1.0 200 OK Approved
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 160
```

```
auth_msg=TEST+APPROVED&ticket_code=XXXXXXXXXXXXXXXX&avs_code=X&aut
h_date=2004-06-
09+22%3A55%3A08&status_code=T&trans_id=109704163690&auth_code=999999&cvv2_code=
M
```

URL encoding is demonstrated in the example above. Parameter-value pairs are separated by ampersands (&), and within each pair, the parameter name and the value are separated by equal signs (=). All non-alphanumeric characters within each value field are substituted with percent signs followed by 2 hexadecimal digits, (%XX) where XX is the hexadecimal value of the encoded ASCII character. The order of the parameter-value pairs doesn't matter.

That's all it takes to execute a basic transaction. In Unix/Linux environments, netcat(1) or nc(1) can be used to recreate this example.

## 2.0 Protocol Request Reference

In this section, all parameters recognized by the system are explained, as well as when they are required.

### 2.1 Required Request Parameters

To use the [Transaction Type Table](#), [Parameter Group Table](#) and [Parameter Definition Table](#) tables, follow these steps:

1. Select the transaction type (**tran\_type**) you need to program.
2. Look at the **Required** parameter box for that transaction type, it will enumerate the minimum required parameters.
3. Whenever a <bracketed> parameter group name is specified, go to the [Parameter Group Table](#).
4. Locate the corresponding parameter group row(s) in the [Parameter Group Table](#).
5. For rows with no **Special condition** the **Required** and **Typical** parameters will always apply.
6. For rows with special conditions, determine whether those rows apply to your transaction based on the **Special condition** column.
7. Each parameter is explained in more detail in the [Parameter Definition Table](#).

For example, the minimum required parameters for a Credit Card Sale are: account\_id, tran\_type, pay\_type, amount, card\_number, card\_expire, card\_cvv2, bill\_name1, bill\_name2, bill\_street, bill\_zip and bill\_country.

<b>Transaction Type Table</b>			
tran_type =	Required	Optional, commonly used	Special
"A" (auth <sup>1</sup> )	<a href="#">account id</a> <a href="#">tran type</a> <payment information> <customer information>	<a href="#">trans id</a> <purchase information> <shipping information> <level 2 information>	<a href="#">dynip sec code</a> <membership> <sup>3</sup> <recurring billing> <sup>3</sup>
"S" (sale <sup>1</sup> )	<a href="#">account id</a> <a href="#">tran type</a> <payment information> <customer information>	<a href="#">trans id</a> <purchase information> <shipping information> <level 2 information>	
"R" (refund <sup>1</sup> )	<a href="#">account id</a> <a href="#">tran type</a> <a href="#">orig id</a>	<a href="#">trans id</a> <a href="#">amount</a> <purchase information>	
"C" (credit <sup>1</sup> )	<a href="#">account id</a> <a href="#">tran type</a> <payment information> <customer information>	<a href="#">trans id</a> <purchase information>	

"D" (capture <sup>1</sup> )	<a href="#">account_id</a> <a href="#">tran_type</a> <a href="#">orig_id</a>	<a href="#">amount</a>	
"B" (batch settle <sup>2</sup> )	<a href="#">account_id</a> <a href="#">tran_type</a> <a href="#">pay_type</a>		

Transaction Type Table - Table 2.1.1

1 = Post to /gw/sas/direct3.1

2 = Post to /gw/sas/settle3.1 - notice that normally settlements are performed automatically, and this operation is not necessary.

3 = Only supported for tran\_type "A" or "S"

Parameter Group Table			
Parameter group	Special condition	Required parameters	Typical
<payment information>		<a href="#">pay_type</a> <a href="#">amount</a>	<a href="#">cisp_storage</a> <a href="#">disable_fraud_checks</a> <a href="#">disable_negative_db</a> <a href="#">disable_email_receipts</a>
	pay_type=C (credit card)	<a href="#">card_number</a> <a href="#">card_expire</a> <a href="#">card_cvv2</a> or <a href="#">card_track1</a> or <a href="#">card_track2</a>	<a href="#">force_code</a> <a href="#">disable_avs</a> <a href="#">disable_cvv2</a> <a href="#">3ds_cavv</a> <a href="#">3ds_xid</a>
	pay_type=K (check / ACH)	<a href="#">account_number</a>	<a href="#">bill_photo_id_no</a> <a href="#">bill_photo_id_state</a> <a href="#">bill_tax_id_no</a> <a href="#">bill_birth_date</a>
	pay_type=S (stored value)	<a href="#">card_number</a> <a href="#">card_pin</a>	
	CISP repeat billing	<a href="#">account_number</a>	
<purchase information>			<a href="#">site_tag</a> <a href="#">description</a> <a href="#">misc_info</a> <a href="#">user_data</a>
<customer information>		<a href="#">bill_name1</a> <a href="#">bill_name2</a> <a href="#">bill_street</a> <a href="#">bill_zip</a>	<a href="#">bill_city</a> <a href="#">bill_state</a> <a href="#">cust_email</a> <a href="#">cust_phone</a>

		<a href="#"><u>bill_country</u></a>	<a href="#"><u>cust_ip</u></a> <a href="#"><u>cust_host</u></a> <a href="#"><u>cust_browser</u></a>
<shipping information>			<a href="#"><u>ship_name1</u></a> <a href="#"><u>ship_name2</u></a> <a href="#"><u>ship_street</u></a> <a href="#"><u>ship_city</u></a> <a href="#"><u>ship_state</u></a> <a href="#"><u>ship_zip</u></a> <a href="#"><u>ship_country</u></a> <a href="#"><u>courier_tracking</u></a>
<level 2 information>			<a href="#"><u>tax_amount</u></a> <a href="#"><u>ship_amount</u></a> <a href="#"><u>purch_order</u></a>
<membership>		<a href="#"><u>site_tag</u></a> <a href="#"><u>member_username</u></a> <a href="#"><u>member_duration</u></a>	<a href="#"><u>member_password</u></a> <a href="#"><u>member_memo</u></a>
<recurring billing>		<a href="#"><u>recurring_amount</u></a> <a href="#"><u>recurring_period</u></a>	<a href="#"><u>recurring_count</u></a> <a href="#"><u>recurring_prorate</u></a>

Parameter Group Table - Table 2.1.2

## 2.2 Request Parameter Definitions

Request Parameter Definition Table	
<b>General parameters</b>	
account_id	This is the number of your merchant or agent account, as a 12 digit string. Required for all transactions.
site_tag	The site tag of your website or retail site, as was configured within the system. This field is optional. It is used to select which email receipt templates will be used, as well as for accounting purposes.  When using the membership/subscription features, site_tag is required and determines which site the new member will belong to. It can optionally contain a comma separated list of multiple site_tags. The new member will be a primary member of the first site_tag in the list, and a secondary member of each additional site specified.
dynip_sec_code	This field is used when a direct mode client needs to connect to the server from a dynamic or unknown IP address. Normally direct mode clients are authenticated

	based on a trusted IP address configured on the account's access security settings page. The dynamic IP security code can also be configured on the same page. If the correct code is sent in this field, transactions will be accepted regardless of the IP.
pay_type	This is required for all transactions. Possible values: C (credit card) K (check / ACH) S (stored value card)
tran_type	This is required for all transactions. Possible values: A - authorize funds only, no funds will be transferred S - sale, funds will be transferred R - refund a previous sale, a partial amount can be specified. C - credit money back when there is no previous sale. D - capture a previous authorization, effectively converting it into a sale. B - batch settlement, all outstanding sale, refund and credit transactions are settled with the bank.
trans_id	Transaction ID for transaction request. This field is optional. When used, the value MUST have been generated by querying getid3.1. This field allows tracking of a transaction, even if a response is never received due to network latency, etc. If no value is supplied, the system will generate a unique ID and returned in the response message.
orig_id	Transaction ID of the original transaction. I.e. trans_id of original Sale or Auth.
amount	This is the total amount of the transaction, including tax and shipping, if any. No spaces, commas or currency signs. It should include all tax and shipping amounts, even if those amounts are specified separately.
tax_amount	This is the tax amount of the transaction. Required to qualify for Purchase/Commercial Card Level-2 processing. No spaces, commas or currency signs.
ship_amount	This is the shipping cost of the transaction. Required to qualify for Purchase/Commercial Card Level-2 processing. No spaces, commas or currency signs.
purch_order	This is the Purchase Order number. Required to qualify for Purchase/Commercial Card Level-2 processing.
courier_tracking	The courier designation, "FEDEX", "UPS", "USPS", "TNT", or "DHL" followed by a colon and the tracking

	number (no spaces.) <i>For example:</i> "FEDEX:844477771111". Specifying this enables the system to provide automatic links directly to the tracking information for the carrier.
<b>Customer information parameters</b>	
bill_name1	First name of paying customer.
bill_name2	Last name of paying customer.
bill_street	Street address of paying customer. Must match the credit card billing address, or ACH bank statement address.
bill_city	City of paying customer.
bill_state	2 character US state code of paying customer, or foreign state or province. Do not specify spelled out state names for US states.
bill_zip	5 or 9 digit US zip code or foreign postal code of paying customer. Must match the credit card billing address, or ACH bank statement address. Format: "55555" or "55555-4444" for US zip codes.
bill_country	The 2 letter ISO3166 country code of paying customer. "US" for the United States, "GB" for the United Kingdom, etc. See <a href="http://www.iso.org/iso/en/prods-services/iso3166ma/index.html">http://www.iso.org/iso/en/prods-services/iso3166ma/index.html</a>
ship_name1	First name of shipping recipient.
ship_name2	Last name of shipping recipient.
ship_street	Street address of shipping recipient.
ship_city	City of shipping recipient.
ship_state	2 character US state code or foreign state or province of shipping recipient. Do not specify spelled out state names for US states.
ship_zip	5 or 9 digit US zip code, or foreign postal code of shipping recipient. Format: "55555" or "55555-4444" for US zip codes.
ship_country	The 2 letter ISO3166 country code of shipping recipient. "US" for the United States, "GB" for the United Kingdom, etc. See <a href="http://www.iso.org/iso/en/prods-services/iso3166ma/index.html">http://www.iso.org/iso/en/prods-services/iso3166ma/index.html</a>
cust_email	Email address of the customer. Optional, but necessary for the system to send email receipts.

cust_phone	Phone number of the customer.
cust_ip	IP address of the customer for web originating transactions.
cust_host	The resolved DNS host name of the customer for web originating transactions.
cust_browser	The User-Agent customer browser header for web originating transactions.
<b>Purchase information parameters</b>	
description	A description or order reference for the order. 4000 characters max.
user_data	Arbitrary user data that will be stored by the system. 4000 characters max. It can be used to store attribute value pairs, separated by newline (\n) characters. For example:  Customer-Number: 1234 Order-Number: 123
misc_info	Arbitrary miscellaneous information that will be stored by the system. 4000 characters max.
<b>Fraud control and special feature parameters</b>	
disable_avs	Set this flag to "1" to disable AVS (Address Verification System) for this credit card transaction.
disable_cvv2	Set this flag to "1" to disable CVV2 (Card Verification Value 2) for this credit card transaction.
disable_fraud_checks	Set this flag to "1" to disable all fraud protection other than AVS/CVV2. This implies disable_negative_db.
disable_negative_db	Set this flag to "1" to disable only the negative database component of the fraud protection system.
disable_email_receipts	Set this flag to "1" to disable automatic sending of both merchant and customer email receipts.
cisp_storage	Set this flag to "1" to indicate that this transactions should be stored securely, and be accessible for future repeat billing. In order to issue another transaction on this card in the future, use "CS:trans_id:last5digits" as the card number for the future transaction, where trans_id is the id returned for the original transaction, and last5digits (optional) is the last 5 digits of the card number.

<b>Credit card parameters</b>	
card_number	Credit card number used for the transaction. No spaces or dashes, digits only.
card_expire	Credit card expiration date in MMY format (no slash). Required whenever card_number is required.
card_cvv2	Credit card CVV2/CVC value, which is the 3 or 4 digit code on the back of the credit card used for fraud prevention.
card_track1	Magnetic stripe data for track 1 used in retail environments. When sending magnetic stripe data, the card_number and card_expire fields are optional. Either track1, track2 or both can be sent. Track 1 is alphanumeric, and will begin with '%' and end with '?'. If both tracks are available, we recommend sending track 1.
card_track2	Magnetic stripe data for track 2 used in retail environments. When sending magnetic stripe data, the card_number and card_expire fields are optional. Either track1, track2 or both can be sent. Track 2 is numeric, and will begin with ';' and end with '?'. 
force_code	Authorization force code (usually 6 digits), typically obtained by phone authorization. This code is used to force a through a sale that would otherwise fail with a CALL CENTER message.
3ds_cavv	Required when using Verified by Visa™ (also known as 3D Secure.) Format: 40 hexadecimal digits in ASCII, encoding a 20 byte binary value, or 28 characters of Base64 code. This value is provided by the MPI (Merchant Plug-In).
3ds_xid	Optional even when using Verified by Visa™. Format: 40 hexadecimal digits in ASCII, encoding a 20 byte binary value, or 28 characters of Base64 code. This value is provided by the MPI (Merchant Plug-In).
<b>ACH/check parameters</b>	
account_number	Checking account number in <aba>:<account_no> format, e.g. "123456789:9999999999" where 123456789 is the bank's 9 digit ABA (routing) number, and 9999999999 is the checking account number (both are printed at the bottom of all checks)
bill_photo_id_no	Paying customer's state issued drivers license or photo

	ID number. Typically only used for ACH/check payments.
bill_photo_id_state	Paying customer's 2 letter state code for the state that issued the photo id above. if the photo id is federal, this field should be "US". Typically only used for ACH/check payments.
bill_tax_id_no	Paying customer's Tax ID or Social Security number. Format: 333-22-4444 for SS#, 22-7777777 for TID#. Typically only used for ACH/check payments.
bill_birth_date	Paying customer's birth date in YYYY-MM-DD format. Typically only used for ACH/check payments.
<b>Stored value card parameters</b>	
card_pin	The PIN code for for the stored value cardholder.
<b>Hotel industry parameters</b>	
hotel_checkin_date	Actual or anticipated checkin date YYMMDD format for hotel industry transactions.
hotel_checkout_date	Actual or anticipated checkout date YYMMDD format for hotel industry transactions.
hotel_flags	<p>This field contains a string of one or more single-letter flags pertaining to the transaction:</p> <p><u>None, One or Several of the following flags can be specified</u></p> <p>G - This transaction includes Gift ship purchases  L - This transaction includes payment for Laundry services  M - This transaction includes payment for Mini-bar purchases  T - This transaction includes payment for Telephone charges  R - This transaction includes payment for Restaurant purchases  O - This transaction includes payment for other services, such as Spa or Room service.  N - This transaction was for a No-show reservation</p> <p><u>None or One of the following flags can be specified</u></p> <p>D - Visa prestigious property program participation, \$500 limit  B - Visa prestigious property program participation, \$1000 limit</p>

	<p>S - Visa prestigious property program participation, \$1500 limit</p> <p>When specifying multiple flags, simply append them together without spaces. The order is not important.</p>
hotel_room_rate	The daily room rate. No spaces, commas or currency signs.
mcc_override	<p>4 digit MCC (Merchant Category Code) override. Can be used to indicate that a specific transaction was not a hotel-industry transaction, even though the account is setup for hotel processing by default.</p> <p>This field is used to indicate the appropriate industry code for a separate restaurant, gift shop or other purchase, that is not processed together with hotel charges.</p>
<b>Membership and Recurring Billing</b>	
member_username	Name of the subscribing member. This is the name that will be used by the member to login to the subscription site. It must be unique with respect to other members of the same site_tag. If a name is already in use, the request will fail and no transaction will be processed.
member_duration	The duration in days of the initial membership.
member_password	The password the member will use to login to the subscription site.
member_memo	An optional merchant memo to attach to this member.
recurring_amount	The amount to bill the subscriber for at the end of the initial membership. If a recurring billing is successful, the subscriber's expiration date is automatically pushed forward until the time of the next recurring billing.
recurring_period	The time period between subsequent recurring billings. This is specified either as a number of days, or a special expression as generated by the Button Editor in the Online Administration can be used.
recurring_count	Stop the recurring billing after this number of successful recurring billings have been processed. The default is no limit.
recurring_prorate	When set to "true", the next recurring billing will occur either at the end of the initial membership period (member_duration), or at the next matching

	<p>recurring_period special expression, whichever is sooner. The amount of the first recurring billing will be reduced based on how many days early the recurring billing occurred compared to the number of days specified in member_duration.</p> <p>Example: member_duration=30 &amp; recurring_period=last_day%28sysdate%29%2B1 &amp; recurring_prorate=true</p> <p>This recurring_period expression specifies that this subscriber should be billed on the 1st of every month. If the subscriber joined on the 18th of February, the first recurring billing would occur on the 1st of March. The amount would be reduced by 2/3rd of the initial transaction amount, because the initial payment was for a 30 day membership, but the first recurring billing was processed after only 10 days, thus a credit is given for 20 days. From then on, the subscriber will always be billed the amount specified in recurring_amount on the 1st of each month.</p>
--	---

Parameter Definition Table - Table 2.2.1

### 3.1 Protocol Response Reference

In this section, all response parameters returned by the system are explained.

#### 3.1 Transaction Response Parameter Definitions

All standard transactions (auth, sale, refund, credit, capture) return URL encoded ("application/x-www-form-urlencoded" content-type) response data.

Transaction Response Parameter Definition Table	
status_code	<p>Single character status code.</p> <p>1 - Successful monetary transaction  I - Pending transaction, such as an unfunded ACH payment  T - Successful auth only transaction  0 - Failed transaction. Consult <b>auth_msg</b> and <b>reason_code2</b>  F - Settlement failure or returned ACH transaction. Not applicable at authorization time  D - Duplicate transaction. The <b>trans_id</b> of the original transaction will be returned</p>

	Any unexpected status code other than "0" and "F" should be interpreted as a successful transaction.
trans_id	If trans_id was supplied in the request, the same code will be returned (except for Duplicate transaction errors.) If trans_id was not supplied in the request, a unique system generated ID will be returned.
auth_code	A 6 digit bank authorization code for the transaction.
auth_date	This is the date and time in GMT (UTC) when the transaction was authorized by the system. "YYYY-MM-DD HH:MM:SS" format.
auth_msg	A human readable approval message, for example "APPROVED 123456", or "DECLINED 05". These messages are processor specific, may change, and are not intended to be machine readable.
avs_code	Single character AVS result code. See the AVS popup window in the online administration system for a list of codes.
cvv2_code	Single character CVV2 result code. See the CVV2 popup window in the online administration system for a list of codes.
ticket_code	Typically not used. Processor specific additional transaction reference code.
reason_code2	Typically not used. Processor specific additional DECLINE-reason code. Consult the transaction details screen for a declined transaction in the online administration system for verbose reason code messages.
member_id	Returned when a new subscriber/member was added.
recurring_id	Returned when a recurring billing record was established for a new subscriber/member.

Table 3.1.1

### 3.2 Batch Settle Response Parameter Definitions

Unlike for standard transactions, batch settlements return CSV encoded ("text/comma-separated-values" content-type) response data. This is done because for some processors (including ACH) more than one batch record can result from a single settlement request. In those cases all records are returned, one per line, in the CSV encoded HTTP body. The first line always defines the fields on the following lines. For example:

```
"STATUS","PAY_TYPE","ID","REPORT_DATE","CLOSE_BALANCE","CLOSE_MSG"
"1","C","200522350933","2005-03-07 23:59:19","999.99","TEST BATCH"
"0","K","","","",""
```

In this example, 2 records are returned. The first record indicates a successful credit card batch settlement. The second record indicates that no open ACH

transactions are available for settlement. Only the STATUS and PAY\_TYPE fields are guaranteed to always be defined.

Settlement Response Parameter Definition Table	
STATUS	<p>Single character batch settlement status code.</p> <p>1 - (one) Successful batch settlement            0 - (zero) Failed to settle batch            O - (big-O) No currently open transactions to settle</p> <p>Any unexpected status code other than "0" and "O" should be interpreted as a successful settlement.</p>
PAY_TYPE	Single character pay_type, echoed from request.
ID	System ID number uniquely identifying the batch settlement record.
REPORT_DATE	This is the date and time in GMT (UTC) when the batch settlement was performed. "YYYY-MM-DD HH:MM:SS" format.
CLOSE_BALANCE	The settlement balance. Positive for net deposits, negative if there are more refunds than sales in the batch.
CLOSE_MSG	A human readable settlement message, for example "BATCH SETTLED". These messages are processor specific, may change, and are not intended to be machine readable.

Table 3.2.1

## 4.0 Exception Reporting

Declined transactions are not considered exceptions, and are responded to with the normal response message. Exceptions occur when an invalid request is sent to the system, or some other abnormal condition occurs. Exception are reported as HTTP errors, and correspond to how exceptions are used in programming languages.

### 4.1 Exception Example

In this example, we're sending the same request to the system that was used in Section 1.5, but with an incorrect parameter name (**account\_ix** where **account\_id** is expected.)

```
POST /gw/sas/direct3.1 HTTP/1.0
Host: secure.fasttransact.com:1401
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
```

pay\_type=C&tran\_type=A&account\_ix=110006559149&card\_number=4444333322221  
186&card\_expire=0909&amount=5.00

The following will be returned by the system:

HTTP/1.0 604 Missing Parameter (account\_id)  
Connection: close  
Content-Type: text/plain  
Content-Length: 0

All HTTP codes other than 200 (604 in this case) indicates an exception. The HTTP message "Missing Parameter (account\_id)", contains a brief description of the error.

## 4.2 Exception codes and handling

Normally clients will simply treat all non-200 HTTP codes as errors, and return the HTTP message to the user. We do not recommend programming individual handling for all possible exceptions. If your application needs to do special processing for a certain error, determine what the code is for that error when it occurs, and check for that error code in future responses.

Codes 699 and 799 are general codes returned for many different reasons. If you need to do special processing for one of those codes, your application also needs to look at the first 5 digits of the HTTP message in order to match a particular error.

Exception code ranges	
600 - 698	Exception is most likely due to invalid input. The HTTP message contains a human-readable error message.
700 - 798	Exception is most likely due to processing error. The HTTP message contains a human-readable error message.
699, 799	The HTTP message will begin with an addition 5 digit machine-readable code that is used to further narrow down the error in software exception handling. For example: HTTP/1.0 699 20112: Invalid card expiration date 0x09 Connection: close Content-Type: text/plain Content-Length: 0 In this example, 20112 indicates that an invalid card expiration date was given.

Table 4.2.1

## 5.0 Special Feature Reference

In this section optional and advanced system features are described.

## 5.1 ID Generation

The system supports Tagged Transaction Processing, which means that before sending a transaction request to the system, the request is tagged with a known Transaction ID (trans\_id) number. If for any reason the response is lost, you will still be able to check the status of the transaction to find out if it succeeded, or needs to be resubmitted.

**NOTE: You should never send a "home made" trans\_id number. When using this field, all trans\_id numbers must be obtained using the getid3.1 method.**

ID numbers are obtained by a GET or POST request to <http://secure.fasttransact.com:1401/gw/sas/getid3.1>. For high speed applications that wish to take advantage of ID number caching, up to 10 IDs can be requested simultaneously, by passing a number (1-10) in the body of a POST, or in the query string of a GET request. e.g.

<http://secure.fasttransact.com:1401/gw/sas/getid3.1?3>.

The ID numbers are returned in the body of the text/plain response. Multiple IDs are separated by new-line characters.

## 5.2 CISP Compliant Storage, Repeat Billing, Up-Selling and Cross-Selling

The system supports CISP compliant cardholder data storage, which can be used to perform Repeat Billing or Up-Selling at a later time for a previous customer, without having to store the cardholder data at the merchant site. To take advantage of this feature, the cisp\_storage=1 parameter must be specified for the initial transaction. The Transaction ID (trans\_id) for that transaction, together with the last 5 digits of the card number become the virtual account number handle for that customer.

To perform a future repeat billing for the same customer, instead of having to store and specify the card number and billing address again, simply send "CS:121212121212:55555" as the card\_number/account\_number, where 121212121212 is the trans\_id of the original transaction, and 55555 is the last 5 digits of the card number (optional).

To change any of the stored information, such as the billing address, the updated information can be specified when issuing a Repeat Billing, which will over-ride the previously stored information. The new trans\_id for the updated transaction can be used in the future to perform another Repeat Billing using the updated default information.

This feature can be used across accounts, which makes it possible for a merchant's affiliates to bill one of the merchant's customers, without having to know the card number or billing information. This can be used to implement cross-selling partnerships. In order for a merchant to allow its affiliates to cross-bill its customers, the partners must first be authorized by entering their account

numbers on the Credit Card Setup page in the online administration system under CISP Managed Storage, Authorized Partners. The merchant needs to share each appropriate "CS:" virtual account number with the affiliate, in accordance with their mutual agreement.

For example, a merchant may have a checkbox on its payment form for a special offer from one of its affiliates. If a customer selects to take advantage of this offer, after completing the transaction, the merchant will send the virtual "CS:" account number to the affiliate. The affiliate will then proceed to bill the customer for the sale and ship the product.

### **5.3 Subscription and Recurring Billing**

Membership and recurring billing records can be automatically managed by the system, including CISP compliant storage of cardholder data. While these features are primarily intended to automate much of the management for developers of member-based websites, this feature can be used by any merchants that are interested in setting up some form of automated billing for their customers.

All recurring billing records that will be setup for automatic management must be created in association with a membership. The setup of such recurring billing records can be accomplished by including the "site\_tag" parameter as well as all relevant and required "Membership and Recurring Billing" fields with the transaction request (as outlined in section 2.2 of this document). If this option is used all information relating to the membership and the associated recurring billing setup will be stored and processed automatically by the system. For member based websites the system can also manage the login and password information automatically by sending information to a control CGI on your webserver. The specifics of the CGI Membership postings are covered in detail in separate documentation.

Recurring billings setup through Direct Mode can be used to automatically rebill the customer's credit card to renew the membership on a regular basis. Rebilling periods can be specified as either a number of days between recurring billing attempts (ex. 1, 7, 30) or as a special period expression. If specified as a special period expression the initial membership period can also be prorated. This allows membership signups to take place at any time during the month but automatically scale the charge amount to match the possible reduced period of time until the first scheduled recurring billing.

Please note that a membership can be associated with multiple site tags by including a comma separated list of site tags as the value for the "site\_tag" field instead of just a single tag. The first site tag in this list will become the member's primary site while the others will be automatically set as additional sites that membership has been granted access to.