Transaction Storage, Tokenization, and Recurring

Version 3.1

Trine Commerce Systems, Inc.
2613 Wilson Street
Austin, TX 78704
512-586-2736
legal@TrineCS.com
techsupport@TrineCS.com

Version 11/13/2014

Introduction

The **Trine Commerce System Gateway** provides a storage facility for merchant clients to manage recurring transactions and stored card and check information for future transactions. This is not legal for a merchant, but is allowed for a gateway service.

Recurring versus Stored Transaction Information
A *recurring* transaction always has the same amount. For example, if every month, a customer pays for a subscription, then its a recurring transaction. The merchant automatically processes a transaction for the same amount every month. *See the gateway manual for information about how to set up recurring transactions.*

If however, the amount varies then it is considered a "stored transaction." While basic transaction data is retained, an amount for the transaction must be supplied. For example, if you tell your electric company that you would like them to bill you on a credit card every month for the full balance due, then it is a "stored card" transaction. The amount is not known until billing is run. All the other details are known, just not the amount.

How do stored transactions work in general?
Your software creates a "customer account" on our server, and when you refer to that account, the customer's information is filled in. This requires that your software is able to send xml transactions to the gateway for processing.

There are three steps:

Accessing the Storage Processing Function

Account setup:	https://www.trinecs.net/processor/account/xml
Card/Pinless Debit Transactions:	https://www.trinecs.net/processor/transact/xml
Check Transactions:	https://www.trinecs.net/processor/check/xml

Test Server: Use the Test Server URL for integration programming and testing. The hash provided for testing is unique to the testing environment. A new production hash will be provided upon completion of integration programming.

In all cases, replace the live server URL with the following:
https://test.trinecs.com/processor/…

Permissions: Permissions must be set for both the Merchant account and the Terminal which hash code will be used for the transactions. Unless BOTH are set to "Yes", the process will be disallowed.

Step 1: Create Customer Account

This creates a “Customer account” on our server that you can reference in the future.
You reference it by the “merchant_reference” that you provided. You must ensure that the merchant_reference is *unique* for each of the “customer accounts” you create.
If the merchant_reference matches an existing account, the data will be updated.

Required Fields

hash	hash of the terminal to which transactions are posted
merchant_reference	Specify a UNIQUE identifier for each customer account. If the merchant_reference field sent matches an existing account, the original data will be over-written.
action	This specifies what action should be performed by the gateway: ACCOUNT creates new/updates customer account.

Example XML Request (with example text):

```
<request>
<hash_id>dkaloeok1323lk4lskeokc</hash_id>
<action>ACCOUNT</action>
<name>John Jones</name>
<street1>Street 1</street1>
<street2>Street 2</street2>
<city>Portland</city>
<state>CO</state>
<zip>80401</zip>
<email>Email@foo.com</email>
<phone>303-444-4444</phone>
<merchant_reference>666-555-44</merchant_reference>
</request>
```

Example XML Response (with example text):

Success:

```
<response>
<isError>No</isError>
<id>some reference</id>
</response>
```

Failure:

```
<response>
<isError>Yes</isError>
<error>Error Reason</error>
</response>
```

Step 2: Create Account Instrument(s)

Once you have created a “customer account” you must create an “account instrument” (credit card, pinless debit, or checking account) with the appropriate information. Each Customer Account may have more than one account instrument associated with it. Send the following request with the required fields and other related fields, such as card_number, exp_date, etc. (See examples and the appropriate API for fields required for transactions.) The instrument data

will be stored, and the response will provide a TOKEN for the specific transaction instrument. This code is *required* for processing transactions in step 3.

id	The TOKEN data is generated and returned by the gateway in field **<id>**. In order to process transactions, this code MUST be captured and stored for use in future transactions.

Required Fields

hash	hash of the terminal to which transactions are posted
merchant_reference	Specify a UNIQUE identifier for each customer account. If the merchant_reference sent matches an existing account, the data will be updated.
action	This specifies what action should be performed by the gateway: INSTRUMENT creates new transaction type (card, etc.)

Example XML Request (with example text):

For Credit Card:

```
<request>
<hash>dkaloeok1323lk4lskeokc</hash>
<action>INSTRUMENT</action>
<account_type>CARD</account>
<merchant_reference>666-555-44</merchant_reference>
<card_name>Mike Smith</card_name>
<card_number>4111111111111111</card_number>
<card_exp_month>10</card_exp_month>
<card_exp_year>11</card_exp_year>
<card_cvv>123</card_cvv>
</request>
```

For PINLESS Debit Card:

```
<request>
<hash>dkaloeok1323lk4lskeokc</hash>
<account_type>DEBIT</account>
<merchant_reference>666-555-44</merchant_reference>
<card_number>4111111111111111</card_number>
<card_exp_month>10</card_exp_month>
<card_exp_year>11</card_exp_year>
<pinless>Y</pinless>
</request>
```

For Check:

```
<request>
<hash>dkaloeok1323lk4lskeokc</hash>
<account_type>CHECK</account>
<merchant_reference>666-555-44</merchant_reference>
<check_transitroute>123456789</check_transitroute>
<check_account>12312312</check_account>
</request>
```

Example XML Response (with example text):

Success:

```
<response>
<isError>No</isError>
<id>TOKEN</id>
</response>
```

Note: The id is the "TOKEN" that identifies the the instrument set up for later transactions.

Failure:

```
<response>
<isError>Yes</isError>
<error>Error Reason</error>
</response>
```

Step 3: Send Transaction(s)

Once the Customer Account and Account Instruments are created, you can do transactions using that information. To do a transaction, pass the following: merchant_reference, account_instrument, and amount.

URLs

For Credit Cards and Pinless debit: https://www.trinecs.net/processor/transact/xml

For Checks: https://www.trinecs.net/processor/check/xml

Required Fields

hash	Hash code of the terminal to which transactions are posted
merchant_reference	The unique identifier assigned to the customer
account_instrument	The kind of transaction (credit card, check, pinless debit, etc.)
amount	Amount of the transaction
transaction_type	The type of transaction valid types are: Credit cards SALE, REFUND, AUTH, FORCE Checks ACHDEBIT, ACHCREDIT

Other fields available but not required may also be passed to be captured by the gateway for review in the terminal's transaction history. See the TCS APIs for Credit Card and Check integration for a list of these fields.

Example XML Request (with example text):

Sample Request:

```
<request>
<hash>dkaloeok1323lk4lskeokc</hash>
<merchant_reference>666-555-44</merchant_reference>
<account_instrument>TOKEN<account_instrument>
<transaction_type>SALE</transaction_type>
<amount>12.34</amount>
</request>
```

Responses from the network are the same as a normal transaction and are detailed in the TCS APIs for Credit Card and Check integration.