CyberSource Payment Manager™

API Reference Guide

July 2001

CyberSource®

CyberSource Contact Information

CPM Contact Information

www.cybersource.com
Visit our website for information about our company, products, and services.

info@cybersource.com
Request information regarding CyberSource Payment Manager.

tech-docs@cybersource.com
Ask questions about the CyberSource Payment Manager technical documentation. Report problems, omissions, or request changes in the documentation.

sales@cybersource.com
Place an order for any CyberSource product.

sales-support@cybersource.com
Potential customers may ask our sales support staff questions about our products and services.

software-support@cybersource.com
Request help with setup and installation or ask questions about maintenance updates or new releases regarding CyberSource Payment Manager.

To receive the best possible support from CyberSource Customer Support, email your requests to software-support@cybersource.com or fax your requests to 314.692.0805. Please include the following information:

- Contact Name
- Company and Department
- CyberSource or merchant customer number (where applicable)
- Your phone number and email address

ICS Contact Information

www.cybersource.com
Visit our website for information about our company, products, and services.

icsinfo@cybersource.com
Email requests for information regarding any Internet Commerce Suite product or call 800.530.9095 650.965.6000.

www.cybersource.com/support
Visit the Support Center website (password required) regarding any Internet Commerce Suite product issues.

ics_support@cybersource.com (USA)
eu_support@cybersource.com(International)
Email requests for support, or call 800.709.7779 (USA) or +44(0) 1932 871530.

Financial Processors Help Desk Information

First Data Merchant Services (Nashville Platform)

800.647.3722
Call FDMS division for FDMS Nashville platform-specific questions.

First Data Merchant Services (South Platform)

800.326.7985
Call FDMS division for FDMS South platform-specific questions.

Midwest Payment Systems

800.278.6888
Call Midwest Payment Systems for MPS-specific questions.

NDC Merchant Services (East Platform)

800.622.2318
Call NDC Merchant Services for NDC East platform-specific questions.

NDC Merchant Services (West Platform)

800.240.0562
Call National Data Processing Services for NDC West platform-specific questions.

Paymentech Merchant Services

603.896.8333
Call Paymentech's Merchant Services division for Paymentech merchant-specific questions from 8am to 8pm Monday through Friday Eastern time.

603.896.8320
Call Paymentech's Operations division for Paymentech transaction processing-specific questions 24 hours a day, seven days a week.

VISANet – Vital Merchant Help Desk

800.847.2772
Call the VISANet – Vital Merchant Help Desk for VISA and Vital-specific questions.

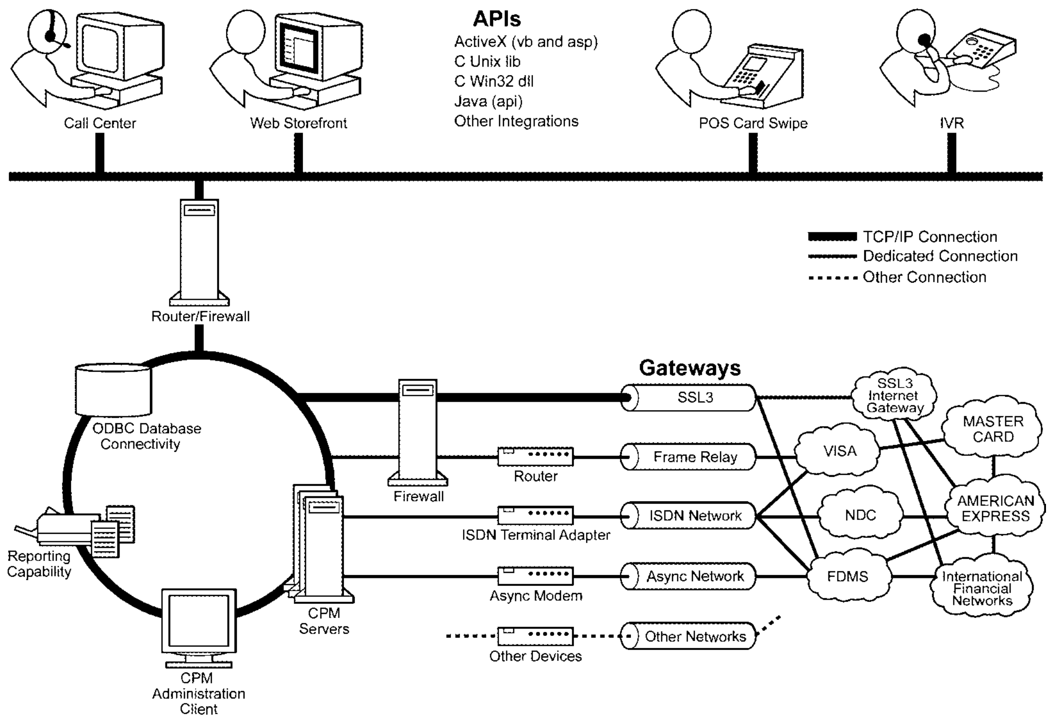
Security Warning

To secure the financial and personal information processed by the CPM Server, stored in the database, and routed through networks, the ecommerce system must be placed behind a firewall. We suggest you consult with providers of firewall and other information technology security solutions to protect and secure the transaction data of your customers, the CPM Server, and the database used by the CPM Server.

When processing transactions through the Internet between a web storefront and the CPM Server, the information is transmitted in plain text unless Secure Socket Layer (SSL) encryption is enabled. For a web storefront, we recommend that you enable the SSL certificate request security feature provided by CyberSource. This feature encrypts the financial and personal information transmitted between the web storefront and the CPM Server for secure transmission.

Out of concern for security, we also recommend against remote administration of the CPM Server through a remote access server (RAS). Administering the CPM Server through a RAS requires strong user authentication methods and strong data encryption. Always administer the CPM Server using the CPM Administration Client installed on a network computer behind a firewall. Refer to the illustration below as an example of a secure network setup.

Figure 1 CPM Architecture



Copyright and Restricted Rights Legends

CPM API Reference Guide

Copyright

Restricted Rights Legends

Copyright 2001 CyberSource Corporation. All rights reserved.
Document Number: 7030
Date Printed: July 2001

Trademarks and Document Conventions

Trademarks

CyberSource, the Power Behind the Buy Button, the CyberSource logo, and PaylinX are registered trademarks in the U.S. and other countries. PaylinX FRAUDefenZ, PaylinX EnterpriZ Engine, CPM Payment Manager, and CyberSource Fraud Manager are trademarks of CyberSource Corporation. CyberSource eCommerce Transaction Suite is a service mark of CyberSource Corporation. All other brands and product names are trademarks or registered trademarks of their respective owners.

Document Conventions

bold Palatino	Function names, messages, main menus, and submenus appear in bold Palatino font. For example, the function **SetSessionId** sets the session ID for a transaction.
italic Palatino	Terms and book titles appear in an italic Palatino font. For example, Section7, Setup the CPM Database, in the *CPM Setup Guide* contains information about CPM database setup.
Arial	Directories, paths, and file names appear in Arial font. For example, on Solaris systems the lcc.jar file may be located in the /lcc/java directory.
`Courier`	Screen text and example code appear in a Courier font. For example, a line in the configuration file appears as `MerchantID=demo`

Contents

Chapter 1

Introduction

The CPM API Reference Guide provides a developer the basic information necessary to create an API for maintaining and extracting merchant information (when stored in the Windows Registry or a configuration file) or to create an interface for performing credit card and Automated Clearing House (ACH) functions on a CyberSource Payment Manager (CPM) Server. The CPM API Reference Guide is written for developers with experience in programming. If you need programming assistance, please consult any of the available publications on ActiveX, C++, and Java, programming techniques.

You can use the CPM Merchant Editor for managing the merchant information stored in the Windows registry used by Windows-based CPM APIs when communicating with the CPM Server. For APIs using a configuration file, you can use the CPM Merchant Editor or a text editor such as Windows Notepad or the vi or pico editors in Unix to manage the merchant information. Developing an integration to the CPM API allows you to maintain merchant information without performing the changes manually on every client computer.

Chapter One provides a general overview of the available functions. Chapter Two discusses the CPM API. Included is a brief discussion of credit card functions, detailed descriptions of the API fields, and lists of the API fields passed to and from the financial processor for each credit card function. Chapter Three provides implementation information and sample code for ActiveX, Batch, C AIX, C Solaris, C Win32, and Java environments. Chapter Four provides test values so you can test your new API. Chapter Five contains the V3.5 database schema and maps the database fields to the API fields.

Anyone developing an API that communicates with the CPM Server or maintains the CPM database should read this document.

Note The term LCC (Loadable Client Cartridge) is synonymous to CPM API.

API basics

Generic function overview

The CPM API consists of a small set of functions that allow the developer to create, execute, and examine transactions. This chapter describes the basic concepts of the CPM API functions. This overview does not apply to the Batch API.

Generic flow chart of functions

To perform a transaction, the developer need only perform the following steps. Please refer to the appropriate API chapter of this guide for the function names.

1. Initialize the CPM API.

   The **Start up** function initializes the CPM programming API. You must call the **Start up** function before any other CPM API calls. This function is not used by ActiveX and Java.

2. Specify the CPM API configuration file. (optional)

   Call the **Set configuration file** function to identify which configuration file the CPM API should use. If the configuration is not specified, the CPM API uses the registry information or the default configuration file.

3. Create the transaction handle.

   Call the **Open transaction** function to allocate memory for the transaction data. This function's handle is the identifier with which to perform subsequent CPM API calls for this transaction. This function is not used by Java.

4. Set the session identifier.

   If transaction security is enabled, call the **Set session ID** function to set the session identifier.

5. Set transaction fields.

   The **Set value** function sets the values of the transaction fields. To set each transaction field, you must have one call to the **Set value** function per field.

6. Perform the transaction.

   Call the **Run transaction** function to start the transaction. The **Run transaction** function returns the transaction response code.

7. Examine the results of the transaction.

   Use the **Get value** function to retrieve output fields from the transaction.

8. Examine the session identifier.

   If transaction security is enabled, call the **Get session ID** function to retrieve the session identifier.

9. Destroy the transaction handle.

   Call the **Close transaction** function to clear the memory used by the transaction. This function is not used by Java.

10. Shutdown the CPM API.

    Shutdown the CPM API after performing all transactions. Call the **Shut down** function to terminate the CPM API. This function is not used by ActiveX and Java.

Generic function detail

Start up

Initializes the CPM API. You must call this function before performing any other CPM API calls. This function is not used by ActiveX and Java.

Shut down

Terminates the CPM API. Call this function after performing all CPM API calls. This function is not used by ActiveX and Java.

Set configuration file

Enables the CPM API to read from the specified configuration file other than the Windows registry for C Win dll and ActiveX or the default configuration file.

Open connection

This function establishes a persistent connection to the CPM Server that allows the transmission of multiple transactions over one connection. This function may increase processing time especially if you are using SSL encryption.

Close connection

This function closes a connection to the CPM Server that was opened with the OpenConnection function.

Open transaction

Opens a unique transaction handle. The transaction uses this handle for identification. Other function calls use this handle to manipulate the transaction. This function is not used by Java.

Close transaction

Closes the handle corresponding to the specified transaction. This function is not used by Java.

Set connection information

This function sets the connection information for the specified transaction at run time. This function overrides the settings in the configuration file or the Windows registry.

Run transaction

Sends a transaction to the CPM Server for execution.

Set session ID

Sets the session ID for a transaction. You must use this function when the CPM Server is configured to use security.

Get session ID

Retrieves the session ID for a transaction. The session ID is used when the CPM Server is configured to use security.

Set value

Sets the value of a field for a transaction.

Get value

Retrieves a field's value for a transaction.

Get value length

Returns the length of a field's value for a transaction.

Get value pointer

Retrieves the pointer value to a field's value for a transaction.

Note This function is not supported for all programming languages, such as Visual Basic.

Clear values

Removes all the values associated with an open transaction.

Dump values

Dumps all the values associated with the specified transaction into lcc_test.txt.

Print values

Prints all the values associated with the specified transaction to stdout.

API chart

The following table lists the APIs by platform currently supported in CPM Version 3.5.

Table 1 CPM API chart

Version 3.5
ActiveX (vb)
Batch
C (AIX, Unix, Win32)
Java (api)
Other integrations

Chapter 2

CPM Transaction API

The CPM Transaction API provides a software developer with an easy-to-use interface to perform all credit card and ACH functions on a CPM Server. For more information, refer to lcc.h and lcc.bas files on the CPM installation CD.

CPM API operations

Credit card transactions

Authorization

The Authorization transaction claims a portion of credit on a customer's credit card without actually transferring funds. The merchant's bank holds the authorization for a set number of days before the credit automatically returns to the customer. The merchant's bank determines the credit holding duration. When the purchase occurs, the merchant performs a Capture transaction to place the authorization into the batch for settlement.

If the authorization results in a **call** response, the merchant's bank requires verbal authorization. A client application must be able to accept a six-character authorization code when a call response occurs. The client application then calls the Manual Authorization transaction to update the system accordingly. If verbal authorization results in a denial, no action is required. Customer self-service points of sale, such as a self service website store front, cannot support the call response and the subsequent manual authorization. The call response functionality can only be supported in such points of sale where a merchant representative enters the order for the customer in call centers and customer service centers.

If the authorized amount is greater than the actual purchase amount, perform a Reversal to correct the authorized amount. Then perform a Capture with the new amount. If the authorized amount is not used, perform a Reversal for the full amount. The merchant can only perform authorization increments by making a second authorization for the difference.

The results of an Address Match and Zip Match check may be returned from the financial processor in an Authorize and Capture transaction if you subscribe to these financial processor services and provide the required AVS input information with the transaction.

Capture

The Capture transaction marks an authorization in a batch for settlement. The CPM Server's database holds these transactions until the next settlement occurs. At settlement time, the CPM Server reads the transactions out of the database and sends them to the banks for processing. The banks move funds from the customer's account to the merchant's account.

Authorize and Capture

The Authorize and Capture transaction is a shortcut version of the Authorization transaction and the Capture transaction. This transaction performs the authorization and, if successful, immediately captures the authorization for settlement. This transaction is helpful in Point-of-Sale environments where there is no time between the credit card authorization and when the customer receives their goods.

If the authorization results in a **call** response, voice authorization is required. The person performing the authorization calls the card issuing bank for instructions. If the issuing bank authorizes the purchase, the client application should accept that authorization number and perform a Capture transaction to place the transaction back into the CPM Server database with the approval code.

Reversal

The Reversal transaction lowers or negates previously authorized credit to a customer's account. Many processors request that a merchant uses this transaction to match the authorized amount against the settlement amount before settlement. Not doing so may result in higher merchant fees. The merchant can perform this transaction only once on each authorization. Not all credit card companies and card issuing banks currently support the reversal transaction, but CyberSource recommends implementation in preparation of industry acceptance of the transaction.

Note A Reversal works on an authorization that has not been settled. If an authorization has been settled, perform a Return transaction to return the money to the cardholder's account.

Return

The Return transaction returns money to a cardholder's account. The CPM Server database holds these transactions until the next settlement occurs. At settlement time, the CPM Server reads the transactions out of the database and sends them to the banks for processing. The banks move funds from the merchant's account to the customer's account.

Void

The Void transaction causes the CPM Server to void a captured transaction marked for settlement.

Note Void works on a captured authorization that has not been settled. If an authorization has been settled, perform a Return transaction to return the money to the cardholder's account.

Predial

The Predial transaction causes the CPM Server to open the communication channel before transmitting the transaction. This transaction is helpful for card-issuing bank processors using dial-up devices such as ISDN and analog modems for communication. This transaction is not mandatory.

Manual Authorization

The Manual Authorization transaction updates the CPM Server database after receiving a **call** response from an authorization. If the user receives voice authorization from the card issuing bank, the Manual Authorization transaction updates the original authorization transaction response in the database with the correct approval code.

Note A Manual Authorization is a database update only. If settlement is required, a Capture transaction still must be performed.

Lookup

A Lookup transaction retrieves an existing transaction from the CPM Server database and returns the credit card transaction to the user. The lookup transaction views transaction information in the CC_TRANSACTION table of the CPM database only.

ACH transactions

ACH is a type of Electronic Funds Transfer (EFT). The ACH transaction is verified or rejected by the Federal Reserve Automated Clearing House network. Use of ACH instantly verifies the customer's check reducing the chances of fraud. Further, the funds are instantly transferred from the customer's checking account to the merchant's acquiring account. The use of ACH is gaining in popularity quickly in all point of sale types.

ACH Verify

The ACH Verify transaction is only applicable to US banks. The ACH Verify transaction allows the merchant to compare each transaction to an external negative file maintained by third party vendors to locate accounts which have a history of bad checks outstanding or are closed for cause. These negative file databases, which are usually located on the processor system, are updated daily.

The ACH Verify transaction does not check if sufficient funds exist in the account nor does it guarantee that the money will be present at the time of settlement. The ACH verify transaction is not required before performing a deposit or refund. The ACH verify transaction occurs in real time and is the only ACH real-time transaction.

ACH Deposit

The ACH Deposit transaction is processed as part of a settlement batch and is sent to the financial processor. The financial processor, when processing the ACH Deposit transaction, takes funds from the customer’s account and places those funds in the merchant's account. The settlement of a ACH deposit transaction can only fail if there is syntactically incorrect information in the transaction. If insufficient funds exist to perform the transaction, the financial processor or acquiring bank calls the merchant directly.

ACH Refund

The ACH Refund transaction is processed as part of a settlement batch and is sent to the financial processor. The financial processor, when processing the ACH Refund transaction, takes funds from the merchant's account and places it in the customer’s account. The settlement of a ACH refund transaction can only fail if there is syntactically incorrect information in the transaction. If insufficient funds exist to perform the transaction, the financial processor or acquiring bank calls the merchant directly.

ACH Void

An ACH Deposit or ACH Refund transaction can be voided in the CPM database as long as the transaction has not already be been assigned to a settlement batch. The state of an ACH transaction can be determined using the ACH Lookup transaction or by reviewing the state of the ACH transaction in the EFT_TRANSACTION table in the CPM database.

ACH Lookup

The ACH Lookup transaction retrieves an existing ACH transaction from the CPM Server database and returns the ACH/EFT transaction to the user. This information is returned from the EFT_TRANSACTION database table.

Generic CPM API operations

Begin Session

The Begin Session function is used for security only. If security is enabled, the user must log into the CPM Server with their username and password to obtain their session identifier. This session identifier is the user's key to performing future transactions with the CPM Server. All subsequent transactions must contain this identifier to obtain access to the CPM Server.

Note If server security is disabled, Begin Session can be ignored and the session identifier field in the transaction function is left NULL.

End Session

The End Session function logs a user out of the CPM Server by *turning in* their session ID.

Predial

The Predial function is used for dial-up Gateways such as ISDN and modem Gateway types. The predial function send a command to the dial up device just prior to sending the transaction to negate the connection process for the dial up device.

Other functions

Other functions for working with the API are described in the Environment and Implementation chapter for each API by platform elsewhere in this guide.

Using the operations

To gain an understanding when various credit card transaction are performed using the CPM API functions, let's follow Sam as he makes several credit card transactions.

Example 1. Begin Session, Authorize and Capture, End Session At a local bookstore Sam presents a credit card to pay for his purchase. The CPM Server is installed at this store with security enabled. Because security is enabled on the CPM Server, when the sales clerk initiates Sam's transaction, a *begin session* function takes place. This begin session requires the sales clerk to login to the CPM Server with a username and password to obtain a session identifier. The sales clerk can then perform an *authorize and capture* so that authorization for the sale is immediately captured for settlement. This method ensures the payment in this situation where the sales clerk hands over the purchased books to the customer. After Sam leaves the store, the CPM Server is idle for several minutes because the sales clerk has no other customers. The CPM Server automatically performs an *end session* function to log out the sales clerk. The session is ended and no other transactions can take place until the sales clerk logs into the CPM Server and starts another session.

That same night, the CPM Server reads all the transactions out of the database and sends them to the bank for settlement. The bank moves funds from Sam's account to the merchant's bank account.

Example 2. Authorization At a call center for catalog sales the CPM Server is installed with the security featured enabled. The sales representative initiates a begin session and logs in to the CPM Server with her username and password at the start of her shift.

Sam calls in to the call center, the sales representative answers Sam's call, then enter's Sam's purchase and credit card information into the CPM Server through a transaction client. The sales representative selects the *authorization* transaction. Authorization claims a portion of credit available on Sam's credit card without transferring funds.

Example 3. Reversal The next day Sam decides he does not want one of the items ordered in Example 2. Sam calls the call center to cancel the item. Because capture has not occurred, the clerk uses a *reversal* transaction. The CPM API reversal function lowers the previously authorized credit amount to Sam's credit card account.

A few days later, when Sam's corrected order ships, the CPM Server is notified and performs a *capture* transaction. The capture marks the authorization in a batch for settlement. That same night, the CPM Server reads all the transactions out of the database and sends it to the bank for settlement.

During the days in between, the sales representative began and ended sessions with the CPM Server.

Example 4. Void At a hardware store Sam presents a credit card to pay for his purchase. The CPM Server is installed at this store with security disabled. The sales clerk performs an authorize and capture. After Sam leaves the store, he decides he does not want the item purchased. Because capture occurred and settlement has not occurred, the sales clerk must perform a *void* transaction.

The CPM API void function cancels a transaction marked for settlement.

Example 5. Return Sam decides to return one of the books to the bookstore. Because settlement occurred, the sales clerk must perform a *return* transaction.

The CPM API return function returns the purchase amount of the book plus any tax to Sam's credit card account when the CPM Server reads the return transaction out of the database and settles with the merchant's bank.

API fields

The CPM API is ordered into groups reflecting transaction and business functionality. The CPM API provides the developer with an interface to the financial processor. Strict adherence to these specifications allows the user to switch between financial processors without impact to the CPM API integration.

The CPM API fields are divided into groups. The groups comprise API fields based on transaction type and credit card or ACH transaction function. Not all groups are necessary for a transaction.

Note When building a CPM API integration, include all fields for an API group or a transaction type to be supported by the integration. This ensures the integration is compatible with all CPM Gateways and financial processor requirements. Fields are listed as required or optional when implementing the API fields at the point of sale or transaction processing client using the CPM API.

Working with financial processor specifications

The information provided in the following tables describe the types of data passed to the CPM API. However, if building a custom CPM API integration or modifying a CPM API integration provided by CyberSource, the financial processor may place additional restrictions and requirements on these fields. Refer to the financial processor’s specifications to maintain character compatibility, field length, and ensure correct transaction billing.

Building the CPM API integration

This functionality is only limited by the calling application’s API and the business functionality an integration supports.

A CPM API integration may not need full transaction functionality. For example, if an integration only supports a web store front, the integration may only need to support the authorization and authorize and capture transaction. However, to perform a return, reversal, or void on a transaction, the CPM Client or other customer CPM API client must by used. Order entry software may need to be manually updated to indicate an order change.

At the very minimum, to perform a credit card transaction, include the following CPM API groups in an integration:

- Base group
- Extended information group

Additional information and integration functionality, such as AVS and CVV, reduces financial processor interchange rates for the merchant. For more information, contact CyberSource Customer Support.

Field descriptions

The CPM database schema allows the last four positions of an amount to be interpreted as the decimal (cents). However, the CPM API and CPM Server only interpret the last two positions as the decimal.

When an amount is recorded to the CPM Server database, an implied decimal is added. As the amount is read from the database and then submitted back to the CPM API calling application or the financial processor, the decimal is appended to the amount.

Base group

The base group contains the basic information for most transactions. This group includes information about the type of credit card used and transaction. We recommend all CPM API integrations include all fields in the base group.

Table 2 Base group field descriptions

Field	Size	Description
Merchant Identifier	32	Identifies which merchant profile to utilize on the CPM Server. The CPM API merchant identifier (Merchant ID) must exactly match the merchant identifier on the CPM Server. In the CPM system, the merchant identifier coordinates transaction between the point of sale (POS) and the financial processor. The Merchant ID is set up in the CPM Administration client.
Merchant Name	32	Identifies which merchant profile to utilize on the CPM Server by referencing the Merchant Name in the configuration information used by the CPM API. The CPM API merchant name must be properly matched to a CPM Merchant ID as set in the configuration information used by the CPM API. This field is required with every function.
Account Number	28	The credit card number for the function. Do not include spaces or dashes.
Expiration Date	4	The credit card expiration date for the function. Use *MMYY* as the format.
Amount	12	The amount for the transaction. Use *DDDDDDDDDDCC* as the format. **Note** Do not include a decimal point in the transaction amount.

Table 2 Base group field descriptions

Field	Size	Description
Card Type	3	The type of credit card used in the transaction. CyberSource recommends that this field be filled by the user. The CPM Server performs syntax checks for the Account Number against the contents of this field. The user must enter the card type. The Card Type field is a required field for authorizations, authorize and capture, reversal, and return transactions. List of card types:

Code	Credit Card
000	Other
001	VISA
002	MasterCard
003	American Express
004	Discover
005	Diner’s Club
006	Carte Blanche
007	Japanese Credit Bank
008	Optima
009	Switch
010	GE Capital
011	General Electric Credit Corporation (GECC)
012	Beneficial
013	CitiBank Encryption Program
022	Liz Claiborne

Table 2 Base group field descriptions

Field	Size	Description
Sequence Number	15	A unique number assigned to each transaction. The Sequence Number is updated with the output of every transaction. The sequence number gives a developer an index to refer to the transaction and tracks the detail fields of an authorization. By inserting the sequence number of an authorization into the Sequence Number field of a reversal, capture, or void, the CPM Server looks up all detail fields of the authorization and copies them to the corresponding fields of the subsequent transaction. Refer to the Using the Sequence Number chapter in this guide for more information.
Approval Code	9	Contains a code of up to nine-characters assigned by the financial processors to approved authorizations. The data returned in this field from authorizations is required input into subsequent reversals and captures based on the authorization.

Table 2 Base group field descriptions

Field	Size	Description
Authorization Response Code	1	A processor independent response for all authorization requests. This field is only used for authorizations, reversals, and authorize and capture. The contents of this field are as follows: **Code** **Definition** **Description** A Approval Authorization is approved. C Call Voice authorization is required. Call the processor. D Decline The approval was declined. Check the Authorization Response Message or Processor Authorization Response Code for details. P Pick Up Card A problem exists with this credit card. Remove the card from the cardholder. Check the Authorization Response Message or Processor Authorization Response Code for details. X Expired Card This credit card is expired. E Error A processing error occurred. Check the Authorization Response Message or Processor Authorization Response Code for details.
Authorization Response Message	20	A text message supplied by the financial processor or CPM Server describing the results of the authorization request. This field is only supplied for authorizations, reversal, and authorize and capture requests.
Return Code Message	40	A description of the return code supplied from the CPM API function call.

Electronic fund transfer group

The electronic funds transfer group facilitates EFT and ACH transactions by the CPM Server.

When implementing a CPM API integration supporting EFT and ACH transaction processing, you must include all fields in the following CPM API groups:

- Base group
- Extended information group
- PS/2000 group
- Billing information group
- User defined fields group

Note Only the Paymentech Frame Gateway supports EFT and ACH processing.

Table 3 ACH group field descriptions

Field	Size	Description
Bank Account Number	17	The bank account number in which to credit or debit funds.
Bank ID	9	The transit routing number of the bank holding the target account. This field is also known as the ABA number or RDFI number.
Account Type	1	Type of bank account used in this transaction. Check the financial processor specification for the proper variables used for this field. These values are set at the point of sale.
Verification Result	1	CPM independent field detailing the result of the verification for this transaction. **Code** **Description** A Verification successful. D Verification rejected.
Processor Response Code	4	Financial processor dependent result code of the verification.
Processor Response Message	4	Financial processor dependent result message of the verification.

Extended information group

The extended information group provides detailed information returned with an authorization. Processors return many identifiers with every authorization to help them track the authorization request from authorization through settlement. Typically, the merchant does not need to be concerned with the content of these fields. However, the merchant must ensure that the processor received the extended information from an authorization during the reversal or capture of that transaction. The CPM Server assists the merchant in this function with the Sequence Number. Refer to the Using the Sequence Number section in this guide for more information. We recommend all CPM API integrations include all fields in the extended information group.

Table 4 Extended information group field descriptions

Field	Size	Description
Transaction Identifier	15	Returns the authorization system's Transaction Identifier. Supply this field with any subsequent capture or reversal functions. Financial processors use this field to associate authorization, reversal, and capture transactions.
Transaction Date	6	The date the transaction occurred. Use *YYMMDD* as the format. If the Transaction Date fields are set in the API, that date is sent to the processor when required. This data is stored in the LOCAL_DATE_TIME database field.

Transaction	Description
Authorization	Date of the authorization
Capture	Date of the authorization
Authorize and Capture	Date of the authorization and capture
Return	Date of the return
Reversal	Date of the reversal

Table 4 Extended information group field descriptions

Field	Size	Description
Transaction Time	6	The time the transaction occurred. Use *YYMMDD* as the format. If the Transaction Time fields are set in the API, that date is sent to the processor when required. This data is stored in the LOCAL_DATE_TIME database field.

Transaction	Description
Authorization	Time of the authorization
Capture	Time of the authorization
Authorize and Capture	Time of the authorization and capture
Return	Time of the return
Reversal	Time of the reversal

Field	Size	Description
Validation Code	4	Returns the issuing bank's Validation Code for this transaction. Supply this field with any subsequent capture or reversal functions. Financial processors use this field to associate authorization, reversal, and capture transactions.
Original Transaction Amount	12	This field contains the amount authorized in the original authorization. Use this field only with Reversal functions. Use *DDDDDDDDDDCC* as the format. **Note** Do not include a decimal point in the transaction amount.
Response Indicator	2	Details information about this authorization.
Returned ACI	1	The value of the requested transaction's Custom Payment Services qualification status. Typically this field is handled by the CPM Server and should not be altered.
Requested ACI	1	The value of the returned transaction Custom Payment Services qualification status. Typically this field is handled by the CPM Server and should not be altered.
POS Mode Code	2	Details point-of-sale mode about this authorization.
Market Specific Indicator	2	Details market specific identifier about this authorization.
Retrieval Reference Number	12	Returns the authorization system's Retrieval Reference Number. Supply this field with any subsequent capture or reversal functions. Financial processors use this field to associate authorization, reversal, and capture transactions.

Table 4 Extended information group field descriptions

Field	Size	Description
Account Data Source	1	The processor specific representation of the source of the customer data that was entered.
Card Holder ID	1	The processor specific representation of the method used to verify card holder identity.
Authorization Source Code	1	The processor specific representation of the authorization code.
Current Amount	12	This field contains the amount authorized in the original authorization. Use this field only with Reversal functions. Use *DDDDDDDDDDCC* as the format. **Note** Do not include a decimal point in the transaction amount.
Transaction Attribute	2	This field is reserved for future use.
Current Tax Amount	12	The current tax amount for the transaction. This field is added to the purchase amount field for the total purchase amount. Use *DDDDDDDDDDCC* as the format. **Note** Do not include a decimal point in the transaction amount.

Address verification service response (AVS) group

The address verification service response group provides the merchant two output fields with independent results to address verification requests. Not all financial processors provide this service. AVS verifies the cardholder's address and the shipping address.

When implementing a CPM API integration supporting AVS, you must include all fields in the following CPM API groups:

- Base group.
- Extended information group.
- AVS response group.
- PS/2000 group.
- Purchasing card group.
- Extended customer information group.
- Billing information group.
- Fraud group.

Note The results of address verification do not affect the results of the authorization. The merchant receives authorization if the credit card has enough available credit, but the address information is incorrect. The merchant must make the decision to accept a transaction where address verification failed.

Table 5 Address verification response group field descriptions

Field	Size	Description
Address Match	1	The results of the address portion of the address verification check. **Code** **Description** Y Match N No Match X Server Unavailable G Global AVS Service Unavailable U Domestic AVS Service not available.

Table 5 Address verification response group field descriptions

Field	Size	Description
Zip Match	1	The results of the zip code portion of the address verification check. **Code** **Description** Y Match N No Match X Service Unavailable

PS/2000 group

The PS/2000 group contains three input fields that assist direct marketing merchants in lowering their processing costs in *card not present* situations. Included in this group are the address verification fields. These fields allow the merchant to check a cardholder's billing address against a given address. On viewing the results of the address verification, the merchant can decide to accept or decline an authorization. We recommend all CPM API integrations include all fields in the PS/2000 group.

Table 6 PS/2000 group field descriptions

Field	Size	Description
Order Number	25	A merchant assigned order number.
Customer Street	20	The cardholder's billing street address. The syntax is <Street Number><SPACE><Street Name> for this field.
Customer Zip	9	The cardholder's billing zip code. The syntax for this field is *NNNNN* for a five-digit zip code or *NNNNNNNNN* for a nine-digit zip code.

Purchasing card group

Some companies use special credit cards, called purchasing cards, to facilitate the purchasing process. Issuing banks provide detailed reports to their customers about the usage of the card in the previous period. The CPM Server supports Purchasing Card level II support. Level two support provides the customer with a customer supplied purchase order number and tax amount.

Note The purchasing card support does not provide a merchant with additional features. A merchant can provide Purchasing Card level II support to their customers. Use these fields only if the credit card presented is a purchasing card.

Table 7 Purchasing card group field descriptions

Field	Size	Description
Purchase Card Order Number	16	The customer supplied identifier associated with a purchase.
Tax Amount	12	The tax amount for the transaction. This field is added to the purchase amount field for the total purchase amount. Use *DDDDDDDDDDCC* as the format. **Note** Do not include a decimal point in the transaction amount.
Commercial Card Type	2	The type of corporate card used. **Code** **Description** 00 Not a commercial card 01 Purchasing Card 02 Corporate Card 03 Business Card 04 Unknown
Ship To Zip Code	9	The zip code to where the product is shipped. Can be used as input for ship to zip code AVS.

CVV information group

Card Verification Value (CVV) is a unique number printed on a credit card that identifies the card holder in conjunction with the credit card account number. This number is not embedded in the magnetic strip, so it is never transferred during a card swipe. Instead, the number should only be known by the actual owner of the card as it is printed on the embossed signature area on the back of all credit cards. Some credit cards also include the card holder CVV value on the front of the credit card. The CVV value is passed through the CPM API to the financial processor who can return several results. The CVV value is matched with other information in the transaction. The specific CVV result returned depends on the card type used in the transaction and the financial processor.

You can use these fields in the API as parameters for American Express Card Identifier (CID) information and VISA Card Verification Value (CVV2) information, but the parameters are not stored in the CPM database for security reasons. The CPM database only stores the processor's response to the CVV request in the CVV Result field.

Table 8 CVV group field descriptions

Field	Size	Description
CVV	3 (VISA) 4 (American Express)	The contents of the CVV.
CVV Result Code	4	The result of the CVV.

Extended customer information group

The extended customer information group contains additional fields for storing customer information. We recommend all CPM API integrations include all fields in the extended customer information group.

Table 9 Extended customer information group field descriptions

Field	Size	Description
Customer Name	26	The customer's name. The syntax is <First Name><SPACE><Middle Initial><SPACE><Last Name> for this field.
Customer Phone	14	The customer's phone number.
Customer City	20	The city in which the customer resides.
Customer State	2	The state in which the customer resides.

Billing information group

The billing information group provides data for processors to change the merchant information that appears on the card holder's statement. Not all processors have this feature. We recommend all CPM API integrations include all fields in the billing information group.

Table 10 Billing information group field descriptions

Field	Size	Description
Merchant Billing Name	25	The merchant's billing name. If used for ACH/EFT transaction, additional information in an 'M' record that appears on the customer's statement.
Merchant Billing State	2	The merchant's billing state.
Merchant Billing Location	13	The merchant's billing location. If used for ACH/EFT transaction, additional information in an 'M' record that appears on the customer's statement.

User defined group

The CPM Server provides seven user fields for CPM API integration software developers to save additional transaction information to the CC_TRANACTION database table with every transaction performed. The API passes these fields to the CPM Server database where they are saved. These fields are NOT passed through to the financial processor.

Table 11 User defined group field descriptions

Field	Size	Description
User Sequence Number	50	Field usage defined by merchant. Can store a merchant-generated sequence number.
User Defined 1	50	Field usage defined by merchant.
User Defined 2	50	Field usage defined by merchant.
User Defined 3	50	Field usage defined by merchant.
User Defined 4	50	Field usage defined by merchant.
User Defined 5	50	Field usage defined by merchant.
User Source Name	31	Field usage defined by merchant.

CPM reserved group

These fields are reserved for future use with the CPM Server. Do not used these fields.

Table 12 Payment server reserved group field descriptions

Field	Size	Description
Reserved 1		Reserved field. Do not use.
Reserved 2		Reserved field. Do not use.

Magnetic track group (Card swiper data)

The magnetic track group passes data to the CPM Server in Card Present situations when the credit card is swiped through a card magnetic stripe card reader. When implementing a CPM API integration supporting the magnetic track group, you must include all fields in the following CPM API group:

- Terminal setup group.

Table 13 Magnetic track group field descriptions

Field	Size	Description
Track 1 Data	76	
Track 2 Data	37	

Ecommerce group

The Ecommerce group is for transactions originating from websites. You must include this group in all integrations supporting web based transactions.

Table 14 Ecommerce group field descriptions

Field	Size	Description	
Ecommerce Type	2	Electronic Commerce Flag for web transactions.	
		Code	**Description**
		Null Value (empty).	Not a web based transaction. At the point of sale, if not a web based transaction, this field should be empty
		01	Not secure. Channel encryption was not used between the browser and web server.
		02	Secure. Channel encryption was used between the browser and web server.

Terminal setup group

The terminal setup group describes the authorization environment for the authorization transaction request to the financial processor. This group is typically used with retail point of sale devices in card present transactions but can be used in any point of sale environment. When implementing a CPM API integration supporting card present transaction, you must include all fields in the following CPM API group:

- Magnetic track group

Table 15 Terminal setup group field descriptions

Field	Size	Description		
Card Present Flag	1	Indicates if the card is present at the time of the transaction.		
			Code	**Description**
			0	The card is not present (call center or IVR)
			1	The card is present (retail POS)
			2	Unknown
Terminal Capability	1	Indicates POS terminal capability used in transaction.		
			Code	**Description**
			0	Unknown
			1	Terminal has a magnetic stripe reader and manual entry capabilities
			2	Magnetic stripe reader
			3	No magnetic stripe reader

Table 15 Terminal setup group field descriptions

Field	Size	Description		
Terminal Type	1	Indicates POS terminal type used in transaction.		
			Code	**Description**
			0	Unknown.
			1	Standalone, credit card terminal.
			2	Electronic Cash Register/POS system.
			3	Unattended device.
POS Entry Mode	1	Indicates entry method of credit card information into POS terminal used in transaction.		
			Code	**Description**
			0	Unknown.
			1	Read from credit card magnetic track 1 (card swiper).
			2	Read from credit card magnetic track 1 (card swiper).
			3	Credit card number manually keyed in to POS terminal.

Table 15 Terminal setup group field descriptions

Field	Size	Description
Customer Present Flag	1	Indicates if the cardholder if present at the time of the transaction. The recurring value is typically used in environments supporting recurring transactions. For example, a recurring shipment of product based on merchant's customer agreement that results in recurring charges on the customer's credit card bill. **Code** / **Description** 0 / Customer present 1 / Customer not present 2 / Recurring **Note** If you are using the FDMS Nashville Frame Gateway, the Customer Present Flag value is always 1, Customer Not Present.

Processor specific response group

The CPM Server provides a processor independent interface to perform credit card functions. The processor specific return codes are available to the user through these fields. We recommend all CPM API integrations include all fields in the processor specific response group.

Table 16 Processor specific response group field descriptions

Field	Size	Description
Processor AVS Result	3	The processor specific address verification result codes.
Processor Authorization Response Code	4	The processor specific authorization response codes.

Security group

The security group contains input fields for logging into the CPM Server. Use these fields when transaction security is enabled on the CPM Server. We recommend all CPM API integrations include all fields in the security group to support CPM security.

Table 17 Security group field descriptions

Field	Size	Description
User Name	31	The CPM Server logon username.
Password	128	The password associated with the username.

Fraud group

The fraud group contains fields for fraud protection and input fields used in cardholder AVS and shipping to AVS. If the CPM API integration is to support compatibility with web based transaction or a fraud verification service, include all fields in this group.

Table 18 Fraud group field descriptions

Field	Size	Description
Ship to Address 1	20	The street address to where the product is shipped.
Ship to Address 2	20	The street address to where the product is shipped.
Ship to City	20	The city to where the product is shipped.
Ship to State	2	The state to where the product is shipped.
Ship to Phone	14	The phone number of the location to where the product is shipped.
Customer IP Address	30	The customer’s IP address if applicable to the transaction type or POS type.
Customer Email	50	The customer’s email address if applicable to the transaction type or POS type..
Fraud Reason Code	6	Three two-digit values, concatenated together, when a fraud alert is triggered during a fraud check. The two-digit value correspond to the type of rule and check violated when the fraud product encounters an offending transaction.
Fraud Score	4	The fraud score for a transaction. A numeric value output by a neural net model.
Fraud Response Code	256	A text message describing the results of the fraud check.
Skip Fraud Check	1	Skips the fraud check for an Authorization and an Authorize and Capture transaction.

Level III purchasing card group

Purchase Card Level III provides line item detail of all items obtained with the current purchasing card. The Purchase Card Level III uses line item information that can be repeated multiple times for each item in the purchase.

The Level III purchasing card group contains fields for additional tax, shipping, and handling charges.

If the CPM API integration is to support Purchase Card III compatibility, include all API fields in the following API groups:

- Level III purchase card group
- Level III purchase card line item detail group

To enhance business to business functionality, also include the following API group:

- Purchase card group

Table 19 Level III purchasing card group field descriptions

Field	Size	Description
Freight Amount	20	Total freight or shipping and handling charges. Use *DDDDDDDDDDCC* as the format.
Duty Amount	20	Total of any import or export duties for this transaction. Use *DDDDDDDDDDCC* as the format.
Ship from Zip Code	9	The zip code from which the product is shipped.
Discount Amount Applied	20	Total amount of the discount applied to the merchant for this transaction. Use *DDDDDDDDDDCC* as the format.
VAT Tax Amount	20	VAT or other tax included in this transaction. Use *DDDDDDDDDDCC* as the format.
VAT Tax Rate	20	Rate of VAT or other tax.
Alternative Tax ID	20	Tax identifier for the alternate tax included in this transaction.
Alternative Tax Amount	20	Total amount of the alternate tax included in this transaction. Use *DDDDDDDDDDCC* as the format.
Line Item Detail Count	4	The number of line item detail records in this purchase.

Level III purchasing card line item detail group

The Level III purchasing card line item detail group contains fields for information regarding each item in a purchase.

Table 20 Level III purchasing card line item detail group field descriptions

Field	Size	Description
Item Description	35	Text description of the item purchased.
Item Product Code	12	Product code of the item purchased.
Item Quantity	12	Number of units of the item purchased.
Item Unit of Measure	12	Unit of measure of measure code for the item purchased.
Item Tax Amount	12	Tax amount for item purchased. Use *DDDDDDDDDDCC* as the format.
Item Tax Rate	5	Tax rate applied to item purchased.
Item Total Amount	12	Total amount charged for item purchased. Use *DDDDDDDDDDCC* as the format.
Item Discount Amount	12	Amount of discount applied to this line item. Use *DDDDDDDDDDCC* as the format.
Item Commodity Code	12	Commodity code used to classify the item purchased.
Item Unit Cost	12	Unit cost of the item purchased. Use *DDDDDDDDDDCC* as the format.
Item Discount Indicator	1	Indicates if a discount was applied to the item purchased.
Item Tax Type Applied	4	Type of tax applied to the item purchased.
Item Tax Applied	1	Tax applied.
Item Tax Exempt	1	Tax exempt.

Private label card group

General Electric Capital Corporation (GECC) and Beneficial Finance Corporation are financial processors as well as a private label card issuers. GECC and Beneficial enable participating businesses to brand the card before issuing the card to their customers. These branded cards are used for promotions, sales, or specific customer plans. For example, an airline may issue a card under its private label to a frequent flyer customer, and the customer may be able to obtain free travel after accumulating frequent flyer miles. The GECC and Beneficial private label card provides fields and identifiers to handle this specialized processing.

The workflow for private label cards is variable when implementing the fields in a CPM API integration. It is necessary to coordinate with the private card issuer, the financial processor, and the acquiring bank for the correct use of the fields for your business type.

Note Only the Paymentech Frame Gateway supports GECC and Beneficial private label card processing.

Table 21 Private label card detail group

Field	Size	Description
Promotional Plan	1 (numeric)	This field is defined by GECC.
Promotional End Date	4 (numeric)	This field is defined by GECC.
Sale Type	1 (numeric)	This field is defined by GECC.
Line item 1	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 2	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 3	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 4	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 5	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 6	4 (numeric)	GECC private card line item detail. Defined by GECC.
Line item 7	4 (numeric)	GECC private card line item detail. Defined by GECC.

Table 21 Private label card detail group

Field	Size	Description
Microfiche Sequence Number	8 (numeric)	GECC Microfiche sequence number associated with the transaction.
Plan Number	5 (numeric)	This field is defined by GECC.
Credit Plan	5 (numeric)	This field’s use is defined by the merchant and Beneficial.
Department Codes	4 (numeric)	This field’s use is defined by the merchant and Beneficial.
SKU Number	9 (numeric)	The Stop Keeping Unit (SKU) number associated with the transaction based on the merchant’s unique SKU schema.
Item Description	40 (numeric)	Description of item associated with the transaction. Defined by merchant.
Store Number	5 (numeric)	This field’s use is defined by the merchant and Beneficial.

Using the Sequence Number

The Sequence Number field is more than an output field providing an index to the transaction table in the database. The sequence number field has two important functions when used as an input field. First, the sequence number associates a series of related transactions in the database. Second, the sequence number aids in transferring authorization detail information to subsequent transactions. The use of the Sequence Number is not required, but highly recommended.

The process of transferring funds with credit card functions can take a series of steps. For example, an authorization may need a reversal, leading to a capture. You can tie these transactions together by saving the Sequence Number of the original transaction and inputting this Sequence Number into the Sequence Number field of the next transaction based on the original authorization. The original transaction field is stored in the database so that the system can perform any audit trail necessary from the authorization through the capture.

The second use of the Sequence Number helps reduce the work necessary by the developer by decreasing the number of fields needed in reversals and captures. The CPM Server must often resubmit information received from an authorization to the processor during a reversal or capture. If the user loads the sequence number of an authorization in the Sequence Number field of an associated reversal or capture, the CPM Server retrieves the authorization information from the database and loads the input fields to the transaction properly.

When using the Sequence Number field to copy authorization information to subsequent transactions, the user still has the ability to override the information in the database. Information passed to the CPM Server through the CPM API has priority over the information in the database.

For example, a merchant performs an authorization for $100 and the authorization returns an approval code of 123456 and a Sequence Number of 0012. When the merchant performs the corresponding capture, the merchant realizes he/she only wants to use $90 of the original $100. The merchant can submit the capture with the following data: Purchase Amount = $90, Sequence Number = 0012. The CPM Server looks up the approval code of 123456 and purchase amount of $100. However, only the approval code is copied to the capture transaction as the purchase amount of $90 has priority.

CPM API field usage in a retail environment

When implementing the CPM API for a retail, card-present workflow environment, you must consider how the data is obtained and passed through to the CPM API. In many retail environments, a credit card magnetic stripe reader is used. This device is typically called a card swiper. When a card swiper reads encoded information from the credit card magnetic stripe, it must pass the data directly to the CPM Server through the API. For merchants accepting transactions in a retail environment, always populate the magnetic track group as read by the card swiper in the CPM API to reduce interchange fees. Consult with your acquiring bank and the card issuer for the best use of the magnetic track group. Refer elsewhere in this guide for more information on these fields.

The data from your card present point of sale is transmitted through the Magnetic track group of the CPM API but is stored in the credit card number, expiration date, and track one and track two fields in the CC_TRANSACTION table of the CPM database.

When developing your implementation, a method that allows a sales representative to enter the credit card number and expiration date. The API must be reset to accept the manually entered mode as set in the POS entry mode API fields. Send the data directly to the credit card number and expiration date fields in the CPM API.

Various rules apply that are unique for each credit card that your implementation accepts. For example, if a VISA card is used in the transaction, VISA only permits settlement of a transaction after the service completes or the product ships. Contact each credit card company for more information on usage requirements as you develop your implementation for the various card types.

Contact CyberSource Customer Support for more information on adapting the CPM API to your business workflow and POS types used by your business. To implement an integration to the CPM, contact CyberSource Global Professional Services.

CPM API transaction type identifiers

When running a CPM API transaction function, you must provide a transaction type identifier. The supported identifiers are in the CPM Server header files, lcc.h for C/C++ and lcc.bas for VB/ActiveX and lcc.java for Java, and listed in the following table.

Table 22 CPM API transaction type identifiers

Field	Size	Description
100	ID_AUTHORIZATION	Authorization
101	ID_REVERSAL	Reversal
102	ID_CAPTURE	Capture
103	ID_RETURN	Return
104	ID_AUTH_AND_CAPTURE	Authorization and Capture
105	ID_MANUAL_AUTHORIZATION	Manual Authorization
106	ID_VOID_TRANSACTION	Void Transaction
108	ID_ACH_VERIFY	ACH Verify
109	ID_ACH_DEPOSIT	ACH Deposit
110	ID_ACH_REFUND	ACH Refund
111	ID_ACH_VOID	ACH Void
112	ID_ACH_LOOKUP	ACH Lookup
120	ID_PREDIAL	Predial
121	ID_LOOKUP	Lookup
150	ID_BEGIN_SESSION	Begin Session
151	ID_END_SESSION	End Session

CPM API field identifiers

In order to create a transaction, you must set the proper fields required by the transaction. The fields are set through a single function call, **LCC_SetValue**. This function uses an ID, provided as an argument, to determine which field to set. You can use field identifiers to retrieve response fields in a similar manner. Common field identifiers are in the CPM Server header files, lcc.h for C/C++ and lcc.bas for VB/ActiveX and lcc.java for Java, and listed in the following table. Different field identifiers may be required by different Gateways.

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
51	ID_MERCHANT_ID	Merchant ID
100	ID_MERCHANT_NAME	Merchant Name for ID lookup
101	ID_ACCOUNT_NUMBER	Account Number
102	ID_EXPIRATION_DATE	Expiration Date
103	ID_AMOUNT	Transaction Amount
104	ID_CARD_TYPE	Card Type
105	ID_SEQUENCE_NUMBER	Sequence Number
106	ID_APPROVAL_CODE	Approval Code
107	ID_AUTH_RESPONSE_CODE	Authorization Response Code
108	ID_AUTH_RESPONSE_MESSAGE	Authorization Response Message
109	ID_RETURN_CODE_MESSAGE	Return Code Message
110	ID_BANK_ACCOUNT_NUMBER	Bank Account Number
111	ID_BANK_ID	Bank ID
112	ID_ACCOUNT_TYPE	Account Type
113	ID_VERIFICATION_RESULT	Verification Result
114	ID_PROCESSOR_RESPONSE_CODE	Processor Response Code
115	ID_PROCESSOR_RESPONSE_ MESSAGE	Processor Response Message
120	ID_TRANSACTION_ID	Transaction ID

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
121	ID_TRANSACTION_DATE	Transaction Date
122	ID_TRANSACTION_TIME	Transaction Time
123	ID_VALIDIATION_CODE	Validation Code
124	ID_ORIGINAL_AMOUNT	Original Amount
125	ID_RESPONSE_INDICATOR	Response Indicator
126	ID_RETURNED_ACI	Returned ACI
127	ID_REQUESTED_ACI	Requested ACI
128	ID_POS_MODE_CODE	POS Mode Code
129	ID_MARKET_SPECIFIC_INDICATOR	Market Specific Indicator
130	ID_RETRIEVAL_REFERENCE_ NUMBER	Retrieval Reference Number
131	ID_ACCOUNT_DATA_SOURCE	Account Data Source
132	ID_CARD_HOLDER_ID	Card Holder ID
133	ID_AUTHORIZATION_SOURCE_CODE	Authorization Source Code
134	ID_CURRENT_AMOUNT	Current Transaction Amount
135	ID_TRANS_ATTRIBUTE	Transaction Attribute
136	ID_CURRENT_TAX_AMOUNT	Current Tax Amount
140	ID_ADDRESS_MATCH	Address Match
141	ID_ZIP_MATCH	Zip Match
150	ID_ORDER_NUMBER	Order Number
151	ID_CUSTOMER_STREET	Street Number, Street Name
152	ID_CUSTOMER_ZIP	Zip Code
155	ID_PURCHASE_CARD_ORDER_ NUMBER	Purchase Card Order Number
156	ID_TAX_AMOUNT	Tax Amount
157	ID_COMMERCIAL_CARD_TYPE	Commercial Card Type

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
158	ID_SHIP_TO_ZIP_CODE	Ship To Zip Code
165	ID_CVV	CVV
166	ID_CVV_RESULT	CVV Result
170	ID_CUSTOMER_NAME	Customer Name
171	ID_CUSTOMER_PHONE	Customer Phone Number
172	ID_CUSTOMER_CITY	Customer City
173	ID_CUSTOMER_STATE	Customer State
190	ID_MERCHANT_BILLING_NAME	Merchant Billing Name
191	ID_MERCHANT_BILLING_STATE	Merchant Billing State
192	ID_MERCHANT_BILLING_LOCATION	Merchant Billing Location
200	ID_USER_SEQUENCE_NUMBER	User Sequence Number
201	ID_USER_DEFINED_1	User Defined 1
202	ID_USER_DEFINED_2	User Defined 2
203	ID_USER_DEFINED_3	User Defined 3
204	ID_USER_DEFINED_4	User Defined 4
205	ID_USER_DEFINED_5	User Defined 5
222	ID_RESERVED_1	Reserved - Do not use!
223	ID_RESERVED_2	Reserved - Do not use!
250	ID_TRACK_1_DATA	Track 1 Data
251	ID_TRACK_2_DATA	Track 2 Data
296	ID_USER_SOURCE_NAME	User Source Name
300	ID_E_COMMERCE_TYPE	Ecommerce Type
350	ID_CARD_PRESENT_FLAG	Card Present Flag
351	ID_TERMINAL_CAPABILITY	Terminal Capability
352	ID_TERMINAL_TYPE	Terminal Type

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
353	ID_POS_ENTRY_MODE	POS Entry Mode
354	ID_CUSTOMER_PRESENT_FLAG	Customer Present Flag
400	ID_PROCESSOR_AVS_RESULT	Processor AVS Result
401	ID_PROCESSOR_AUTH_RESPONSE_ CODE	Processor Authorization Response Code
450	ID_USERNAME	Username for begin session
451	ID_PASSWORD	Password for begin session
460	ID_SHIP_TO_ADDRESS_1	Ship to address1
461	ID_SHIP_TO_ADDRESS_2	Ship to address 2
462	ID_SHIP_TO_CITY	Ship to city
463	ID_SHIP _TO_STATE	Ship to state
464	ID_SHIP_TO_PHONE	Ship to phone number
465	ID_CUSTOMER_IP_ADDRESS	Customer IP address
466	ID_CUSTOMER_EMAIL	Customer email address
467	ID_FRAUD_REASON_CODE	Fraud reason code
468	ID_FRAUD_SCORE	Fraud score
469	ID_FRAUD_RESPONSE_CODE	Fraud response code
470	ID_SKIP_FRAUD_CHECK	Skip fraud check
500	ID_FREIGHT_AMOUNT	Freight amount
501	ID_DUTY_AMOUNT	Duty amount
503	ID_SHIP_FROM_ZIP_CODE	Ship from zip code
504	ID_DISCOUNT_AMOUNT_APPLIED	Discount amount applied
505	ID_VAT_TAX_AMOUNT	VAT tax amount
506	ID_VAT_TAX_RATE	VAT tax rate
507	ID_ALTERNATIVE_TAX_ID	Alternative tax ID

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
508	ID_ALTERNATIVE_TAX_AMOUNT	Alternative tax amount
509	ID_LINE_ITEM_DETAIL_COUNT	Line item detail count
600	ID_GECC_PROMOTIONAL_PLAN	Promotional Plan
601	ID_GECC_PROMOTIONAL_END_DATE	Promotional End Date
602	ID_GECC_SALE_TYPE	Sale Type
603	ID_GECC_LINE_ITEM_1	Line item 1
604	ID_GECC_LINE_ITEM_2	Line item 2
605	ID_GECC_LINE_ITEM_3	Line item 3
606	ID_GECC_LINE_ITEM_4	Line item 4
607	ID_GECC_LINE_ITEM_5	Line item 5
608	ID_GECC_LINE_ITEM_6	Line item 6
609	ID_GECC_LINE_ITEM_7	Line item 7
610	ID_GECC_MICROFICHE_SEQUENCE_NUM	Microfiche Sequence Number
611	ID_GECC_PLAN_NUMBER	Plan Number
650	ID_BENEFICIAL_CREDIT_PLAN	Credit Plan
651	ID_BENEFICIAL_DEPARTMENT_CODE	Department Codes
652	ID_BENEFICIAL_SKU_NUMBER	SKU Number
653	ID_BENEFICIAL_ITEM_DESCRIPTION	Item Description
654	ID_BENEFICIAL_STORE_NUMBER	Store Number
10000	ID_ITEM_DESCRIPTION	Item description
11000	ID_ITEM_PRODUCT_CODE	Item product code
12000	ID_ITEM_QUANTITY	Item quantity
13000	ID_ITEM_UNIT_OF_MEASURE	Item unit of measure
14000	ID_ITEM_TAX_AMOUNT	Item tax amount

Table 23 CPM API field identifiers

Numerical ID	Text Name	Description
15000	ID_ITEM_TAX_RATE	Item tax rate
16000	ID_ITEM_TOTAL_AMOUNT	Item total amount
17000	ID_ITEM_DISCOUNT_AMOUNT	Item discount amount
18000	ID_ITEM_COMMODITY_CODE	Item commodity code
19000	ID_ITEM_UNIT_COST	Item unit cost
20000	ID_ITEM_DISCOUNT_INDICATOR	Item discount indicator
21000	ID_ITEM_TAX_TYPE_APPLIED	Item tax type applied
22000	ID_ITEM_TAX_APPLIED	Item tax applied
23000	ID_ITEM_TAX_EXEMPT	Item tax exempt

Error identifiers

The **LCC_RunTransaction** function returns a value corresponding to one of the error identifiers. The error identifiers are in the CPM Server header files, lcc.h for C/C++ and lcc.bas for VB/ActiveX and lcc.java for Java, and listed in the following table. It may be necessary in the integration or the point of sale to convert the CPM API numerical ID and text message error into meaningful terms for customer or merchant representative.

Table 24 Error identifier descriptions

Numerical ID	Text message	Description
-114	ERR_MISSING_CONNECTION	Connection information was not specified.
-113	ERR_MERCHANT_NOT_SPECIFIED	No merchant name specified.
-112	ERR_CONFIG_BAD_ENTRY	Bad formatted entry in configuration file.
-111	ERR_CONFIG_LOOKUP_FAILED	Could not find merchant in configuration file.
-110	ERR_CONFIG_OPEN_FAILED	Failed to open configuration file.
-105	ERR_OPEN_DEBUG_FILE	Failed to open lcc_test.txt.
-104	ERR_VALUE_TOO_LARGE	Value too large for input buffer.

Table 24 Error identifier descriptions

Numerical ID	Text message	Description
-103	ERR_VALUE_NOT_FOUND	Value ID not set.
-102	ERR_CONNECTION_INVALID	Invalid Connection Handle.
-101	ERR_INVALID_HANDLE	Invalid Transaction Handle.
-100	ERR_NOT_INITIALIZED	CPM API Not Initialized.
-20	ERR_SSL_GENERAL	General SSL error.
-19	ERR_SSL_NEG	SSL negotiation error.
-18	ERR_SSL_LIB_INIT	Failed to load SSL modules.
-17	ERR_ENCRYPTION_FAILED	Encryption failed.
-16	ERR_SOCK_READ	Error reading from socket.
-15	ERR_SOCK_WRITE	Error writing to socket.
-14	ERR_SOCK_CONN_REFUSED	Connection refused.
-13	ERR_SOCK_CONNECT	Failed to connect to server.
-12	ERR_SOCK_CREATE	Failed to create socket.
-10	ERR_SOCK_LIB_INIT	Failed to start winsock.
0	ERR_SUCCESS	The transaction was successful. This is a global CPM API success message.
11	ERR_NOT_ACCEPTING_TX	Payment server is not accepting transactions.
13	ERR_INVALID.FIELD_25_NO_MAP	Unable to map to 2.5 error code.
30	ERR_TRANSACTION_TYPE	Incorrect transaction type.
101	ERR_INVALID_FIELD	Missing or invalid field.
102	ERR_FIELD_TOO_LONG	Field too long.
135	ERR_COULD_NOT_CONNECT_ PROCESSOR	Could not connect to the processor’s network.
136	ERR_TIME_OUT	Too much time between the transmission and receiving of data, the connection was dropped.

Table 24 Error identifier descriptions

Numerical ID	Text message	Description
137	ERR_UNIDENTIFIABLE	Unable to identify error. Contact CPM Product Support for more information.
138	ERR_INVALID_FIELD_VALUE	The data for the field is either too large or too small.
139	ERR_INVALID_DATA_TYPE	The data for the field does not match the field's data type.
141	ERR_INVALID_RECORD_SEQUENCE	Invalid sequence number.
142	ERR_INVALID_DIVISION_NUMBER	The merchant configuration is invalid on the Server.
143	ERR_CC_MOP_MISMATCH	The credit card number doesn't match the credit card type.
144	ERR_INVALID_MOP_FOR_DIVISION	The merchant doesn't accept this credit card type.
145	ERR_INVALID_TRANSACTION_TYPE	The transaction is not supported by the processor.
146	ERR_DUPLICATE_PURCHASE _ IDENTIFER	The purchase ID is already in use.
200	ERR_UNKNOWN	Unknown error.
230	ERR_SEQUENCE_NUMBER_NOT_ FOUND	The sequence number is missing.
231	ERR_MULTIPLE_SEQUENCE_ NUMBERS_FOUND	The sequence number is already in use.
232	ERR_VOID_FAILED_TX_SETTLED	The void function failed because the transaction has already settled.
233	ERR_VOID_FAILED_TX_NOT_FOUND	Cannot find the transaction to void.
234	ERR_SEQUENCE_NUMBER_ LOOKUP_TIMEOUT	Database timeout on sequence number lookup.
235	ERR_HTTP_SSL_COMMUNICATIONS_ FAILURE	As error status other than 200 from Vital.
1000	ERR_SERVICE_NEED_TCB_PRIV	The CPM Server needs to have *Act as Operating System* rights.

Table 24 Error identifier descriptions

Numerical ID	Text message	Description
1001	ERR_INVALID_USERNAME_ PASSWORD	An invalid username or password was entered.
1002	ERR_SESSION_INVALID	The session ID specified is invalid.
1003	ERR_SESSION_TIMEOUT	The session has timed out.
1004	ERR_SESSION_SOURCE_INVALID	Invalid session ID.
1005	ERR_MERCHANT_ID	Could not find merchant ID on the server.
1006	ERR_SESSION_ACCESS_DENIED	Access denied.
1007	ERR_CREATE_SESSION_FAILURE	Could not create a user session.
1008	ERR_SESSION_AMOUNT_LIMIT_ REACHED	Access denied. The total batch amount has reached or exceded the limit.
1009	ERR_SESSION_RETURN_AMOUNT_ LIMIT_REACHED	Access denied. The total return amount has reached or exceded the limit.
1010	ERR_LICENSE_KEY_VIOLATION	The invalid license key for this client.
1011	ERR_DATABASE_LOG_FAILURE	The transaction was not added to the database.
1012	ERR_DUPLICATE_TRANSACTION_ CHECK	The transaction is already in the database.
1015	ERR_VOID_TX_FAILURE	Cannot void transaction that has already settled.
1016	ERR_NO_LPC_AVAILABLE	No Gateway available to handle transactions for this merchant.
1017	ERR_LPC_NOT_ENABLED	The Gateway for this merchant is not enabled.
1018	ERR_TX_NOT_VOIDABLE	Can only void Captures, Authorize and Captures, and Returns.

Refer to the *CPM Messages and Processor Codes* guide for more information.

Input/output requirements per API function

This section lists all the possible inputs and outputs to each of the API functions. API fields requirements are recommendations at the point of sale. For example, if your business accepts Purchase Card Level III, enable acceptance if this transaction information at your points of sale. Only some field outputs are appropriate for the customer or merchant representative at the point of sale. Refer to the API fields description in this section in this guide and CyberSource Customer Support and your financial processor for more information. The CPM Server will not process transaction if the required fields are not submitted from the CPM API to the CPM Server.

Authorization

Input

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Account Number	Required.	Either Account Number and Expiration Date or Track 1 Data, or Track 2 Data is required.
Expiration Date	Required.	
Track 1 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Track 2 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Amount	Required.	
CVV	Recommended if processor requests this field.	
Card Type	Required.	
Order Number	Recommended for direct marketing merchants.	
Requested ACI	Not required.	Refer to the API fields section in this guide for more information.
Address Field	Recommended for direct marketing merchants.	

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
Zip Code	Recommended for direct marketing merchants.	
Customer Name	Not required.	
Customer Phone	Not required.	
Customer City	Not required.	
Customer State	Not required.	
Card Present Flag	Not required.	
Customer Present Flag	Not required.	
Terminal Type	Not required.	
Terminal Capability	Not required.	
POS Entry Mode	Not required.	
Ecommerce Type	Required for web-based transactions.	
Purchase Card Order Number	Required for purchasing cards only.	
Tax Amount	Required for purchasing cards only.	
Commercial Card Type	Required for purchasing cards only.	
Ship To Zip Code	Required for purchasing cards only.	
User Sequence Number	Not required.	
Merchant Billing Name	Not required.	
Merchant Billing State	Not required.	
Merchant Billing Location	Not required.	

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	
User Source Name	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Ship to address 1	Not required.	
Ship to address 2	Not required.	
Ship to city	Not required.	
Ship to state	Not required.	
Ship to phone	Not required.	
Customer IP address	Not required.	
Customer email	Not required.	
Skip fraud check	Not required.	When called, use *Y* to skip fraud check for an Authorization.
Freight Amount	Required for purchasing card level III only.	
Duty Amount	Required for purchasing card level III only.	
Ship from Zip Code	Required for purchasing card level III only.	
Discount Amount Applied	Required for purchasing card level III only.	

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
VAT Tax Amount	Required for purchasing card level III only.	
VAT Tax Rate	Required for purchasing card level III only.	
Alternative Tax ID	Required for purchasing card level III only.	
Alternative Tax Amount	Required for purchasing card level III only.	
Line Item Detail Count	Required for purchasing card level III only.	
Item Description	Required for purchasing card level III only.	
Item Product Code	Required for purchasing card level III only.	
Item Quantity	Required for purchasing card level III only.	
Item Unit of Measure	Required for purchasing card level III only.	
Item Tax Amount	Required for purchasing card level III only.	
Item Tax Rate	Required for purchasing card level III only.	
Item Total Amount	Required for purchasing card level III only.	
Item Discount Amount	Required for purchasing card level III only.	
Item Commodity Code	Required for purchasing card level III only.	
Item Unit Cost	Required for purchasing card level III only.	
Item Discount Indicator	Required for purchasing card level III only.	

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
Item Tax Type Applied	Required for purchasing card level III only.	
Item Tax Applied	Required for purchasing card level III only.	
Item Tax Exempt	Required for purchasing card level III only.	
Promotional Plan	Required for GECC private label card.	
Promotional End Date	Required for GECC private label card.	
Sale Type	Required for GECC private label card.	
Line item 1	Required for GECC private label card.	
Line item 2	Required for GECC private label card.	
Line item 3	Required for GECC private label card.	
Line item 4	Required for GECC private label card.	
Line item 5	Required for GECC private label card.	
Line item 6	Required for GECC private label card.	
Line item 7	Required for GECC private label card.	
Microfiche Sequence Number	Required for GECC private label card.	
Plan Number	Required for GECC private label card.	
Credit Plan	Required for Beneficial private label card.	

Table 25 Authorization input requirements

Field	Point of sale field requirement	Description
Department Codes	Required for Beneficial private label card.	
SKU Number	Required for Beneficial private label card.	
Item Description	Required for Beneficial private label card.	
Store Number	Required for Beneficial private label card.	

Output

Table 26 Authorization output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Authorization Response Message	Refer to the API fields section in this guide for more information.
Authorization Response Code	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Identifier	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.
Transaction Time	Refer to the API fields section in this guide for more information.

Table 26 Authorization output fields

Field	**Description**
Validation Code	Refer to the API fields section in this guide for more information.
Response Indicator	Refer to the API fields section in this guide for more information.
Returned ACI	Refer to the API fields section in this guide for more information.
Requested ACI	Refer to the API fields section in this guide for more information.
POS Mode Code	Refer to the API fields section in this guide for more information.
Market Specific Indicator	Refer to the API fields section in this guide for more information.
Retrieval Reference Number	Refer to the API fields section in this guide for more information.
Account Data Source	Refer to the API fields section in this guide for more information.
Card Holder ID	Refer to the API fields section in this guide for more information.
Authorization Source Code	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Card Present Flag	Refer to the API fields section in this guide for more information.
Terminal Capability	Refer to the API fields section in this guide for more information.
Terminal Type	Refer to the API fields section in this guide for more information.
POS Entry Mode	Refer to the API fields section in this guide for more information.
Customer Present Flag	Refer to the API fields section in this guide for more information.
CVV Result	Refer to the API fields section in this guide for more information.
Ecommerce Type	Refer to the API fields section in this guide for more information.
Purchase Card Order Number	Refer to the API fields section in this guide for more information.
Tax Amount	Refer to the API fields section in this guide for more information.

Table 26 Authorization output fields

Field	Description
Commercial Card Type	Refer to the API fields section in this guide for more information.
Ship To Zip Code	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Track 1 Data	Refer to the API fields section in this guide for more information.
Track 2 Data	Refer to the API fields section in this guide for more information.
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Processor AVS Result	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.
Address Field	Refer to the API fields section in this guide for more information.
Zip Code	Refer to the API fields section in this guide for more information.
Fraud reason code	Refer to the API fields section in this guide for more information.
Fraud score	Refer to the API fields section in this guide for more information.
Fraud response code	Refer to the API fields section in this guide for more information.

Capture

Input

Table 27 Capture input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Account Number	Required. This field is filled in if Sequence Number is used.	
Expiration Date	Required. This field is filled in if Sequence Number is used.	
Amount	Required. This field is filled in if Sequence Number is used.	
Sequence Number	Not required. Highly recommended.	
Approval Code	Required. This field is filled in if Sequence Number is used.	
Transaction Identifier	Required. This field is filled in if Sequence Number is used.	
Transaction Date	Required. This field is filled in if Sequence Number is used.	
Transaction Time	Required. This field is filled in if Sequence Number is used.	
Validation Code	Required. This field is filled in if Sequence Number is used.	
Response Indicator	Required. This field is filled in if Sequence Number is used.	
Returned ACI	Required. This field is filled in if Sequence Number is used.	
Requested ACI	Required. This field is filled in if Sequence Number is used.	
POS Mode Code	Required. This field is filled in if Sequence Number is used.	
Market Specific Indicator	Required. This field is filled in if Sequence Number is used.	

Table 27 Capture input requirements

Field	Point of sale field requirement	Description
Retrieval Reference Number	Required. This field is filled in if Sequence Number is used.	
Account Data Source	Not required. This field is filled in if Sequence Number is used.	
Card Holder ID	Not required. This field is filled in if Sequence Number is used.	
Authorization Source Code	Not required. This field is filled in if Sequence Number is used.	
Order Number	Recommended for direct marketing merchants. This field is filled in if Sequence Number is used.	
CVV Result	Required if CVV check was performed. This field is filled in if Sequence Number is used.	
Card Present Flag	Not required. This field is filled in if Sequence Number is used.	
Customer Present Flag	Not required. This field is filled in if Sequence Number is used.	
Terminal Type	Not required. This field is filled in if Sequence Number is used.	
Terminal Capability	Not required. This field is filled in if Sequence Number is used.	
POS Entry Mode	Not required. This field is filled in if Sequence Number is used.	
Ecommerce Type	Required for web-based transactions. This field is filled in if Sequence Number is used.	
Purchase Card Order Number	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
Tax Amount	Required for purchasing cards only. This field is filled in if Sequence Number is used.	

Table 27 Capture input requirements

Field	Point of sale field requirement	Description
Commercial Card Type	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
Ship To Zip Code	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
User Sequence Number	Not required. This field is filled in if Sequence Number is used.	
User Defined 1	Not required. This field is filled in if Sequence Number is used.	
User Defined 2	Not required. This field is filled in if Sequence Number is used.	
User Defined 3	Not required. This field is filled in if Sequence Number is used.	
User Defined 4	Not required. This field is filled in if Sequence Number is used.	
User Defined 5	Not required. This field is filled in if Sequence Number is used.	
User Source Name	Not required. This field is filled in if Sequence Number is used.	
Processor AVS Result	Required. This field is filled in if Sequence Number is used.	
Processor Authorization Response Code	Required. This field is filled in if Sequence Number is used.	
Merchant Billing Name	Not required. This field is filled in if Sequence Number is used.	
Merchant Billing State	Not required. This field is filled in if Sequence Number is used.	
Merchant Billing Location	Not required. This field is filled in if Sequence Number is used.	

Table 27 Capture input requirements

Field	Point of sale field requirement	Description
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Ship to address 1	Not required.	
Ship to address 2	Not required.	
Ship to city	Not required.	
Ship to state	Not required.	
Ship to phone	Not required.	
Customer IP address	Not required.	
Customer email	Not required.	
Freight Amount	Required for purchasing card level III only.	
Duty Amount	Required for purchasing card level III only.	
Ship from Zip Code	Required for purchasing card level III only.	
Discount Amount Applied	Required for purchasing card level III only.	
VAT Tax Amount	Required for purchasing card level III only.	
VAT Tax Rate	Required for purchasing card level III only.	
Alternative Tax ID	Required for purchasing card level III only.	
Alternative Tax Amount	Required for purchasing card level III only.	
Line Item Detail Count	Required for purchasing card level III only.	

Table 27 Capture input requirements

Field	**Point of sale field requirement**	**Description**
Item Description	Required for purchasing card level III only.	
Item Product Code	Required for purchasing card level III only.	
Item Quantity	Required for purchasing card level III only.	
Item Unit of Measure	Required for purchasing card level III only.	
Item Tax Amount	Required for purchasing card level III only.	
Item Tax Rate	Required for purchasing card level III only.	
Item Total Amount	Required for purchasing card level III only.	
Item Discount Amount	Required for purchasing card level III only.	
Item Commodity Code	Required for purchasing card level III only.	
Item Unit Cost	Required for purchasing card level III only.	
Item Discount Indicator	Required for purchasing card level III only.	
Item Tax Type Applied	Required for purchasing card level III only.	
Item Tax Applied	Required for purchasing card level III only.	
Item Tax Exempt	Required for purchasing card level III only.	
Promotional Plan	Required for GECC private label card.	
Promotional End Date	Required for GECC private label card.	

Table 27 Capture input requirements

Field	Point of sale field requirement	Description
Sale Type	Required for GECC private label card.	
Line item 1	Required for GECC private label card.	
Line item 2	Required for GECC private label card.	
Line item 3	Required for GECC private label card.	
Line item 4	Required for GECC private label card.	
Line item 5	Required for GECC private label card.	
Line item 6	Required for GECC private label card.	
Line item 7	Required for GECC private label card.	
Microfiche Sequence Number	Required for GECC private label card.	
Plan Number	Required for GECC private label card.	
Credit Plan	Required for Beneficial private label card.	
Department Codes	Required for Beneficial private label card.	
SKU Number	Required for Beneficial private label card.	
Item Description	Required for Beneficial private label card.	
Store Number	Required for Beneficial private label card.	

Output

Table 28 Capture output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Original Transaction Amount	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Authorization Response Code	Refer to the API fields section in this guide for more information.
Authorization Response Message	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Identifier	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.
Transaction Time	Refer to the API fields section in this guide for more information.
Response Indicator	Refer to the API fields section in this guide for more information.
Returned ACI	Refer to the API fields section in this guide for more information.
Requested ACI	Refer to the API fields section in this guide for more information.
POS Mode Code	Refer to the API fields section in this guide for more information.
Market Specific Indicator	Refer to the API fields section in this guide for more information.

Table 28 Capture output fields

Field	Description
Retrieval Reference Number	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Card Present Flag	Refer to the API fields section in this guide for more information.
Terminal Capability	Refer to the API fields section in this guide for more information.
Terminal Type	Refer to the API fields section in this guide for more information.
POS Entry Mode	Refer to the API fields section in this guide for more information.
Customer Present Flag	Refer to the API fields section in this guide for more information.
Ecommerce Type	Refer to the API fields section in this guide for more information.
Purchase Card Order Number	Refer to the API fields section in this guide for more information.
Tax Amount	Refer to the API fields section in this guide for more information.
Commercial Card Type	Refer to the API fields section in this guide for more information.
Ship To Zip Code	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Track 1 Data	Refer to the API fields section in this guide for more information.
Track 2 Data	Refer to the API fields section in this guide for more information.

Table 28 Capture output fields

Field	Description
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Processor AVS Result	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.
Validation Code	Refer to the API fields section in this guide for more information.
Address Field	Refer to the API fields section in this guide for more information.
Zip Code	Refer to the API fields section in this guide for more information.

Authorize and Capture

Input

Table 29 Authorize and Capture input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Account Number	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Expiration Date	Required.	
Track 1 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Track 2 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Amount	Required.	
CVV	Recommended if processor requests this field.	
Card Type	Required.	

Table 29 Authorize and Capture input requirements

Field	Point of sale field requirement	Description
Order Number	Recommended for direct marketing merchants.	
Address Field	Recommended for direct marketing merchants.	
Zip Code	Recommended for direct marketing merchants.	
Customer Name	Not required.	
Customer Phone	Not required.	
Customer City	Not required.	
Customer State	Not required.	
Card Present Flag	Not required.	
Customer Present Flag	Not required.	
Terminal Type	Not required.	
Terminal Capability	Not required.	
POS Entry Mode	Not required.	
Ecommerce Type	Required for web-based transactions.	
Purchase Card Order Number	Required for purchasing cards only.	
Tax Amount	Required for purchasing cards only.	
Commercial Card Type	Required for purchasing cards only.	
Ship To Zip Code	Required for purchasing cards only.	
Merchant Billing Name	Not required.	
Merchant Billing State	Not required.	

Table 29 Authorize and Capture input requirements

Field	Point of sale field requirement	Description
Merchant Billing Location	Not required.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	
User Source Name	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Ship to address 1	Not required.	
Ship to address 2	Not required.	
Ship to city	Not required.	
Ship to state	Not required.	
Ship to phone	Not required.	
Customer IP address	Not required.	
Customer email	Not required.	
Skip fraud check	Not required.	When called, use *Y* to skip fraud check for an Authorization.
Freight Amount	Required for purchasing card level III only.	
Duty Amount	Required for purchasing card level III only.	

Table 29 Authorize and Capture input requirements

Field	Point of sale field requirement	Description
Ship from Zip Code	Required for purchasing card level III only.	
Discount Amount Applied	Required for purchasing card level III only.	
VAT Tax Amount	Required for purchasing card level III only.	
VAT Tax Rate	Required for purchasing card level III only.	
Alternative Tax ID	Required for purchasing card level III only.	
Alternative Tax Amount	Required for purchasing card level III only.	
Line Item Detail Count	Required for purchasing card level III only.	
Item Description	Required for purchasing card level III only.	
Item Product Code	Required for purchasing card level III only.	
Item Quantity	Required for purchasing card level III only.	
Item Unit of Measure	Required for purchasing card level III only.	
Item Tax Amount	Required for purchasing card level III only.	
Item Tax Rate	Required for purchasing card level III only.	
Item Total Amount	Required for purchasing card level III only.	
Item Discount Amount	Required for purchasing card level III only.	
Item Commodity Code	Required for purchasing card level III only.	

Table 29 Authorize and Capture input requirements

Field	**Point of sale field requirement**	**Description**
Item Unit Cost	Required for purchasing card level III only.	
Item Discount Indicator	Required for purchasing card level III only.	
Item Tax Type Applied	Required for purchasing card level III only.	
Item Tax Applied	Required for purchasing card level III only.	
Item Tax Exempt	Required for purchasing card level III only.	
Promotional Plan	Required for GECC private label card.	
Promotional End Date	Required for GECC private label card.	
Sale Type	Required for GECC private label card.	
Line item 1	Required for GECC private label card.	
Line item 2	Required for GECC private label card.	
Line item 3	Required for GECC private label card.	
Line item 4	Required for GECC private label card.	
Line item 5	Required for GECC private label card.	
Line item 6	Required for GECC private label card.	
Line item 7	Required for GECC private label card.	
Microfiche Sequence Number	Required for GECC private label card.	

Table 29 Authorize and Capture input requirements

Field	Point of sale field requirement	Description
Plan Number	Required for GECC private label card.	
Credit Plan	Required for Beneficial private label card.	
Department Codes	Required for Beneficial private label card.	
SKU Number	Required for Beneficial private label card.	
Item Description	Required for Beneficial private label card.	
Store Number	Required for Beneficial private label card.	

Output

Table 30 Authorize and Capture output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Authorization Source Code	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Authorization Response Code	Refer to the API fields section in this guide for more information.

Table 30 Authorize and Capture output fields

Field	Description
Authorization Response Message	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Identifier	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.
Transaction Time	Refer to the API fields section in this guide for more information.
Validation Code	Refer to the API fields section in this guide for more information.
Response Indicator	Refer to the API fields section in this guide for more information.
Returned ACI	Refer to the API fields section in this guide for more information.
Requested ACI	Refer to the API fields section in this guide for more information.
POS Mode Code	Refer to the API fields section in this guide for more information.
Market Specific Indicator	Refer to the API fields section in this guide for more information.
Retrieval Reference Number	Refer to the API fields section in this guide for more information.
Account Data Source	Refer to the API fields section in this guide for more information.
Card Holder ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Card Present Flag	Refer to the API fields section in this guide for more information.
Terminal Capability	Refer to the API fields section in this guide for more information.
Terminal Type	Refer to the API fields section in this guide for more information.
POS Entry Mode	Refer to the API fields section in this guide for more information.
Customer Present Flag	Refer to the API fields section in this guide for more information.
CVV Result	Refer to the API fields section in this guide for more information.

Table 30 Authorize and Capture output fields

Field	Description
Ecommerce Type	Refer to the API fields section in this guide for more information.
Purchase Card Order Number	Refer to the API fields section in this guide for more information.
Tax Amount	Refer to the API fields section in this guide for more information.
Commercial Card Type	Refer to the API fields section in this guide for more information.
Ship To Zip Code	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Processor AVS Result	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.
Address Field	Refer to the API fields section in this guide for more information.
Zip Code	Refer to the API fields section in this guide for more information.
Fraud reason code	Refer to the API fields section in this guide for more information.
Fraud score	Refer to the API fields section in this guide for more information.
Fraud response code	Refer to the API fields section in this guide for more information.

Reversal

Input

Table 31 Reversal input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Account Number	Required. This field is filled in if Sequence Number is used.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Expiration Date	Required. This field is filled in if Sequence Number is used.	
Track 1 Data	Required. This field is filled in if Sequence Number is used.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Track 2 Data	Required. This field is filled in if Sequence Number is used.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Original Transaction Amount	Required. This field is filled in if Sequence Number is used.	
Amount	Required.	
Card Type	Required. This field is filled in if Sequence Number is used.	
Sequence Number	Not required. Highly recommended.	
Approval Code	Required. This field is filled in if Sequence Number is used.	
Transaction Identifier	Required. This field is filled in if Sequence Number is used.	
Transaction Date	Required. This field is filled in if Sequence Number is used.	
Transaction Time	Required. This field is filled in if Sequence Number is used.	
Validation Code	Required. This field is filled in if Sequence Number is used.	

Table 31 Reversal input requirements

Field	Point of sale field requirement	Description
Response Indicator	Required. This field is filled in if Sequence Number is used.	
Returned ACI	Required. This field is filled in if Sequence Number is used.	
Requested ACI	Required. This field is filled in if Sequence Number is used.	
POS Mode Code	Required. This field is filled in if Sequence Number is used.	
Market Specific Indicator	Required. This field is filled in if Sequence Number is used.	
Retrieval Reference Number	Required. This field is filled in if Sequence Number is used.	
Account Data Source	Not required. This field is filled in if Sequence Number is used.	
Card Holder ID	Not required. This field is filled in if Sequence Number is used.	
Authorization Source Code	Not required. This field is filled in if Sequence Number is used.	
Order Number	Not required.	
Address Field	Not required.	
Zip Code	Not required.	
Card Present Flag	Required. This field is filled in if Sequence Number is used.	
Customer Present Flag	Required. This field is filled in if Sequence Number is used.	
Terminal Type	Required. This field is filled in if Sequence Number is used.	
Terminal Capability	Required. This field is filled in if Sequence Number is used.	
POS Entry Mode	Required. This field is filled in if Sequence Number is used.	

Table 31 Reversal input requirements

Field	Point of sale field requirement	Description
Ecommerce Type	Required for web-based transactions. This field is filled in if Sequence Number is used.	
Purchase Card Order Number	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
Tax Amount	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
Commercial Card Type	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
Ship To Zip Code	Required for purchasing cards only. This field is filled in if Sequence Number is used.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	
User Source Name	Not required.	
Processor AVS Result	Required. This field is filled in if Sequence Number is used.	
Processor Authorization Response Code	Required. This field is filled in if Sequence Number is used.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Table 31 Reversal input requirements

Field	Point of sale field requirement	Description
Ship to address 1	Not required.	
Ship to address 2	Not required.	
Ship to city	Not required.	
Ship to state	Not required.	
Ship to phone	Not required.	
Customer IP address	Not required.	
Customer email	Not required.	

Output

Table 32 Reversal output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Authorization Response Code	Refer to the API fields section in this guide for more information.
Authorization Response Message	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Identifier	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.

Table 32 Reversal output fields

Field	Description
Transaction Time	Refer to the API fields section in this guide for more information.
Validation Code	Refer to the API fields section in this guide for more information.
Original Transaction Amount	Refer to the API fields section in this guide for more information.
Response Indicator	Refer to the API fields section in this guide for more information.
Returned ACI	Refer to the API fields section in this guide for more information.
Requested ACI	Refer to the API fields section in this guide for more information.
POS Mode Code	Refer to the API fields section in this guide for more information.
Market Specific Indicator	Refer to the API fields section in this guide for more information.
Retrieval Reference Number	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Card Present Flag	Refer to the API fields section in this guide for more information.
Terminal Capability	Refer to the API fields section in this guide for more information.
Terminal Type	Refer to the API fields section in this guide for more information.
POS Entry Mode	Refer to the API fields section in this guide for more information.
Customer Present Flag	Refer to the API fields section in this guide for more information.
Ecommerce Type	Refer to the API fields section in this guide for more information.
Purchase Card Order Number	Refer to the API fields section in this guide for more information.
Tax Amount	Refer to the API fields section in this guide for more information.
Ship To Zip Code	Refer to the API fields section in this guide for more information.
Commercial Card Type	Refer to the API fields section in this guide for more information.

Table 32 Reversal output fields

Field	Description
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Processor AVS Result	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.

Return

Input

Table 33 Return input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Account Number	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Expiration Date	Required.	
Track 1 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Track 2 Data	Required.	Either Account Number and Expiration Date, or Track 1 Data, or Track 2 Data is required.
Amount	Required.	

Table 33 Return input requirements

Field	Point of sale field requirement	Description
Card Type	Required.	
Card Present Flag	Not required.	
Customer Present Flag	Not required.	
Terminal Type	Not required.	
Terminal Capability	Not required.	
POS Entry Mode	Not required.	
Purchase Card Order Number	Required for purchasing cards only.	
Tax Amount	Required for purchasing cards only.	
Commercial Card Type	Required for purchasing cards only.	
Ship To Zip Code	Required for purchasing cards only.	
Ecommerce Type	Required for web-based transactions.	
Merchant Billing Name	Not required.	
Merchant Billing State	Not required.	
Merchant Billing Location	Not required.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	

Table 33 Return input requirements

Field	Point of sale field requirement	Description
User Source Name	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Ship to address 1	Not required.	
Ship to address 2	Not required.	
Ship to city	Not required.	
Ship to state	Not required.	
Ship to phone	Not required.	
Customer IP address	Not required.	
Customer email	Not required.	

Output

Table 34 Return output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.

Table 34 Return output fields

Field	Description
Transaction Time	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Track 1 Data	Refer to the API fields section in this guide for more information.
Track 2 Data	Refer to the API fields section in this guide for more information.
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Processor AVS Result	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.

Void

Input

Table 35 Void input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Sequence Number	Required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 36 Void output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Session ID	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.

Manual Authorization

Input

Table 37 Manual Authorization input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Sequence Number	Required.	
Approval Code	Required.	
Session ID	Required if transaction security it enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 38 Manual Authorization output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.

Lookup

Input

Table 39 Lookup input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Sequence Number	Required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 40 Lookup output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Ecommerce Type	Refer to the API fields section in this guide for more information.
Account Number	Refer to the API fields section in this guide for more information.
Expiration Date	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Original Transaction Amount	Refer to the API fields section in this guide for more information.
Purchase Card Order Number	Refer to the API fields section in this guide for more information.
Card Type	Refer to the API fields section in this guide for more information.
Tax Amount	Refer to the API fields section in this guide for more information.
Commercial Card Type	Refer to the API fields section in this guide for more information.
Sequence Number	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Authorization Response Code	Refer to the API fields section in this guide for more information.
Authorization Response Message	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.
Transaction Identifier	Refer to the API fields section in this guide for more information.
Transaction Date	Refer to the API fields section in this guide for more information.
Transaction Time	Refer to the API fields section in this guide for more information.
Validation Code	Refer to the API fields section in this guide for more information.

Table 40 Lookup output fields

Field	Description
Response Indicator	Refer to the API fields section in this guide for more information.
Returned ACI	Refer to the API fields section in this guide for more information.
Requested ACI	Refer to the API fields section in this guide for more information.
POS Mode Code	Refer to the API fields section in this guide for more information.
Market Specific Indicator	Refer to the API fields section in this guide for more information.
Retrieval Reference Number	Refer to the API fields section in this guide for more information.
Address Match	Refer to the API fields section in this guide for more information.
Zip Match	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Card Present Flag	Refer to the API fields section in this guide for more information.
Customer Present Flag	Refer to the API fields section in this guide for more information.
Terminal Capability	Refer to the API fields section in this guide for more information.
Terminal Type	Refer to the API fields section in this guide for more information.
POS Entry Mode	Refer to the API fields section in this guide for more information.
Ship To Zip Code	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
User Source Name	Refer to the API fields section in this guide for more information.
Track 1 Data	Refer to the API fields section in this guide for more information.

Table 40 Lookup output fields

Field	Description
Track 2 Data	Refer to the API fields section in this guide for more information.
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Processor Authorization Response Code	Refer to the API fields section in this guide for more information.
Address Field	Refer to the API fields section in this guide for more information.
Zip Code	Refer to the API fields section in this guide for more information.

ACH Verify

Input

Table 41 ACH Verify input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Bank Account Number	Required.	
Bank ID	Required.	
Order Number	Not required.	
Amount	Required. For Paymentech this must be 0.	
Account Type	Required.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	

Table 41 ACH Verify input requirements

Field	Point of sale field requirement	Description
Sequence Number	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Customer Name	Not required.	
Merchant Billing Name	Not required.	
Merchant Billing Location	Not required.	

Output

Table 42 ACH Verify output requirements

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Bank Account Number	Refer to the API fields section in this guide for more information.
Bank ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Account Type	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.

Table 42 ACH Verify output requirements

Field	Description
Verification Result	Processor code detailing the response to the verification
Sequence Number	Sequence Number associated with this transaction.
Server ID	Refer to the API fields section in this guide for more information.
Transaction Code	Refer to the API fields section in this guide for more information.
LCC Return Message	Refer to the API fields section in this guide for more information.
Batch ID	Refer to the API fields section in this guide for more information.
Draft ID	Refer to the API fields section in this guide for more information.
Transaction Date and Time	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Processor Response Code	Refer to the API fields section in this guide for more information.
Processor Response Message	Refer to the API fields section in this guide for more information.
Customer Name	Refer to the API fields section in this guide for more information.
Bad Field Code	Refer to the API fields section in this guide for more information.
Bad Field Data	Refer to the API fields section in this guide for more information.
Merchant Billing Name	Refer to the API fields section in this guide for more information.
Merchant Billing Location	Refer to the API fields section in this guide for more information.

ACH Deposit

Input

Table 43 ACH Deposit input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Bank Account Number	Required.	
Bank ID	Required.	
Order Number	Optional.	
Amount	Required.	
Account Type	Required.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	
User Defined 4	Not required.	
User Defined 5	Not required.	
Customer Name	Not required.	
Sequence Number	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Merchant Billing Name	Not required.	
Merchant Billing Location	Not required.	

Output

Table 44 ACH Deposit output requirements

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Bank Account Number	Refer to the API fields section in this guide for more information.
Bank ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Account Type	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
Verification Result	Processor code detailing the response to the verification
Sequence Number	Sequence Number associated with this transaction.
Server ID	Refer to the API fields section in this guide for more information.
Transaction Code	Refer to the API fields section in this guide for more information.
LCC Return Message	Refer to the API fields section in this guide for more information.
Batch ID	Refer to the API fields section in this guide for more information.
Draft ID	Refer to the API fields section in this guide for more information.
Transaction Date and Time	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.

Table 44 ACH Deposit output requirements

Field	Description
Processor Response Code	Refer to the API fields section in this guide for more information.
Processor Response Message	Refer to the API fields section in this guide for more information.
Bad Field Code	Refer to the API fields section in this guide for more information.
Bad Field Data	Refer to the API fields section in this guide for more information.
Customer Name	Refer to the API fields section in this guide for more information.
Merchant Billing Name	Refer to the API fields section in this guide for more information.
Merchant Billing Location	Refer to the API fields section in this guide for more information.

ACH Refund

Input

Table 45 ACH Refund input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Bank Account Number	Required.	
Bank ID	Required.	
Order Number	Not required.	
Amount	Required.	
Account Type	Required.	
User Sequence Number	Not required.	
User Defined 1	Not required.	
User Defined 2	Not required.	
User Defined 3	Not required.	

Table 45 ACH Refund input requirements

Field	Point of sale field requirement	Description
User Defined 4	Not required.	
User Defined 5	Not required.	
Customer Name	Required.	
Sequence Number	Not required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.
Merchant Billing Name	Not required.	
Merchant Billing Location	Not required.	

Output

Table 46 ACH Refund output requirements

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Bank Account Number	Refer to the API fields section in this guide for more information.
Bank ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Account Type	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.

Table 46 ACH Refund output requirements

Field	Description
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
Verification Result	Processor code detailing the response to the verification
Sequence Number	Sequence Number associated with this transaction.
Server ID	Refer to the API fields section in this guide for more information.
Transaction Code	Refer to the API fields section in this guide for more information.
LCC Return Message	Refer to the API fields section in this guide for more information.
Batch ID	Refer to the API fields section in this guide for more information.
Draft ID	Refer to the API fields section in this guide for more information.
Transaction Date and Time	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Processor Response Code	Refer to the API fields section in this guide for more information.
Processor Response Message	Refer to the API fields section in this guide for more information.
Bad Field Code	Refer to the API fields section in this guide for more information.
Bad Field Data	Refer to the API fields section in this guide for more information.
Customer Name	Refer to the API fields section in this guide for more information.
Merchant Billing Name	Refer to the API fields section in this guide for more information.
Merchant Billing Location	Refer to the API fields section in this guide for more information.

ACH Void

Input

Table 47 ACH Void input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Sequence Number	Required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 48 ACH Void output requirements

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Bank Account Number	Refer to the API fields section in this guide for more information.
Bank ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Account Type	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
Verification Result	Processor code detailing the response to the verification

Table 48 ACH Void output requirements

Field	Description
Sequence Number	Sequence Number associated with this transaction.
Server ID	Refer to the API fields section in this guide for more information.
Transaction Code	Refer to the API fields section in this guide for more information.
LCC Return Message	Refer to the API fields section in this guide for more information.
Batch ID	Refer to the API fields section in this guide for more information.
Draft ID	Refer to the API fields section in this guide for more information.
Transaction Date and Time	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Processor Response Code	Refer to the API fields section in this guide for more information.
Processor Response Message	Refer to the API fields section in this guide for more information.
Bad Field Code	Refer to the API fields section in this guide for more information.
Bad Field Data	Refer to the API fields section in this guide for more information.
Customer Name	Refer to the API fields section in this guide for more information.
Merchant Billing Name	Refer to the API fields section in this guide for more information.
Merchant Billing Location	Refer to the API fields section in this guide for more information.

ACH Lookup

Input

Table 49 ACH Lookup input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Sequence Number	Required.	
Session ID	Required if transaction security is enabled.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 50 ACH Lookup output requirements

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Bank Account Number	Refer to the API fields section in this guide for more information.
Bank ID	Refer to the API fields section in this guide for more information.
Order Number	Refer to the API fields section in this guide for more information.
Amount	Refer to the API fields section in this guide for more information.
Account Type	Refer to the API fields section in this guide for more information.
User Sequence Number	Refer to the API fields section in this guide for more information.
User Defined 1	Refer to the API fields section in this guide for more information.
User Defined 2	Refer to the API fields section in this guide for more information.
User Defined 3	Refer to the API fields section in this guide for more information.
User Defined 4	Refer to the API fields section in this guide for more information.
User Defined 5	Refer to the API fields section in this guide for more information.
Verification Result	Processor code detailing the response to the verification

Table 50 ACH Lookup output requirements

Field	Description
Sequence Number	Sequence Number associated with this transaction.
Server ID	Refer to the API fields section in this guide for more information.
Transaction Code	Refer to the API fields section in this guide for more information.
LCC Return Message	Refer to the API fields section in this guide for more information.
Batch ID	Refer to the API fields section in this guide for more information.
Draft ID	Refer to the API fields section in this guide for more information.
Transaction Date and Time	Refer to the API fields section in this guide for more information.
Approval Code	Refer to the API fields section in this guide for more information.
Processor Response Code	Refer to the API fields section in this guide for more information.
Processor Response Message	Refer to the API fields section in this guide for more information.
Bad Field Code	Refer to the API fields section in this guide for more information.
Bad Field Data	Refer to the API fields section in this guide for more information.
Customer Name	Refer to the API fields section in this guide for more information.
Merchant Billing Name	Refer to the API fields section in this guide for more information.
Merchant Billing Location	Refer to the API fields section in this guide for more information.

Begin Session

Input

Table 51 Begin Session input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Session User Name	Required.	
Session Password	Required.	

Output

Table 52 Begin Session output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Session ID	Refer to the API fields section in this guide for more information.
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.

End Session

Input

Table 53 End Session input requirements

Field	Point of sale field requirement	Description
Merchant Name	Required.	
Session User Name	Not required.	
Session Password	Not required.	
Session ID	Required.	Set the session identifier with the **Set Session ID** function in the CPM API. Refer to the appropriate language in the Environment and Implementation section in this guide for more information.

Output

Table 54 End Session output fields

Field	Description
Merchant Name	Refer to the API fields section in this guide for more information.
Session User Name	Refer to the API fields section in this guide for more information.
Session Password	Refer to the API fields section in this guide for more information.
Return Code Message	Refer to the API fields section in this guide for more information.

Output Overview

The developer can receive three result codes from a transaction. The first is the Transaction Response code from the API call itself. This code details the success or failure of the completion of the function. The second return code is the Authorization Response code. The Authorization Response code tells the merchant if the authorization was approved or denied. This code is only received for authorization, reversal, and authorize and capture functions. The final code is the Address Verification Response code. This code details the results of the Address Verification performed with an authorization.

Transaction Response code

The Transaction Response code is the first value to examine after the completion of a transaction. The **LCC_RunTransaction** function returns the transaction response code. A **0**-response code means that the transaction completed successfully. Any non-**0** value means that an error occurred during processing.

Note A **0**-response does not mean that an authorization was approved. This response indicates only that the transaction was successfully communicated from the client, through the CPM Server, to the credit card issuing bank processors, and back.

Authorization Response code

For authorization, reversal, and authorization and capture functions, the second field to examine is the Authorization Response code. The Authorization Response code details the success or failure of an authorization. The Authorization Response contains two fields. The first field is the Authorization Result code field. This field contains the processor independent results of the transaction.

Table 55 Authorization Responses code

Code	Definition	Description
A	Approval	Authorization is approved.
C	Call	Voice authorization is required. Call the processor.
D	Decline	Authorization was declined. If desired, check the Authorization Response Message or Processor Authorization Response code for details.
P	Pick Up Card	A problem exists with this credit card. Remove the card from the cardholder. If desired, check the Authorization Response Message or Processor Authorization Response code for details.

Table 55 Authorization Responses code

Code	Definition	Description
X	Expired Card	This credit card is expired.
E	Error	A processing error has occurred. Check the Authorization Response Message or Processor Authorization Response code for details.

Note Examine the Authorization Result code field only for authorization and reversal functions.

The second field is the Processor Authorization Result code. This field is the response code returned by the processor. The CPM Server provides this field to assist in possible transaction processing problems; however, do not place any dependence on the field. Writing code based on the results of this field makes changing processors without also having to alter the integration code difficult.

Address verification code

The address verification code fields tell the merchant the results of the address information and the billing address for the account check. The CPM Server fields, Address Match and Zip Match, tell the merchant if the fields match or not.

Note The results of the address verification DO NOT have an impact on the results of an authorization. If a credit card is valid but the address verification information is wrong, the authorization is still processed. The merchants must determine if they want to accept or reject the authorization based on the address verification results.

Chapter 3

Environment and Implementation

ActiveX (vb)

The CPM ActiveX control is an ATL COM (Component Object Model) object. The properties and methods are implemented through the IUknown and IDispatch interfaces. This feature allows integration with Visual Basic, Visual Basic Scripts, Visual C++, and J++ and distribution as described in the Microsoft DNA model.

These ActiveX functions allow the developer to create, execute, and examine transactions. Refer to Chapter 2, CPM Transaction API, in this guide for more information on the transaction type identifiers, field identifiers, and error identifiers.

Merchant Information

The ActiveX control provides three ways to specify merchant information.

- Configuration file
- Server port and IP address
- Registry

If a configuration file is not specified, the ActiveX API uses the Windows Registry.

Configuration file

You can use the default configuration file, lcc_client.cf in the same directory as the CPM ActiveX API lccx.dll, or create your own configuration file. If you use the default file, do not specify a configuration file path and name with the **SetConfigFile** function. To use your own configuration file, include the path and name.

Configuration file format

The format of the file is much like an .ini file. The sections are denoted by brackets ([ ]), and information underneath those sections are simple key=value pairs.

The client configuration file stores the Merchant Identifier, CPM Server TCP/IP Address, CPM Server TCP/IP Port, and SSL Encryption Scheme underneath each merchant name section.

```
[Merchant Name]
MerchantID=Merchant ID
Server Address=IP Address of CPM Server
Server Port=Port on which the CPM Server is listening
Encryption=Encryption Scheme (0 default, 1 SSL)
```

Example

```
[Demonstration Store]
MerchantID=demo
Server Address=localhost
Server Port=1530
Encryption=0
```

Server port and IP address

The API allows you to set the CPM Server TCP/IP port and TCP/IP address. Use these commands with the merchant object. Refer to the sample code in this section for more information.

Command detail

.MerchantName

This command sets the merchant’s name.

For example, `SampMerch.MerchantName=”Demo Store”`

.MerchantID

This command sets the merchant’s identification number.

For example, `SampMerch.MerchantID=”Demonstration Store”`

.ServerPort

This command sets the merchant’s server port.

For example, `SampMerch.ServerPort=1530`

.ServerIP

This command sets the merchant's server IP address.

For example, `SampMerch.ServerIP=123.45.67.89`

.Encryption

This command sets the merchant's mode of encryption.

For example, `SampMerch.Encryption=1`

.Registry

The API allows you to set merchant information in the Windows registry.

Function detail

.Load

This function loads all the merchants in the merchant list.

Inputs

None

Outputs

(char)	merchant information

.AddMerchant(merchant)

This function adds a new merchant to the list.

Inputs

(char)	merchant object to add

Outputs

None

.GetMerchantCount

This function retrieves the number of merchants in the list.

Inputs

None

Outputs

(long)	number of merchants in the list

.GetNextMerchant(bool restart, merchant)

This function loads the next merchant in the list.

Inputs

(bool)	start from beginning of store list if true; otherwise, next merchant
(char)	merchant object

Outputs

(char)	merchant information

.RemoveMerchant(merchant)

This function deletes a merchant from the list.

Inputs

(char)	merchant object to remove

Outputs

None

.UpdateMerchant(merchant)

This function adds a new merchant to the list.

Inputs

(char)	edited merchant information

Outputs

None

.Save

This function commits the changes made to the list.

Inputs

None

Outputs

None

Environment set up

Make sure you set the following environment settings. Refer to your compiler documentation for more information.

- Include lcc.bas
- Include ATLPayment 1.0 Type Library

Binding

You can bind operations in two ways: early binding and late binding. Early binding allows binding of the control to occur when you compile the API and is the preferred method. Late binding allows binding of the control to occur when the API executes.

Early binding To bind the interfaces in the control, include the following declarations in the .bas or .frm modules.

```
Public payment As LCCPayment
Public merchant As LCCMerchant
Public merchantinfo As LCCMerchatList
```

Once the declarations are included in the modules, you can explicitly call the modules as shown below.

```
Set merchant = New LCCMerchant
Set payment = New LCCPayment
Set merchantinfo = New LCCMerchantList
```

Properly clean up memory associated with the objects with the statements below.

```
If Not merchant Is Nothing Then Set merchant = Nothing
If Not payment Is Nothing Then Set payment = Nothing
If Not merchantinfo Is Nothing Then Set merchantinfo = Nothing
```

Late binding To bind the interfaces in the control include the following statements.

```
Set merchant = CreateObject(“LCC.Merchant”)
Set merchantinfo = CreateObject(“LCC.MerchantList”)
Set payment = CreateObject(“LCC.Payment”)
```

Properly clean up memory associated with the objects with the statements below.

```
If Not merchant Is Nothing Then Set merchant = Nothing
If Not payment Is Nothing Then Set payment = Nothing
If Not merchantinfo Is Nothing Then Set merchantinfo = Nothing
```

Function detail

This section describes the functions.

The object created when ActiveX API is called lcc.lcc.1.

.AddMerchant(pdispMerchant As object)

This function adds a new merchant to the list.

Inputs

(pdispmerchant)	merchant object to add

Outputs

None

.GetNextMerchant(Restart As long, objMerchant As object)

This function loads the next merchant in the list.

Inputs

(bool)	start from beginning of store list if true; otherwise, next merchant
(merchant)	merchant object

Outputs

(merchant)	merchant information

.RemoveMerchant(strMerchant as string)

This function deletes a merchant from the list.

Inputs

(string)	merchant object to remove

Outputs

None

.UpdateMerchant(objMerchant As Object)

This function adds a new merchant to the list.

Inputs

(pdispmerchant)	edited merchant information

Outputs

None

.SetConfigFile(strConfigFile As String)

This function sets the path of the merchant configuration file. The current directory and lcc_client.cf are the default settings.

Inputs

(string)	full path to the API configuration file

Output

(long)	0 if successful; an identifier corresponding to an error otherwise

.OpenConnection(strServer As String, nPort As Long, nEncryption As Long) As Long

This function establishes a persistent connection to the CPM Server that allows the transmission of multiple transactions over one connection. This function may increase processing time especially if you are using SSL encryption.

Inputs

(string)	network address of the connection socket
(long)	port of the CPM Server application (Usually 1530)
(long)	type of encryption; 0 is the default, 1 is for SSL

Output

(long)	connection handle is successful; an identifier corresponding to an error otherwise

.SetConnectionInformation(strServer As String, nPort As Long, nEncryption As Long)

This function sets the connection information for the specified transaction at run time. This function overrides the settings in the configuration file or the Windows Registry.

Inputs

(string)	network address of the connection socket
(long)	port of the CPM Server application
(long)	type of encryption; 0 is the default, 1 is for SSL

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

.SetConnectionHandle(hTransaction As Long)

This function sets a handle corresponding to the specified transaction. Other functions use this handle to manipulate the transaction.

Inputs

(long)	the handle of the transaction

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

.CloseConnection(hConnection As Long)

This function closes a connection to the CPM Server that was opened with the OpenConnection function.

Inputs

(long)	connection handle of an existing connection

Output

(long)	0 if successful; an identifier corresponding to an error otherwise

.GetSessionId

This function retrieves the session ID for a transaction. The session ID is used when the CPM Server is configured to use security.

Inputs

None

Outputs

(long)	the session ID

.SetSessionId(SessionID As Long)

This function sets the session ID for a transaction. You must use this function when the CPM Server is configured to use security.

Inputs

(long)	the session ID

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

.SetValue(FieldId As Long, strValue As String)

This function sets the value of a field for a transaction.

Inputs

(long)	identifier of the field to set
(string)	the value to set the field to

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

.GetValue(FieldId As Long) As String

This function retrieves a field's value for a transaction.

Inputs

(long)	identifier of the field to set

Outputs

(string)	field value

.RunTransaction(TransactionId As Long) As Long

This function executes a transaction. Use **GetValue** to retrieve the returned fields.

Inputs

(long)	transaction identifier

Outputs

(int)	transaction value

Sample code

Early binding

```
Private Sub Form_Load()
Set LCCMerchant = CreateObject("LCC.Merchant")
Set LCCMerchantList = CreateObject("LCC.MerchantList")

lbMerchant.Clear

LCCMerchantList.Load
cMerchants = LCCMerchantList.GetMerchantCount
For x = 1 To cMerchants
    If (x = 1) Then
      LCCMerchantList.GetNextMerchant True, LCCMerchant
    Else
      LCCMerchantList.GetNextMerchant False, LCCMerchant
    End If
    lbMerchant.AddItem LCCMerchant.MerchantName
Next

End Sub
```

Late binding

```
Private Sub pbAuth_Click()
Set LCCPayment = CreateObject("LCC.Payment")

LCCPayment.SetValue ID_MERCHANT_NAME, lbMerchant.Text
LCCPayment.SetValue ID_ACCOUNT_NUMBER, ebAccount
LCCPayment.SetValue ID_EXPIRATION_DATE, ebExpire
LCCPayment.SetValue ID_AMOUNT, ebAmount

nResult = LCCPayment.RunTransaction(ID_AUTHORIZATION)
ebReturnCodeMsg = LCCPayment.GetValue(ID_RETURN_CODE_MESSAGE)

If (nResult = 0) Then
    ebSequence = LCCPayment.GetValue(ID_SEQUENCE_NUMBER)
    ebApproval = LCCPayment.GetValue(ID_APPROVAL_CODE)
    ebResponse = LCCPayment.GetValue(ID_AUTH_RESPONSE_MESSAGE)
End If
```

Batch API

The CPM Batch Processor Interface is an *optional* component that takes files containing transactions from legacy transaction processing systems and sends each transaction to the CPM Server. A single line in the batch file of fixed formatted fields represents each transaction.

You can send the following transaction types to the CPM Server through the batch interface.

- Authorization
- Capture
- Authorize and Capture
- Line Item Detail
- Void
- Reversal
- Return
- Begin Session
- End Session

You can send several different transaction types to the CPM Server through the batch interface. Each line of the batch file represents a single transaction. Three configuration files define the fields for the transaction types. The input files need only the fields required for the credit card function. The output files supply all output information as well as a copy of the input fields.

Configuration files

LCC_BATCH.CFG file

The lcc_batch.cfg file defines the input and output fields for each transaction record. The transactions are denoted by brackets ([]), and information underneath the transactions are simple key=value lists.

Note If you are processing Purchasing Card Level III transactions, enter the line item record prior to the transaction record.

The Begin Session and End Session transactions do not have outputs.

LCC_BATCH_IDS.CFG file

The lcc_batch_ids.cfg file pairs the field names defined in the lcc_batch.cfg file with API field identifiers. The lcc_batch_ids.cfg file defines what API fields can be used by the Batch API.

Note Each field listed in the lcc_batch.cfg file must have an API field identifier defined in the lcc_batch_ids.cfg file.

LCC_BATCH_LENGTHS.CFG file

The lcc_batch_lengths.cfg file allows you to define the length of each input field. If a field length is set to an amount greater than that allowed by the CPM API, the CPM Server generates the error 102 ERR_FIELD_TOO_LONG and places the transaction in the .err file.

Note Each field listed in the lcc_batch.cfg file must have a length defined in the lcc_batch_lengths.cfg file.

Input file description

The input file consists of fixed format records, one record per line, and one transaction per record. The first three characters of each line hold a number that identifies the transaction type. If a field is unused, fill that field with spaces. Left justify all fields and fill any unused portion with spaces. The carriage return is used as the delimiter between records.

All amount fields formatted in the CPM API are 12 characters in length with a format of *DDDDDDDDDDCC*. Do not include any formatting characters. For example, enter $25.67 as 2567 followed by 8 spaces.

If CPM Server security is enabled, send a Begin Session transaction to log into the CPM Server and obtain a session identifier. Send an End Session transaction to log out of the CPM Server.

Implementing Purchase Card level III usage

If you are processing Purchasing Card Level III transactions, enter the line item information (transaction type identifier 999) immediately preceding the transaction information.

LCC_BATCH_LAYOUT.TXT file

The lcc_batch_layout.txt file provides the record layout for each transaction type defined in lcc_batch.cfg. The file lists the input and output fields for each transaction, the starting position of each field, and the field length.

To generate the lcc_batch_layout.txt file,

1. At the command prompt, change to the CPM/lcc_batch subdirectory. For example:
   ```
   C:\>cd CPM\lcc_batch
   ```
2. Enter the following command.
   ```
   lcc_batch -layout
   ```

Run the batch input file

1. At the command prompt, change to the CPM/lcc_batch subdirectory. For example,
   ```
   C:\>cd CPM\lcc_batch
   ```
2. Enter the following command.
   ```
   lcc_batch <batch input file name>
   ```
   The batch processor begins processing the transactions. For each transaction a period (.) appears on the screen and three output files are generated.

Output files

Approval output file

The approval output file, *. app, lists all the transactions in the batch file that were approved.

Decline output file

The decline output, *.den, file lists all the transactions in the batch file that were not approved.

Error output file

The error output file, *.err, lists all the transactions in the batch file that were not processed because of errors.

Working with output files

If you intend to run another batch input file of the same name, all three output files will have the same name and will overwrite the output files from the previous session. Move the output files to a different directory or rename the *. app, *.den, and the *.err from the previous session. We suggest you establish naming conventions for your batch input and batch output files to maintain batch file organization.

Sample code

LCC_BATCH.CFG file

```
[Authorization]

Input=merchant_name, account_number, expiration_date, amount

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code


[Capture]

Input = merchant_name, sequence_number, amount

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code


[Auth_And_Capture]

Input = merchant_name, account_number, expiration_date, amount

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code


[Reversal]

Input = merchant_name, sequence_number, amount

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code


[Return]

Input = merchant_name, account_number, expiration_date, amount

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code


[Manual_Authorization]

Input = merchant_name, sequence_number, approval_code

Output= merchant_name, account_number, expiration_date, amount, sequence_number,
auth_response_code, auth_response_message, approval_code
```

[Void_Transaction]

Input = merchant_name, sequence_number

Output= merchant_name, account_number, expiration_date, amount, sequence_number, auth_response_code, auth_response_message, approval_code

[Lookup]

Input = merchant_name, sequence_number

Output= merchant_name, account_number, expiration_date, amount, sequence_number, auth_response_code, auth_response_message, approval_code

[Predial]

Input = merchant_name

Output= merchant_name, return_code_message

[Begin_Session]

Input = merchant_name, username, password

[End_Session]

Input = merchant_name

[Line_Item_Detail]

Input = item_description, item_product_code, item_quantity, item_total_amount

Output = item_description, item_product_code, item_total_amount

LCC_BATCH_IDS.CFG file

```
merchant_name=100
account_number=101
expiration_date=102
amount=103
card_type=104
sequence_number=105
approval_code=106
auth_response_code=107
auth_response_message=108
return_code_message=109
transaction_id=120
transaction_date=121
transaction_time=122
validation_code=123
original_amount=124
response_indicator=125
returned_aci=126
requested_aci=127
pos_mode_code=128
market_specific_indicator=129
retrieval_reference_number=130
account_data_source=131
card_holder_id=132
authorization_source_code=133
current_amount=134
trans_attribute=135
current_tax_amount=136
address_match=140
zip_match=141
order_number=150
customer_street=151
customer_zip=152
purchase_card_order_number=155
tax_amount=156
```

```
commercial_card_type=157
ship_to_zip_code=158
cvv=165
cvv_result=166
customer_name=170
customer_phone=171
customer_city=172
customer_state=173
merchant_billing_name=190
merchant_billing_state=191
merchant_billing_location=192
user_sequence_number=200
user_defined_1=201
user_defined_2=202
user_defined_3=203
user_defined_4=204
user_defined_5=205
user_source_name=296
reserved_1=222
reserved_2=223
track_1_data=250
track_2_data=251
e_commerce_type=300
card_present_flag=350
terminal_capability=351
terminal_type=352
pos_entry_mode=353
customer_present_flag=354
processor_avs_result=400
processor_auth_response_code=401
username=450
password=451
ship_to_address_1=460
ship_to_address_2=461
ship_to_city=462
```

```
ship_to_state=463
ship_to_phone=464
customer_ip_address=465
customer_email=466
fraud_reason_code=467
fraud_score=468
fraud_response_code=469
skip_fraud_check=470
freight_amount=500
duty_amount=501
ship_from_zip_code=503
discount_amount_applied=504
vat_tax_amount=505
vat_tax_rate=506
alternative_tax_id=507
alternative_tax_amount=508
line_item_detail_count=509
item_description=10000
item_product_code=11000
item_quantity=12000
item_unit_of_measure=13000
item_tax_amount=14000
item_tax_rate=15000
item_total_amount=16000
item_discount_amount=17000
item_commodity_code=18000
item_unit_cost=19000
item_discount_indicator=20000
item_tax_type_applied=21000
item_tax_applied=22000
item_tax_exempt=23000
```

LCC_BATCH_LENGTHS.CFG file

```
merchant_id=32
merchant_name=32
account_number=28
expiration_date=4
amount=12
card_type=4
sequence_number=15
approval_code=9
auth_response_code=1
auth_response_message=20
return_code_message=40
transaction_id=15
transaction_date=6
transaction_time=6
validation_code=4
original_amount=12
response_indicator=2
returned_aci=1
requested_aci=1
pos_mode_code=2
market_specific_indicator=2
retrieval_reference_number=12
account_data_source=1
card_holder_id=1
authorization_source_code=1
current_amount=12
trans_attribute=2
current_tax_amount=12
address_match=1
zip_match=1
order_number=25
customer_street=20
customer_zip=9
purchase_card_order_number=16
```

```
tax_amount=12
commercial_card_type=2
ship_to_zip_code=9
cvv=4
cvv_result=4
customer_name=26
customer_phone=14
customer_city=20
customer_state=2
merchant_billing_name=25
merchant_billing_state=2
merchant_billing_location=13
user_sequence_number=50
user_defined_1=50
user_defined_2=50
user_defined_3=50
user_defined_4=50
user_defined_5=50
user_source_name=31
reserved_1=50
reserved_2=50
track_1_data=76
track_2_data=37
e_commerce_type=2
card_present_flag=1
terminal_capability=1
terminal_type=1
pos_entry_mode=1
customer_present_flag=1
processor_avs_result=3
processor_auth_response_code=4
username=31
password=12
ship_to_address_1=20
ship_to_address_2=20
```

```
ship_to_city=20
ship_to_state=2
ship_to_phone=14
customer_ip_address=30
customer_email=50
fraud_reason_code=6
fraud_score=4
fraud_response_code=256
skip_fraud_check=1
freight_amount=20
duty_amount=20
ship_from_zip_code=9
discount_amount_applied=20
vat_tax_amount=20
vat_tax_rate=20
alternative_tax_id=20
alternative_tax_amount=20
line_item_detail_count=3
item_description=15
item_product_code=6
item_quantity=4
item_unit_of_measure=12
item_tax_amount=12
item_tax_rate=5
item_total_amount=12
item_discount_amount=12
item_commodity_code=12
item_unit_cost=12
item_discount_indicator=1
item_tax_type_applied=4
item_tax_applied=1
item_tax_exempt=1
```

LCC_BATCH_LAYOUT.TXT file

```
        AUTH_AND_CAPTURE
INPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12

OUTPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12
sequence_number                    80   15
auth_response_code                 95   1
auth_response_message              96   20
approval_code                      116  9


        AUTHORIZATION
INPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12

OUTPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12
sequence_number                    80   15
auth_response_code                 95   1
```

```
auth_response_message             96   20
approval_code                     116  9


        BEGIN_SESSION
INPUT:
transaction_type                  1    3
merchant_name                     4    32
username                          36   31
password                          67   12


        CAPTURE
INPUT:
transaction_type                  1    3
merchant_name                     4    32
sequence_number                   36   15
amount                            51   12

OUTPUT:
transaction_type                  1    3
merchant_name                     4    32
account_number                    36   28
expiration_date                   64   4
amount                            68   12
sequence_number                   80   15
auth_response_code                95   1
auth_response_message             96   20
approval_code                     116  9


        END_SESSION
INPUT:
transaction_type                  1    3
merchant_name                     4    32
```

```
        LINE_ITEM_DETAIL
INPUT:
transaction_type                   1    3
item_description                   4    35
item_product_code                  39   12
item_quantity                      51   12
item_total_amount                  63   12

OUTPUT:
transaction_type                   1    3
item_description                   4    35
item_product_code                  39   12
item_total_amount                  51   12


        LOOKUP
INPUT:
transaction_type                   1    3
merchant_name                      4    32
sequence_number                    36   15

OUTPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12
sequence_number                    80   15
auth_response_code                 95   1
auth_response_message              96   20
approval_code                      116  9
```

```
        MANUAL_AUTHORIZATION
INPUT:
transaction_type                   1    3
merchant_name                      4    32
sequence_number                    36   15
approval_code                      51   9

OUTPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12
sequence_number                    80   15
auth_response_code                 95   1
auth_response_message              96   20
approval_code                      116  9


        PREDIAL
INPUT:
transaction_type                   1    3
merchant_name                      4    32

OUTPUT:
transaction_type                   1    3
merchant_name                      4    32
return_code_message                36   40


        RETURN
INPUT:
transaction_type                   1    3
merchant_name                      4    32
account_number                     36   28
expiration_date                    64   4
amount                             68   12
```

```
OUTPUT:
transaction_type                  1    3
merchant_name                     4    32
account_number                    36   28
expiration_date                   64   4
amount                            68   12
sequence_number                   80   15
auth_response_code                95   1
auth_response_message             96   20
approval_code                     116  9


        REVERSAL
INPUT:
transaction_type                  1    3
merchant_name                     4    32
sequence_number                   36   15
amount                            51   12

OUTPUT:
transaction_type                  1    3
merchant_name                     4    32
account_number                    36   28
expiration_date                   64   4
amount                            68   12
sequence_number                   80   15
auth_response_code                95   1
auth_response_message             96   20
approval_code                     116  9


        VOID_TRANSACTION
INPUT:
transaction_type                  1    3
merchant_name                     4    32
sequence_number                   36   15
```

```
OUTPUT:
transaction_type                  1    3
merchant_name                     4    32
account_number                    36   28
expiration_date                   64   4
amount                            68   12
sequence_number                   80   15
auth_response_code                95   1
auth_response_message             96   20
approval_code                     116  9
```

Sample Input file

```
150Demonstration Store             abmnan                         123est98
999Notebook       2510068875     4935516
999Pencil         2510118932     4560325
100Demonstration Store             60118811888888880101200000000203000000004513
101Demonstration Store             40128811888888880101100000000203000560004510
102Demonstration Store             40128811888888880101100000000203001110004511
103Demonstration Store             40128811888888880101300001100203000000004512
104Demonstration Store             40128811888888880101400001100203000000204516
105Demonstration Store             40128811888888880101100000000203000888004517
106Demonstration Store             40128811888888880101100001100203000000004512
120Demonstration Store             40128811888888880101100001100203000000004512
121Demonstration Store             40128811888888880101100001100203000000004512
151Demonstration Store
```

C API (AIX, Solaris, Win32 dll)

C AIX

The C AIX is complied for AIX V4.4.3. The C AIX consists of a transaction interface and a class. The transaction interface, lcc.h, contains the constants for the transaction type identifiers, field identifiers, and error identifiers. The class contains a set of functions that allows the developer to create, execute, and examine transactions. Refer to Chapter 2, CPM Transaction API, in this guide for more information on the transaction type identifiers, field identifiers, and error identifiers.

C Solaris

The C Solaris consists of a transaction interface and a class. The transaction interface, lcc.h, contains the constants for the transaction type identifiers, field identifiers, and error identifiers. Refer to Chapter 2, CPM Transaction API, in this guide for more information on the transaction type identifiers, field identifiers, and error identifiers. The class contains a set of functions that allows the developer to create, execute, and examine transactions.

C Win32 dll

The C Win32 dll consists of a transaction interface and a class. The transaction interface, lcc.dll, contains the constants for the transaction type identifiers, field identifiers, and error identifiers. Refer to Chapter 2, CPM Transaction API, in this guide for more information on the transaction type identifiers, field identifiers, and error identifiers. The class contains a set of functions that allows the developer to create, execute, and examine transactions.

Merchant Information

The C API allows you to enter merchant information using the configuration file.

Configuration file

You can use the default configuration file or create your own configuration file. To use the default file, do not specify a configuration file path and name with the **SetConfigFile** function. To use your own configuration file, include the path and name.

Configuration file format

The format of the file is much like an .ini file. The sections are denoted by brackets ([]), and information underneath those sections are simple key=value pairs.

The client configuration file stores the Merchant Identifier, Server Address, Server Port, and encryption scheme underneath each merchant name section.

```
[Merchant Name]
MerchantID=Merchant Id
Server Address=IP Address of CPM Server
Server Port=Port on which the CPM Server is listening
Encryption=Encryption Scheme (0 default, 1 SSL)
```

Example

```
[Demonstration Store]
MerchantId=demo
Server Address=localhost
Server Port=1530
Encryption=1
```

Environment set up

C AIX

Make sure you set the following environment settings. Please refer to your compiler documentation for more information.

- Include the lcc.h file
- Link to the lcc.aix.a or lcc.aix.so file

C Solaris

Make sure you set the following environment settings. Please refer to your compiler documentation for more information.

- Include the lcc.h file
- Link to the lcc.a or lcc.so file

C Win32 dll

Make sure you set the following environment settings. Please refer to your compiler documentation for more information.

- Include the lcc.h file
- Link to the lcc.lib file

Function detail

This section describes the C API class and functions.

int LCC_Startup()

The function initializes the CPM API. You must call this function before performing any other API calls.

Inputs

None

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_Shutdown()

This function deleted the transaction corresponding to the specified handle.

Inputs

(long)	the unique handle of the transaction to be destroyed

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

long LCC_SetConfigFile(const char *sConfigFileName)

This function sets the path of the merchant configuration file. The current directory and lcc.cf are the default settings.

Inputs

(char *)	full path to the API configuration file

Outputs

(long)	0 if successful; an identifier corresponding to an error otherwise

long LCC_OpenConnection(const char *sServerAddress, int nPort, int nEncryption)

This function establishes a persistent connection to the CPM Server that allows the transmission of multiple transactions over one connection. This function may increase processing time especially if you are using SSL encryption.

Inputs

(char *)	network address of the connection socket
(int)	port of the CPM Server application
(int)	type of encryption; 0 is the default, 1 is for SSL

Outputs

(long)	connection handle is successful; an identifier corresponding to an error otherwise

long LCC_CloseConnection(long hConnectionHandle)

This function closes a connection to the CPM Server that was opened with the OpenConnection function.

Inputs

(long)	connection handle of an existing connection

Outputs

(long)	0 if successful; an identifier corresponding to an error otherwise

int LCC_SetConnectionInformation(long hTransaction, const char *sServerAddress, int nPort, int nEncryption)

This function sets the connection information for the specified transaction at run time. This function overrides the settings in the configuration file or the Windows Registry.

Inputs

(long)	the handle of the transaction
(char *)	network address of the connection socket
(int)	port of the CPM Server application
(int)	type of encryption; 0 is the default, 1 is for SSL

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_SetConnectionHandle(long hTransaction, long hConnection)

This function sets a handle corresponding to the specified transaction. Other functions use this handle to manipulate the transaction.

Inputs

(long)	the handle of the transaction
(long)	the connection handle

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

long LCC_OpenTransaction()

This function opens a unique transaction handle. The transaction uses this handle for identification. Other function calls use this handle to manipulate the transaction.

Inputs

None

Outputs

(long)	a unique handle assigned to the new transaction

int LCC_CloseTransaction(long hTransaction)

This function closes the handle corresponding to the specified transaction.

Inputs

(long)	the handle of the transaction

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

long LCC_RunTransaction(long hTransaction, long nTransactionId)

This function sends a transaction to the CPM Server for execution.

Inputs

(long)	the handle of the transaction to be run
(long)	the identifier for the transaction type

Outputs

(long)	0 if successful; an identifier corresponding to an error otherwise

int LCC_SetSessionId(long hTransaction, long nSessionId)

This function sets the session ID for a transaction. You must use this function when the CPM Server is configured to use security.

Inputs

(long)	the handle of the transaction
(long)	the session ID

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_GetSessionId(long hTransaction, long* val)

This function retrieves the session ID for a transaction. The session ID is used when the CPM Server is configured to use security.

Inputs

(long)	the handle of the transaction
(long*)	the session ID

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_SetValue(long hTransaction, long nFieldId, const char* sValue)

This function sets the value of a field for a transaction.

Inputs

(long)	handle corresponding to the transaction
(long)	identifier of the field to set
(const char *)	the value to set the field to

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_GetValue(long hTransaction, long nFieldId, char* pValue, int cbValue)

This function retrieves a field's value for a transaction.

Inputs

(long)	handle corresponding to the transaction
(long)	identifier of the field to get
(char *)	buffer that the value will be copied to
(int)	length of the buffer

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

const char *LCC_GetValuePtr(long hTransaction, long nFieldId)

This function retrieves a pointer value to a field's value for a transaction.

Note This function does not work properly for all programming languages, such as Visual Basic.

Inputs

(long)	handle corresponding to the transaction
(long)	identifier of the field to get

Outputs

(const char *)	a pointer to the value if successful; an identifier corresponding to an error otherwise

long LCC_GetValueLength(long hTransaction, long nFieldId)

This function returns the length of a field's value for a transaction.

Inputs

(long)	handle corresponding to the transaction
(long)	identifier of the field length to get

Outputs

(long)	the length of the field's value; an identifier corresponding to an error otherwise

int LCC_ClearValues(long hTransaction)

This function removes all the values associated with an open transaction.

Inputs

(long)	handle corresponding to the transaction

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

int LCC_DumpValues(long hTransaction)

This function dumps all the values associated with the specified transaction into lcc_test.txt.

Inputs

(long)	handle corresponding to the transaction

Outputs

(int)	1 if successful; 0 or ERR_OPEN_DEBUG_FILE if unsuccessful

int LCC_PrintValues(long hTransaction)

This function prints all the field/value pairs set by **SetValue** or by the server after a **RunTransaction**.

Inputs

(long)	handle corresponding to the transaction

Outputs

(int)	1 if successful; 0 or ERR_OPEN_DEBUG_FILE if unsuccessful

Sample code

```
/*
* lcc_test.c
* Simple test program to demonstrate various features of the LCC.
*/

/* Include the necessary header file to utilize the API */
#include "lcc.h"

#include "stdio.h"
#include <time.h>

/* Define some defaults for the examples that bypass a config file */
#define TEST_SERVER_ADDRESS "0.0.0.0"
#define TEST_MERCHANT_NAME"Demonstration Store"
#define TEST_MERCHANT_ID"demo"
#define TEST_LOGIN"carl"
#define TEST_PASSWORD"torconi"


/*
  Demonstrates the ability to do multiple transactions over a single
  connection.  Also bypasses the use of a configuration file by using
  the MERCHANT_ID field as opposed to the MERCHANT_NAME field.
*/
long DoMultiTrans()
{
  long nRet = 0;/* Return Code*/
  long hConnection;/* Connection Handle*/
  long hTransaction;/* Transaction Handle*/

  int i = 0;/* Counter */

  /* Open the connection */
  hConnection = LCC_OpenConnection(TEST_SERVER_ADDRESS, 1530, 0 );
  if ( hConnection < 0 )
  {
   fprintf(stderr, "Failed to Open Connection.  Reason: %ld\n", hConnection );
   return -1;
  }

  /* Create a transaction */
  hTransaction = LCC_OpenTransaction();
  if ( hTransaction < 0 )
  {
   fprintf(stderr, "Failed to Create Transaction.  Reason: %ld\n", hTransaction );
   return -1;
  }

  /* Associate the Transaction with the connection */
  LCC_SetConnectionHandle( hTransaction, hConnection );
  /* Perform a couple of transactions over the same connection */
  for ( i = 0; i < 2; i++ )
  {
   /* Dummy up some values and do an authorization */
   LCC_SetValue( hTransaction, ID_MERCHANT_ID, TEST_MERCHANT_ID);
   LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
   LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
   LCC_SetValue( hTransaction, ID_AMOUNT, "423" );

   /* Run the transaction */
   nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );

   /* Print the fields that came back */
   LCC_PrintValues( hTransaction );

   /* If the response of the first transaction is good, then let’s do another. */
   /* Clear out all of the values in the Transaction first. */
   LCC_ClearValues( hTransaction );
  }

  /* Clean up resources we created */
  LCC_CloseTransaction( hTransaction );
  LCC_CloseConnection( hConnection );

  return 0;
}

/*
  Does a single transaction without using a configuration file.  The
  RunTransaction will handle connecting to the server.
*/
long DoTransWithoutConfig()
{
  long nRet = 0;/* Return Code */
  long hTransaction;/* Transaction Handle */

  /* Create a transaction */
  hTransaction = LCC_OpenTransaction();
  if ( hTransaction < 0 )
  {
   fprintf(stderr, "Failed to Create Transaction.  Reason: %ld\n", hTransaction );
   return -1;
  }

  /*
  *To bypass the use of the configuratio file, we must set the connection
  *information.
  */
  LCC_SetConnectionInformation(hTransaction, TEST_SERVER_ADDRESS, 1530, 1);

  /*
  *Also set the MERCHANT_ID as opposed to the MERCHANT_NAME to
  *bypass the configuration file
  */
  LCC_SetValue( hTransaction, ID_MERCHANT_ID, TEST_MERCHANT_ID);

  /* Set the rest of the fields */
  LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
  LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
  LCC_SetValue( hTransaction, ID_AMOUNT, "424" );

  /* Run the transaction */
  nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );

  /* Print the fields that came back */
  LCC_PrintValues( hTransaction );

  /* Clean up the resources we created */
  LCC_CloseTransaction( hTransaction );

  return nRet;
}

/*
  Perform a transaction with the config info coming from the default
  configuration file.
*/
long DoTransWithDefaultConfig()
{
  long nRet = 0;/* Return Code */
  long hTransaction;/* Transaction Handle */

  /* Create a transaction */
  hTransaction = LCC_OpenTransaction();
  if ( hTransaction < 0 )
  {
   fprintf(stderr, "Failed to Create Transaction.  Reason: %ld\n", hTransaction );
   return -1;
  }

  /*
  *We set the MERCHANT_NAME here (used for display purposes).
  *The merchant id, source address, port and encrpytion info
  *will get read from the config file when RunTransaction is called.
  */
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
  LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
  LCC_SetValue( hTransaction, ID_AMOUNT, "420" );

  /* Run the transaction */
  nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
  /* Print the fields that came back. */
  LCC_PrintValues( hTransaction );

  /* Clean up the resources we created. */
  LCC_CloseTransaction( hTransaction );

  return nRet;
}

/*
  Perform a transaction with a configuration file set from the code.
*/
long DoTransWithSetConfig()
{
  long nRet = 0;/* Return Code */
  long hTransaction;/* Transaction Handle */

  /* Create a transaction */
  hTransaction = LCC_OpenTransaction();
  if ( hTransaction < 0 )
  {
   fprintf(stderr, "Failed to Create Transaction.  Reason: %ld\n", hTransaction );
   return -1;
  }

  /*
  *To bypass the use of the configuration file, we must set the connection
  *information.
  */
  LCC_SetConfigFile("/etc/lcc_config.cf");

  /*
  *We set the MERCHANT_NAME here (used for display purposes).
  *The merchant id, source address, port and encrpytion info
  *will get read from the config file when RunTransaction is called.
  */
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
  LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
  LCC_SetValue( hTransaction, ID_AMOUNT, "424" );

  /* Run the transaction. */
  nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );

  /* Print the fields that came back. */
  LCC_PrintValues( hTransaction );

  /* Clean up the resources we created */
  LCC_CloseTransaction( hTransaction );

  return nRet;
}
```

```
void DoSleep(int iSec)
{
  long t=time(NULL)+iSec;
  while (time(NULL) < t);
}

void DoTestSecurity(void)
{
  long nRet = 0;/* Return Code*/
  long hConnection;/* Connection Handle*/
  long hTransaction;/* Transaction Handle*/
  long hSession;

  int i = 0;/* Counter */

  /* Open the connection. */
  hConnection = LCC_OpenConnection(TEST_SERVER_ADDRESS, 1530, 0 );
  if ( hConnection < 0 )
  {
   fprintf(stderr, "Failed to Open Connection.  Reason: %ld\n", hConnection );
  }

  /* Create a transaction. */
  hTransaction = LCC_OpenTransaction();
  if ( hTransaction < 0 )
  {
   fprintf(stderr, "Failed to Create Transaction.  Reason: %ld\n", hTransaction );
  }

  /* Associate the transaction with the connection. */
  LCC_SetConnectionHandle( hTransaction, hConnection );


  /* Run the transaction with out begin session. */
  LCC_ClearValues(hTransaction);
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
  LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
  LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
  nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
  printf("No begin session: 1002=%d\n", nRet);

  /* Begin a session as admin.*/
  LCC_ClearValues(hTransaction);
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetValue( hTransaction, ID_USERNAME, TEST_LOGIN );
  LCC_SetValue( hTransaction, ID_PASSWORD, TEST_PASSWORD );
  nRet = LCC_RunTransaction( hTransaction, ID_BEGIN_SESSION );
  LCC_GetSessionId(hTransaction, &hSession);
  printf("begin session: 0=%d %d\n", nRet, hSession);
```

```
/* Run an auth (should be ok). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "100000" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $1000.00: 0=%d\n", nRet);

/* Run an auth (should be ok). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "50001" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $500.01: 0=%d\n", nRet);

/* Run an auth (). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "100001" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $1000.01: 1008=%d\n", nRet);

/* Run a return (admins, no limit). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "100424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_RETURN );
printf("Return $1004.24: 0=%d\n", nRet);

/* Run a predial (not auth). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_PREDIAL );
printf("Predial: 1006=%d\n", nRet);

DoSleep(65); /* sleep about 1 minute */
/* Run an auth (should be ok). */
```

```
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24 (sleep 1 minute): 0=%d\n", nRet);

DoSleep(65); /* sleep about 1 minute */
DoSleep(65); /* sleep about 1 minute */

/* Run an auth (force a database lookup). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24 (sleep 2 mins): 0=%d\n", nRet);


DoSleep(65); /*sleep about 1 minute*/
DoSleep(65); /*sleep about 1 minute*/
DoSleep(65); /*sleep about 1 minute*/
DoSleep(65); /*sleep about 1 minute*/

/* Run an auth (fail - timeout). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24 (sleep 4 mins): 1003=%d\n", nRet);


/* End a session. */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_END_SESSION );
printf("end session: 0=%d\n", nRet);

/* Begin a session as joe user. */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_USERNAME, "joeuser" );
LCC_SetValue( hTransaction, ID_PASSWORD, "joeuser" );
nRet = LCC_RunTransaction( hTransaction, ID_BEGIN_SESSION );
```

```
LCC_GetSessionId(hTransaction, &hSession);
printf("begin session (joe user): 0=%d %d\n", nRet, hSession);


/* Run an auth (should be ok). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24: 0=%d\n", nRet);

/* Run an auth (should be ok). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "50001" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $500.01: 1008=%d\n", nRet);

/* Run an auth (). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "100424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $1004.24: 1008=%d\n", nRet);

/* Run a return (admins, no limit). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "100424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_RETURN );
printf("Return $1004.24: 1009=%d\n", nRet);

/* Run a return (admins, no limit). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "50001" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_RETURN );
```

```
printf("Return $500.01: 1009=%d\n", nRet);

/* Run a return (admins, no limit). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "50000" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_RETURN );
printf("Return $500.00: 0=%d\n", nRet);

/* Run a predial (not auth). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_PREDIAL );
printf("Predial: 1006=%d\n", nRet);

DoSleep(65); /*Sleep about 1 minute. */
/* Run an auth (should be ok). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24 (sleep 1 minute): 0=%d\n", nRet);

DoSleep(65); /*Sleep about 1 minute. */
DoSleep(65); /*Sleep about 1 minute. */

/* Run an auth (force a database lookup). */
LCC_ClearValues(hTransaction);
LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
LCC_SetSessionId(hTransaction, hSession);
nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
printf("Auth $4.24 (sleep 2 mins): 0=%d\n", nRet);


DoSleep(65); /*Sleep about 1 minute. */
DoSleep(65); /*Sleep about 1 minute. */
DoSleep(65); /*Sleep about 1 minute. */
DoSleep(65); /*Sleep about 1 minute. */
```

```
  /* Run an auth (fail - timeout). */
  LCC_ClearValues(hTransaction);
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetValue( hTransaction, ID_ACCOUNT_NUMBER, "4012881188888888" );
  LCC_SetValue( hTransaction, ID_EXPIRATION_DATE, "1299" );
  LCC_SetValue( hTransaction, ID_AMOUNT, "424" );
  LCC_SetSessionId(hTransaction, hSession);
  nRet = LCC_RunTransaction( hTransaction, ID_AUTHORIZATION );
  printf("Auth $4.24 (sleep 4 mins): 1003=%d\n", nRet);


  /* End a session. */
  LCC_ClearValues(hTransaction);
  LCC_SetValue( hTransaction, ID_MERCHANT_NAME, TEST_MERCHANT_NAME);
  LCC_SetSessionId(hTransaction, hSession);
  nRet = LCC_RunTransaction( hTransaction, ID_END_SESSION );
  printf("end session (joe user): 0=%d\n", nRet);




  /* Clean up the resources we created. */
  LCC_CloseTransaction( hTransaction );
  LCC_CloseConnection( hConnection );

}

/*
* Run a few different styles of test transactions.
*/
int main(int argc, char* argv[])
{
  /* Initialize the LCC */
  long nRet = LCC_Startup();
  if ( nRet != 0 )
  {
   fprintf(stderr, "Failed to Initialize LCC.  Reason: %ld\n", nRet );
   return -1;
  }

  DoMultiTrans();
  DoTransWithoutConfig();
  DoTransWithDefaultConfig();
  DoTransWithSetConfig();
  DoTestSecurity();

  /* Free up the resources used created by the LCC */
  LCC_Shutdown();

  return 0;
}
```

Java (api)

The Java API consists of a transaction interface and a class. The transaction interface, lcc.japi.lcc, contains the constants for the transaction type identifiers, field identifiers, and error identifiers. Refer to Chapter 2, CPM Transaction API, in this guide for more information on the transaction type identifiers, field identifiers, and error identifiers. The class contains a set of functions that allows the developer to create, execute, and examine transactions.

Merchant Information

The Java API provides two ways to enter merchant information.

- Configuration file
- Server port and IP address

Configuration file

You can use the default configuration file or create your own configuration file. To use the default file, do not specify a configuration file path and name with the **SetConfigFile** function. To use your own configuration file, include the path and name.

Configuration file format

The format of the file is much like an .ini file. The sections are denoted by brackets ([]), and information underneath those sections are simple key=value pairs.

The client configuration file stores the Merchant Identifier, Server Address, Server Port, and encryption scheme underneath each merchant name section.

Note The Java API sends SSL transactions only over port 1531.

```
[Merchant Name]
MerchantId=Merchant Id
Server Address=IP Address of CPM Server
Server Port=Port on which the CPM Server is listening
Encryption=Encryption Scheme (0 default, 1 SSL)
```

Example 1

```
[Demonstration Store]
MerchantId=demo
Server Address=localhost
Server Port=1530
Encryption=0
```

Example 2

```
[Demonstration Store]
MerchantId=demo
Server Address=localhost
Server Port=1531
Encryption=1
```

Server port and IP address

The Java API allows you to set merchant name, merchant ID, merchant server port, IP address, and encryption method using **SetValue** and **SetConnectionInformation**. For example, to set the demonstration store without SSL encryption, enter the following commands.

```
.SetValue(LCC.ID_MERCHANT_ID, “demo”);
.SetConnectionInformation(“123.45.67.89”, 1530, 0);
```

To set the demonstration store with SSL encryption, enter the following commands.

```
.SetValue(LCC.ID_MERCHANT_ID, “demo”);
.SetConnectionInformation(“123.45.67.89”, 1531, 1);
```

Environment set up

Make sure you set the following environment settings. Please refer to your compiler documentation for more information.

- Include the lcc.jar, jcert.jar, jnet.jar, and jsse.jar files in the class path. For example, `classpath=C:\LCC\class\lcc.jar;C:\LCC\class\jcert.jar;C:\LCC\class \jnet.jar;C:\LCC\class\jsse.jar`
- Import lcc.japi.* (include in source code)
- Call the variables to access the static ID's (for example, `LCC.ID_AUTHORIZATION`)

Function detail

This section describes the Java API class and functions.

LCCTransaction()

This function creates a new transaction.

Inputs

None

Outputs

None

SetSessionId(int nSessionID)

This function sets the session ID for a transaction. You must use this function when the CPM Server is configured to use security.

Inputs

(int)	the session ID

Outputs

None

OpenConnection(string sServerAddress, int nPort, int nEncryption)

This function establishes a persistent connection to the CPM Server that allows the transmission of multiple transactions over one connection. This function may increase processing time especially if you are using SSL encryption.

Inputs

(string)	network address of the connection socket
(int)	port of the CPM Server application, 1531 is for SSL
(int)	type of encryption; 0 is the default, 1 is for SSL

Output

(long)	an identifier corresponding to an error

SetConnectionInformation(string sServerAddress, int nPort, int nEncryption)

This function sets the connection information for the specified transaction at run time. This function overrides the settings in the configuration file or the Windows Registry.

Inputs

(string)	network address of the connection socket
(int)	port of the CPM Server application, 1531 is for SSL
(int)	type of encryption; 0 is the default, 1 is for SSL

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

GetSessionId()

This function retrieves the session ID for a transaction. The session ID is used when the CPM Server is configured to use security.

Inputs

None

Outputs

(int)	the session ID

SetValue(int nFieldId, String sValue)

This function sets the value of a field for a transaction.

Inputs

(int)	identifier of the field to set
(string)	the value to set the field to

Outputs

None

GetValue(int nFieldId)

This function retrieves a field's value for a transaction.

Inputs

(int)	identifier of the field of the value to return

Outputs

(int)	field value

GetValueLength(int nFieldId)

This function returns the length of a field's value for a transaction.

Inputs

(int)	identifier of the field length to get

Outputs

(int)	the length of the field's value; an identifier corresponding to an error otherwise

ClearValues()

This function clears the values for a transaction. The session identifier is not reset.

Inputs

None

Outputs

None

RunTransaction(int nTransactionId)

This function executes a transaction. Use **GetValue** to retrieve the returned fields.

Inputs

(int)	transaction identifier

Outputs

(int)	0 if successful; an identifier corresponding to an error otherwise

SetConfigFile(string sConfigFileName)

This function sets the path of the merchant configuration file. The current directory and lcc_client.cf are the default settings.

Inputs

(string)	path to the merchant configuration file

Outputs

None

CloseConnection()

This function closes a connection to the CPM Server that was opened with the OpenConnection function.

Inputs

None

Output

(long)	0 if successful; an identifier corresponding to an error otherwise

PrintFields()

This function prints all the field/value pairs set by **SetValue** or by the CPM Server after a **RunTransaction**.

Inputs

None

Outputs

None

Sample code

```
/**
 * LCCExample demonstrates has code to demonstrate most of the transactions
 * allowed to the payment server. Please look through the entire file to
 * get you familiar with what it does.
 *
 * * * * * * * * * * * * * * * * * * * * * * * *
 * Configuration Prior to running application.
 *
 * You need the following for this application to run.
 * 1) JDK 1.2.x  (Tested with JDK 1.2.2) http://java.sun.com/products/jdk/1.2/
 * 2) JSSE 1.0.x (Tested with JSSE 1.0.1) http://java.sun.com/products/jsse/
 *
 * Configuration Prior to running transactions:
 * -Add the jdk and jsse jar files to the classpath.
 *
 * The following classes must be added to your class path.
 * lcc.jar
 * sslplus3.1.5.jar
 * eccpresso_cfg.jar
 * eccpresso_ssl.jar
 * sslcrvs.jar
 * jcert.jar
 * jnet.jar
 * jsse.jar
 *
 * For example: if you have lcc.jar in C:\LCC\class, you must append
C:\LCC\class\lcc.jar to your CLASSPATH.
 *
 * Example:
C:\LCC\class\lcc.jar;C:\LCC\class\sslplus3.1.5.jar;C:\LCC\class\eccpresso_cfg.jar;
C:\LCC\class\eccpresso_ssl.jar;C:\LCC\class\sslcrvs.jar
 *C:\jsse1.0.1\lib\jcert.jar;C:\jsse1.0.1\lib\jnet.jar;C:\jsse1.0.1\lib\jsse.jar
 *
 * NOTE: You will need to look through each of the functions with this example to
meet your needs.
 * Example: modifying the card types, account number, merchant name, merchant id,
amount, etc.
 *
 * Here are most of the default configuration for this example:
 * MERCHANT_ID = "demo"
 * MERCHANT_NAME = "Demonstration Store"
 * ACCOUNT_NUMBER = "4012881188888888"
 *
 *
 * For Secure SSL Transactions the following must apply to the configuration file.
 * -Port Address is: 1531 (Note: This is the default port for ssl transactions
only.)
 * -Encryption is: 1
 *
 * For NON-Secure Transacations the following must apply to the configuration file.
 * -Port Address is: 1530 (Note: This is the default port for ssl transactions
only.)
 * -Encryption is: 0
 *
 * The above can either be specified in the configuration file or called directly.
See the following code for examples.
 *
 */


// IMPORTANT: Import the LCC Java API classes. Make sure lcc.jar is in your
classpath.
import lcc.japi.*;

class LCCExample
{

  public static void main(String args[])
  {

   LCCTransaction trans = CreateTransaction();



   //Comment in the next line if you are using "Require User Authentication" on the
payment server.
   //RequireUserAuthentication(trans);

   //////////////////////////////////////////
   //Choose the configuration (configuration file or manual set connection)
   //RunUsingConfigurationFile(trans);
   RunBypassingConfigurationFile(trans);
   //////////////////////////////////////////



   //////////////////////////////////////////
   //Do you need Purchasing Card information.
   //AddPCardInfo(trans, 15); //Add PCard Information to the transaction. (trans
object, num of records)
   //////////////////////////////////////////
   ////////////////////////////////////////////
   RunAuthorizationTransaction(trans);
   //RunAuthAndManualTransaction(trans);
   //RunManualTransaction(trans);
   //RunCaptureTransaction(trans);
   //RunReversalTransaction(trans);
   //RunLookupTransaction(1000);
   ////////////////////////////////////////////




   //Comment in to run multiple transaction of the same connection.
   //RunMultipleTransactionsOverSingleConnection(100);


   System.exit(0);
 }



 private static LCCTransaction CreateTransaction()
 {
   // Create an instance of a LCC Transaction
   LCCTransaction trans = new LCCTransaction();


   // Set the necessary fields for the transaction you intend to perform
   trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
   trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
   trans.SetValue(LCC.ID_EXPIRATION_DATE, "1014" );
   trans.SetValue(LCC.ID_AMOUNT, "1500" ); // $15.00
   trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
   trans.SetValue(LCC.ID_USER_DEFINED_1, "special 1");
   trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
   trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");

   return trans;
 }



 private static void RunTransaction( LCCTransaction trans )
```

```
 {
   int rCode = trans.RunTransaction(LCC.ID_AUTHORIZATION);

   System.out.println("\n\nAuthorization Returned with:");

   // Check the return status
   // It should be 0, if the transaction was performed successfully
   // A 0 return status does NOT mean the transaction was approved
   if ( rCode != 0 )
   {
    // If the return status != 0, get the return code message
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
   }
   else
   {
    // If the authorization was approved...
    //String sTemp = trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE);


    if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("A") )
    {
    // Show some of the fields returned by this transaction
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));

    // Show a complete list of field id/value pairs returned by the server for this
transaction...
    //trans.PrintFields();
    }
    else
    {
    System.out.println("Transaction Denied - " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
```

```
      }
     }

     //The values must be cleared after each transaction ran.
     trans.ClearValues();

   }


   /**
    * This function gets a session for the transaction. This is required when
requiring "User Authentication" on the payment server.
    */
   private static LCCTransaction RequireUserAuthentication(LCCTransaction trans)
   {
     int nCode = 0;
     int nSession = 0;

     //Required Fields to run a begin session transaction.
     trans.SetValue(LCC.ID_MERCHANT_ID, "demo");
     trans.SetValue(LCC.ID_USERNAME, "demo");
     trans.SetValue(LCC.ID_PASSWORD, "abc123");

     //Must set connection to run the "begin session" transaction.
     trans.SetConnectionInformation("localhost", 1530, 0);

     nCode = trans.RunTransaction(LCC.ID_BEGIN_SESSION);

     if (nCode != 0)
     {
      System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
     }
     else
     {
      nSession = trans.GetSessionId();
     }

     //Clear the transaction object since a transaction was ran.
     trans.ClearValues();

     //Set Session ID since we are using authentication.
     trans.SetSessionId(nSession);

     return trans;
```

```
  }



  /**
   * This function uses the configuration file (lcc_client.cf) to read the merchant
information.
   */
  private static LCCTransaction RunUsingConfigurationFile(LCCTransaction trans)
  {
    System.out.println("********** RunUsingConfigurationFile ***********");

    //Note calling ID_MERCHANT_NAME for retrieving the merchant from the
configuration file.
    trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store");

    return trans;

  }

  /**
   * This function bypasses the configuration by calling SetConnection Information.
   */
  private static LCCTransaction RunBypassingConfigurationFile(LCCTransaction trans)
  {
    int nSession = 0;

    System.out.println("********** RunBypassingConfigurationFile ***********");

    //Port 1530 for transactions over a non-encrypted connection.
    //Port 1531 for transactions over an encrypted SSL connection.
    trans.SetConnectionInformation("localhost", 1530, 0);

    return trans;

  }


  private static void RunMultipleTransactionsOverSingleConnection(int
nNumberOfTransactions)
  {
    System.out.println("********** RunMultipleTransactionsOverSingleConnection
***********");
```

```
  // Create an instance of a LCC Transaction
  LCCTransaction trans = new LCCTransaction();


  String sTemp = "";

  //Port 1530 for transactions over a non-encrypted connection.
  //Port 1531 for transactions over an encrypted SSL connection.
  int rCode = trans.OpenConnection("localhost", 1530, 0);
  if ( rCode == 0 )
  {
   for ( int j = 0; j < nNumberOfTransactions; j++)
   {


   trans.SetValue(LCC.ID_MERCHANT_ID, "demo");

   // Set the necessary fields for the transaction you intend to perform
   trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
   trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
   trans.SetValue(LCC.ID_EXPIRATION_DATE, "1200" );
   trans.SetValue(LCC.ID_AMOUNT, "2000" ); // $20.00
   trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
   trans.SetValue(LCC.ID_USER_DEFINED_1, "special 1");
   trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
   trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");

   AddPCardInfo(trans, 10); //Add purchasing card information to the transaction.

   RunTransaction(trans);

   }

  }
  else
  {
   System.out.println("Failed to connect to server localhost");
  }

  trans.CloseConnection();

}

/**
```

```
   * Runs a manual authorization transaction only.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   */
  private static void RunManualTransaction(LCCTransaction trans)
  {


   String sSequenceNumber = "";
   String sMerchantID = "";
   int rCode = -1;

   //This is hard coded since it is a manual transaction. Modify to fit your needs.
   trans.SetValue(LCC.ID_SEQUENCE_NUMBER, "000100000361635"); //
   trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store"); //Adding merchant 
name to set value initiates using the configuration file.
   trans.SetValue(LCC.ID_APPROVAL_CODE, "123456");

   rCode = trans.RunTransaction(LCC.ID_MANUAL_AUTHORIZATION);

   System.out.println("\n\nManual Authorization Returned with:" + rCode);

   if (rCode != 0)
   {
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
   }
   else
   {
    System.out.println("Return Code Message = "+ 
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " + 
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " + 
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " + 
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " + 
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " + 
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));
   }

  }
  /**
   * Runs a authorization and a manual authorization transaction.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   */
  private static void RunAuthAndManualTransaction(LCCTransaction trans)
  {

    // Set the necessary fields for the transaction you intend to perform
    trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
    trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
    trans.SetValue(LCC.ID_EXPIRATION_DATE, "1014" );
    trans.SetValue(LCC.ID_AMOUNT, "1500" ); // $15.00  //Must be between 100 and 200
for a call response.
    trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
    trans.SetValue(LCC.ID_USER_DEFINED_1, "special 1");
    trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
    trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");


    int rCode = trans.RunTransaction(LCC.ID_AUTHORIZATION);

    System.out.println("\n\nAuthorization Returned with:" + rCode);

    // Check the return status
    // It should be 0, if the transaction was performed successfully
    // A 0 return status does NOT mean the transaction was approved
    if ( rCode != 0 )
    {
     // If the return status != 0, get the return code message
     System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    }
    else
    {
     // If the authorization was approved...
     //String sTemp = trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE);


     if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("A") )
     {
     // Show some of the fields returned by this transaction
     System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
     System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
     System.out.println("Authorization Response Code = " +
 trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
     System.out.println("Authorization Response Code Message = " +
 trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
     System.out.println("Sequence Number = " +
 trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
     System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
     System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
     System.out.println("Processor Auth Response Code = " +
 trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
     System.out.println("Processor AVS Result = " +
 trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));

     // Show a complete list of field id/value pairs returned by the server for this
 transaction...
     //trans.PrintFields();
     }
     else if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("C") ) //Call
 Response.
     {
     String sSequenceNumber = "";
     String sMerchantID = "";

     sSequenceNumber = trans.GetValue(LCC.ID_SEQUENCE_NUMBER);
     sMerchantID = trans.GetValue(LCC.ID_MERCHANT_ID);

     trans.ClearValues(); //MUST clear values between transacations.


     trans.SetValue(LCC.ID_SEQUENCE_NUMBER, sSequenceNumber);
     trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store"); //Adding merchant
 name to set value initiates using the configuration file.
     trans.SetValue(LCC.ID_MERCHANT_ID, sMerchantID);
     trans.SetValue(LCC.ID_APPROVAL_CODE, "987654");

     rCode = trans.RunTransaction(LCC.ID_MANUAL_AUTHORIZATION);

     System.out.println("\n\nManual Authorization Returned with:" + rCode);

     if (rCode != 0)
     {
     System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
     }
     else
     {
```

```
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));
    }


    }
    else
    {
    System.out.println("Transaction Denied - " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    }
   }

   //The values must be cleared after each transaction ran.
   trans.ClearValues();



  }



  /**
   * Runs a authorization and a manual authorization transaction.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   */
  private static void RunAuthorizationTransaction(LCCTransaction trans)
  {

   // Set the necessary fields for the transaction you intend to perform
   //trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store");
   trans.SetValue(LCC.ID_MERCHANT_ID, "demo");

   trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
   trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
   trans.SetValue(LCC.ID_EXPIRATION_DATE, "1014" );
   trans.SetValue(LCC.ID_AMOUNT, "1500" ); // $15.00  //Must be between 100 and 200
for a call response.
   trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
   trans.SetValue(LCC.ID_USER_DEFINED_1, "special 1");
   trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
   trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");


   int rCode = trans.RunTransaction(LCC.ID_AUTHORIZATION);

   System.out.println("\n\nAuthorization Returned with:" + rCode);

   // Check the return status
   // It should be 0, if the transaction was performed successfully
   // A 0 return status does NOT mean the transaction was approved
   if ( rCode != 0 )
   {
    // If the return status != 0, get the return code message
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
   }
   else
   {
    // If the authorization was approved...
    //String sTemp = trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE);


    if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("A") )
    {
    // Show some of the fields returned by this transaction
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
```

```
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));

    // Show a complete list of field id/value pairs returned by the server for this
transaction...
    //trans.PrintFields();
    }
    else
    {
    System.out.println("Transaction Denied - " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    }
   }

   //The values must be cleared after each transaction ran.
   trans.ClearValues();


  }



  /**
   * Runs a reversal transaction. First is runs an authorization to get the sequence
number and then the reversal.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   */
  private static void RunReversalTransaction(LCCTransaction trans)
  {

   // Set the necessary fields for the transaction you intend to perform
   trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
   trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
   trans.SetValue(LCC.ID_EXPIRATION_DATE, "1014" );
   trans.SetValue(LCC.ID_AMOUNT, "15000" ); // $150.00  //Must be between 100 and
200 for a call response.
   trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
   trans.SetValue(LCC.ID_USER_DEFINED_1, "special 1");
   trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
   trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");


   int rCode = trans.RunTransaction(LCC.ID_AUTHORIZATION);
   System.out.println("\n\nAuthorization Returned with:" + rCode);

   // Check the return status
   // It should be 0, if the transaction was performed successfully
   // A 0 return status does NOT mean the transaction was approved
   if ( rCode != 0 )
   {
    // If the return status != 0, get the return code message
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
   }
   else
   {
    // If the authorization was approved...
    //String sTemp = trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE);


    if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("A") )
    {
    // Show some of the fields returned by this transaction
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));

    // Show a complete list of field id/value pairs returned by the server for this
transaction...
    //trans.PrintFields();
    }
    else if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("C") ) //Call
Response.
    {
    String sSequenceNumber = "";
    String sMerchantID = "";

    sSequenceNumber = trans.GetValue(LCC.ID_SEQUENCE_NUMBER);
    sMerchantID = trans.GetValue(LCC.ID_MERCHANT_ID);

    //trans.ClearValues(); //MUST clear values between transacations.

    trans.SetValue(LCC.ID_SEQUENCE_NUMBER, sSequenceNumber);
    trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store"); //Adding merchant
name to set value initiates using the configuration file.



    rCode = trans.RunTransaction(LCC.ID_REVERSAL);

    System.out.println("\n\nReversal Returned with:" + rCode);

    if (rCode != 0)
    {
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    }
    else
    {
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));
    }

    }
    else
    {
    System.out.println("Transaction Denied - " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    }
   }

   //The values must be cleared after each transaction ran.
```

```
    trans.ClearValues();


  }

  /**
   * Runs a authorization and a capture transaction. Note to be confused with a one
(auth and capture) transaction.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   */
  private static void RunCaptureTransaction(LCCTransaction trans)
  {

    // Set the necessary fields for the transaction you intend to perform
    trans.SetValue(LCC.ID_ACCOUNT_NUMBER, "4012881188888888");
    trans.SetValue(LCC.ID_CARD_TYPE, "001");  // Visa Card
    trans.SetValue(LCC.ID_EXPIRATION_DATE, "1014" );
    trans.SetValue(LCC.ID_AMOUNT, "15000" ); // $150.00  //Must be between 100 and
200 for a call response.
    trans.SetValue(LCC.ID_TAX_AMOUNT, "250" );  // $2.50
    trans.SetValue(LCC.ID_USER_DEFINED_1, "User Defined 1");
    trans.SetValue(LCC.ID_CUSTOMER_STREET, "100 Main St.");
    trans.SetValue(LCC.ID_CUSTOMER_ZIP, "90210");


    int rCode = trans.RunTransaction(LCC.ID_AUTHORIZATION);

    System.out.println("\n\nAuthorization Returned with:" + rCode);

    // Check the return status
    // It should be 0, if the transaction was performed successfully
    // A 0 return status does NOT mean the transaction was approved
    if ( rCode != 0 )
    {
     // If the return status != 0, get the return code message
     System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    }
    else
    {
     // If the authorization was approved...
     //String sTemp = trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE);


     if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("A") )
     {
     // Show some of the fields returned by this transaction
     System.out.println("Return Code Message = "+
 trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
     System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
     System.out.println("Authorization Response Code = " +
 trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
     System.out.println("Authorization Response Code Message = " +
 trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
     System.out.println("Sequence Number = " +
 trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
     System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
     System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
     System.out.println("Processor Auth Response Code = " +
 trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
     System.out.println("Processor AVS Result = " +
 trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));

     // Show a complete list of field id/value pairs returned by the server for this
 transaction...
     //trans.PrintFields();
     }
     else if ( trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE).equals("C") ) //Call
 Response.
     {
     String sSequenceNumber = "";
     String sMerchantID = "";

     sSequenceNumber = trans.GetValue(LCC.ID_SEQUENCE_NUMBER);
     sMerchantID = trans.GetValue(LCC.ID_MERCHANT_ID);

     trans.ClearValues(); //MUST clear values between transacations.



     trans.SetValue(LCC.ID_SEQUENCE_NUMBER, sSequenceNumber);
     trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store"); //Adding merchant
 name to set value initiates using the configuration file.
     trans.SetValue(LCC.ID_MERCHANT_ID, sMerchantID);
     trans.SetValue(LCC.ID_APPROVAL_CODE, "123456");

     rCode = trans.RunTransaction(LCC.ID_CAPTURE);

     System.out.println("\n\nCapture Returned with:" + rCode);

     if (rCode != 0)
     {
```

```
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    }
    else
    {
    System.out.println("Return Code Message = "+
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    System.out.println("Approval Code = " + trans.GetValue(LCC.ID_APPROVAL_CODE));
    System.out.println("Authorization Response Code = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    System.out.println("Authorization Response Code Message = " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_MESSAGE));
    System.out.println("Sequence Number = " +
trans.GetValue(LCC.ID_SEQUENCE_NUMBER));
    System.out.println("Address Match = " + trans.GetValue(LCC.ID_ADDRESS_MATCH));
    System.out.println("zip Match = " + trans.GetValue(LCC.ID_ZIP_MATCH));
    System.out.println("Processor Auth Response Code = " +
trans.GetValue(LCC.ID_PROCESSOR_AUTH_RESPONSE_CODE));
    System.out.println("Processor AVS Result = " +
trans.GetValue(LCC.ID_PROCESSOR_AVS_RESULT));
    }

    }
    else
    {
    System.out.println("Transaction Denied - " +
trans.GetValue(LCC.ID_AUTH_RESPONSE_CODE));
    }
   }

   //The values must be cleared after each transaction ran.
   trans.ClearValues();

  }


  /**
   * Runs a lookup transaction.
   *
   * @param nNumOfLookups.
   */
  private static void RunLookupTransaction(int nNumOfLookups)
  {

   for ( int i = 0; i < nNumOfLookups; i++ )
   {
    LCCTransaction trans = new LCCTransaction();

    //You must change this value.
    trans.SetValue(LCC.ID_SEQUENCE_NUMBER, "000100000383776");
    trans.SetValue(LCC.ID_MERCHANT_NAME, "Demonstration Store");


    int rCode = trans.RunTransaction(LCC.ID_LOOKUP);

    //System.out.println("\n\nLookup Returned with:");

    if (rCode != 0)
    {
    System.out.println(trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    }
    else
    {
    //System.out.println("Return Code Message = " +
trans.GetValue(LCC.ID_RETURN_CODE_MESSAGE));
    //System.out.println("Merchant ID = " + trans.GetValue(LCC.ID_MERCHANT_ID));
    //System.out.println("Merchant Name = " +
trans.GetValue(LCC.ID_MERCHANT_NAME));
    //System.out.println("Account Number = " +
trans.GetValue(LCC.ID_ACCOUNT_NUMBER));

    }

   }

  }



  /**
   * Adds the Purchasing Card Information to the transaction object before
processing.
   *
   * @param LCCTransaction trans. The transaction object from the calling function.
   * @param nNumOfLineItems The number of line items that you want to process.
  */
  private static void AddPCardInfo(LCCTransaction trans, int nNumOfLineItems)
  {

   Integer Intit = new Integer(0); //For Testing purposes only.
   String sTemp = ""; //For Testing purposes only.
   //Note the -1. We need the first record to start at 0. i++ will increment on
first loop.
   for ( int i = 0; i < nNumOfLineItems; i++ )
   {


    sTemp = Intit.toString(i);


    trans.SetValue(LCC.ID_ITEM_DESCRIPTION+i, "Test" + sTemp);
    trans.SetValue(LCC.ID_ITEM_PRODUCT_CODE+i, "1" + sTemp);
    trans.SetValue(LCC.ID_ITEM_QUANTITY+i, "556" + sTemp);
    trans.SetValue(LCC.ID_ITEM_UNIT_OF_MEASURE+i, "1234" + sTemp);
    trans.SetValue(LCC.ID_ITEM_TAX_AMOUNT+i, "558" + sTemp);
    trans.SetValue(LCC.ID_ITEM_TAX_RATE+i, "38" + sTemp);
    trans.SetValue(LCC.ID_ITEM_TOTAL_AMOUNT+i, "383" + sTemp);
    trans.SetValue(LCC.ID_ITEM_DISCOUNT_AMOUNT+i, "55");
    trans.SetValue(LCC.ID_ITEM_COMMODITY_CODE+i, "ISO383" + sTemp);
    trans.SetValue(LCC.ID_ITEM_UNIT_COST+i, "500" + sTemp);
    trans.SetValue(LCC.ID_ITEM_DISCOUNT_INDICATOR+i, "1");
    trans.SetValue(LCC.ID_ITEM_TAX_TYPE_APPLIED+i, "1" + sTemp);
    trans.SetValue(LCC.ID_ITEM_TAX_APPLIED+i, "1");
    trans.SetValue(LCC.ID_ITEM_TAX_EXEMPT+i, "0");

   }


  }

}
```

Chapter 4

CPM Server test mode emulation

Test Gateway

The Test Gateway returns specific Authorization Response Codes, Address Match Codes, and Zip Match Codes for specific input value ranges.

Approval codes

Table 56 Possible approval code return values

Input Amount	Code	Description
0 to 100.00	A	Approval
100.01 to 200.00	C	Call
200.01 to 300.00	D	Decline
300.01 to 400.00	P	Pick up card
400.01 to 500.00	X	Expired card
500.01 to 600.00	E	Error
>600.00		Generates random approval responses.

Address match

Table 57 Possible address match return values

Customer Street	Address Match	Description
0 to 100	Y	Match
101 to 200	N	No match
201 to 300	X	The service is unavailable
>300		Generates random address responses.

Zip code match

Table 58 Possible zip code match return values

Customer Zip	Zip Match	Description
00000-0000 to 10000-1000	Y	Match
10000-1001 to 20000-2000	N	No match
20000-2001 to 30000-3000	X	The service is unavailable
>30000-3001		Generates random zip code responses.

Restrictions

- Do not install two CPM Servers on the same computer.
- Do not use the CPM Server in test mode for load testing or statistic gathering.

Chapter 5

CPM Server Database Schema

The CyberSource Payment Manager (CPM) Database contains transaction information. As with any database, you should implement security precautions to prevent unauthorized access.

Refer to your database documentation for technical issues relating to the setup and use. The CPM Server uses the Open Database Connectivity (ODBC) standard which converts Structured Query Language (SQL) statements into the proprietary interface of your chosen database. This enables the CPM Server to work with many different databases such as Oracle and Microsoft Sequel Server.

Proper maintenance of the CPM database

Warning Only perform the Purge feature in CPM Database Utility on an active CPM database during your lowest period of transaction activity. To purge old transaction information that has been reported, set the Aging and Volume parameters in the Storage tab of the CPM Server properties. You must coordinate your reporting practices prior to transaction purging.

Database management recommendations

Good database management practices include monitoring database access activity, monitoring the number of records in the database, following a database backup schedule, following a purging old records schedule, and having a crisis/recovery plan in place.

Database Administrator We strongly suggest employing a database administrator for establishing and maintaining the database environment for CPM software.

Database sizing Take database sizing into account for your anticipated transaction volume when establishing a CPM database.

Monitor database activity Identify peak and low database access times based on your transaction throughput. This information is important for scheduling database maintenance activity.

Monitor database size The size of your database is limited by the amount of hard drive space available. This information is important for scheduling the purging of old records.

Follow a backup schedule Backup copies of a database reduce the amount of lost data when a database problem occurs.

Follow a schedule to purge old database records Old records in a database should be removed to prevent the database from becoming too large. The purging process requires many CPU cycles from the database server and should be performed when access activity is low. CPM recommends performing this procedure on the CPM database in such a way that no more than 10,000 records are purged at one time.

Crisis/recovery plan Hard drives fail, data becomes corrupt, and lightening strikes. Fortunately, these are rare events. But because they do happen, careful planning can reduce the amount of data lost and the time required to get the database back up and running.

CPM and SQL92 database compliancy

The relational database management system (RDBMS) you choose for the CPM database MUST be SQL92 compliant.

CPM API and CPM Server database amount fields

The CPM API limits the length of all amount fields to 12 characters.

The lengths of the amount fields in the CPM database are typically longer than the CPM API amount lengths. Do NOT alter any amount field in the CPM Server database.

CPM Database tables

CC_TRANSACTION table

Description This table stores all credit card transaction information

Usage Inserts a record for every credit card transaction that communicates with the processor, updated with every response from the processor for online transactions.

Indices

1 SEQUENCE_NUMBER (primary key)
2 BATCH_ID, DRAFT_ID
3 MERCHANT_ID
4 TRANS_DATE_TIME
5 ACCOUNT_NUMBER
6 ACCOUNT_EXTENSION

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
TRANSACTION_CODE	3 (alphanumeric)	Indicates transaction type with three letter code. **Code** / **Description** 100 Authorization 101 Reversal 102 Capture 103 Return 104 Authorize and Capture 105 Manual Authorization 106 Void Transaction
SERVER_ID	4 (alphanumeric)	Identifier representing the CPM Server that issued the transaction. If multiple CPM Servers use one database, this number must be unique to each Server.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
LPC_TYPE	4 (alphanumeric)	Type of Gateway used in the transaction.
MERCHANT_ID	32 (alphanumeric)	The MERCHANT_ID is a merchant identifier associated with each transaction. The identifier may contain alphabetical and numerical characters.
SEQUENCE_NUMBER	15 (alphanumeric)	Primary Key. Transaction Sequence Number unique to each transaction.
PARENT_SEQ_ NUMBER	15 (alphanumeric)	The Sequence Number of the transaction’s parent. For example, a capture transaction’s Parent Sequence Number is the corresponding authorization transaction’s Sequence Number.
LCC_RETURN_CODE	4 (alphanumeric)	Return code from the CPM API.
LCC_RETURN_MSG	40 (alphanumeric)	Text Description of the return code from the CPM Server.
TRANSACTION_ STATE	1 (alphanumeric)	Indicates transaction state. **Code** — **Transaction** O — Open V — Void C — Captured S — Settled (Reserved for future use) B — Failed to Settle
TRANS_DATE_TIME	Date Time (*MM/ DD/YY HHMMSS AM* or *PM*)	Time the transaction occurred.
LOCAL_DATE_TIME	Date Time (*MM/ DD/YY HHMMSS AM* or *PM*)	Time parameter in the API function. This parameter may be different than current time. For example, a direct marketing capture should contain the date the authorization occurred.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
AUTH_RESP_CODE	1 (alphanumeric)	Indicates authorization response. **Code** — **Description** A — Approval C — Call D — Decline P — Pick Up Card X — Expired Card E — Error
PROC_AUTH_RSP_ CODE	4 (alphanumeric)	Processor dependent authorization response code.
AUTH_RESPONSE_ MSG	20 (alphanumeric)	Processor dependent authorization response message.
APPROVAL_CODE	9 (alphanumeric)	Processor assigned approval code.
CVV_RESPONSE_ CODE	4 (alphanumeric)	Result of Card Verification Value (CVV) check
ACCOUNT_NUMBER	28 (alphanumeric)	Credit card number used in transaction.
ACCOUNT_ EXTENSION	80 (alphanumeric)	Encrypted credit card information.
EXPIRATION_DATE	4 Date (*MMYY*)	Month and year when credit card expires.
AUTH_CURRENCY	3 (alphanumeric)	Currency code used in authorization transaction.
SETTLE_CURRENCY	3 (alphanumeric)	Currency code used in settlement transaction.
CLEAR_CURRENCY	3 (alphanumeric)	Currency code used in clear transaction.
AUTH_COUNTRY	3 (alphanumeric)	Country code where authorization occurred.
SETTLE_COUNTRY	3 (alphanumeric)	Country code where settlement occurred.
CLEAR_COUNTRY	3 (alphanumeric)	Country code where clear occurred.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
AMOUNT	16 (numeric)	Amount for the transaction. For authorization, the amount to authorize. For returns, the amount to give back. For captures, the amount to settle. For reversals, the new amount to be authorized (lower than the current authorized amount). For incrementals, the new amount to be authorized and added to the original authorized amount. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
ORIGINAL_AMOUNT	16 (numeric)	The amount authorized in the first authorization. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
CURRENT_AMOUNT	16 (numeric)	The current amount authorized after any subsequent functions with a transaction. The original authorization's field is updated after any transactions acting on it. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
CURRENT_TAX_ AMOUNT	16 (numeric)	The current tax amount authorized after any subsequent function. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
ORDER_NUMBER	25 (alphanumeric)	Order Number assigned by the merchant.
MERCH_BILL_NAME	25 (alphanumeric)	Merchant's Name.
MERCH_BILL_LOC	13 (alphanumeric)	Merchant's Location (can be by city).
MERCH_BILL_STATE	2 (alphanumeric)	Merchant's location by state.
BATCH_ID	12 (alphanumeric)	Identification number given to batch by the CPM Server.
DRAFT_ID	8 (alphanumeric)	Identification number given to draft in batch by the CPM Server.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
CARD_TYPE	4 (alphanumeric) **Note** The API data input size is three characters.	Indicated card type. The corresponding API field, ID_CARD_TYPE accepts only three characters for field length. **Code** — **Credit Card** 001 — VISA 002 — MasterCard 003 — American Express 004 — Discover 005 — Diner's Club 006 — Carte Blanche 007 — Japanese Credit Bank 008 — Optima 009 — Switch 010 — GE Capital 011 — General Electric Credit Corporation (GECC) 012 — Beneficial 013 — CitiBank Encryption Program 022 — Liz Claiborne
CUSTOMER_NAME	26 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_PHONE	14 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_STREET	20 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_CITY	20 (alphanumeric)	Provided for storing additional customer information.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
CUSTOMER_STATE	2 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_ZIP	9 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_EMAIL	50 (alphanumeric)	Provided for storing additional customer information.
CUSTOMER_IP_ADDR	30 (alphanumeric)	Provided for storing additional customer information.
ADDRESS_MATCH	1 (alphanumeric)	One letter AVS result indicator. **Code** **Description** Y Match N No match X Service Unavailable G Global AVS Service Unavailable U Domestic AVS Service not available.
ZIP_MATCH	1 (alphanumeric)	One letter AVS result indicator. **Code** **Description** Y Match N No match X Service Unavailable
PROC_AVS_RESULT	3 (alphanumeric)	AVS result information. Code is processor dependent.
ACCT_DATA_ SOURCE	1 (alphanumeric)	Account data source. Detail transaction information.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
RET_REFERENCE_ NUM	12 (alphanumeric)	Retrieval reference number.
CARDHOLDER_ID_ CODE	1 (alphanumeric)	Cardholder ID Code. Embossed on card.
AUTH_SOURCE_ CODE	1 (alphanumeric)	Authorization source code.
TRANSACTION_ID	15 (alphanumeric)	Transaction ID number.
VALIDATION_CODE	4 (alphanumeric)	Processor dependent validation code. Refer to processor specifications for reference.
POS_MODE_CODE	2 (alphanumeric)	Type of point-of-sale device used initiating transaction. Detail transaction information.
MARKET_SPEC_IND	2 (alphanumeric)	Market specific indicator. Detail transaction information.
RETURNED_ACI	1 (alphanumeric)	The value of the returned transaction’s Custom Payment Services qualification status.
REQUESTED_ACI	1 (alphanumeric)	The value of the requested transaction’s Custom Payment Services qualification status.
RESPONSE_ INDICATOR	2 (alphanumeric)	Details information about this transaction.
TRANS_ATTRIBUTE	2 (alphanumeric)	Was reversal sent or not for this authroization. The corresponding API field, ID_TRANS_ATTRIBUTE, accepts only one character field length.

Code	Description
0	Reversal not sent
1	Reversal Sent

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
E_COMMERCE_TYPE	2 (alphanumeric)	Electronic Commerce Flag for web transactions. **Code** — **Description** Null Value (empty) — Not a web based transaction. 01 — Not secure. Channel encryption was not used between the browser and web server. 02 — Secure. Channel encryption was used between the browser and web server.
RECUR_TRANS_ NUMBER	2 (alphanumeric)	Recurring transaction information.
RECUR_TRANS_ COUNT	2 (alphanumeric)	Recurring transaction information.
USER_DEFINED_1	50 (alphanumeric)	User Defined field.
USER_DEFINED_2	50 (alphanumeric)	User Defined field.
USER_DEFINED_3	50 (alphanumeric)	User Defined field.
USER_DEFINED_4	50 (alphanumeric)	User Defined field.
USER_DEFINED_5	50 (alphanumeric)	User Defined field.
USER_SEQUENCE_ NUM	50 (alphanumeric)	User Defined field.
USER_SOURCE_ NAME	31 (alphanumeric)	User Defined field.
RESERVED_1	50 (alphanumeric)	CPM Reserved field. Do not use.
RESERVED_2	50 (alphanumeric)	CPM Reserved field. Do not use.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
SOURCE_IP_ ADDRESS	15 (alphanumeric)	Requested transaction source Internet Protocol Address.
TAX_AMOUNT	16 (numeric)	Tax amount added to order. Purchasing card information. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
P_CARD_ORDER_ NUM	16 (alphanumeric)	Purchasing card Information.
COM_CARD_TYPE	2 (alphanumeric)	Two character code indicating commercial card type. **Code** — **Description** 00 — Not a commercial card 01 — Purchasing Card 02 — Corporate Card 03 — Business Card 04 — Unknown
FRAUD_REASON_ CODE	4 (alphanumeric)	Result code of the fraud check.
FRAUD_SCORE	6 (alphanumeric)	Fraud score of the fraud check.
FRAUD_RESP_ CODE	255 (alphanumeric)	Text message describing the results of the fraud check.
SHIP_TO_ADDRESS_ 1	20 (alphanumeric)	The street address to where the product is shipped.
SHIP_TO_ADDRESS_ 2	20 (alphanumeric)	The street address to where the product is shipped.
SHIP_TO_CITY	20 (alphanumeric)	The city to where the product is shipped.
SHIP_TO_STATE	2 (alphanumeric)	The state to where the product is shipped.

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
SHIP_TO_PHONE	14 (alphanumeric)	The phone number of the location to where the product is shipped.
FREIGHT_AMOUNT	20 (alphanumeric)	Total freight or shipping and handling charges.
DUTY_AMOUNT	20 (alphanumeric)	Total of any import or export duties for this transaction.
LINE_ITEM_COUNT	4 (alphanumeric)	The number of line item detail records in this purchase.
DISCOUNT_AMOUNT	20 (alphanumeric)	Total amount of the discount applied to the merchant for this transaction.
VAT_TAX_AMOUNT	20 (alphanumeric)	VAT or other tax included in this transaction.
VAT_TAX_RATE	20 (alphanumeric)	Rate of VAT or other tax.
ALT_TAX_ID	20 (alphanumeric)	Tax identifier for the alternate tax included in this transaction.
ALT_TAX_AMOUNT	20 (alphanumeric)	Total amount of the alternate tax included in this transaction.
CARD_PRESENT_ FLAG	1 (alphanumeric)	Indicates if the card is present. **Code** / **Description**: 0 — The card is not present (call center or IVR); 1 — The card is present (retail POS); 2 — Unknown

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition		
TERM_CAPABILITY	1 (alphanumeric)	Indicates POS terminal capability.		
			Code	**Description**
			0	Unknown
			1	Terminal has a magnetic stripe reader and manual entry capabilities
			2	Magnetic stripe reader
			3	No magnetic stripe reader
TERMINAL_TYPE	1 (alphanumeric)	Indicates POS terminal type used in transaction.		
			Code	**Description**
			0	Unknown
			1	Standalone, credit card terminal
			2	Electronic Cash Register/POS system
			3	Unattended device

Table 59 CC_TRANASCTION table

Field Name	Characters in Field	Definition
POS_ENTRY_MODE	1 (alphanumeric)	Indicates entry method of credit card information into POS terminal used in transaction. **Code** — **Description** 0 — Unknown. 1 — Read from credit card magnetic track 1 (card swiper). 2 — Read from credit card magnetic track 1 (card swiper). 3 — Credit card number manually keyed in to POS terminal.
CUST_PRESENT_ FLAG	1 (alphanumeric)	Indicates if the cardholder if present at the time of the transaction. **Code** — **Description** 0 — Customer present 1 — Customer not present 2 — Recurring
BAD_FIELD_CODE	3 (alphanumeric)	Failed settlement information. Indicates a field containing incorrect data in the transaction.
BAD_FIELD_DATA	30 (alphanumeric)	Failed settlement information. Indicates the incorrect data in the field of the transaction causing failed settlement.

CC_LINEITEM_DETAIL table

Description Stores all Purchasing Card Level III data.

Usage Inserts one or more rows for every Purchasing Card Level III transaction.

Index

1 SEQUENCE_NUMBER (primary key) and LINE_ITEM_NUMBER (primary key)

Table 60 CC_LINE_ITEM_DETAIL table

Field Name	Characters in Field	Definition
SEQUENCE_NUMBER	15 (alphanumeric)	Primary Key. Transaction Sequence Number unique to each transaction.
LINE_ITEM_NUMBER	4 (alphanumeric)	Primary Key. Count of the line item number.
DESCRIPTION	35 (alphanumeric)	Text description of the item purchased.
PRODUCT_CODE	12 (alphanumeric)	Product code of the item purchased.
QUANTITY	12 (alphanumeric)	Number of units of the item purchased.
UNIT_OF_MEASURE	12 (alphanumeric)	Unit of measure of measure code for the item purchased.
TAX_AMOUNT	20 (alphanumeric)	Tax amount for item purchased.
TAX_RATE	20 (alphanumeric)	Tax rate applied to item purchased.
TOTAL_AMOUNT	20 (alphanumeric)	Total amount charged for item purchased.
DISCOUNT_AMOUNT	20 (alphanumeric)	Amount of discount applied to this line item.
COMMODITY_CODE	12 (alphanumeric)	Commodity code used to classify the item purchased.
UNIT_COST	20 (alphanumeric)	Unit cost of the item purchased.
TAX_TYPE_APPLIED	4 (alphanumeric)	Type of tax applied to the item purchased.
TAX_APPLIED	1 (alphanumeric)	Tax applied.
TAX_EXEMPT	1 (alphanumeric)	Tax exempt.
DISCOUNT_INDICATOR	1 (alphanumeric)	Indicates if a discount was applied to the item purchased.

CC_SETTLEMENT table

Description This table stores the results of batch settlement.

Usage Insert a record for every batch, update with current batch status.

Indices

1 MERCHANT_ID (primary key) and BATCH_ID (primary key)

Table 61 CC_Settlement table

Field Name	Characters in Field	Definition
SERVER_ID	4 (numeric)	CPM Server ID. Used to identify unique CPM Servers when connected to the same database.
MERCHANT_ID	32 (alphanumeric)	Primary Key. Merchant for this batch.
BATCH_ID	12 (numeric)	Primary Key. Number identifying batch sent for settlement.
LPC_TYPE	4 (alphanumeric)	Indicates what Gateway was used to connect to processor.
CURRENCY_CODE	3 (alphanumeric)	Indicates the national currency used by the merchant.
SETTLE_DATE_TIME	Date Time (*MM/DD/YY HHMMSS AM* or *PM*)	Date and time of batch settlement.
BATCH_STATUS	4 (alphanumeric)	Current status of batch at any given time.
BATCH_RESPONSE_MSG	30 (alphanumeric)	Text message of batch response.
PROCESSOR_BATCH_ID	12 (alphanumeric)	Processor assigned batch identifier.

CC_SETTLEMENT_LOCK table

Description This table keeps multiple servers from settling the same batch simultaneously.

Usage Inserts one row for every merchant being settled and deletes the row upon completion of the settlement for that merchant.

Indices

1 BATCH_ID (primary key)

Table 62 CC_SETTLEMENT_LOCK table

Field Name	Characters in Field	Definition
SERVER_ID	4 (numeric)	CPM Server ID. Used to identify unique CPM Servers when connected to the same database.
MERCHANT_ID	32 (alphanumeric)	Primary Key. Merchant holding the lock.
BATCH_ID	12 (numeric)	Number identifying batch sent for settlement.
LPC_TYPE	4 (alphanumeric)	Indicates what Gateway was used to connect to processor.
DATE_TIME	Date Time (*MM/DD/YY HHMMSS AM* or *PM*)	The date and time the lock was placed on the batch.

PX_MESSAGE table

Description This table stores warning/error messages generated.

Usage Inserts one record for every message.

Table 63 PX_MESSAGE table

Field Name	Characters in Field	Definition
SERVER_ID	4 (numeric)	CPM Server Identification Number.
DATE_TIME	Date Time (*MM/DD/YY HHMMSS AM* or *PM*)	Date and time of message.
CODE	5 (alphanumeric)	Error Code.
MESSAGE	200 (alphanumeric)	Server Message.

PX_SESSION table

Description This table stores information on every security session on the server system table.

Usage Inserts a record for every new security session, cleared out automatically by the server.

Index

1 SESSION_ID (primary key)

Table 64 PX_SESSION table

Field Name	Characters in Field	Definition
SESSION_ID	12 (alphanumeric)	Primary Key. Unique identifier of the user's session.
MERCHANT_ID	32 (alphanumeric)	Primary Key. Merchant identifier for this session.
BITMASK	12 (alphanumeric)	
AMOUNT_LIMIT	16 (numeric)	Transaction amount limit for this user. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.
RETURN_LIMIT	16 (numeric)	Return limit for this user. Use *DDDDDDDDDDDDCCCC* as the format. **Note** Do not use a decimal point in the transaction amount.

Table 64 PX_SESSION table

Field Name	Characters in Field	Definition
VALID_DATE_TIME	Date/Time	The date and time of the initial logon. The value is updated with the date and time of any activity that occurs during the session.
IP_ADDRESS	15 (alphanumeric)	The IP address of the client.
SESSION_TIMEOUT	12 (alphanumeric)	The maximum amount of time, in minutes, that a session remains open before timing out.

RPT_MERCHANT_LIST table

Description This table associates the Merchant ID to the Merchant Name for reporting purposes.

Usage Inserts a record for each Merchant ID.

Index

1 MERCHANT_INDEX (primary key)

Table 65 RPT_MERCHANT_LIST table

Field Name	Characters in Field	Definition
MERCHANT_INDEX	5 (numeric)	A unique identifier for each merchant.
MERCHANT_ID	32 (alphanumeric)	Primary Key. Merchant Identification number associated with each transaction.
MERCHANT_NAME	50 (alphanumeric)	The display name for the merchant.

DB_STATUS table

Description This table stores the total transactions and the oldest transaction in the database.

Usage Counts transactions by unit of measure.

Table 66 DB_STATUS table

Field Name	Characters in Field	Definition
TOTAL_ TRANSACTIONS	8 (numeric)	Records total number of transaction in database.
OLDEST_ TRANSACTION	8 (numeric)	Records oldest transaction stored in CPM Server database.

PLCARD_GECC table

Description This table stores transaction information for the General Electric Capitol Corporation (GECC) private label credit card.

Usage Insert a record for every transaction using the GECC private label credit card, update with response from processor.

Indices

1 SEQUENCE_NUMBER (primary key)

Table 67 PLCARD_GECC table

Field Name	Characters in Field	Definition
SEQUENCE_NUMBER	15 (alphanumeric)	Primary Key. CPM sequence number associated with a GECC private label card transaction.
PROMOTIONAL_PLAN	1 (alphanumeric)	This field is defined by GECC.
PROMO_END_DATE	4 (alphanumeric)	This field is defined by GECC.
SALE_TYPE	1 (alphanumeric)	This field is defined by GECC.
LINE_ITEM_1	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
LINE_ITEM_2	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
LINE_ITEM_3	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.

Table 67 PLCARD_GECC table

Field Name	Characters in Field	Definition
LINE_ITEM_4	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
LINE_ITEM_5	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
LINE_ITEM_6	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
LINE_ITEM_7	4 (alphanumeric)	GECC private card line item detail. Defined by GECC.
MICROFICHE_SEQ_NUM	8 (alphanumeric	Microfiche sequence number associated with the transaction.
PLAN_NUMBER	5 (alphanumeric)	This field is defined by GECC.

PLCARD_BENEFICIAL table

Description This table stores transaction information for the Beneficial private label credit card.

Usage Insert a record for every transaction using the Beneficial private label card, update with response from processor.

Indices

1 SEQUENCE_NUMBER (primary key)

Table 68 PLCARD_BENEFICIAL table

Field Name	Characters in Field	Definition
SEQUENCE_NUMBER	15 (numeric)	Primary Key. CPM sequence number associated with a Beneficial private label card transaction
CREDIT_PLAN	5 (alphanumeric)	This field’s use is defined by the merchant and Beneficial.
DEPARTMENT_CODE	4 (alphanumeric)	This field’s use is defined by the merchant and Beneficial.
SKU_NUMBER	9 (alphanumeric)	The Stop Keeping Unit (SKU) number associated with the transaction based on the merchant’s unique SKU schema.

Table 68 PLCARD_BENEFICIAL table

Field Name	Characters in Field	Definition
ITEM_DESCRIPTION	40 (alphanumeric)	Description of item associated with the transaction. Defined by merchant.
STORE_NUMBER	5 (alphanumeric)	This field's use is defined by the merchant and Beneficial.

EFT_TRANSACTION table

Description This table stores ACH transaction information.

Usage Insert a record for every ACH transaction function, update with response from the processor. This table is populated for every CPM API ACH function except ACH lookup. In addition to the field in the Electronic fund transfer API writing to this table, other fields from other CPM API groups write to this table. The CPM API groups are:

- Base group
- Extended information group
- PS/2000 group
- Billing information group
- User defined fields group

Indices

SEQUENCE_NUMBER (primary key)

Table 69 EFT_TRANSACTION table

Field Name	Characters in Field	Definition
SERVER_ID	4 (alphanumeric)	The CPM Server ID used in this transaction.
TRANSACTION_CODE	3 (alphanumeric)	ACH transaction code
LCC_RETURN_CODE	4 (alphanumeric)	CPM API return code.
LCC_RETURN_MSG	40 (alphanumeric)	CPM API return message.
BATCH_ID	12 (alphanumeric)	Indicates which settlement batch contains this transaction.
DRAFT_ID	8 (alphanumeric)	Draft ID.
MERCHANT_ID	32 (alphanumeric)	Merchant ID used in this transaction.

Table 69 EFT_TRANSACTION table

Field Name	Characters in Field	Definition
TRANSACTION_ STATE	1 (alphanumeric)	Transaction state.
TRANS_DATE_TIME	Date/Time	Date and time of transaction.
BANK_ACCT_NUM	17 (alphanumeric)	The bank account number in which to credit or debit funds.
BANK_ID	9 (alphanumeric)	The transit routing number of the bank holding the target account. This field is also known as the ABA number or RDFI number.
ORDER_NUMBER	25 (alphanumeric)	Order number for this transaction.
ACCOUNT_TYPE	1 (alphanumeric)	Type of bank account used in this transaction. Check the financial processor specification for the proper variables used for this field. These values are set at the point of sale.
AMOUNT	16,4 (numeric)	Transaction amount.
VERIFICATION_ RESULT	1 (alphanumeric)	CPM independent field detailing the result of the verification for this transaction. **Code** **Description** A Verification successful D Verification rejected
APPROVAL_CODE	9 (alphanumeric)	Financial processor approval code.
PROC_RSP_CODE	4 (alphanumeric)	Financial processor dependent result of the verification.
PROC_RSP_MSG	20 (alphanumeric)	Processor response message.
USER_SEQ_NUMBER	40 (alphanumeric)	User sequence number.
USER_DEFINED_1	50 (alphanumeric)	User defined field.
USER_DEFINED_2	50 (alphanumeric)	User defined field.
USER_DEFINED_3	50 (alphanumeric)	User defined field.

Table 69 EFT_TRANSACTION table

Field Name	Characters in Field	Definition
USER_DEFINED_4	50 (alphanumeric)	User defined field.
USER_DEFINED_5	50 (alphanumeric)	User defined field.
SEQUENCE_NUMBER	15 (alphanumeric)	Transaction Sequence Number unique to each transaction. Primary key.
BAD_FIELD_CODE	3 (alphanumeric)	Code from financial processor indicating ACH transaction failed during settlement.
BAD_FIELD_DATA	30 (alphanumeric)	Data indicating bad field causing failed settlement.
MERCH_BILL_NAME	25 (alphanumeric)	Merchant billing name.
MERCH_BILL_LOC	13 (alphanumeric)	Merchant billing location.
CUSTOMER_NAME	26 (alphanumeric)	Provided for storing additional customer information.

Database field and CPM API field mapping

CC_TRANSACTION table

Table 70 CC_TRANSACTION fields and API fields mapping table

Database field name	API field name
MERCHANT_ID	ID_MERCHANT_ID
SEQUENCE_NUMBER	ID_SEQUENCE_NUMBER
AUTH_RESP_CODE	ID_AUTH_RESPONSE_CODE
PROC_AUTH_RSP_CODE	ID_PROCESSOR_AUTH_RESPONSE_CODE
AUTH_RESPONSE_MSG	ID_AUTH_RESPONSE_MESSAGE
APPROVAL_CODE	ID_APPROVAL_CODE
ACCOUNT_NUMBER	ID_ACCOUNT_NUMBER
EXPIRATION_DATE	ID_EXPIRATION_DATE
AMOUNT	ID_AMOUNT
ORIGINAL_AMOUNT	ID_ORIGINAL_AMOUNT
TRANS_DATE_TIME	ID_TRANSACTION_DATE
TRANS_DATE_TIME	ID_TRANSACTION_TIME
CURRENT_AMOUNT	ID_CURRENT_AMOUNT
ORDER_NUMBER	ID_ORDER_NUMBER
MERCH_BILL_NAME	ID_MERCHANT_BILLING_NAME
MERCH_BIL_LOC	ID_MERCHANT_BILLING_LOCATION
MERCH_BILL_STATE	ID_MERCHANT_BILLING_STATE
CARD_TYPE	ID_CARD_TYPE
CUSTOMER_NAME	ID_CUSTOMER_NAME
CUSTOMER_PHONE	ID_CUSTOMER_PHONE
CUSTOMER_STREET	ID_CUSTOMER_STREET
CUSTOMER_CITY	ID_CUSTOMER_CITY

Table 70 CC_TRANSACTION fields and API fields mapping table

Database field name	API field name
CUSTOMER_STATE	ID_CUSTOMER_STATE
CUSTOMER_ZIP	ID_CUSTOMER_ZIP
CUSTOMER_EMAIL	ID_CUSTOMER_EMAIL
CUSTOMER_IP_ADDR	ID_CUSTOMER_IP_ADDRESS
ADDRESS_MATCH	ID_ADDRESS_MATCH
ZIP_MATCH	ID_ZIP_MATCH
PROC_AVS_RESULT	ID_PROCESSOR_AVS_RESULT
ACCT_DATA_SOURCE	ID_ACCOUNT_DATA_SOURCE
RET_REFERENCE_NUM	ID_RETRIEVAL_REFERENCE_NUMBER
CARDHOLDER_ID_CODE	ID_CARD_HOLDER_ID
AUTH_SOURCE_CODE	ID_AUTHORIZATION_SOURCE_CODE
TRANSACTION_ID	ID_TRANSACTION_ID
VALIDATION_CODE	ID_VALIDATION_CODE
POS_MODE_CODE	ID_POS_MODE_CODE
MARKET_SPEC_IND	ID_MARKET_SPECIFIC_INDICATOR
RETURNED_ACI	ID_RETURNED_ACI
REQUESTED_ACI	ID_REQUESTED_ACI
RESPONSE_INDICATOR	ID_RESPONSE_INDICATOR
TRANS_ATTRIBUTE	ID_TRANS_ATTRIBUTE
E_COMMERCE_TYPE	ID_E_COMMERCE_TYPE
USER_DEFINED_1	ID_USER_DEFINED_1
USER_DEFINED_2	ID_USER_DEFINED_2
USER_DEFINED_3	ID_USER_DEFINED_3
USER_DEFINED_4	ID_USER_DEFINED_4
USER_DEFINED_5	ID_USER_DEFINED5

Table 70 CC_TRANSACTION fields and API fields mapping table

Database field name	API field name
USER_SEQUENCE_NUM	ID_USER_SEQUENCE_NUMBER
USER_SOURCE_NAME	ID_USER_SOURCE_NAME
RESERVED_1	ID_RESERVED_1
RESERVED_2	ID_RESERVED_2
TAX_AMOUNT	ID_TAX_AMOUNT
P_CARD_ORDER_NUM	ID_PURCHASE_CARD_ORDER_NUMBER
COM_CARD_TYPE	ID_COMMERCIAL_CARD_TYPE
FRAUD_REASON_CODE	ID_FRAUD_REASON_CODE
FRAUD_SCORE	ID_FRAUD_SCORE
FRAUD_RESP_CODE	ID_FRAUD_RESPONSE_CODE
SHIP_TO_ADDRESS_1	ID_SHIP_TO_ADDRESS_1
SHIP_TO_ADDRESS_2	ID_SHIP_TO_ADDRESS_2
SHIP_TO_CITY	ID_SHIP_TO_CITY
SHIP_TO_STATE	ID_SHIP_TO_STATE
SHIP_TO_PHONE	ID_SHIP_TO_PHONE
SHIP_TO_ZIP_CODE	ID_SHIP_TO_ZIP_CODE
SHIP_FROM_ZIP_CODE	ID_SHIP_FROM_ZIP_CODE
FREIGHT_AMOUNT	ID_FREIGHT_AMOUNT
DUTY_AMOUNT	ID_DUTY_AMOUNT
LINE_ITEM_COUNT	ID_LINE_ITEM_DETAIL_COUNT
DISCOUNT_AMOUNT	ID_DISCOUNT_AMOUNT_APPLIED
VAT_TAX_AMOUNT	ID_VAT_TAX_AMOUNT
VAT_TAX_RATE	ID_VAT_TAX_RATE
ALT_TAX_ID	ID_ALTERNATIVE_TAX_ID
ALT_TAX_AMOUNT	ID_ALTERNATIVE_TAX_AMOUNT

Table 70 CC_TRANSACTION fields and API fields mapping table

Database field name	API field name
CARD_PRESENT_FLAG	ID_CARD_PRESENT_FLAG
TERM_CAPABILITY	ID_TERMINAL_CAPABILITY
TERMINAL_TYPE	ID_TERMINAL_TYPE
POS_ENTRY_MODE	ID_POS_ENTRY_MOED
CUST_PRESENT_FLAG	ID_CUSTOMER_PRESENT_FLAG

CC_LINEITEM_DETAIL table

Table 71 CC_LINEITEM_DETAIL fields and API fields mapping table

Database field name	API field name
SEQUENCE_NUMBER	ID_SEQUENCE_NUMBER
DESCRIPTION	ID_ITEM_DESCRIPTION
PRODUCT_CODE	ID_ITEM_PRODUCT_CODE
QUANTITY	ID_ITEM_QUANTITY
UNIT_OF_MEASURE	ID_ITEM_UNIT_OF_MEASURE
TAX_AMOUNT	ID_ITEM_TAX_AMOUNT
TOTAL_AMOUNT	ID_ITEM_TOTAL_AMOUNT
DISCOUNT_AMOUNT	ID_ITEM_DISCOUNT_AMOUNT
COMMODITY_CODE	ID_ITEM_COMMODITY_CODE
UNIT_COST	ID_ITEM_UNIT_COST
TAX_TYPE_APPLIED	ID_ITEM_TAX_TYPE_APPLIED
TAX_APPLIED	ID_ITEM_TAX_APPLIED
TAX_EXEMPT	ID_ITEM_TAX_EXEMPT
DISCOUNT_INDICATOR	ID_ITEM_DISCOUNT_INDICATOR

PLCARD_BENEFICIAL table

Table 72 PLCARD_BENEFICIAL fields and API fields mapping table

Database field name	API field name
SEQUENCE_NUMBER	ID_SEQUENCE_NUMBER
CREDIT_PLAN	ID_BENEFICIAL_CREDIT_PLAN
DEPARTMENT_CODE	ID_BENEFICIAL_DEPARTMENT_CODE
SKU_NUMBER	ID_BENEFICIAL_SKU_NUMBER
ITEM_DESCRIPTION	ID_BENEFICIAL_ITEM_DESCRIPTION
STORE_NUMBER	ID_BENEFICIAL_STORE_NUMBER

PLCARD_GECC table

Table 73 PLCARD_GECC fields and API fields mapping table

Database field name	API field name
SEQUENCE_NUMBER	ID_SEQUENCE_NUMBER
PROMOTION_PLAN	ID_GECC_PROMOTIONAL_PLAN
PROMO_END_DATE	ID_GECC_PROMOTIONAL_END_DATE
SALE_TYPE	ID_GECC_SALE_TYPE
LINE_ITEM_1	ID_GECC_LINE_ITEM_1
LINE_ITEM_2	ID_GECC_LINE_ITEM_2
LINE_ITEM_3	ID_GECC_LINE_ITEM_3
LINE_ITEM_4	ID_GECC_LINE_ITEM_4
LINE_ITEM_5	ID_GECC_LINE_ITEM_5
LINE_ITEM_6	ID_GECC_LINE_ITEM_6
LINE_ITEM_7	ID_GECC_LINE_ITEM_7
MICROFICHE_SEQ_NUM	ID_GECC_MICROFICHE_SEQUENCE_NUM
PLAN_NUMBER	ID_GECC_PLAN_NUMBER

EFT_TRANSACTION table

Table 74 EFT_TRANSACTION fields and API fields mapping table

Database field name	API field name
TRANSACTION_CODE	ID_TRANSACTION_ID
LCC_RETURN_MSG	ID_RETURN_CODE_MESSAGE
TRANS_DATE_TIME	ID_TRANSATION_DATE
TRANS_DATE_TIME	ID_TRANSACTION_TIME
BANK_ACCT_NUM	ID_BANK_ACCOUNT_NUMBER
BANK_ID	ID_BANK_ID
ACCOUNT_TYPE	ID_ACCOUNT_TYPE
VERIFICATION_RESULT	ID_VERIFICATION_RESULT
PROC_RSP_CODE	ID_PROCESSOR_RESPONSE_CODE
PROC_RSP_MSG	ID_PROCESSOR_RESPONSE_MESSAGE
MERCHANT_ID	ID_MERCHANT_ID
ORDER_NUMBER	ID_ORDR_NUMBER_ID
AMOUNT	ID_AMOUNT_ID
APPROVAL_CODE	ID_APPROVAL_CODE
USER_SEQ_NUMBER	ID_USER_SEQUENCE_NUMBER
USER_DEFINED_1	ID_USER_DEFINED_1
USER_DEFINED_2	ID_USER_DEFINED_2
USER_DEFINED_3	ID_USER_DEFINED_3
USER_DEFINED_4	ID_USER_DEFINED_4
USER_DEFINED_5	ID_USER_DEFINED_5
SEQUENCE_NUMBER	ID_SEQUENCE_NUMBER
MERCH_BILL_NAME	ID_MERCHANT_BILLING_NAME
MERCH_BILL_LOC	ID_MERCHANT_BILLING_LOCATION
CUSTOMER_NAME	ID_CUSTOMER_NAME

Index