



PCCharge DevKit

Electronic Payment Processing Software
White Paper
9th Edition

GOsoftware
an **FTI** company

Copyright 2002, GO Software Inc.

Notice

Copyright © October, 2002 GO Software Inc. All rights reserved. active-Charge, active-Charge SDK, **PCCharge** Payment Server, **PCCharge** Pro, **PCCharge** DevKit, Virtual-Charge, IP-Charge are trademarks and PC-Charge is a registered trademark of GO Software Inc.

Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

© GO Software Inc.

Parkway Business Center

5000 Business Center Drive, Suite 1000

Savannah, Georgia 31405

Development Support: (877) 659-8983

Technical Support: (877) 659-8981

Fax: (912) 527-4596

Table of Contents

Introduction	4
Product Components.....	5
PCCharge DevKit.....	5
PCCharge Payment Server and PCCharge Pro.....	5
PCCharge Virtual Terminal	6
Web Interface Kit.....	6
Integration Techniques.....	7
OLE/COM Method.....	7
ActiveX OCX Method	8
ActiveX DLL Method	9
TCP Interface	9
File Method.....	10
Other Methods	10
PCCharge DevKit Development Support.....	11
Sample API	12
PSCharge.dll	12

Introduction

Thank you for purchasing the **PCCharge** DevKit with its associated suite of products.

We at GO Software, Inc. are committed to providing you with the most comprehensive electronic payment processing solutions available today. If you have any suggestions as to how we can improve our products, support, or documents, please call us at (877) 659-8983 or e-mail us at dev_support@gosoftware.com.

This chapter is an introduction to the **PCCharge** DevKit. It will familiarize you with the various components and typical uses of this suite of products.

The **PCCharge** DevKit is a bundle of applications, tools, code examples, and documents that can assist you in a smooth integration of electronic payments into your application.

The following applications are included in the development kit:

- **PCCharge** Payment Server
- **PCCharge** Pro
- **PCCharge** Virtual Terminal
- Web Interface Kit

Recently, GO Software, Inc. changed the names of some of its software products. The following table lists the old names and corresponding new names.

PCCharge Product Name Changes

Old Product Name	New Product Name	Executable
active-Charge SDK	PCCharge DevKit	N/A
active-Charge	PCCharge Payment Server	Active-Charge.exe
PC-Charge	PCCharge Pro	Pccw.exe

Product Components

PCCharge DevKit

The **PCCharge** DevKit is this manual and a group of examples of the various interface methods available to integrate payment processing into your applications. It includes sample code for FoxPro, VB5, VB6, C++, Access, Java, and Delphi 3/4 to demonstrate the OLE (COM), OCX (ActiveX), DLL (ActiveX), TCP/IP, and File Methods of integration.

Also, there are files containing test account and test transaction information to test your application in a simulated live environment. **Note:** Not all processing networks provide distributable test merchant information. If test merchant is not provided in our DevKit, please call the processor for a test account.

We have included a number of interesting sample applications and examples in the **PCCharge** DevKit. Browse through the DevKit directory and see what we have to offer. There are Perl, ASP, and Cold Fusion examples in the Web Interface Kit as well as sample JAVA applications. The DevKit also includes an application to install the test merchant information, and a utility to decode and/or create `.inp`, `.pro`, and `.out` files (**PCCharge's** proprietary file formats).

PCCharge Payment Server and PCCharge Pro

Included with the DevKit are evaluation copies of **PCCharge** Payment Server and **PCCharge** Pro. They are meant to act as the "payment-processing engines" for your transactions. When given transaction information, either product will handle formatting the request, communicating with the processor, and receiving the processor's response to the transaction request.

Note: If you have purchased the DevKit Suite, you are entitled to activate an unlimited user license version of either **PCCharge** Payment Server or **PCCharge** Pro.

WARNING: Neither **PCCharge** Payment Server nor **PCCharge** Pro was designed to be used as a Windows NT/2000 service. GO Software does NOT support the use of either of these programs as a service; doing so may result in unpredictable results.

The **PCCharge** Payment Server engine is a fully integratable component that has no user interface by default and, thus, requires no manual intervention. **PCCharge** Pro also is integratable; however, it:

- has a default user interface
- is primarily used as a stand-alone program to process transactions
- requires user intervention in response to error messages

Both engines can process credit card, EBT, check, gift card, and debit transactions. **PCCharge** Pro has the additional capability of processing recurring billing contracts. However, we do not currently support interfacing to this feature.

The copies included in the **PCCharge** DevKit are evaluation copies for developing and testing integrations. Licenses for distribution to resellers and customers are available by contacting your sales representative or by calling 800-725-9264.

PCCharge Virtual Terminal

PCCharge Virtual Terminal is a collection of web pages (Active Server Pages) and files that act as an interface to PCCharge Payment Server or PCCharge Pro. By configuring and running PCCharge Virtual Terminal on a web server, you can access and manage your merchant account in PCCharge Payment Server or PCCharge Pro from any browser, anywhere in the world.

Note: When integrating, it is recommended that you use PCCharge Payment Server instead of PCCharge Pro unless the ability to process recurring billing contracts is a requirement. Note that any recurring billing processing must occur through the PCCharge graphical user interface.

Web Interface Kit

The Web Interface Kit provides components that allow real-time, online transactions from your website, using either PCCharge Payment Server or PCCharge Pro.

The Web Interface Kit contains a CGI/Perl sample and a Java Applet Sample. . These editable samples communicate transaction information from your web pages to the payment-processing engine via TCP.

The Web Interface Kit also includes ASP pages, Cold Fusion tags, and PSCharge.dll for use with those sites based on Active Server Pages or Cold Fusion. This DLL contains both a charge class and batch class for communicating transaction information to the payment-processing engine.

Integration Techniques

There are five primary techniques to integrate **PCCharge** Payment Server into another application: the OLE/COM method, the ActiveX OCX method, the ActiveX DLL method, the TCP Interface, and the File Method. Every technique creates a file that is recognized by **PCCharge**. Thus, all techniques are essentially sending transaction information to **PCCharge** Payment Server via the File Method. However, the different techniques can appear quite different to your application, and you should consider carefully which method to use. Please note that the TCP Method depends on a socket connection and can send either a fixed-width file stream or Name/Value pairs.

OLE/COM Method

The exposed OLE/COM objects method makes it possible to completely hide the **PCCharge** Payment Server interface from the user. All aspects from setup and configuration to processing transactions can be done through the integrator interface or programmatically. It is possible to make calls to set properties or show **PCCharge** Payment Server forms by accessing the objects exposed through OLE.

Several exposed OLE objects in **PCCharge** Payment Server allow you to perform various processing functions. Before you can use these objects, you will need to make a reference to the **PCCharge** Payment Server executable. For example, in Microsoft Visual Basic 6, follow the procedure below:

1. Choose **Project | References** from the Visual Basic menu bar.
2. Click the **Browse** button.
3. Switch the **Files of Type** drop down list to "Executable Files" and then find the file "Active-Charge.exe".
4. Select it and click **Open**.
5. The "Active-Charge Payment Server" reference will be displayed in the list of available references.
6. Check the box next to "Active-Charge Payment Server" and click **OK**.
7. Create an instance of any of the OLE objects by including the following line in your code: "Set <instance name> = New <object name>"

You can also view all of the objects through the object browser. If you are not using MS VB6, refer to your language documentation for instructions on adding OLE references.

A program example of how to use OLE objects in your application is located in the directory C:\Program Files\active-charge SDK\OLE\. It is written in Visual Basic and should be followed closely when beginning your integration project.

ActiveX OCX Method

If you are programming in a visual environment that supports ActiveX technology, and all of the client machines that generate transactions are Windows platforms, then consider using the OCXs. The OCXs are visual wrappers around code to create a flat text file in the correct `.inp` format and to automatically parse the `.out` file created by PCCharge Payment Server.

Before you can use one of these controls, you must add the control to your visual programming language toolbox or equivalent. For example, in Microsoft Visual Basic 6, follow the procedure below:

1. Right click on the toolbar, and select **Components**.
2. After the **Components** window opens, click the **Browse** button.
3. Select the control you want from the directory `C:\Windows\system\`.
4. After the control is displayed in the list of components, check the box next to it and click **OK**.
5. After the control is on your toolbar, you can add it by dragging and dropping it in the desired area on your form.

The user will never see the control. It is only displayed during design-time so that you can click on it and modify its properties. If you are not using MS VB6, refer to your language documentation for instructions on using ActiveX OCX Controls.

When integrating using the ActiveX method, particular care should be taken to familiarize yourself with the properties and methods associated with the various controls. There is sample code for transactions in the `SampleCode.txt` document located in the directory `C:\Program Files\active-charge SDK\`.

Note to Visual FoxPro 6 Users: The instruction `"application.autoyield = .F."` must be executed before you call the `Send` method for any of the ActiveX controls. A sample Visual FoxPro 6 application is included. It is located in the directory `C:\Program Files\active-charge SDK\Ocx\FoxPro\`.

Note to Delphi 3/4 Users: There is a version of the `Charge` OCX specifically designed for Delphi 3/4. The Delphi version provides `Get` and `Set` methods for all of the properties. Otherwise, these OCXs are the same as the VB versions, except where noted in section **Charge.OCX** (see page **Error! Bookmark not defined.**). A sample Delphi 3/4 application is included. It is located in the directory `C:\Program Files\active-charge SDK\Ocx\Delphi3\` (or `Delphi4\`).

Note to Visual C++ Users: The standard versions of the OCX controls can be used in a Visual C++ project. The control can be added by going to **Project | Components | Add Registered Controls**. Visual C++ will provide a wrapper that will give your project access to the various properties and methods in the control. A sample Visual C++ application is included. It is located in the directory `C:\Program Files\active-charge SDK\Ocx\Cpps\`.

Note to All Users: A sample algorithm for using the OCX control (or DLL) to do credit card transactions is included in the file `SampleCode.txt`. This document can be found in the directory `C:\Program Files\active-charge SDK\`. Be sure to follow it closely when beginning your integration.

ActiveX DLL Method

If you are programming in a Windows programming environment that does not support the ActiveX OCX technology, then consider using the DLL Method. The DLL is a wrapper around code to create a flat text file in the correct `.inp` format and to automatically parse the `.out` file created by **PCCharge** Payment Server.

Before you can use the DLL, you must add a reference to the DLL to your programming language. For example, in Microsoft Visual Basic 6, follow the procedure below:

1. Choose **Project | References** from the Visual Basic menu bar.
2. After the Components window opens, click the **Browse** button.
3. Select the DLL you want from the list that appears (it appears as "GO Software's ActiveX Payment Server DLL").
4. Check the box next to it and click **OK**.
5. You may now reference the classes in the DLL.
6. If you are not using MS VB6, refer to your language documentation for instructions on using ActiveX DLLs.

Note to All Users: A sample algorithm for using the DLL to do credit card transactions is included in the file `SampleCode.txt`. This document can be found in the `C:\Program Files\active-charge SDK` folder. Be sure to follow it closely when beginning your integration.

TCP Interface

The primary advantage to using the TCP Interface Method is that it is operating system independent. The only requirement is that **PCCharge** Payment Server must be running on a Windows machine somewhere on the network. Client machines can be running on any operating system. In a client/server environment, the client machines must have TCP connectivity to the server on which **PCCharge** resides.

The TCP Interface Method consists of the following:

1. Creating and sending a properly formatted TCP transaction request stream to **PCCharge** Payment Server.
2. Retrieving the result stream that **PCCharge** Payment Server returns.

Not many things can go wrong when using this method. The obvious errors are in the format of the stream, not parsing the results correctly, or not interpreting the response correctly. Consult the section **TCP Interface** for detailed file format information (see page **Error! Bookmark not defined.**).

File Method

The primary advantage to using the File Method is that, like the TCP Interface Method, it is operating system independent. The only requirement is that **PCCharge** Payment Server must be running on a Windows machine somewhere on the network. Client machines can be running on any operating system. In a client/server environment, the directory in which **PCCharge** Payment Server resides must be shared so that the clients that are generating transactions have read, write, and execute permissions.

The File Method consists of the following:

1. Create a flat text (plain ASCII) file in a specific format.
2. Put that file into the **PCCharge** Payment Server's directory with a `.inp` extension.
3. Retrieve a flat text file with an `.out` extension that **PCCharge** Payment Server returns containing the results of your request.

Not many things can go wrong when using this method. The obvious errors are in the format of the `.inp` file, not parsing the `.out` file contents correctly, or not interpreting the response correctly. Consult the section **File Method** for detailed file format information (see page **Error! Bookmark not defined.**). The `.inp` file you create must be named `<user>.inp`, where `<user>` is a registered user in **PCCharge** Payment Server (unless you have an activated unlimited user license).

Other Methods

Ultimately, **PCCharge** Payment Server does not care how transactions are created or delivered. Any method that creates a transaction in the correct format or can use the OLE/COM, ActiveX OCX, or ActiveX DLLs is workable. This opens a whole range of possibilities to the integrator. Transaction information can be gathered through web pages, from database files, or any other resource imaginable. Integrators can use ASP, HTML, XML, CGI, Perl, or any other programming language available.

PCCharge DevKit Development Support

You are entitled to three free hours of telephone developer support within twelve months. After that, you may purchase a DevKit upgrade in order to receive three additional hours of support. Alternately, telephone/e-mail developer support is provided at a charge of \$75.00 per ½ hour billed in advance in ½-hour increments. Consultative coding support is \$1200 per day onsite at the GO Software, Inc. office in Savannah, Georgia or \$1200 per day plus travel and expenses at your location.

Developer Support Contact Information	
Toll-free support hotline	(877) 659-8983
Local contact number	(912) 527-4595
E-mail address	dev_support@pccharge.com

Developer Support Hours
9:00 AM to 5:00 PM ET Weekdays
During normal working hours, requests for callbacks and e-mail responses are fulfilled within 24 hours. Weekend, holiday and off-shift support can be pre-arranged during normal business hours.

Sample API

PSCharge.dll

This is an example of what is available within the **PCCharge** DevKit.

Charge Class Properties

Property	Description	Data Type	Notes
Action	Code identifying transaction. Consult the section DevKit Constants for a description of values (see DevKit Manual).	Long	1 = Sale 2 = Credit 3 = Void 4 = Pre-Auth 5 = Post-Auth 6 = Void Credit 7 = Void Post-Auth
Amount	Amount of transaction	String	Format: DDDDDD.CC
Card	Credit card number	Character	
CardPresent	Used for restaurant authorization transactions -- indicates if card is physically present at point of sale	String	1 -- Card present, 0 -- Card not present
CheckCard	Flag to activate credit card validity testing.	Boolean	0 = bypasses test 1 = tests card number
CustCode	Purchasing Card customer code	String	
CVV2	CVV2 Value on Card	String	Fraud protection value on back or front of credit card. 3 or 4 digits
Demo	Sets DLL in Demo mode	Boolean	
ExpDate	Expiration date.	Character	Format: MMY
Index	Returns/Sets the index of the merchant the transaction is to be run on.	Long	
LastValidDate	Last year to be considered as in the 21 st century	String	Format: YY
Manual	Manual vs. swiped transaction	Long	TRUE = swiped FALSE = manual
MCSC	Multiple count sequence count (1 of MCSC)	String	MCSC is used for non-restaurant recurring transactions. This field is the "5" in 1 of 5. This number is the total number of installment payments to be received.
MCSN	Multiple count sequence number (MCSN of 5)	String	MCSN is used for non-restaurant recurring transactions. This field is the "1" in 1 of 5. This is the number of the installment payment being made at the time of the current transaction.
Member	Credit cardholder name	Character	20 characters maximum.
MerchantNumber	Merchant number associated with this transaction	Character	Required if more than one number is entered in payment-processing engine
MTS	Microsoft Transaction Server Flag	Boolean	
Multi	Returns/Sets the flag that indicates to PCCharge that there are multiple transactions	String	Set to "1" to leave the modem port open in between transactions otherwise blank
OffLine	Offline transaction Flag	String	Set to "1" to indicate an offline transaction otherwise blank

Charge Class Properties -- Continued

Property	Description	Data Type	Notes
Path	Path where payment-processing engine is running	Character	100 characters maximum. Must end with "\".
PeriodicPayment	Periodic payment transaction flag	String	Set to "1" to indicate a periodic payment transaction. Otherwise blank
PrintReceipts	Number of receipts to print	String	Set to "0" to disable receipt printing
Processor	Code for electronic transaction processing company	Character	4 characters maximum. Consult the section DevKit Constants for a description of values (see DevKit Manual).
Reference	Reference number for Post-Authorization transactions	Character	Used for Post-Authorizations and Voids
Street	Credit cardholder billing address street number	Character	20 characters maximum (FDC: use first 5 digits only)
TaxAmt	Amount of tax in the transaction. For purchasing cards only.	String	
Ticket	Ticket or invoice number	Character	For internal referencing by merchant (8 characters maximum)
TimeOut	Maximum amount of time control will wait for result before returning timeout response	Long	Time in seconds (required field)
TotalAmount	Total authorized amount	Character	Format: DDDDDD.CC (required for post-authorizations)
Track	Track II credit card information	Character	Required for swiped transactions

Charge Class Methods

Method	Description	Data Type	Notes
TRANSID	Transaction identifier	Character	Post-Authorization = TransID from Pre-Authorization. With RBOC, for a refund transaction = Auth number from original sale. Restaurant Sale/Pre-Auth = Estimated Gratuity. Restaurant Post-Auth = Actual Gratuity.
User	User name	Character	DOS filename format
Zip	Credit cardholder billing address zip code	Character	5 or 9 digits
AddMatch	Returns a string associated with the AVS response	String	
Cancel	Cancels transaction in process.	Method	
CVV2Match	Returns a string associated with the CVV2 response	String	
DeleteUserFiles	Deletes temporary files used by PCCharge	Method	
GetACI	Returns the ACI of the transaction	String	
GetAuth	Authorization number for approved transactions	Character	Must be recorded for use with associated post-authorization
GetAVS	Address verification response for approved transactions.	Character	Y = Address + 5-digit Zip X = Address + 9-digit Zip A = Address Match Z = 5-digit Zip Match W = 9-digit Zip Match N = No Match U = Address Match Not Available R = Retry, System Not Available S = Service Not Supported E = Address Match Error

Charge Class Methods -- Continued

Method	Description	Data Type	Notes
GetCaptured	Transaction result	Boolean	TRUE = captured FALSE = not captured
GetCompanyCity	Returns the company city that is in PCCharge	String	
GetCompanyName	Returns the company name that is in PCCharge	String	
GetCompanyState	Returns the company state that is in PCCharge	String	
GetCompanyStreet	Returns the company street that is in PCCharge	String	
GetCompanyZip	Returns the company zip that is in PCCharge	String	
GetCreditCardType	Credit card type.	Character	Returns: VISA, MasterCard, etc. Uses the card number to determine the card type. Consult the section DevKit Constants for descriptions of values (see DevKit Manual).
GetCVV2	CVV2 Response	String	Returns M or N
GetErrorCode	Transaction error code	Long	Use in conjunction with the GetErrorDesc method to determine what the error code means
GetErrorDesc	Verbal description of error	Character	Returns current error code. Consult the section DevKit Constants (see DevKit Manual).
GetIND	CPS Information	String	
GetIndex	Returns the index of the merchant number in PCCharge	Long	
GetMSI	CPS Information	String	
GetPEM	POS entry mode	Character	
GetRefNumber	Reference number for approved transactions	Character	With host-based systems, used to void transactions
GetResult	Transaction result	Character	Captured, Not Captured, Processed (for offline transactions)
GetRET	Retrieval reference number	Character	Response property
GetTBatch	Transaction batch number	Character	Response property
GetTicket	Invoice or ticket number	Character	8 characters maximum
GetTICode	Transaction identifier code	Character	Returned with pre-authorizations. Required for post-authorizations.
GetTIM	Transaction time	Character	Response property
GetTitem	Transaction item. Not available with all processors.	String	
GetTransNum	Payment-processing engine transaction number	Character	Response property
PccSysExists	Checks to see if condition exists where payment-processing engine cannot perform transactions	Boolean	TRUE = problem, examine Pcc.sys FALSE = no problem
Send	Initiates the transaction	Integer	
ValidCardLength	Returns TRUE if the card is of the correct length	Boolean	
VerifyCreditCard	Verifies that credit card is correct	Boolean	TRUE = credit card is good FALSE = credit card is formatted incorrectly