Developer's SDK: Handbook for Integration Using Rossware's *Virtual Terminal ActiveX* Control

At Rossware Computing, we first developed this control for use within several of our own products. Actually (and to be more precise), its immediate predecessor was developed as a VB6 form and associated code, which were simply incorporated (as such) into each of our apps as needed it. The advent of required PA-DSS certification spawned new thinking. We realized it would be much more expedient to certify a single ActiveX component — as opposed to several applications that each, independently, implement a particular code set.

For such reason, we converted our already well-proven form and code into an ActiveX control (sometimes hereinafter abbreviated as "VT"). Now, any application that wants to use this Virtual Terminal needs only to incorporate the ActiveX control, and make appropriate calls thereto. VT handles all card-related info, conducts the desired transactions, and thoroughly sequesters sensitive data within its own processes. Since an incorporating application does not in itself handle (or even begin to touch) such data, it is taken out of scope so far as PA-DSS certification is concerned.

Any and every developer who so chooses is free to use VT, and without any charge or obligation except one. From the end-user's perspective, VT may not be implemented to work in conjunction with any merchant account, except as setup under the auspices of Rossware Computing as the registered reseller, and with Merchant Warehouse (hereinafter "MW") as the gateway company.

For end-user instruction as concerns retail use of Rossware's Virtual Terminal, please see the *user manual* at:

http://rossware.net/MiniManuals/VirtualTerminalHandbook.pdf.

Chapter 1

Adding the Virtual Terminal OCX to Your Project

The VirtualTerminal.ocx control may be acquired from Rossware, as one of a few files contained in a .zip folder, obtainable at:

http://rossware.net/downloads/VT/vt_sdk.zip.

Like any ActiveX component, by design VT should be capable of incorporation within applications written in a wide variety of programming languages and environments. For purposes of simplicity within this Handbook, we will assume you are using VB6. If otherwise, please adapt as appropriate.

Steps for incorporation are as follows:

1. Add VirtualTerminal.ocx to your c:\Windows\System32 (or other appropriately applicable) folder.

2. Add any other needed Windows runtimes. In particular, you will need msxml3.dll (which is also provided in the vt_sdk folder). You will also need standard VB6 runtimes, which most platforms already have installed.

3. Open or create the project in which you want to add VT.

4. From the *Add Components* tab, find and add the previously installed instance of VirtualTerminal.ocx.

5. Open or create the form in which you want the VT interface to appear. From the VB6 Toolbox, select the VirtualTerminal, and add it to your form.

6. Position the control as desired, and size its dimensions as appropriate to accommodate its contents.

7. Write code to access the control's methods, as per programming purposes and intent.

Details, as applicable to Step 7, are what we turn to now.

Chapter 2

Writing Code to Access Methods

VirtualTerminal.ocx has just three methods, as follows:

1. **.ShowForGeneralOpen**

   Use this method to activate the VT interface for general user interaction. In other words, the interface is activated in a mode where the user can run as many transactions and in whatever amount as wanted. There are no arguments in the call, and no information (i.e., regarding transactions conducted) comes back to the calling client.

   This method makes the interface into the equivalent, more or less, of a stand-alone terminal.

2. .ShowForParticularXaction

Use this method to activate the VT interface for a particular transaction that, typically, is advance-specified by the calling client as to Amount and Ticket Reference. The method is a *Function*, and returns as *True* if the transaction is successful, *False* if it is not. It incorporates the following arguments:

Amnt	Used by the calling client to specify the amount desired for the transaction
TcktNumbr	Used by the calling client to specify a string value that's used, within the transaction, as the merchant's identifier for the transaction (typically this consists of an InvoiceNumber)
rtrnAmount	VT returns the amount actually transacted (negative value designates a credit)
rtrnRefNmbr	VT returns the ReferenceNumber, as provided by MW, on the transaction
AllowZeroStart	Optional. Calling client should set as true if wanting VT interface to accept an initial *Amnt* specification of -0-
CNm	Optional. Calling client may use this argument to specify probable name on card, as potentially used for keyed entry (the VT interface provides user with means of easy insertion on verification name is actually what's on card)
CStrtNmbr	Optional. Calling client may use this argument to specify the *number portion* of the customer's street address (similar purpose to *CNm*, above)
CZip	Optional. Calling client may use this argument to specify the zipcode of the customer's street address (similar purpose to *CNm* and *CStrtNmbr*, above).

Please note that, in contrast to .ShowForGeneralOpen, this method makes the VT interface into a specific transaction agent. In this mode, the latter fulfills a request as requested by the client caller, and reports back on the result.

3. .EndOpenState

This method (like the one just described) is also a Function. It is used to assure VT closes, as a process, when it should, and that it does so in a good and proper manner. It returns as *True* when closing is complete. You should write your application so that neither it, nor any form that displays the VT closes until this method is first called, and returned as True.

The reason this is important is because there is typically a period of one to three seconds (sometimes more), after a transaction has been initiated by the user, and prior to the response being returned by Merchant Warehouse. Absent the precaution as provided by this method, it's possible a calling client could close itself while the Virtual Terminal is still waiting for a response from MW. It's possible, furthermore, the response could come back indicating a successful transaction — while in the meantime the application (and/or its applicable processes) have already closed, and therefore fail to register the transaction.

Please follow good practices and assure you do not let your calling interface close until after this call is successfully completed.

For programming with VirtualTerminal.ocx, all you must do is write the very simple code as needed to call (and utilize per your own programming intent) these methods.

Chapter 3

Error Handling

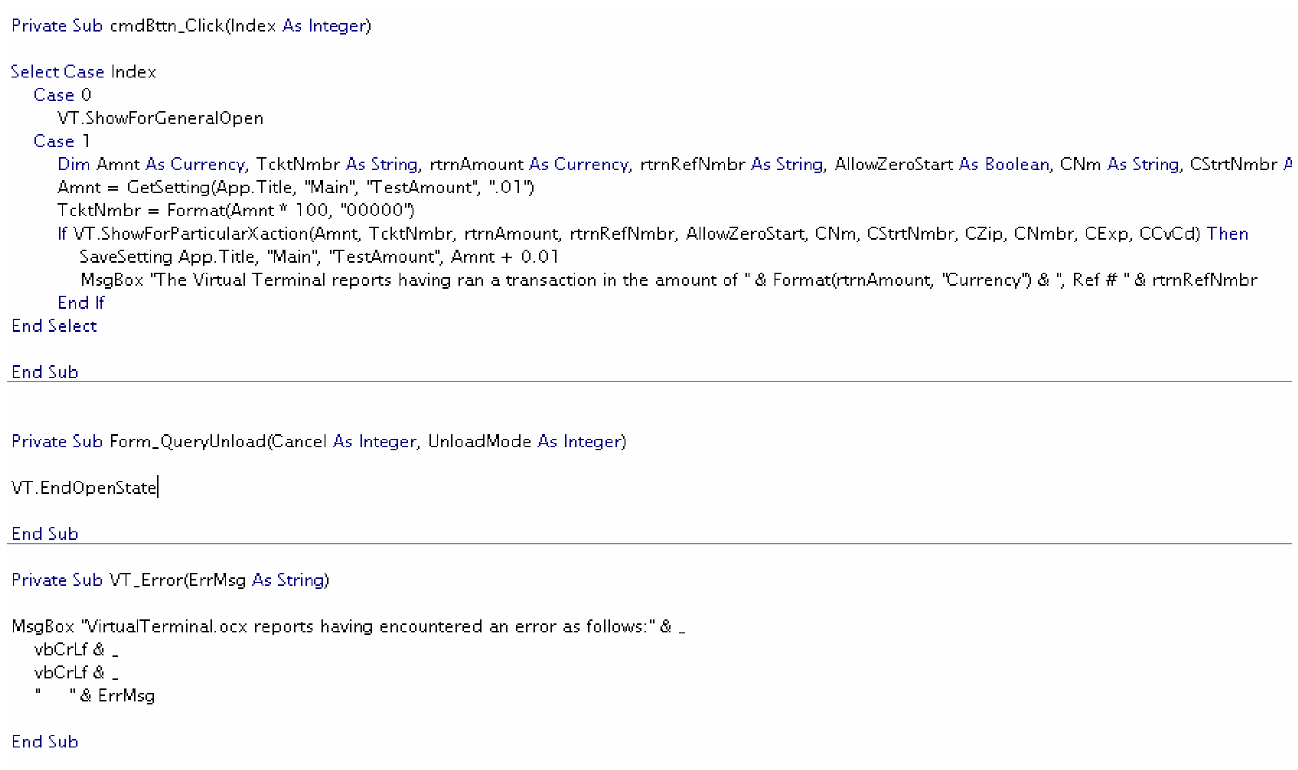
The Virtual Terminal has one event that's deliberately provided and exposed. It is the **.Error** event. The event will trigger when and if the Virtual Terminal encounters an error in execution. For optimum programming, you should place a handler in the *VirtualTerminal_Error* procedure.

Chapter 4

The "VT-Test" Sample Program

Also provided in the vt_sdk.zip folder is very simple VB6 project that invokes the methods of VirtualTerminal.ocx. It is a simple program. It has no modules, no classes, and no user controls. There is only one, very basic form containing three objects: the VirtualTerminal (internally named "VT"), and two command buttons.

There is so little code involved as to be nearly comical. Following is a snapshot that shows (aside from what's cutoff at the far-right) its entirety.



```
Private Sub cmdBttn_Click(Index As Integer)

Select Case Index
   Case 0
      VT.ShowForGeneralOpen
   Case 1
      Dim Amnt As Currency, TcktNmbr As String, rtrnAmount As Currency, rtrnRefNmbr As String, AllowZeroStart As Boolean, CNm As String, CStrtNmbr A
      Amnt = GetSetting(App.Title, "Main", "TestAmount", ".01")
      TcktNmbr = Format(Amnt * 100, "00000")
      If VT.ShowForParticularXaction(Amnt, TcktNmbr, rtrnAmount, rtrnRefNmbr, AllowZeroStart, CNm, CStrtNmbr, CZip, CNmbr, CExp, CCvCd) Then
         SaveSetting App.Title, "Main", "TestAmount", Amnt + 0.01
         MsgBox "The Virtual Terminal reports having ran a transaction in the amount of " & Format(rtrnAmount, "Currency") & ", Ref # " & rtrnRefNmbr
      End If
End Select

End Sub
```

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)

VT.EndOpenState

End Sub
```

```
Private Sub VT_Error(ErrMsg As String)

MsgBox "VirtualTerminal.ocx reports having encountered an error as follows:" & _
   vbCrLf & _
   vbCrLf & _
   "    " & ErrMsg

End Sub
```

That's right. The above is *all* the code as strictly required for an application that harnesses all three VT methods, and appropriately responds to the .Error event. For a person with even minimal programming skills, it should be child's play to see how this code works, and to fathom how one might similarly implement within his or her own application.

This sample program is also provided in *compiled* format (Test_VT.exe). This allows you — using just two files, *Test_VT.exe* and *VirtualTerminal.ocx* — to fully run the VT interface, and to experiment with functionality (at least assuming your platform is otherwise appropriately runtime-equipped and that you possess an appropriate MW merchant test account for testing purposes). In fact, with a real merchant account you could even run real/live transactions — using nothing more that's specific to the purpose than the two files here described.

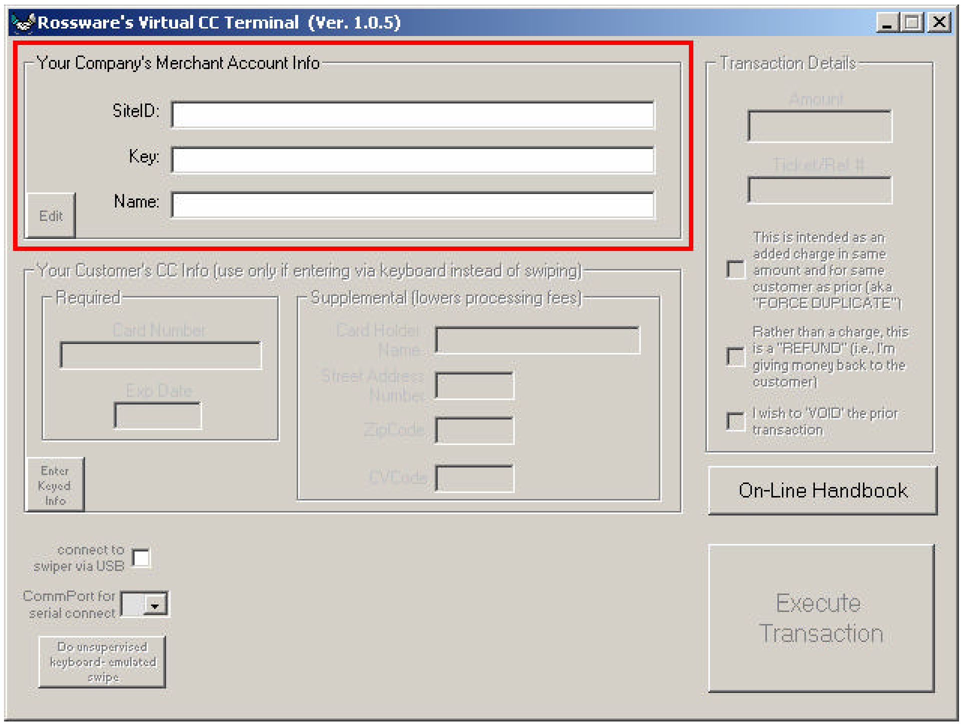
Chapter 5

Description of Processes within the Virtual Terminal

This section describes underlying structure and processes within the VT, and is not directly needed for programming integration. If your sole interest is in creating the integration, this section (indeed, the remainder of this handbook) may be skipped.

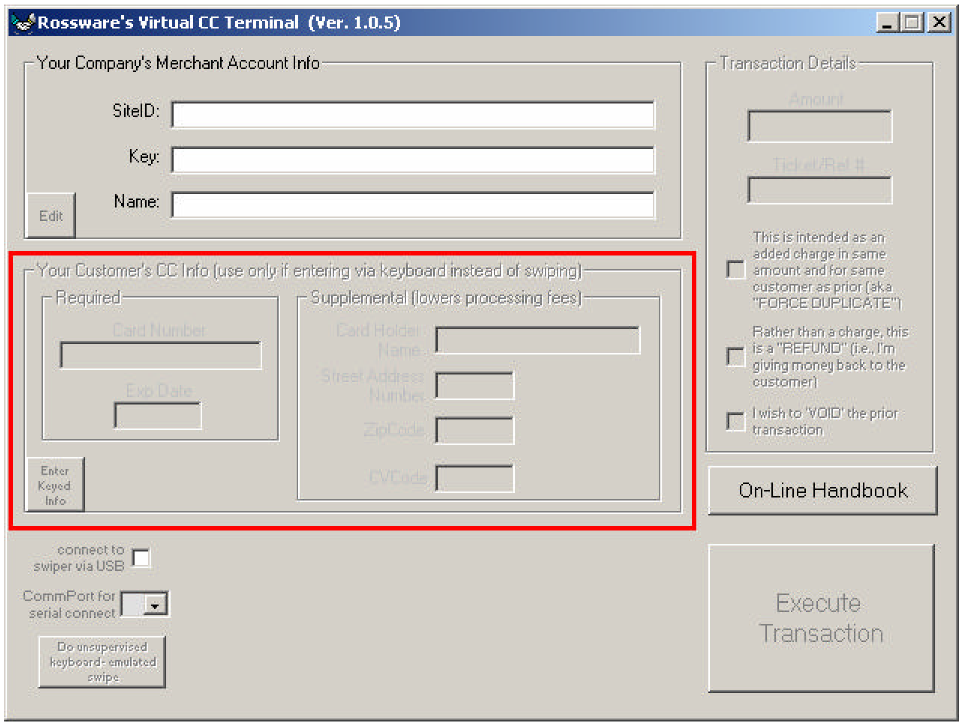
The VT GUI is divided into six functional areas.

The <u>first</u> provides textboxes for the user to insert his or her Merchant Credentials.


Rossware's Virtual CC Terminal (Ver. 1.0.5)
Your Company's Merchant Account Info
SiteID:
Key:
Name:
Edit
Transaction Details
Amount
Ticket/Ref #
This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")
Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)
I wish to 'VOID' the prior transaction
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info
On-Line Handbook
connect to swiper via USB
CommPort for serial connect
Do unsupervised keyboard-emulated swipe
Execute Transaction

These are the particular strings that identify and authenticate the user/merchant to MW. Since there are no major security concerns as connected to Merchant Credentials, VT direct-stores them (i.e., without encryption) in the platform's system registry.

The second section is where the user may type cardholder information for a keyed (card-not-present) transaction.


Rossware's Virtual CC Terminal (Ver. 1.0.5)
Your Company's Merchant Account Info
SiteID:
Key:
Name:
Edit
Transaction Details
Amount
Ticket/Ref #
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info
On-Line Handbook
connect to swiper via USB
CommPort for serial connect
Do unsupervised keyboard-emulated swipe
Execute Transaction

This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")

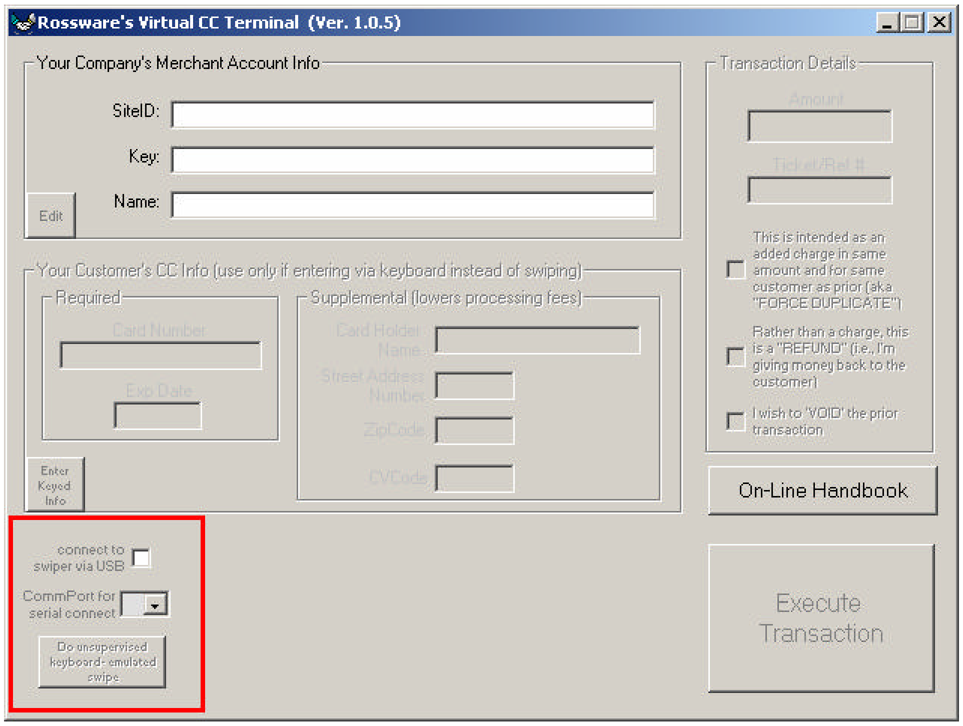
Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)

I wish to 'VOID' the prior transaction

The user may activate this section by clicking on the provided button. Information as typed here is never stored anywhere. It exists solely as text in the displayed textboxes – meaning when removed from those boxes (or when the boxes themselves are closed) the data is extinguished in its entirety — at least so far as any involvement by VT and/or its operating platform is concerned. This data's presence in/on the platform is thoroughly transitory, in other words, and is never in any circumstance written to non-volatile memory.

Of course, if used in a transaction with MW, the applicable text will be forwarded (as part of the transaction, and via a Secure Socket Layer connection) to that entity, and presumably may persist in MW systems to whatever extent MW allows.

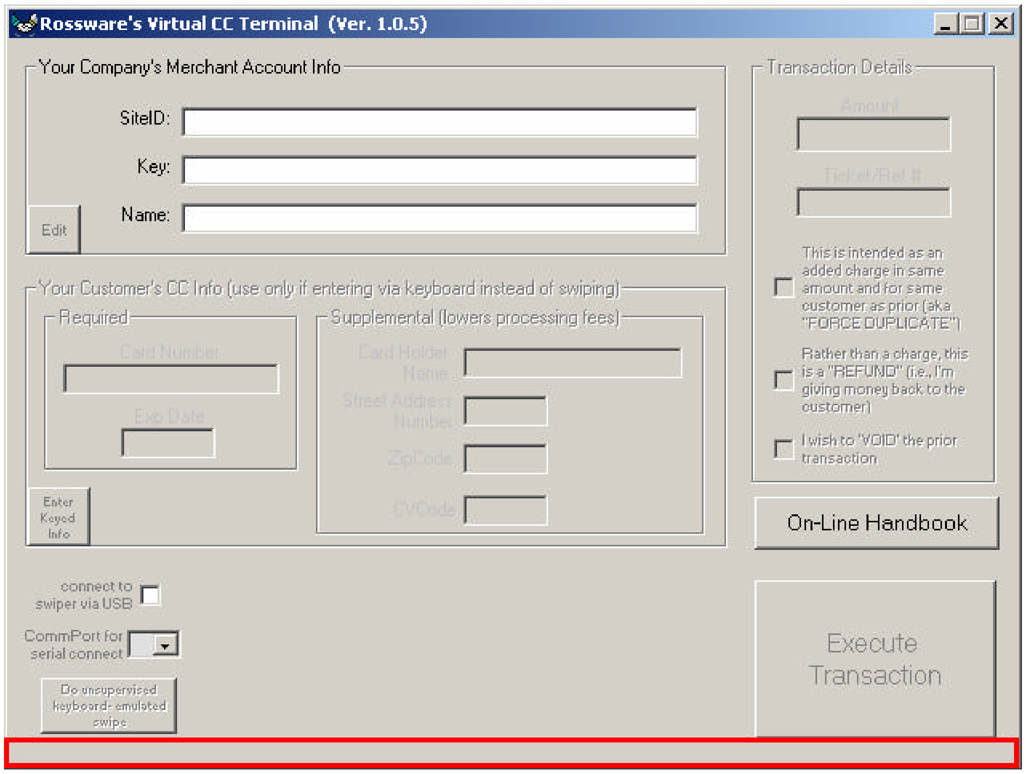
The third functional area is used in conjunction with connecting to an MCR device for card swiping.


Rossware's Virtual CC Terminal (Ver. 1.0.5)
Your Company's Merchant Account Info
SiteID:
Key:
Name:
Edit
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info
Transaction Details
Amount
Ticket/Ref #
This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")
Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)
I wish to 'VOID' the prior transaction
On-Line Handbook
connect to swiper via USB
CommPort for serial connect
Do unsupervised keyboard-emulated swipe
Execute Transaction

If the device connects via USB, the end-user needs to check the applicable button. In reaction, the VT scans for and connects to the device. If the device connects via serial port, the end-user should provide the correct CommPort number within the dropdown provided (and in reaction the VT will appropriately connect). The third option in this section (provided by a button that contains rather small print) is used to initiate a swipe with any MCR device that mimics keyboard-based input.

Much like Merchant Credentials, settings in this section are saved in the platform's system registry.

Though not visible as such in the illustration, the fourth functional area becomes a momentary-display textbox during and/or immediately after a swipe.


Rossware's Virtual CC Terminal (Ver. 1.0.5)
Your Company's Merchant Account Info
SiteID:
Key:
Name:
Edit
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info
Transaction Details
Amount
Ticket/Ref #
On-Line Handbook
connect to swiper via USB
CommPort for serial connect
Do unsupervised keyboard- emulated swipe
Execute Transaction

This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")

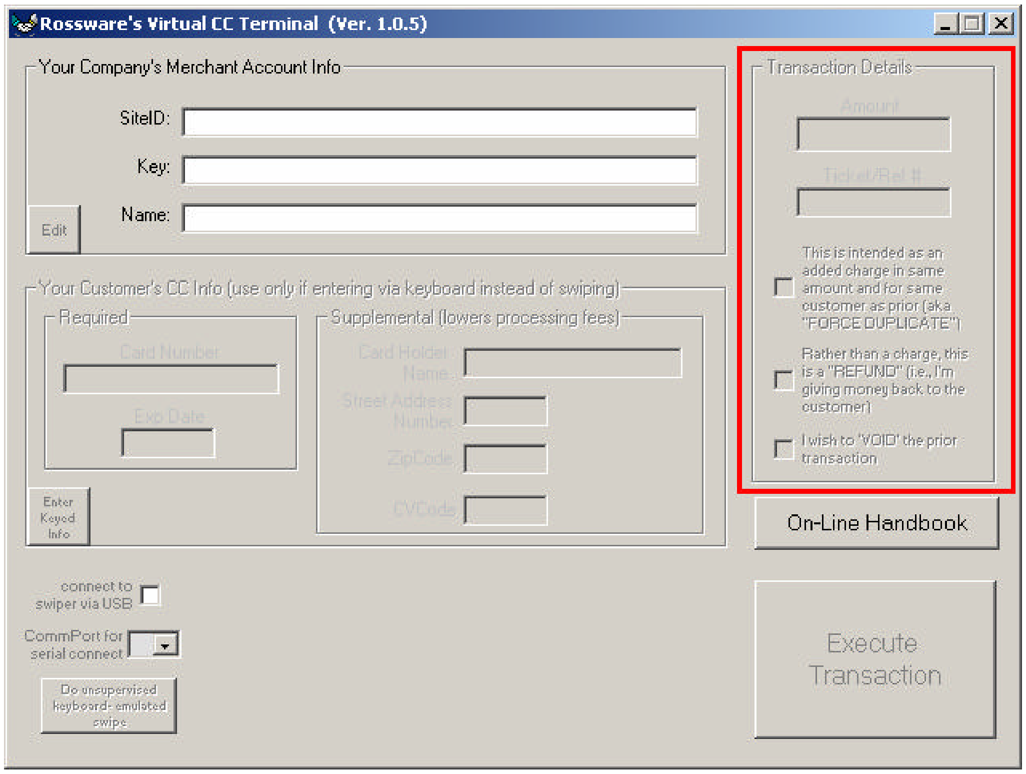
Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)

I wish to 'VOID' the prior transaction

This textbox contains actual data as swiped from the card (albeit in disguised-to-the-user format, with actual/true characters replaced by asterisks, much as in a typical password-input interface). Aside from when momentarily included in an actual transaction message (via Secure Socket Layer), this textbox is the only place such swiped data is ever placed by VT.

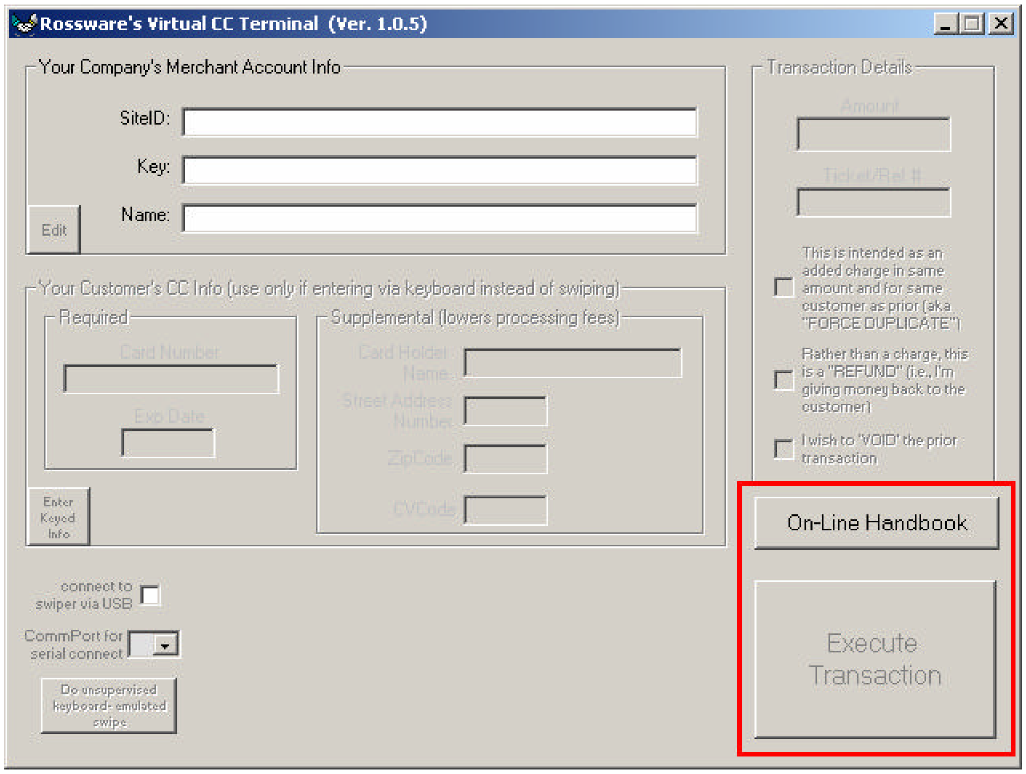
Just like with keyed-in data, this data is never “stored” anywhere. As soon as its containing box is closed (an object that’s maintained purely in volatile ram), the data is totally and thoroughly extinguished.

The <u>fifth</u> functional area is for specifying details about the desired transaction (i.e., amount, type, etc.).


Rossware's Virtual CC Terminal (Ver. 1.0.5)
Your Company's Merchant Account Info
SiteID:
Key:
Name:
Edit
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info
Transaction Details
Amount
Ticket/Ref #
This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")
Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)
I wish to 'VOID' the prior transaction
On-Line Handbook
connect to swiper via USB
CommPort for serial connect
Do unsupervised keyboard-emulated swipe
Execute Transaction

Though unrelated to security purposes, nothing in this section is stored. It is used solely for the duration (and as needed) for a particular transaction.

The <u>sixth</u> functional area contains two buttons.


Rossware's Virtual CC Terminal (Ver. 1.0.5)

Your Company's Merchant Account Info

SiteID:

Key:

Name:

Edit

Your Customer's CC Info (use only if entering via keyboard instead of swiping)

Required

Card Number

Exp Date

Supplemental (lowers processing fees)

Card Holder Name

Street Address Number

ZipCode

CVCode

Enter Keyed Info

connect to swiper via USB

CommPort for serial connect

Do unsupervised keyboard-emulated swipe

Transaction Details

Amount

Ticket/Ref #

This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")

Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)

I wish to 'VOID' the prior transaction

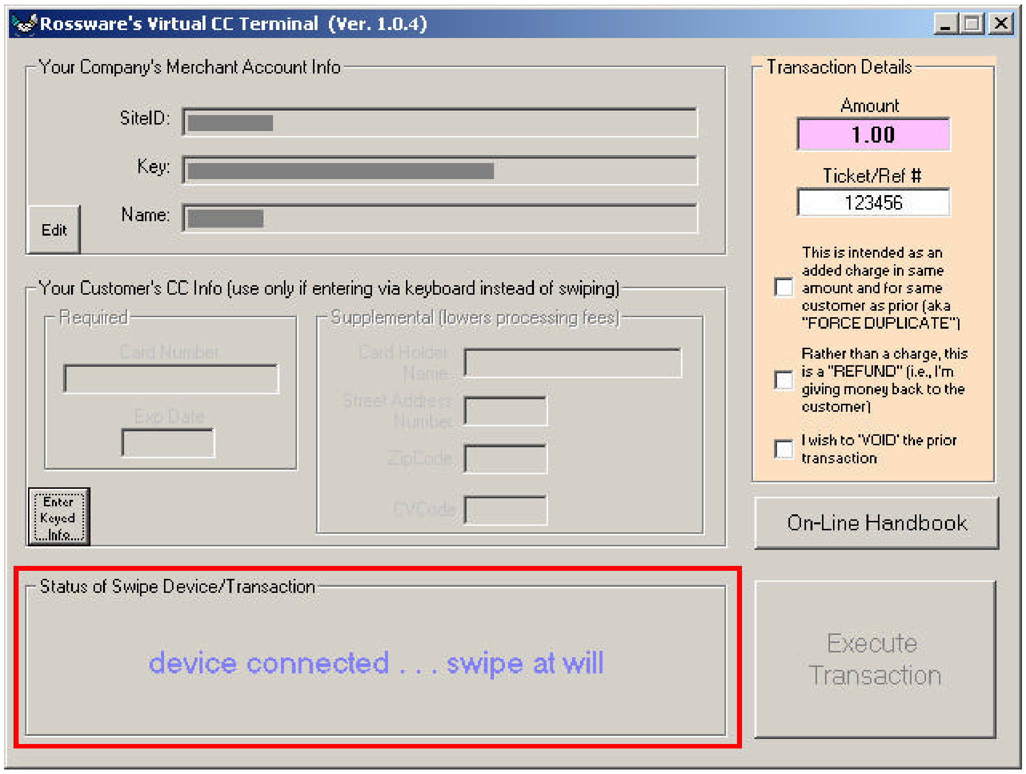
On-Line Handbook

Execute Transaction

One is for opening the user handbook (same as referenced on this document's first page).

The other will *activate*, and turn to an *operative green color*, when other elements in the GUI are appropriately setup for a transaction. The user clicks on it, of course, to initiate a transaction.

The <u>sixth</u> functional area (note that when functionally appropriate, it will supersede the area consumed by the *third* area) is used to communicate with the user, indicating such matters as when the VT is scanning for an MCR device, when one is located, when the setup is ready for a swipe, and so on.


Rossware's Virtual CC Terminal (Ver. 1.0.4)
Your Company's Merchant Account Info
Edit
Your Customer's CC Info (use only if entering via keyboard instead of swiping)
Required
Card Number
Exp Date
Supplemental (lowers processing fees)
Card Holder Name
Street Address Number
ZipCode
CVCode
Enter Keyed Info...
Transaction Details
On-Line Handbook
Execute Transaction

SiteID: [redacted]

Key: [redacted]

Name: [redacted]

Amount: 1.00

Ticket/Ref #: 123456

☐ This is intended as an added charge in same amount and for same customer as prior (aka "FORCE DUPLICATE")

☐ Rather than a charge, this is a "REFUND" (i.e., I'm giving money back to the customer)

☐ I wish to 'VOID' the prior transaction

Status of Swipe Device/Transaction: device connected . . . swipe at will

Additionally, once a successful transaction has been conducted, this area displays information concerning the result.

Chapter 6

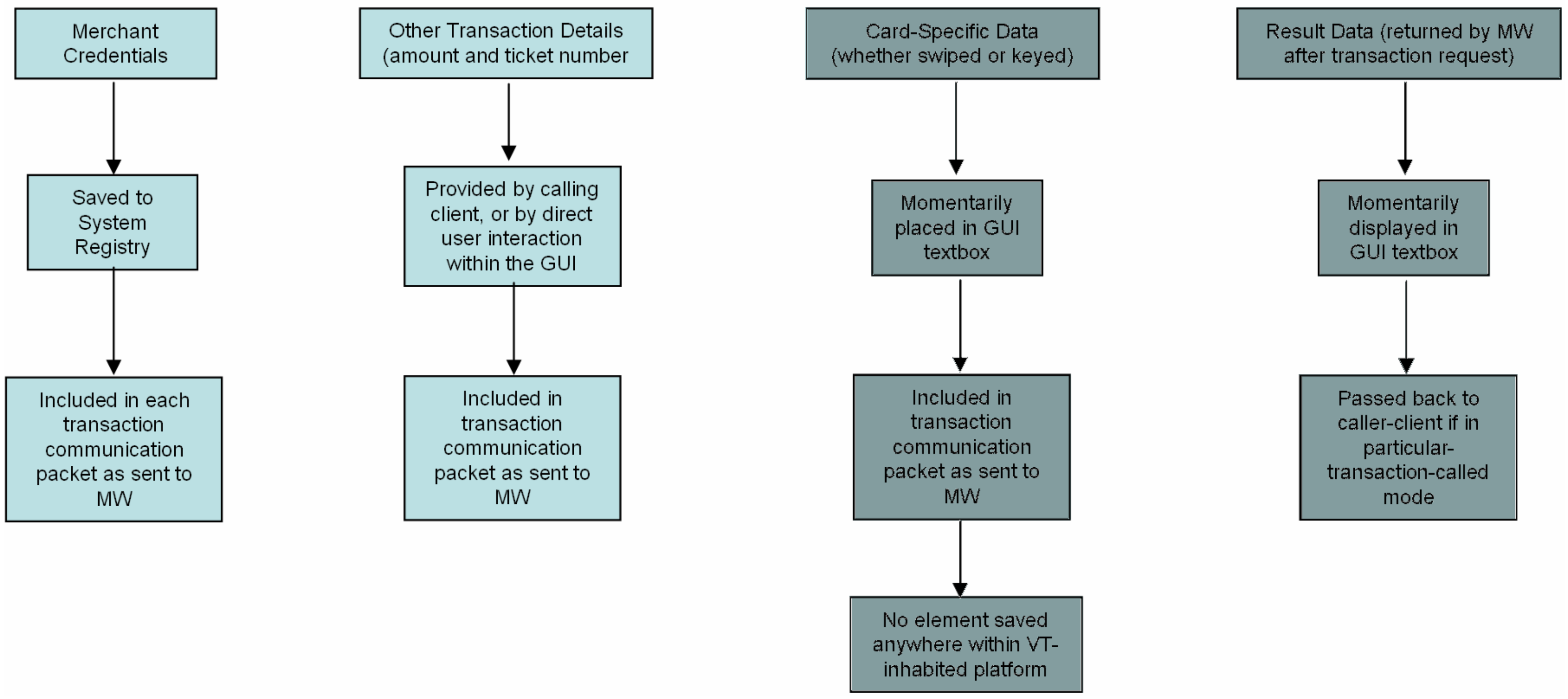
Under the Hood: How VT Completes a Transaction

This is another section that is not truly needed if your sole interest is in integrative use of VT. Much as it's easy to drive a car without knowing how the engine works, details here are not required reading.

VT uses MW's *Merchantware 3.0* interface.

Under that system, communication between VT and MW (client and gateway) is conducted using web services via an *https secure socket layer* (ssl) connection. This connection encrypts the communication stream, assuring impossibility of any meaningful interception.

Data Flow Diagram for VirtualTerminal.ocx


Merchant Credentials
Saved to System Registry
Included in each transaction communication packet as sent to MW
Other Transaction Details (amount and ticket number
Provided by calling client, or by direct user interaction within the GUI
Included in transaction communication packet as sent to MW
Card-Specific Data (whether swiped or keyed)
Momentarily placed in GUI textbox
Included in transaction communication packet as sent to MW
No element saved anywhere within VT-inhabited platform
Result Data (returned by MW after transaction request)
Momentarily displayed in GUI textbox
Passed back to caller-client if in particular-transaction-called mode