**EXISTING PIN PAD IMPLEMENTATIONS**

Used for Ingenico Ipp320 (Paymentec) and Ingenico Ipp320 (Moneris).

The same model used for both is basically the same, the back-end AMS DLL is different.

Overview:

* CreditCardInput.htx calls Javascript, which calls applet (APDInterfaceApplet,) which sends request through JNIDotNetBridge which handles communications between our Java applet and AMS DLL.

Existing Pin Pad Interfaces to AMS DLL:

* APDInterface -> JNIDotNetBridge
* APDInterfaceMoneris -> JNIDotNetBridge

Parameters passed to existing AMS DLL from JNIDotNetBridge:

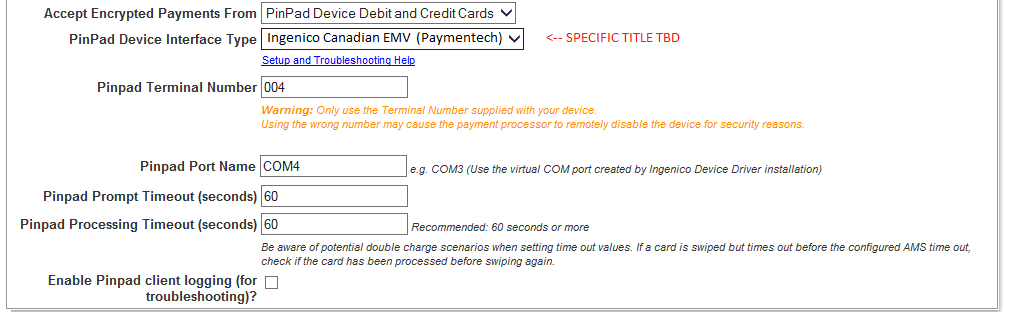
* TransactionType: SALE / AUTH / CREDIT / CAPTURE / VOID / WALLET SAVE
* TenderType: DEBIT / CC\_RETAIL (CREDIT CARD)
* AMSURL: From Financial Configuration (or site override)
* AMSPort: From Financial Configuration
* AMSTimeout: From Financial Configuration
* CardAcceptorName: Org name (CreditAuthorization.organizationName)
* TerminalLocationAddress: Workstation site address1 + address2
* TerminalLocationCity: Workstation site city
* TerminalLocationState: Workstation site state
* TerminalLocationZIP: Workstation site zipcode
* MerchantDescriptor: AMS merchant name from Financial Configuration (or site override)
* MerchantUser: AMS logon from Financial Configuration (or site override)
* MerchantPassword: AMS password from Financial Configuration (or site override)
* TerminalNumber: From workstation pinpad configuration (if applicable)
* USBPortName: From workstation pinpad configuration
* ClientTransactionID: Unique request id (randomUUID)
* CCAddress: Payer (cardholder) address
* CCZIP: Payer (cardholder) zipcode
* OrderAmount: Amount of charge (or $0.11 when getting wallet ID)
* OrderDescriptor: Org name (CreditAuthorization.organizationName)
* OrderID: Receipt header ID (CreditAuthorization.getNewOrderId(receipt\_header\_id)
* RetentionDate: TBD (not used?)
* AccountID: Existing wallet ID (if applicable, for credit/capture/etc.)
* ReferenceID: Previous AMS reference ID (if applicable, for credit/capture/etc.)
* ManualEntry: Did user request to manually key in value on pinpad rather than swipe
* PinpadDataEntryTimeout: From workstation pinpad configuration
* PinpadPINEntryTimeout: From workstation pinpad configuration
* PinpadSwipeCardTimeout: From workstation pinpad configuration
* WriteToLog: From workstation pinpad configuration

Response data received back from existing AMS DLL:

* TransactionMessage: Result data
* CCGResultCode: Result data
* CCGAUTHCode: Result data
* CCGAVSAddress: Result code for AVS address check
* CCGAVSZIP: Result code for AVS zipcode check
* CCGCSC: Result data
* TransactionResultCode: Result data
* AMSTransactionID: Result data
* CardType: Visa/MC/etc.
* LastFourDigits: Stored for display on receipts, etc.
* AccountID: Wallet ID
* MerchantCTR: Payment transaction “receipt text”, will be displayed in applet log, and captured to record in ICVERIFYLOG.ICVERIFYRESPONSE. Not used otherwise.
* CustomerCTR: Not used.

**INTEGRATION WITH NEW AMS DLL (ActiveEVM.DLL)**

New ActiveNet Workstation configuration:

****

**ActiveEVM DLL:**

REQUEST DATA

* Merchant (merchant parameters):

public String PaymenytPlatformURL { get; set; }

public int PaymentPlatformPort { get; set; }

public int PaymentPlatformTimeout { get; set; }

public String MerchantDescriptor { get; set; }

public String MerchantUser { get; set; }

public String MerchantPassword { get; set; }

//merchant should not have terminal number

//public string TerminalNumber { get; set; }

public string MerchantNumber { get; set; } 🡨 TBD, what is this and is it required? Should be optional, not set by Test App

public string ClientNumber { get; set; } 🡨 TBD, what is this and is it required? Should be optional, not set by Test App

* ActiveTerminal (terminal parameters):

public PaymentDeviceType Type { get; set; } 🡨 should default to PaymentDeviceType.IPP320

public PaymentDeviceIssuer Issuer { get; set; } 🡨 From Raymond: Please only use Paymentech

public bool IsUnattended { get; set; } 🡨 won’t use

public String CommPortName { get; set; }

public int MaxSwipes { get; set; } 🡨 won’t use

public bool UseManualEntry { get; set; }

public bool PromptForTip { get; set; } 🡨 won’t use

public bool E2EEEnabled { get; set; } 🡨 From Raymond: end to end encryption configuration. It is mandatory for this delivery. We have to turn it on always.

public int CommandTimeout { get; set; }

public int PromptTimeout { get; set; }

public String LocationName { get; set; }

public String LocationAddress { get; set; }

public String LocationCity { get; set; }

public String LocationState { get; set; }

public String LocationZIP { get; set; }

public String TerminalNumber { get; set; }

public Object CachedObject { get; set; } 🡨 probably won’t need

//public bool NeedWriteToFile { get; set; }

//public String LogPath { get; set; }

public PaymentDeviceLanguage Language { get; set; } 🡨 won’t use initially

* TransactionRequest (transaction parameters):

public string TransactionId { get; set; }

public string RefTransactionId { get; set; }

public string ClientTransactionId { get; set; }

public RequestType RequestType { get; set; } 🡨 sale, auth, credit, etc.

public string Amount { get; set; }

public string CardType { get; set; } 🡨 TBD, do we need to pass this in? Doesn’t look like it’s set by Test App

//public String RetentionDate { get; set; }

public string OrderDescriptor { get; set; }

public string OrderId { get; set; }

public bool IsManualEntry { get; set; }

public Merchant Merchant { get; set; }

public string AccountId { get; set; }

RESPONSE DATA

* TransactionResponse:

public String TransactionMessage { get; set; }

public String TransactionResultCode { get; set; }

public String ClientTransactionId { get; set; }

public String TransactionId { get; set; }

public String CCGResultCode { get; set; }

public String CCGAUTHCode { get; set; }

public String CCGTransactionId { get; set; }

public String CCGErrorMessage { get; set; }

public String CCGResponseMessage { get; set; }

public String LastFourDigits { get; set; }

public String CCExpMonth { get; set; } 🡨 NEW: CAPTURE THIS RESPONSE DATA

public String CCExpYear { get; set; } 🡨 NEW: CAPTURE THIS RESPONSE DATA

public String AccountID { get; set; }

public String Amount { get; set; }

public String MerchantCTR { get; set; }

public String CustomerCTR { get; set; }

public String CardType { get; set; }

public bool NeedVerify { get; set; } 🡨 probably won’t need

Integration Notes:

* Parameters for the new AMS ActiveEVM DLL seem to line up with parameters for the other AMS DLLs that ActiveNet integrates with.
* Looks like ActiveEVM DLL will return the card expiration month/year, if so we should probably capture this data, and at least record in ICVERIFYLOG if nothing else
* It would be simplest if we kept the same model (Java Applet -> new version of JNIDotNetBridge -> new AMS ActiveEVM DLL) as used for earlier implementations.
* Given that we use the same model, it would be simplest if the applet and JNIDotNetBridge structure (request/response and parameters) were left the same as the existing versions. Only the JNIDotNetBridge code that actually calls the new AMS DLL needs to be modified to map parameters as necessary, call the correct functions, and map response data as necessary.
* It looks like the previous AMS DLLs had simple REQUEST and RESPONSE objects. But the new ActiveEVM DLL has TransactionRequest, Merchant, and ActiveTerminal objects.
* TBD: Because of early deficiencies in the ActivePaymentDevice interface (used for the Ingenico ipp320 Paymentech pinpad integration, we had to implement the UI model where the receipt is finished FIRST, generating a temporary payment plan to be paid off, and then the UI is displayed to capture the card input and process the payment, which is applied against that temporary payment plan. The problem was that the AMS library didn’t initially allow us to issue an AUTH transaction in the client, followed by a SALE transaction from the server during normal receipt commit processing. So, since we didn’t want to process the user credit card until AFTER we were sure the receipt commit would work, we implemented the UI described. I believe the new AMS ActiveEVM library would let us issue an AUTH client side, and then server side receipt commit processing could issue the SALE transaction (or void the AUTH if necessary) during normal processing without having to create this temporary payment plan and collect payment AFTER the commit is finished. TBD: Should we just keep the model used for the other Ingenico pinpad devices, or refine it for this one?