ChargeItPro EasyIntegrator API – Card Storage Example

ChargeItPro can now be used to store credit card and ACH account numbers securely.

Transaction Types Acceptable for Use with Stored Cards

CreditSale
CreditReturn
ACHSale
ACHReturn
GenericVoid

Modes – Interactive or Silent

Interactive Mode is to be used in situations where a merchant wants to store a customer's account number so the merchant doesn't have to ask for the account number when the customers calls to place another order. In this mode, your system is responsible for storing a CustomerRef to identify the customer. The API will display a screen to allow the merchant to manage the cards attached to this customer.

Silent Mode is to be used in situations where a merchant needs to store accounts numbers so he/she can process transactions at the end of a billing cycle. Examples include gym memberships or Wine club subscriptions. In this mode, your system will be responsible for storing a CustomerRef and a PaymentID.

The examples below are divided between the different modes. Please see the appropriate section.

Steps of an Interactive Transaction

1. Configure Setup
2. Create New Customer
3. Process Transaction
4. Print Receipt/Record Transaction

Steps of a Silent Transaction

1. Configure Setup
2. Create New Customer
3. Add Payment
4. Process Transaction
5. Print Receipt/Record Transaction

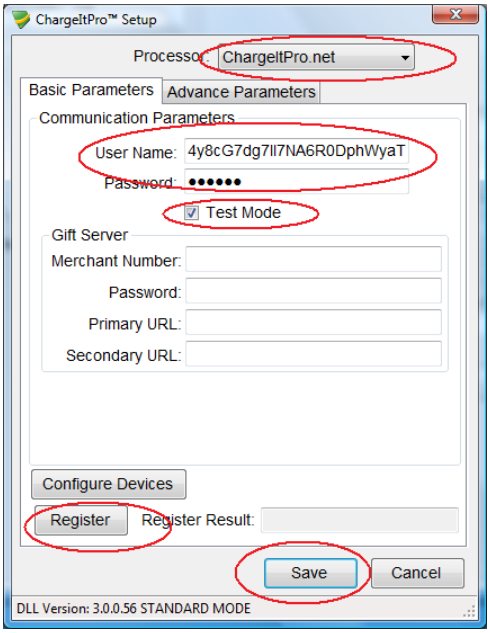
Step 1. Configure Setup - Interactive Mode

The ChargeItPro EasyIntegrator API has up until now communicated with a ChargeItPro Server running as a desktop application on a merchant's local network. In order to store credit card account numbers to use later, the EasyIntegrator API has to communicate directly with our hosted gateway. Follow the steps below to point EasyIntegrator to the gateway. If you are currently using the Setup screen, there is no need to modify your code.

The string returned by Setup should be saved per PC as it may contain settings for hardware specific to an individual computer.

```
StringSaveSetup = easyIntegrator.Setup(StringSaveSetup)
```

1. Change Processor to *ChargeItPro.Net.*
2. Enter your *Username* and *Password*.
3. Check *Test Mode*.
4. Use the *Register Button* to validate the Username and Password.
5. Click *Save*.

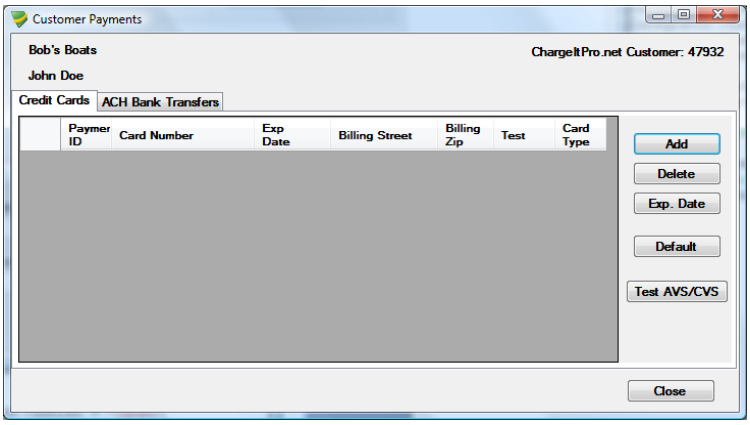


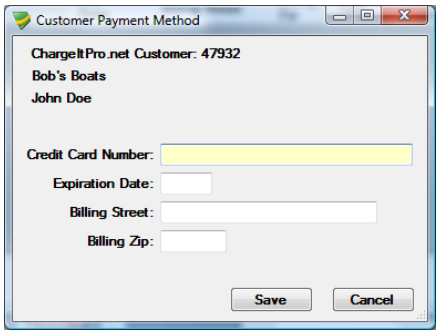
Step 2. Create New Customer - Interactive Mode

The following code will create a new customer on the gateway and display the screen below to allow the merchant to manage credit card account numbers or ACH Bank Transfer information.

Note: All fields for an existing customer get overwritten every time the UpdatePayments function is called. For example if you don't populate FirstName when you call UpdatePayment for the customer below the 2nd time, "John" would get deleted from the gateway.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
'Company or LastName is required.
easyIntegrator.Customer.CustomerFields.Company = "Bob's Boats"
easyIntegrator.Customer.CustomerFields.FirstName = "John"
easyIntegrator.Customer.CustomerFields.LastName = "Doe"
easyIntegrator.Customer.CustomerFields.CustomerID = "YourID 123"
easyIntegrator.Customer.CustomerFields.Phone = "123-456-7890"
easyIntegrator.Customer.CustomerFields.EMail = "ansonl@chargeitpro.com"
'use 0 to create new customer, otherwise pass number of customer to update
easyIntegrator.Customer.UpdatePayments("0")
'save to use later to process a payment for this customer
StringSavedCustomerRef = easyIntegrator.Customer.CustomerRef
```





Delete Customer- Interactive Mode

The following code will delete a customer from the gateway.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
easyIntegrator.customer.DeleteCustomer(StringSavedCustomerRef)
```

Step 3. Process Transaction - Interactive Mode

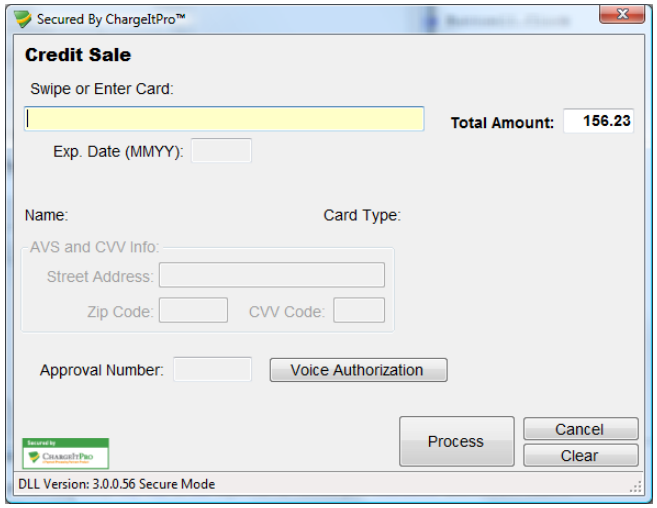
This code will display either of the 2 screens below depending on if a CustomerRef is populated. The first screen is the traditional screen where a cashier would swipe a customer's credit card. The 2nd screen allows the merchant to choose an account to process against. CreditSale returns a True or False depending on if the transaction was approved or not.

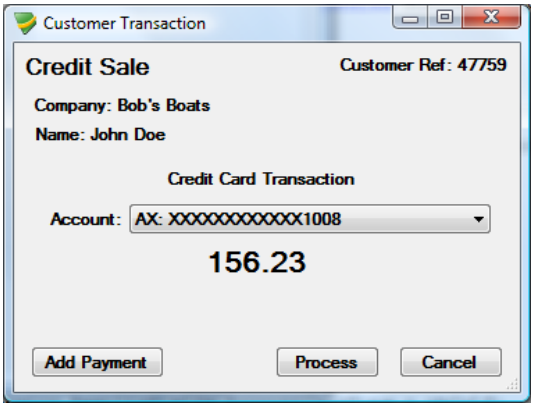
Valid transaction types include:

1. CreditSale
2. CreditReturn
3. GenericVoid
4. ACHSale
5. ACHReturn

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
easyIntegrator.transFields.Cashier = "Janet"
easyIntegrator.TransFields.TransactionReference = "YourReceiptNumber"
easyIntegrator.TransFields.AmountTotal = 156.23
'use 0 for one time payment
easyIntegrator.customer.CustomerRef = StringSavedCustomerRef
'add something similar to the bottom of your receipt
easyIntegrator.CreditSale()

'save to identify transaction to Void
StringSaveToVoid = easyIntegrator.resultsFields.UniqueTransID
```





Void a Previous Transaction

Only transactions in the open batch can be voided. Batches are closed at a user defined time. Voids will not appear on a customer's statement whereas a return will appear.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
'identify transaction to void
easyIntegrator.transFields.UniqueTransRef = StringSaveToVoid
easyIntegrator.GenericVoid()
```

Step 4. Print Receipt/Record Transaction - Interactive Mode

After a transaction as been processed, fields you may want to save or print on the bottom of your receipt are returned in ResultsFields.

The example below is using:
easyIntegrator.ResultsFields.MaskedAccount
easyIntegrator.ResultsFields.AccountCardType
easyIntegrator.ResultsFields.TransactionType
easyIntegrator.ResultsFields.ApprovalNumberResult

```
Acct: XXXXXXXXXXXX1008
CardType: AX
Transaction Type: CreditSale
Approval Code: 034823

   I AGREE TO PAY ABOVE
  TOTAL AMOUNT ACCORDING
 TO CARD ISSUER AGREEMENT


X_____________________________
```

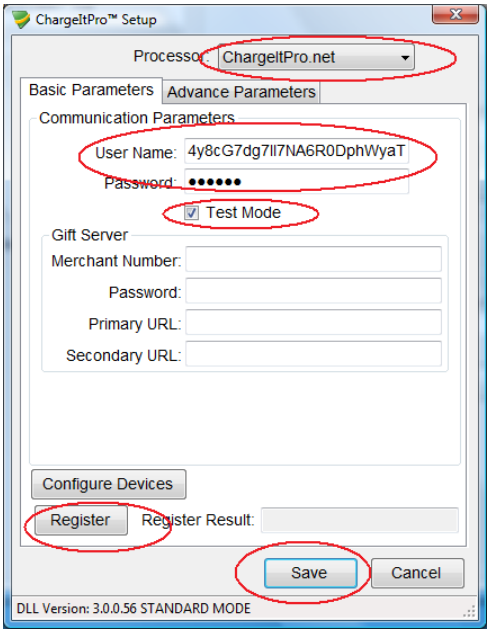
Step 1. Configure Setup - Silent Mode

The ChargeItPro EasyIntegrator API has up until now communicated with a ChargeItPro Server running as a desktop application on a merchant's local network. In order to store credit card account numbers to use later, the EasyIntegrator API has to communicate directly with our hosted gateway. Follow the steps below to point EasyIntegrator to the gateway. If you are currently using the Setup screen, there is no need to modify your code.

The string returned by Setup should be saved per PC as it may contain settings for hardware specific to an individual computer.

```
StringSaveSetup = easyIntegrator.Setup(StringSaveSetup)
```

1. Change Processor to *ChargeItPro.Net.*
2. Enter your *Username* and *Password*.
3. Check *Test Mode*.
4. Use the *Register Button* to validate the Username and Password.
5. Click *Save*.



Step 2. Create New Customer – Silent Mode

The following code will create a new customer on the gateway and not display anything. Unlike Interactive Mode, you will have to create your own screen to manage payments within your system.

Note: All fields for an existing customer get overwritten every time the UpdateCustomer function is called. For example if you don't populate FirstName when you call UpdateCustomer for the customer below the 2nd time, "John" would get deleted from the gateway.

Note: The only difference here between Interactive Mode and Silent Mode is that UpdatePayments is used in Interactive Mode while UpdateCustomer is used in Silent Mode.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
'Company or LastName is required.
easyIntegrator.Customer.CustomerFields.Company = "Bob's Boats"
easyIntegrator.Customer.CustomerFields.FirstName = "John"
easyIntegrator.Customer.CustomerFields.LastName = "Doe"
easyIntegrator.Customer.CustomerFields.CustomerID = "YourID 123"
easyIntegrator.Customer.CustomerFields.Phone = "123-456-7890"
easyIntegrator.Customer.CustomerFields.EMail = "ansonl@chargeitpro.com"
'use 0 to create new customer, otherwise pass number of customer to update
easyIntegrator.Customer.UpdateCustomer("0")
'save to use later to process a payment for this customer
StringSavedCustomerRef = easyIntegrator.Customer.CustomerRef
```

Delete Customer- Silent Mode

The following code will delete a customer from the gateway.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
easyIntegrator.customer.DeleteCustomer(StringSavedCustomerRef)
```

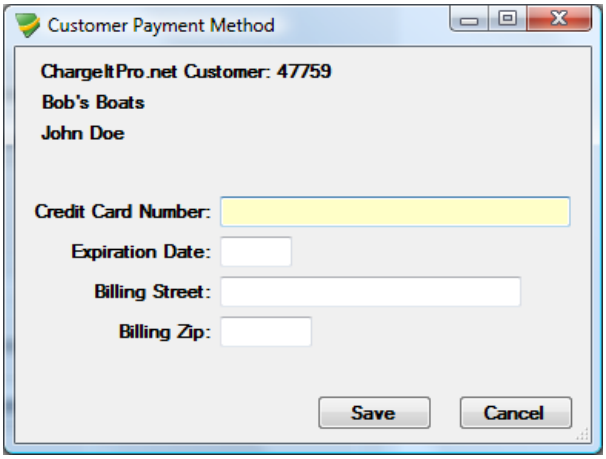
Step 3. Add Payment – Silent Mode

After creating a customer on the gateway and storing the CustomerRef, the next step is to associate a credit card to that customer. A customer can have multiple credit cards and ACH account numbers associated with their account.

Please save PaymentID, AccountCardType and MaskedAccount to use in your customer payment interface and to use to print receipts/transaction reports. Silent Mode will not return the AccountCardType nor MaskedAccount.

To add a ACH account number use AddCheckPayment instead of AddCreditPayment

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
easyIntegrator.customer.AddCreditPayment(StringSavedCustomerRef)
'save these to use later
StringSavePaymentID = easyIntegrator.customer.PaymentID
StringSaveAccountCardType = easyIntegrator.resultsFields.AccountCardType
StringSaveMaskedAccount = easyIntegrator.resultsFields.MaskedAccount
```



Delete Payment

```
easyIntegrator.LoadSetup(TextBoxSetup.Text)
easyIntegrator.Clear()
easyIntegrator.customer.DeletePayment(StringSavedCustomerRef,
TextBoxPaymentID.Text)
```

Edit Expiration Date

```
easyIntegrator.LoadSetup(TextBoxSetup.Text)
easyIntegrator.Clear()
easyIntegrator.customer.UpdateExpirationDate(StringSavedCustomerRef,
StringSavePaymentID)
```

Step 4. Process Transaction - Silent Mode

CreditSale returns a True or False depending on if the transaction was approved or not. Note that once you populate CustomerRef and PaymentID, the transaction will process silently.

Valid transaction types include:

1. CreditSale
2. CreditReturn
3. GenericVoid
4. ACHSale
5. ACHReturn

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
easyIntegrator.transFields.Cashier = "Janet"
easyIntegrator.TransFields.TransactionReference = "YourReceiptNumber"
easyIntegrator.TransFields.AmountTotal = 156.23
'identify customer and card to charge
easyIntegrator.customer.CustomerRef = StringSavedCustomerRef
easyIntegrator.customer.PaymentID = StringSavePaymentID
easyIntegrator.CreditSale()

'save to identify transaction to Void
StringSaveToVoid = easyIntegrator.resultsFields.UniqueTransID
```

Void a Previous Transaction

Only transactions in the open batch can be voided. Batches are closed at a user defined time. Voids will not appear on a customer's statement whereas a return will appear.

```
easyIntegrator.LoadSetup(StringSaveSetup)
easyIntegrator.Clear()
'identify transaction to void
easyIntegrator.transFields.UniqueTransRef = StringSaveToVoid
easyIntegrator.GenericVoid()
```

Step 5. Print Receipt/Record Transaction – Silent Mode

To generate a printed receipt you cannot use
easyIntegrator.ResultsFields.MaskedAccount
easyIntegrator.ResultsFields.AccountCardType

You must use the values you saved in Step 3 when the credit card was stored on the gateway. This was implemented to save a few seconds per transaction.

The example below is using:
StringSaveAccountCardType
StringSaveMaskedAccount
easyIntegrator.ResultsFields.TransactionType
easyIntegrator.ResultsFields.ApprovalNumberResult

Acct: XXXXXXXXXXXX1008
CardType: AX
Transaction Type: CreditSale
Approval Code: 034823

I AGREE TO PAY ABOVE
TOTAL AMOUNT ACCORDING
TO CARD ISSUER AGREEMENT

X_____