Auth Workflow and touchpoints

**Proxy request coming in**

{

"UserToken": "eyJ0eXAiOiJKV1QiLC",

"Request": {

"BuyerID": "12345",

"OrderID": "99996",

"TransactionType": "authCaptureTransaction",

"Amount": 10,

“CorporateCard”:false,

“Assignments”:{

“Users”:[

{“UserID”:”725bh6”}

],

“UserGroups”:[

{“UserGroupID”:”725bh6”}

]

},

"CardDetails": {

"CardholderName": "Bob Tom",

"PaymentID": null,

"CreditCardID": null,

"CardType": "visa",

"CardNumber": "4007000000027",

"ExpirationDate": "1220",

"CardCode": "001"

}

},

"ConfigData": {

"APILoginID": "43aKN8hqh8",

"TransactionKey": "2HbqQx483859rEtS"

}

}

**createCreditCard**

Use this function to create a new credit card for an existing customer profile or create a new credit card and a new payment profile at the same time.

Resources used

* public class CreateCustomerProfile
* public class CreateCustomerPaymentProfile
* /me
* /me/creditcards
* /buyers/[buyerID]/creditcards
* /buyers/[buyerID]/creditcards/assignments
* check GET /me for the xp object AuthorizeNetProfileID
* If null use CreateCustomerProfile in the SDK
  + Write the Four51 userID to Billto.firstName (for tracking it back, merchantCustomerID has a max of 20 characters)
  + Use CC info to createCustomerPaymentProfileRequest
  + In the response use customerProfileID and set /me xp.AuthorizeNetProfileID
  + In response POST to /me/creditcards and write customerPaymentProfileIdList (there should only be one since it’s a new profile)

{

"ID": "IIMHbKD6p0-7u7ej4MwGzg", auto generate

"Token": "645asdf654", customerPaymentProfileIdList

"CardType": "VISA", - type from the request

"PartialAccountNumber": "1234",

"CardholderName": "John Doe",

"ExpirationDate": "2020-01-01T06:00:00+00:00",

"xp": null

}

* If not null use value in AuthorizeNetProfileID
  + Use CC info to createCustomerPaymentProfileRequest
  + In response POST to /me/creditcards

**updateCreditCard**

Use this function to update a credit card for an existing customer profile.

Resources used

* public class UpdateCustomerPaymentProfile
* /me/creditcards/[creditCardID]
* /buyers/[buyerID]/creditcards
* Use the CreditCardID and use perform a GET /me/creditcards/[creditCardID] to find the token value which is customerPaymentProfileID on Authorize.Net.
* Update Authorize.Net
  + If only updating the expiration you will need to pass in a masked value of XXXX and the last 4 of the credit card number. XXXX1234
  + If only updating the CC number you will have to pass a mask of XXXX for expiration date
* Updating Four51
  + After a success response from Authorize.net PATCH me/creditcards/[creditCardID] with the updated information. i.e (expiration, last 4)
* Error Handling
  + If an error is received from Authorize pass back response.
  + If error comes from Four51 then the card at Authorize has to be reverted back using the data from Four51 and the error sent back in the response

**deleteCreditCard**

Use this function to delete a credit card from an existing customer profile

Resources used

* public class DeleteCustomerPaymentProfile
* /me/creditcards/[creditCardID]
* /buyers/[buyerID]/creditcards
* /buyers/[buyerID]/creditcards/[creditCardID]
* Use the CreditCardID and use perform a GET /me/creditcards/[creditCardID] or /buyers/[buyerID]/creditcards/[creditCardID] to find the token value which is customerPaymentProfileID on Authorize.Net.
* Delete card from Authorize
* Delete card from Four51
* If error from Authorize pass response back. If error from Four51 pass back response

**authCaptureTransaction**

Use this method to authorize and capture a credit card payment. A payment profile or a one-time credit card can be used.

Resources used

* public class ChargeCreditCard
* /buyers/[buyerID]/orders/[orderID]/payments
* /buyers/[buyerID]/orders/[orderID]/payments/transactions
* /me
* /me/creditcards/[creditCardID]
* /buyers/[buyerID]/creditcards/[creditCardID]
* Using a one-time use card
  + Authorize.Net
    - Use the credit card number, exp date, card code, and amount from the request to use in the ChargeCreditCard class
    - Use CC number, exp date, card code for var creditCard = new creditCardType
    - Use orderID for var order= new orderType (invoiceNumber)
* Using a saved CC
  + Authorize.Net
    - Use the creditCardID in the request to call GET /me/creditcards/[creditCardID] or /buyers/[buyerID]/creditcards/[creditCardID] to grab the token from the CC object.
* Four51
  + On successful charge use responseCode, transID, transactionResponse.messages.description, and messages.resultCode
  + Create a payment on Four51
    - Type = ‘CreditCard’
    - DateCreated = Now()
    - CreditCardID = if !null creditCardID from request
    - Description = ‘Authorize.Net’
    - Amount = Amount from request
* Create a transaction on the payment
  + Type = transType from request
  + DateExecuted = Now()
  + Amount = amount from request
  + Succeeded = true if resultCode == ‘ok’
  + ResultCode = responseCode
  + ResultMessage = responseCode 1 = Approved 2 = Declined 3 = Error 4 = Held for Review
  + xp.transID = transactionResponse.transID
  + xp.AuthCode = transactionResponse.authCode
  + xp.messages.ResultCode = messages.resultCode
  + xp.Message.code = messages.message.code
  + xp.Message.Text = messages.message.text
* Error handling
  + If there is an error from Authorize pass the response on
  + If there is an error in creating either the payment or transaction the charge in Authorize will need to be voided.(see voidTransaction)

**authOnlyTransaction**

Use this method to authorize a credit card payment. To actually charge the funds you will need to follow up with a capture transaction.

Resources used

* public class AuthorizeCreditCard
* /buyers/[buyerID]/orders/[orderID]/payments
* /buyers/[buyerID]/orders/[orderID]/payments/transactions
* /buyers/[buyerID]/creditcards
* /me
* /me/creditcards/[creditCardID]
* /buyers/[buyerID]/creditcards/[creditCardID]
* Using a one-time use card
  + Authorize.Net
    - Use the credit card number, exp date, card code, and amount from the request to use in the AuthorizeCreditCard class
    - Use CC number, exp date, card code for var creditCard = new creditCardType
    - Use orderID for var order= new orderType (invoiceNumber)
* Using a saved CC
  + Authorize.Net
    - Use the creditCardID in the request to call GET /me/creditcards/[creditCardID] or /buyers/[buyerID]/creditcards/[creditCardID] to grab the token from the CC object.
* Four51
  + On successful authorize use responseCode, transID, transactionResponse.messages.description, and messages.resultCode
  + Create a payment on Four51
    - Type = ‘CreditCard’
    - DateCreated = Now()
    - CreditCardID = if !null creditCardID from request
    - Description = ‘Authorize.Net’
    - Amount = Amount from request
* Create a transaction on the payment
  + Type = transType from request
  + DateExecuted = Now()
  + Amount = amount from request
  + Succeeded = true if resultCode == ‘ok’
  + ResultCode = responseCode
  + ResultMessage = responseCode 1 = Approved 2 = Declined 3 = Error 4 = Held for Review
  + xp.transID = transactionRequest.transID
  + xp.AuthCode = transactionResponse.authCode
  + xp.messages.ResultCode = messages.resultCode
  + xp.Message.code = messages.message.code
  + xp.Message.Text = messages.message.text
* Error handling
  + If there is an error from Authorize pass the response on
  + If there is an error in creating either the payment or transaction the charge in Authorize will need to be voided.(see voidTransaction)

**priorAuthCaptureTransaction**

Use this method to capture funds for a transaction that was previously authorized using authOnlyTransaction.

Resources used

* public class CapturePreviouslyAuthorizedAmount
* /buyers/[buyerID]/orders/[orderID]/payments
* /buyers/[buyerID]/orders/[orderID]/payments/transactions
* /me
* Four51
  + Using the paymentID sent in the request perform a GET on the payment and find the transaction with the Type of authOnlyTransaction and then collect xp.transID to use as the refTransID in the call to Authorize.Net
* Authorize.Net
  + Using the transID from the payment set that to the refTransID in the call for the priorAuthCaptureTransaction
* Four51
  + On successful capture use responseCode, transID, transactionResponse.messages.description, and messages.resultCode
* Create a transaction on the payment using the paymentID from the request
  + Type = transType from request
  + DateExecuted = Now()
  + Amount = amount from request
  + Succeeded = true if resultCode == ‘ok’
  + ResultCode = responseCode
  + ResultMessage = responseCode 1 = Approved 2 = Declined 3 = Error 4 = Held for Review
  + xp.transID = transactionRequest.transID
  + xp.AuthCode = transactionResponse.authCode
  + xp.messages.ResultCode = messages.resultCode
  + xp.Message.code = messages.message.code
  + xp.Message.Text = messages.message.text
* Error handling
  + If there is an error from Authorize pass the response on
  + If there is an error in creating either the transaction the capture in Authorize will need to be voided.(see voidTransaction)

**refundTransaction**

This transaction type is used to refund a customer for a transaction that was originally processed and successfully settled through the payment gateway.

Resources used

* public class RefundTransaction
* /buyers/[buyerID]/orders/[orderID]/payments
* /buyers/[buyerID]/orders/[orderID]/payments/transactions
* /me
* /me/creditcards/[creditCardID]
* /buyers/[buyerID]/creditcards/[creditCardID]
* Using a one-time use card
  + Authorize.Net
    - Use CC last 4 numbers, exp date, and amount from the request to use in the RefundTransaction class
* Using a saved CC
  + Authorize.Net
    - Use the creditCardID in the request to call GET /me/creditcards/[creditCardID] or /buyers/[buyerID]/creditcards/[creditCardID] to grab the token from the CC object.
* Four51
  + Using the paymentID sent in the request perform a GET on the payment and find the transaction with the Type of priorAuthCaptureTransaction or authCaptureTransaction and then collect xp.transID to use as the refTransID in the call to Authorize.Net
* Authorize.Net
  + Using the transID from the payment set that to the refTransID in the call for the refundTransaction
* Four51
  + On successful refund use responseCode, transID, transactionResponse.messages.description, and messages.resultCode
* Create a transaction on the payment using the paymentID from the request
  + Type = transType from request
  + DateExecuted = Now()
  + Amount = amount from request
  + Succeeded = true if resultCode == ‘ok’
  + ResultCode = responseCode
  + ResultMessage = responseCode 1 = Approved 2 = Declined 3 = Error 4 = Held for Review
  + xp.transID = transactionRequest.transID
  + xp.AuthCode = transactionResponse.authCode
  + xp.messages.ResultCode = messages.resultCode
  + xp.Message.code = messages.message.code
  + xp.Message.Text = messages.message.text
* Error handling
  + If there is an error from Authorize pass the response on
  + If there is an error in creating the transaction, the refund in Authorize will need to be voided.(see voidTransaction)

**voidTransaction**

This transaction type can be used to cancel either an original transaction that is not yet settled or an entire order composed of more than one transaction. A Void prevents the transaction or the order from being sent for settlement. A Void can be submitted against any other transaction type.

Resources used

* public class VoidTransaction
* /buyers/[buyerID]/orders/[orderID]/payments
* /buyers/[buyerID]/orders/[orderID]/payments/transactions
* /me
* Four51
  + Using the paymentID sent in the request perform a GET on the payment and find the transaction with the Type of priorAuthCaptureTransaction or authCaptureTransaction.
  + and then collect xp.transID to use as the refTransID in the call to Authorize.Net
* Authorize.Net
  + Using the transID from the payment set that to the refTransID in the call for the voidTransaction
* Four51
  + On successful void use responseCode, transID, transactionResponse.messages.description, and messages.resultCode
* Create a transaction on the payment using the paymentID from the request
  + Type = transType from request
  + DateExecuted = Now()
  + Amount = amount from request
  + Succeeded = true if resultCode == ‘ok’
  + ResultCode = responseCode
  + ResultMessage = responseCode 1 = Approved 2 = Declined 3 = Error 4 = Held for Review
  + xp.transID = transactionRequest.transID
  + xp.AuthCode = transactionResponse.authCode
  + xp.messages.ResultCode = messages.resultCode
  + xp.Message.code = messages.message.code
  + xp.Message.Text = messages.message.text
* Error handling
  + If there is an error from Authorize pass the response on