Pay with Rewards

Kibo eCommerce enables shoppers to pay for products using your rewards program. Shoppers can now use credit earned through your rewards program to pay for products online.

Use Cases

A shopper has registered with your stores' rewards program and wants to use rewards points or vouchers to pay for a product on your Kibo eCommerce site.

Feature Limitations

Currently the Kibo eCommerce system can integrate with external customer rewards systems and process customer rewards and vouchers as payment, but it cannot create or manage customer rewards profiles.

Enable Pay with Rewards

This feature requires a combination of theme changes and API Extension actions to enable.

Update your Theme

Ask your theme developer to make the required changes to your theme to enable this feature, as detailed in the GitHub Pull Request.

These sample theme changes require you to create a storefront order attribute.

API Extension Actions to Implement

Implement the following API Extension actions in order to integrate Kibo eCommerce with your rewards system:

getCredits.before or getCredits.after

Use this code sample as a reference when implementing either getCredits.before (if using only custom purchase rewards on site) or getCredit.after (if using both store credits and custom purchase rewards on site).

```
/*
--Custom purchase rewards functionality leveraging NG store credit model--

This is a sample action to add additional custom rewards to NG store credits
creditType on NG Store credit object will need to be set to 'Custom'
*/
var _ = require('underscore');
var CustomerAccountFactory = require('mozu-node-sdk/clients/commerce/customer/accounts/cust
omerAttribute');
var OrderAttributeFactory = require('mozu-node-sdk/clients/commerce/orders/orderAttribute');

module.exports = function (context, callback) {
var response = context.response.body;
```

```
var response = context.response.body;
var requestParams = context.request.params;
var customerId = null;
console.log("Filter : " + requestParams.filter);

if (!requestParams.filter) {
console.log("No filter present! Calling back..");
callback();
return;
}

var customerIdString = requestParams.filter.match(/(CustomerId eq ["]?\d+["]?)/gi);
if (customerIdString) {
//Check if the reward no. is stored on a customer attribute.
console.log("Found Customer Id in the filter. Looking for a customer attribute to find reward no.");
var id = customerIdString[0].match(/\d+/);
customerId = id[0];
var customerResource = CustomerAccountFactory(context.apiContext);
customerResource.context['user-claims'] = null;
customerResource.getAccountAttribute({ accountId: customerId, attributeFQN: "tenant~purchase-
reward-number" }) //replace attribute FQN with the correct name that you setup
.then(function (attribute) {
var purchaseRewardNo = attribute.values[>0].toString();
console.log("purchase Reward No:" + purchaseRewardNo);
var purchaseRewards = getPurchaseRewards(purchaseRewardNo);
var responseItems = response.items;
var responseItemsWithpurchaseRewards = responseItems.concat(purchaseRewards);
context.response.body.items = responseItemsWithpurchaseRewards;
response.totalCount = response.totalCount + purchaseRewards.length;
callback();
})
.catch(function (err) {
console.error(err);
callback();
});
}
else {

console.log("Found Code comparision filter, extracting purchase reward no.");
var codeString = requestParams.filter.match(/(Code eq ["]?.\w+["]?)/gi)[0].replace(/"/g, "").split(" ")
;
customerId = codeString[2];
console.log("Purchase Reward No:" + customerId);
var purchaseRewards = getPurchaseRewards(customerId);
var responseItems = response.items;
var responseItemsWithPurchaseRewards = responseItems.concat(purchaseRewards);
context.response.body.items = responseItemsWithPurchaseRewards;
response.totalCount = response.totalCount + purchaseRewards.length;
callback();
}
};

function getPurchaseRewards(rewardNo) {

//Call your external system to retreive rewards and map it to the Credit Object on NG.
//Sample rewards with CreditType set to 'Custom'

var rewards = [
{
"code": "135646846",
```

```
"activationDate": "2017-12-03T06:00:00.000Z",
"creditType": "Custom", //This needs to 'Custom' for external rewards
"customCreditType": "AR",
"currencyCode": "USD",
"initialBalance": 10,
"currentBalance": 10,
"expirationDate": "2018-11-14T00:00:00.000Z",
"auditInfo": {
"updateDate": "2017-11-20T22:17:44.218Z",
"createDate": "2017-11-14T18:53:58.736Z",
"updateBy": "88318826167a438ab4bf32679971c561",
"createBy": "355060a60a5e48eeb7f2fb8d92af2ba5"
}
},
{
"code": "135646847",
"activationDate": "2017-12-03T06:00:00.000Z",
"creditType": "Custom",
"customCreditType": "AR",
"currencyCode": "USD",
"initialBalance": 100,
"currentBalance": 100,
"expirationDate": "2018-11-14T00:00:00.000Z",
"auditInfo": {
"updateDate": "2017-11-20T22:17:44.218Z",
"createDate": "2017-11-14T18:53:58.736Z",
"updateBy": "88318826167a438ab4bf32679971c561",
"createBy": "355060a60a5e48eeb7f2fb8d92af2ba5"
}
}
];
return rewards;
}
```

action.before

Use this sample when implementing action.before.

```
/**
* Sample action implementation for purchase rewards
* In case of customer having a specific account no.(reward no.) with external rewards system, this sample action
* is used to pass this information in a databag on payment actiom to use it when capturing store credit.
*/
var OrderAttributeFactory = require('mozu-node-sdk/clients/commerce/orders/orderAttribute');
var _ = require('underscore');

module.exports = function (context, callback) {
var payment = context.get.payment();
var isForOrder = context.get.isForOrder(); //Flag to indicate if the action is running in order context.
//Set reward information on payment data bag
//Assuming external reward no. is stored on a storefront order attribute
if (payment.paymentType === "StoreCredit" && payment.billingInfo.storeCreditType === "Custom" && (payment.status === "New" || payment.status === "Collected") && isForOrder) {
console.log("Order Context");
var orderAttributeResource = OrderAttributeFactory(context.apiContext);
orderAttributeResource.getOrderAttributes({ orderId: context.get.payment().orderId })
.then(function (attributes) {
console.log("Looking for purchase reward order attribute");
var attr = _.findWhere(attributes, { fullyQualifiedName: "tenant~purchaseRewardNumber" });  //replace attribute FQN with the correct name that you setup
var purchaseRewardNo = attr.values[0].toString();
console.log("purchase Reward No:" + purchaseRewardNo);
context.exec.setPaymentData("rewardData", { "purchaseRewardNo": purchaseRewardNo, "creditType": "purchaseReward" });
callback();
})
.catch(function (err) {
console.error(err);
callback();
});
}
else {
callback();
}
};
```

addTransaction.before

Use this sample when implementing addTransaction.before.

```
/**
* This Sample action is used to achieve purhcase rewards functionality
* When capturing a store credit payment, this action bypasses NG store credit capture
* by calling external system to capture.
* Databag set in the payment action before can be used here to retrieve account no.(reward no.),
but the voucher number used for the payment
* itself is available on request param 'code' field.
*/
var CustomerAccountFactory = require('mozu-node-sdk/clients/commerce/customer/accounts/cust
omerAttribute');
var _ = require('underscore');

module.exports = function (context, callback) {
var requestParams = context.request.params;
var creditTransaction = requestParams.creditTransaction;

if (!_.has(creditTransaction.data, "rewardData")) {
console.log("No reward data present! Calling back..");
callback();
return;
}

var rewardData = creditTransaction.data.rewardData;
//Call your external reward system to perform a capture.
console.log("Amount to " + creditTransaction.transactionType + ": " + creditTransaction.impactA
mount + " for voucher no: " + requestParams.code);
var orderId = creditTransaction.orderId;
var amount = creditTransaction.impactAmount;

//Once captured, create a success response
var transaction = {
"transactionType": creditTransaction.transactionType,
"impactAmount": amount,
"orderId": orderId,
"comments": "This is purchaseReward transaction",
"auditInfo": {
"updateDate": "2017-11-20T22:17:44.218Z",
"createDate": "2017-11-14T18:53:58.736Z",
"updateBy": "purchaseRewards Application",
"createBy": "purchaseRewards Application"
}
};
context.response.status = 200;
context.response.body = transaction;

//bypass NG capture
context.response.end();
callback();
};
```