Testing API Extension Applications

The recommended method of testing API Extension applications is to use the Action Simulator in conjunction with thorough integration testing.

Action Simulator

The Action Simulator produces a mock context that you can utilize in your API Extension test files to simulate data from Kibo and verify that your functions run as expected.

You can view the Action Simulator library in the `node_modules/mozu-action-simulator` directory of your project folder. The simulator provides:

- Test fixtures for your actions that you can leverage to create data for your tests.
- Access to the assert module, which helps you devise assertion tests to determine whether your logic executes as expected.

To use the simulator:

1. Open the test file that corresponds to one of the actions you have scaffolded. Test files are located in the `assets/test` directory.
2. Require the `mozu-action-simulator` package in your test file. For example:

   ```
   var Simulator = require('mozu-action-simulator');
   ```
3. Access simulator features, such as `Simulator.context()` or `Simulator.assert`, to facilitate writing tests for your function.

Example of Testing with the Simulator

In this example, the `embedded.commerce.carts.addItem.before` action has been designed to check whether an item added to the cart has a 'Hazardous' product attribute. If the new item is hazardous and the cart already contains an existing hazardous item, the action removes the new item from the cart. The following code block demonstrates how you would test this action within its corresponding test file using the Action Simulator.

`assets/test/embedded.commerce.carts.addItem.before.t.js`

```
/**
* This is a scaffold for unit tests for the custom function for
* `embedded.commerce.carts.addItem.before`.
* Modify the test conditions below. You may:
*  - add special assertions for code actions from Simulator.assert
*  - create a mock context with Simulator.context() and modify it
*  - use and modify mock Mozu business objects from Simulator.fixtures
*/
//Test for:
//embedded.commerce.carts.addItem.before.t.is
```

```
'use strict';

var Simulator = require('mozu-action-simulator');
var assert = Simulator.assert;

var actionName = 'embedded.commerce.carts.addItem.before';

describe('embedded.commerce.carts.addItem.before implementing embedded.commerce.carts.ad
dItem.before', function() {

    var action;

    before(function() {
        action = require('../src/domains/commerce.carts/embedded.commerce.carts.addItem.before')
;
    });

    it('runs successfully', function(done) {

        var callback = function(err) {
            assert.ok(!err, "Callback was called with an error: " + err);
            // more assertions
            done();
        };

        var context = Simulator.context(actionName, callback);

        // modify context as necessary

        context.exec.removeItem = function() {
        };

        Simulator.simulate(actionName, action, context, callback);
        });

        it('checks incoming product for hazard indicator and then removes hazardous items from the
cart if the incoming item is hazardous', function(done) {
            var callback = function(err) {
                assert.ok(!err, "Callback was called with an error: " + err);
                // more assertions
                done();
            };
            //Test setup
            var context = Simulator.context(actionName, callback);

            var cart = context.get.cart();
            var cartItem = context.get.cartItem();
            var cartItemsCount = cart.items.length;

            cart.items[0].product.properties.push({
                "attributeFQN": "tenant~hazardous",
                "name": "Hazardous",
                "dataType": "Bool",
                "isMultiValue": false,
                "values": [
                    {
                        "value": true
                    }
                ]
            });
```

```
        });

        cart.items[1].product.properties.push({
            "attributeFQN": "tenant~hazardous",
            "name": "Hazardous",
            "dataType": "Bool",
            "isMultiValue": false,
            "values": [
                {
                    "value": true
                }
            ]
        });

        cartItem.product.properties.push({
            "attributeFQN": "tenant~hazardous",
            "name": "Hazardous",
            "dataType": "Bool",
            "isMultiValue": false,
            "values": [
                {
                    "value": true
                }
            ]
        });

        context.get.cart = function() {
            return cart;
        }

        context.get.cartItem = function() {
            return cartItem;
        }

        context.exec.removeItem = function() {
            var alteredCart = context.get.cart().items;
            alteredCart.pop();
            assert.equal(alteredCart.length, 1);
        };

        Simulator.simulate(actionName, action, context, callback);

    });
});
```

```
module.exports = function(context, callback) {
    var cart = context.get.cart();
    var cartItem = context.get.cartItem();
    var isHazardousCart = false;

    cart.items.forEach(function(item) {
        item.product.properties.forEach(function(property) {
            if (property.attributeFQN === 'tenant~hazardous') {
                isHazardousCart = true;
            }

            if (isHazardousCart && property.attributeFQN === 'tenant~hazardous') {
                context.exec.removeItem();
            }
        });
    });
    callback();
};
```