Microsoft Dynamics D365 Retail Experts - https://www.meetup.com/Dynamics-User-Group-India/
Sheikhmydheen
https://www.linkedin.com/in/sheikhmydheen/
Sheikhmydheen@gmail.com , d365retail@gmail.com

POS and New Hardware device extension:

In this topic, we will walkthrough how you can integrate POS with new hardware device. This topic is applicable for Dynamics 365 for Finance and Operations or Dynamics 365 for Retail platform update 8.

Scenario: We want to integrate POS with Coin dispenser device, after the transaction completed POS should check if tender change is required in coins then trigger the coin dispenser device to dispense the change.

To call the coin dispenser device at the end of the transaction we can use the POS post end transaction trigger. In this exercise we implement POS post end transaction trigger and call the coin dispenser device to dispense the change.

1. Open visual studio 2015 in administrator mode.
2. Open ModernPOS solution from …\RetailSDK\Code\POS
3. Under the POS.Extensions project create a new folder called CoinDispenserSample.
4. Under CoinDispenserSample, create new folder called TriggersHandlers.
5. In the TriggersHandlers folder, add a new ts (typescript) file and name it has PostEndTransactionTrigger.ts
6. Add the below import statement to import the relevant entities and context.

```
import * as Triggers from "PosApi/Extend/Triggers/TransactionTriggers";
import { ObjectExtensions } from "PosApi/TypeExtensions";
import { ClientEntities } from "PosApi/Entities";
import { HardwareStationDeviceActionRequest, HardwareStationDeviceActionResponse }
from "PosApi/Consume/Peripherals";
```

7. Create a new class called `PostEndTransactionTrigger` and extend it from `PostEndTransactionTrigger`.

```
export default class PostEndTransactionTrigger extends
Triggers.PostEndTransactionTrigger { }
```

8. Implement the trigger execute method to call the coin dispenser device

```
public execute(options: Triggers.IPostEndTransactionTriggerOptions): Promise<void>
{
        this.context.logger.logVerbose("Executing PostEndTransactionTrigger with
options " + JSON.stringify(options) + ".");

        if (ObjectExtensions.isNullOrUndefined(options) ||
ObjectExtensions.isNullOrUndefined(options.receipts)) {
            // This will never happen, but is included to demonstrate how to
return a rejected promise when validation fails.
            let error: ClientEntities.ExtensionError
                = new ClientEntities.ExtensionError("The options provided to the
PostEndTransactionTrigger were invalid.");

            return Promise.reject(error);
        } else {
            let request:
HardwareStationDeviceActionRequest<HardwareStationDeviceActionResponse> =
```

Sheikhmydheen
https://www.linkedin.com/in/sheikhmydheen/
Sheikhmydheen@gmail.com , d365retail@gmail.com

```
                new HardwareStationDeviceActionRequest("COINDISPENSER",
"DispenseChange", {
                    "DeviceName": "Sample device",
                    "DeviceType": "Sample device type",
                    "Amount": 1
                });

            return this.context.runtime.executeAsync(request)
                .then((result:
ClientEntities.ICancelableDataResult<HardwareStationDeviceActionResponse>): void
=> {
                    this.context.logger.logInformational("Hello coin dispenser
completed: " + result.data.response);
                }).catch((reason: any): Promise < void> => {
                    // Resolves to a void result when rejected. This matches
existing POS printing behavior.
                    this.context.logger.logError("PostSuspendTransactionTrigger
execute error: " + JSON.stringify(reason));
                    return Promise.resolve();
                });
        }
       }
```

The overall code should look like this:

```
/**
 * SAMPLE CODE NOTICE
 *
 * THIS SAMPLE CODE IS MADE AVAILABLE AS IS.  MICROSOFT MAKES NO WARRANTIES,
WHETHER EXPRESS OR IMPLIED,
 * OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY OR COMPLETENESS OF RESPONSES,
OF RESULTS, OR CONDITIONS OF MERCHANTABILITY.
 * THE ENTIRE RISK OF THE USE OR THE RESULTS FROM THE USE OF THIS SAMPLE CODE
REMAINS WITH THE USER.
 * NO TECHNICAL SUPPORT IS PROVIDED.  YOU MAY NOT DISTRIBUTE THIS CODE UNLESS YOU
HAVE A LICENSE AGREEMENT WITH MICROSOFT THAT ALLOWS YOU TO DO SO.
 */

import * as Triggers from "PosApi/Extend/Triggers/TransactionTriggers";
import { ObjectExtensions } from "PosApi/TypeExtensions";
import { ClientEntities } from "PosApi/Entities";
import { HardwareStationDeviceActionRequest, HardwareStationDeviceActionResponse }
from "PosApi/Consume/Peripherals";

export default class PostEndTransactionTrigger extends
Triggers.PostEndTransactionTrigger {
    /**
     * Executes the trigger functionality.
     * @param {Triggers.IPostSuspendTransactionTriggerOptions} options The options
provided to the trigger.
     */
    public execute(options: Triggers.IPostEndTransactionTriggerOptions):
Promise<void> {
        this.context.logger.logVerbose("Executing PostEndTransactionTrigger with
options " + JSON.stringify(options) + ".");
```

Sheikhmydheen
https://www.linkedin.com/in/sheikhmydheen/
Sheikhmydheen@gmail.com , d365retail@gmail.com

```
        if (ObjectExtensions.isNullOrUndefined(options) ||
ObjectExtensions.isNullOrUndefined(options.receipts)) {
            // This will never happen, but is included to demonstrate how to
return a rejected promise when validation fails.
            let error: ClientEntities.ExtensionError
                = new ClientEntities.ExtensionError("The options provided to the
PostEndTransactionTrigger were invalid.");

            return Promise.reject(error);
        } else {
            let request:
HardwareStationDeviceActionRequest<HardwareStationDeviceActionResponse> =
                new HardwareStationDeviceActionRequest("COINDISPENSER",
"DispenseChange", {
                    "DeviceName": "Sample device",
                    "DeviceType": "Sample device type",
                    "Amount": 1
                });

            return this.context.runtime.executeAsync(request)
                .then((result:
ClientEntities.ICancelableDataResult<HardwareStationDeviceActionResponse>): void
=> {
                    this.context.logger.logInformational("Hello coin dispenser
completed: " + result.data.response);
                }).catch((reason: any): Promise < void> => {
                    // Resolves to a void result when rejected. This matches
existing POS printing behavior.
                    this.context.logger.logError("PostSuspendTransactionTrigger
execute error: " + JSON.stringify(reason));
                    return Promise.resolve();
                });
        }
    }
       }
```

9. Create a new json file and under the CoinDispenserSample folder and name it as manifest.json.
10. In the manifest.json file, copy and paste the below code:

```
{
  "$schema": "../manifestSchema.json",
  "name": "CoinDispenser_Sample",
  "publisher": "Microsoft",
  "version": "7.2.0",
  "minimumPosVersion": "7.2.0.0",
  "components": {
    "extend": {
      "triggers": [
        {
          "triggerType": "PostEndTransaction",
          "modulePath": "TriggersHandlers/PostEndTransactionTrigger"
        }
      ]
    }
  }
```

Sheikhmydheen
https://www.linkedin.com/in/sheikhmydheen/
Sheikhmydheen@gmail.com , d365retail@gmail.com

```
        }
```

11. Open the extensions.json file under POS.Extensions project and update it with CoinDispenserSample, so that POS during runtime will include this extension.

```
{
  "extensionPackages": [
    {
      "baseUrl": "SampleExtensions2"
    },
    {
      "baseUrl": "POSAPIExtension"
    },
    {
      "baseUrl": "CustomColumnExtensions"
    },
    {
      "baseUrl": "EODSample"
    },
    {
      "baseUrl": "ProdDetailsCustomColumnExtensions"
    },
    {
      "baseUrl": "SerachExtension"
    },
    {
      "baseUrl": "SuspendReceiptSample"
    },
    {
      "baseUrl": "CoinDispenserSample"
    }
  ]
}
```

12. Open the tsconfig.json to comment out the extension package folders from the exclude list. POS will use this file to include or exclude the extension. By default, the list contains all the excluded extensions list, if you want to include any extension part of the POS then you need add the extension folder name and comment the extension from the extension list like below.

```
"exclude": [
"AuditEventExtensionSample",
 "B2BSample",
 "CustomerSearchWithAttributesSample",
 "FiscalRegisterSample",
 "PaymentSample",
 "PromotionsSample",
 "SalesTransactionSignatureSample",
 "SampleExtensions",
 //"SampleExtensions2",
 "StoreHoursSample",
 "SuspendTransactionReceiptSample"
 //"POSAPIExtension",
 //"CustomColumnExtensions",
 //"EODSample",
 //"ProdDetailsCustomColumnExtensions",
```

Sheikhmydheen
https://www.linkedin.com/in/sheikhmydheen/
Sheikhmydheen@gmail.com , d365retail@gmail.com

```
        //"SerachExtension",
        //"SuspendReceiptSample",
        //"CoinDispenserSample"
                ],
```

13. Compile and rebuild the project.

Build Deploy the sample payment project:

14. Open visual studio 2015 in administrator mode.
15. Open HardwareStation.Extension.CoinDispenserSample.csproj from …\RetailSDK\SampleExtensions\HardwareStation\ Extension.CoinDispenserSample
16. Right click and rebuild the project.
17. Navigate to …\ RetailSDK\SampleExtensions\HardwareStation\ Extension.CoinDispenserSample\bin\Debug
18. Copy the Contoso.Commerce.HardwareStation.Extension.CoinDispenserSample.dll and paste it in C:\Program Files (x86)\Microsoft Dynamics 365\70\Retail Modern POS\ClientBroker
19. Open the HardwareStation.Extension.config
20. Add the extension payment library details under the composition section.

```
  <add source="assembly"
value="Contoso.Commerce.HardwareStation.Extension.CoinDispenserSample" />
```

21. Save the file.
22. Close Modern POS if running.
23. Open Task manager and kill all the dllhost.exe

How to validate your extension:

24. Press F5 and deploy the POS to test your customization.
25. Once the POS is launched, login to POS and add any item to transaction.
26. Navigate to the HardwareStation.Extension.CoinDispenserSample.csproj and Click Debug > Attach to Process and select Dllhost.exe from the available processes list and Click Attach.
27. In the POS hit the pay by cash button and complete the payment.
28. The debugger should hit the coin dispenser execute method and you can debug the device.