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

How to create POS Triggers and call existing peripherals APIs:

You can use triggers to capture events that fire before and after any Retail POS operations. You can insert custom logic before the operation runs or after it has completed. There are operation specific triggers and generic trigger called the PreOperationTrigger and PostTOperationTrigger. These triggers run at the beginning and end of all POS operations., depends on the scenario you can implement either pre triggers or post trigger for your operation.

This topic is applicable for Dynamics 365 for Finance and Operations or Dynamics 365 for Retail platform update 8 with retail App update 4 hotfix.

Scenario: We want to print a custom receipt whenever the user suspends a transaction. For this sample we will implement the Post suspend transaction trigger and print the custom receipt using the existing print peripheral API.

1. Open visual studio 2015 in administrator mode.
2. Open ModernPOS solution from …\RetailSDK\POS
3. Under the POS.Extensions project create a new folder called SuspendReceiptSample.
4. Under SuspendReceiptSample, create new folder called TriggersHandlers.
5. In the TriggersHandlers folder, add a new ts (typescript) file and name it has PostSuspendTransactionTrigger.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, ProxyEntities } from "PosApi/Entities";
import { PrinterPrintRequest, PrinterPrintResponse } from
"PosApi/Consume/Peripherals";
import { GetHardwareProfileClientRequest, GetHardwareProfileClientResponse } from
"PosApi/Consume/Device";
import { GetReceiptsClientRequest, GetReceiptsClientResponse } from
"PosApi/Consume/SalesOrders";
```

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

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

8. Implement the trigger execute method to get the receipt profile and print the custom receipt:

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

        if (ObjectExtensions.isNullOrUndefined(options) ||
ObjectExtensions.isNullOrUndefined(options.cart)) {
            // This will never happen, but is included to demonstrate how to
return a rejected promise when validation fails.
            let error: ClientEntities.ExtensionError
```

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

```
                = new ClientEntities.ExtensionError("The options provided to the
PostSuspendTransactionTrigger were invalid.");

            return Promise.reject(error);
        } else {
            return this.context.runtime.executeAsync(new
GetHardwareProfileClientRequest())
                .then((response:
ClientEntities.ICancelableDataResult<GetHardwareProfileClientResponse>)
                    :
    Promise<ClientEntities.ICancelableDataResult<GetReceiptsClientResponse>> => {
                    let hardwareProfile: ProxyEntities.HardwareProfile =
response.data.result;

                    // Gets the receipts.
                    let salesOrderId: string = options.cart.Id;
                        let receiptRetrievalCriteria:
    ProxyEntities.ReceiptRetrievalCriteria = {
                        IsCopy: false,
                        IsRemoteTransaction: false,
                        IsPreview: false,
                        QueryBySalesId: true,
                        ReceiptTypeValue:
ProxyEntities.ReceiptType.CustomReceipt7,
                        HardwareProfileId: hardwareProfile.ProfileId
                    };
                    let getReceiptsClientRequest:
GetReceiptsClientRequest<GetReceiptsClientResponse> =
                        new GetReceiptsClientRequest(salesOrderId,
receiptRetrievalCriteria);

                    return
this.context.runtime.executeAsync(getReceiptsClientRequest);
                })
                .then((response:
ClientEntities.ICancelableDataResult<GetReceiptsClientResponse>)
                    :
Promise<ClientEntities.ICancelableDataResult<PrinterPrintResponse>> => {
                    let receipts: ProxyEntities.Receipt[] = response.data.result;

                    // Prints the receipts.
                    let printerPrintRequest:
PrinterPrintRequest<PrinterPrintResponse> = new PrinterPrintRequest(receipts);

                    return this.context.runtime.executeAsync(printerPrintRequest);
                }).then((): Promise<void> => {
                    // Resolves to a void result when fulfilled.
                    return Promise.resolve();
                }).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();
                });
```

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

```
            }
        }
```

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, ProxyEntities } from "PosApi/Entities";
import { PrinterPrintRequest, PrinterPrintResponse } from
"PosApi/Consume/Peripherals";
import { GetHardwareProfileClientRequest, GetHardwareProfileClientResponse } from
"PosApi/Consume/Device";
import { GetReceiptsClientRequest, GetReceiptsClientResponse } from
"PosApi/Consume/SalesOrders";

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

        if (ObjectExtensions.isNullOrUndefined(options) ||
ObjectExtensions.isNullOrUndefined(options.cart)) {
            // 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
PostSuspendTransactionTrigger were invalid.");

            return Promise.reject(error);
        } else {
            return this.context.runtime.executeAsync(new
GetHardwareProfileClientRequest())
                .then((response:
ClientEntities.ICancelableDataResult<GetHardwareProfileClientResponse>)
```

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

```
                    :
Promise<ClientEntities.ICancelableDataResult<GetReceiptsClientResponse>> => {
                    let hardwareProfile: ProxyEntities.HardwareProfile =
response.data.result;

                    // Gets the receipts.
                    let salesOrderId: string = options.cart.Id;
                    let receiptRetrievalCriteria:
ProxyEntities.ReceiptRetrievalCriteria = {
                        IsCopy: false,
                        IsRemoteTransaction: false,
                        IsPreview: false,
                        QueryBySalesId: true,
                        ReceiptTypeValue: ProxyEntities.ReceiptType.CustomReceipt7,
                        HardwareProfileId: hardwareProfile.ProfileId
                    };
                    let getReceiptsClientRequest:
GetReceiptsClientRequest<GetReceiptsClientResponse> =
                        new GetReceiptsClientRequest(salesOrderId,
receiptRetrievalCriteria);

                    return
this.context.runtime.executeAsync(getReceiptsClientRequest);
                })
                .then((response:
ClientEntities.ICancelableDataResult<GetReceiptsClientResponse>)
                    :
Promise<ClientEntities.ICancelableDataResult<PrinterPrintResponse>> => {
                    let receipts: ProxyEntities.Receipt[] = response.data.result;

                    // Prints the receipts.
                    let printerPrintRequest: PrinterPrintRequest<PrinterPrintResponse>
= new PrinterPrintRequest(receipts);

                    return this.context.runtime.executeAsync(printerPrintRequest);
                }).then((): Promise<void> => {
                    // Resolves to a void result when fulfilled.
                    return Promise.resolve();
                }).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 POSAPIExtension folder and name it as manifest.json.
10. In the manifest.json file, copy and paste the below code:

```
{
  "$schema": "../manifestSchema.json",
  "name": "Pos_Extensibility_SuspendTransactionReceiptSample",
  "publisher": "Microsoft",
```

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

```
  "version": "7.2.0",
  "minimumPosVersion": "7.2.0.0",
  "components": {
    "extend": {
      "triggers": [
        {
          "triggerType": "PostSuspendTransaction",
          "modulePath": "TriggersHandlers/PostSuspendTransactionTrigger"
        }
      ]
    }
  }
      }
```

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

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

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",
```

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

```
            "SampleExtensions",
            //"SampleExtensions2",
            "StoreHoursSample",
            "SuspendTransactionReceiptSample"
            //"POSAPIExtension",
            //"CustomColumnExtensions",
            //"EODSample",
            //"ProdDetailsCustomColumnExtensions",
            //"SerachExtension",
            //"SuspendReceiptSample"
                ],
```

13. Compile and rebuild the project.

Add Custom receipt layout in HQ:

14. Navigate to Dynamics 365 for operation url.
15. Click Retail > Channel setup > POS setup > POS > Receipts formats.
16. Click New in the Receipts formats
17. In the Receipt format filed enter the format name - “Suspend” and in the Receipt type field select the CustomReceiptType7
18. Click the Designer to open the receipt designer.
19. Follow the instructions if prompted to install and enter the AAD credentials to launch the designer.
20. Drag and drop the required field in Header, Lines and Footer Or you can also copy the from existing receipt format by using the copy feature and edit it.
21. Click the Save button to save the changes.
22. Navigate to Retail > Channel setup > POS setup > POS profiles > Receipt profiles
23. Select the Email receipt profile or whichever profile you store is using and Click the Add button in the General tab.
24. In the Receipt type select “CustomReceiptType7” and in the Receipt format select “Suspend”.
25. Navigate to Retail > Retail IT > Distribution schedule
26. Select the Registers (1090) and click the Run now button in the action bar. On prompt click Yes, to run the job. Bottom of Form

Configure XPS printer for quick testing:

27. Navigate to Retail > Channel setup > POS setup > POS profiles > Hardware profiles
28. Select the Hardware profile your device is using. Ex: In the demo data all the Houston devices uses HW002
29. Click Edit in the action bar.
30. Expand the printer fast tab, in the Printer drop down select **Windows driver** and in the device name enter “**Microsoft XPS Document Writer**”
31. Click Save.
32. Navigate to Retail > Retail IT > Distribution schedule
33. Select the Registers (1090) and click the Run now button in the action bar. On prompt click Yes, to run the job. Bottom of Form

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

How to validate your extension:

34. Press F5 and deploy the POS to test your customization.
35. Once the POS is launched, login to POS and add any item to transaction.
36. Suspend the transaction.
37. Custom receipt should be printed.