Microsoft Dynamics D365 Retail Experts - https://www.meetup.com/Dynamics-User-Group-India/

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

Consume POS APIs:

Retail POS API helps you to build extensions or new features to the POS app easily. Ex: If you are extending Retail POS application to add new feature in which to want to get product details or change price or add item to cart in many such scenarios you can consume our APIs which will do the work for you, simply call our APIs to do the work. The POS API simplifies the extension pattern and provide continuous support to build the extensions.

We unified our extension patterns across commerce runtime (CRT), POS and Hardware station (HWS) by following the request/response pattern. All the POS APIs are exposed as request/response like CRT and HWS. This topic is applicable for Dynamics 365 for Finance and Operations or Dynamics 365 for Retail platform update 8 with application update 4 hotfix.

The POS APIs are categorized into three different scenarios:

1. **Consume** – Consume our public APIs in your extension.
2. **Extend** – Extend our public APIs to do some additional logic.
3. **Create** – Create your new APIs using the POS interface exposed which can be used across your extensions.

Consume:

We exposed lot of APIs to be consumed in extensions. Ex: You have scenario where you want to change the price of the item based on an external web service call, in that case you can call our PriceOverrideOperationRequest API to change the price of the item. Within the consume, the APIs are sub categorized by module like Crat, peripherals, store operations etc.

Note: You can get all the API list from the **Pos.Api.d.ts** which is part of the Retail SDK (…Retail SDK\POS\Extensions\Pos.Api.d.ts)

How to consume our APIs in your extension:

To consume our APIs in your extension, follow the below steps:

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 POSAPIExtension.
4. Under POSAPIExtension, create new folder called TriggersHandlers.
5. In the TriggersHandlers folder, add a new ts (typescript) file and name it has PreEndTransactionTrigger.ts
6. Add the below import statement to import the relevant entities and context.

```
import * as Triggers from "PosApi/Extend/Triggers/TransactionTriggers";
import { ClientEntities, ProxyEntities } from "PosApi/Entities";
import { ObjectExtensions, StringExtensions } from "PosApi/TypeExtensions";
import {
    GetCurrentCartClientRequest, GetCurrentCartClientResponse,
    SaveAttributesOnCartClientRequest, SaveAttributesOnCartClientResponse
} from "PosApi/Consume/Cart";
import {
    GetCustomerClientRequest, GetCustomerClientResponse,
} from "PosApi/Consume/Customer";
import { ShowMessageDialogClientRequest, ShowMessageDialogClientResponse } from
"PosApi/Consume/Dialogs";
```

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

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

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

8. Inside the class declare the below variables to declare the attributes names and sample values:

```
 private static CART_ATTRIBUTE_NAME: string = "ATT SAMPLE";
 private static CART_ATTRIBUTE_VALUE_TRUE: string = "True";
 private static CART_ATTRIBUTE_VALUE_FALSE: string = "False";
 private static DIALOG_RESULT_YES: string = "yes";
 private static DIALOG_RESULT_NO: string = "no";
private static DIALOG_YES_BUTTON_ID: string =
"CART_PreEndTransactionTrigger_MessageDialog_Yes";
private static DIALOG_NO_BUTTON_ID: string =
"CART_PreEndTransactionTrigger_MessageDialog_No";
```

9. Implement the trigger execute method and call the existing POS APIs:
   In the execute method we will be calling the below APIS:
   1. Get current cart
   2. Get Current customer
   3. Save attribute on cart

```
public execute(options: Triggers.IPreEndTransactionTriggerOptions):
Promise<ClientEntities.ICancelable> {
        console.log("Executing PreEndTransactionTrigger with options " +
JSON.stringify(options) + ".");

 let currentCart: ProxyEntities.Cart;
        return this.context.runtime.executeAsync<GetCurrentCartClientResponse>(new
GetCurrentCartClientRequest())
            .then((getCurrentCartClientResponse:
ClientEntities.ICancelableDataResult<GetCurrentCartClientResponse>):

Promise<ClientEntities.ICancelableDataResult<GetCustomerClientResponse>> => {

         currentCart = getCurrentCartClientResponse.data.result;

         // Gets the current customer.
                let result:
Promise<ClientEntities.ICancelableDataResult<GetCustomerClientResponse>>;
                if (!ObjectExtensions.isNullOrUndefined(currentCart) &&
!ObjectExtensions.isNullOrUndefined(currentCart.CustomerId)) {
                    let getCurrentCustomerClientRequest:
GetCustomerClientRequest<GetCustomerClientResponse> =
                        new GetCustomerClientRequest(currentCart.CustomerId);
                    result =
this.context.runtime.executeAsync<GetCustomerClientResponse>(getCurrentCustomerCli
entRequest);
```

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

```
                } else {
                    result = Promise.resolve({ canceled: false, data: new
GetCustomerClientResponse(null) });
                }

                return result;
            })
            .then((getCurrentCustomerClientResponse:
ClientEntities.ICancelableDataResult<GetCustomerClientResponse>):

Promise<ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>> =>
{

        let currentCustomer: ProxyEntities.Customer =
getCurrentCustomerClientResponse.data.result;

        let result:
Promise<ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>>;
        if (!ObjectExtensions.isNullOrUndefined(currentCart)
            && !ObjectExtensions.isNullOrUndefined(currentCustomer)) {

            let yesButton: ClientEntities.Dialogs.IDialogResultButton = {
                id: PreEndTransactionTrigger.DIALOG_YES_BUTTON_ID,
                label: "Yes", // "Yes"
                result: PreEndTransactionTrigger.DIALOG_RESULT_YES
            };
            let noButton: ClientEntities.Dialogs.IDialogResultButton = {
                id: PreEndTransactionTrigger.DIALOG_NO_BUTTON_ID,
                label: "No", // "No"
                result: PreEndTransactionTrigger.DIALOG_RESULT_NO
            };
                    let showMessageDialogClientRequestOptions:
ClientEntities.Dialogs.IMessageDialogOptions = {
                title: "Save attribute - Sample",
                subTitle: StringExtensions.EMPTY,
                       message: "Save attribute ?",
                button1: yesButton,
                button2: noButton
            };
                    let showMessageDialogClientRequest:
ShowMessageDialogClientRequest<ShowMessageDialogClientResponse> =
                        new
ShowMessageDialogClientRequest(showMessageDialogClientRequestOptions);

                    result =
this.context.runtime.executeAsync<ShowMessageDialogClientResponse>(showMessageDial
ogClientRequest);
                } else {
                    result = Promise.resolve({ canceled: false, data: new
ShowMessageDialogClientResponse(null) });
                }
```

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

```
                return result;
            })
            .then((showMessageDialogClientResponse:
ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>):

Promise<ClientEntities.ICancelableDataResult<SaveAttributesOnCartClientResponse>>
=> {

                // Save the attribute value depending on the dialog result.
                let messageDialogResult:
ClientEntities.Dialogs.IMessageDialogResult =
showMessageDialogClientResponse.data.result;

                let result:
Promise<ClientEntities.ICancelableDataResult<SaveAttributesOnCartClientResponse>>;
                if (!ObjectExtensions.isNullOrUndefined(messageDialogResult)) {

                    let attributeValue: ProxyEntities.AttributeTextValue = new
ProxyEntities.AttributeTextValueClass();
                    attributeValue.Name =
PreEndTransactionTrigger.CART_ATTRIBUTE_NAME;
                    attributeValue.TextValue = messageDialogResult.dialogResult
=== PreEndTransactionTrigger.DIALOG_RESULT_YES ?
                        PreEndTransactionTrigger.CART_ATTRIBUTE_VALUE_TRUE :
PreEndTransactionTrigger.CART_ATTRIBUTE_VALUE_FALSE;
                    let attributeValues: ProxyEntities.AttributeValueBase[] =
[attributeValue];
                    let saveAttributesOnCartRequest:
SaveAttributesOnCartClientRequest<SaveAttributesOnCartClientResponse> =
                        new SaveAttributesOnCartClientRequest(attributeValues);
                    result =
this.context.runtime.executeAsync(saveAttributesOnCartRequest);
                } else {
                    result = Promise.resolve({ canceled: false, data: new
SaveAttributesOnCartClientResponse(null) });
                }

                return result;
            });
        }
```

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.
```

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

```
 * 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 { ClientEntities, ProxyEntities } from "PosApi/Entities";
import { ObjectExtensions, StringExtensions } from "PosApi/TypeExtensions";
import {
    GetCurrentCartClientRequest, GetCurrentCartClientResponse,
    SaveAttributesOnCartClientRequest, SaveAttributesOnCartClientResponse
} from "PosApi/Consume/Cart";
import {
    GetCustomerClientRequest, GetCustomerClientResponse,
} from "PosApi/Consume/Customer";
import { ShowMessageDialogClientRequest, ShowMessageDialogClientResponse } from
"PosApi/Consume/Dialogs";

/**
 * Example implementation of an PreEndTransactionTrigger trigger that logs to the
console.
 */
export default class PreEndTransactionTrigger extends
Triggers.PreEndTransactionTrigger {
    private static CART_ATTRIBUTE_NAME: string = "ATT SAMPLE";
    private static CART_ATTRIBUTE_VALUE_TRUE: string = "True";
    private static CART_ATTRIBUTE_VALUE_FALSE: string = "False";
    private static DIALOG_RESULT_YES: string = "yes";
    private static DIALOG_RESULT_NO: string = "no";
           private static DIALOG_YES_BUTTON_ID: string =
       "CART_PreEndTransactionTrigger_MessageDialog_Yes";
       private static DIALOG_NO_BUTTON_ID: string = "
CART_PreEndTransactionTrigger_MessageDialog_No";

    /**
     * Executes the trigger functionality.
     * @param {Triggers.IPreEndTransactionTriggerOptions} options The options
provided to the trigger.
     */
    public execute(options: Triggers.IPreEndTransactionTriggerOptions):
Promise<ClientEntities.ICancelable> {
        console.log("Executing PreEndTransactionTrigger with options " +
JSON.stringify(options) + ".");

        let currentCart: ProxyEntities.Cart;
        return this.context.runtime.executeAsync<GetCurrentCartClientResponse>(new
GetCurrentCartClientRequest())
            .then((getCurrentCartClientResponse:
ClientEntities.ICancelableDataResult<GetCurrentCartClientResponse>):
```

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

```
Promise<ClientEntities.ICancelableDataResult<GetCustomerClientResponse>> => {

                currentCart = getCurrentCartClientResponse.data.result;

                // Gets the current customer.
                let result:
Promise<ClientEntities.ICancelableDataResult<GetCustomerClientResponse>>;
                if (!ObjectExtensions.isNullOrUndefined(currentCart) &&
!ObjectExtensions.isNullOrUndefined(currentCart.CustomerId)) {
                    let getCurrentCustomerClientRequest:
GetCustomerClientRequest<GetCustomerClientResponse> =
                        new GetCustomerClientRequest(currentCart.CustomerId);
                    result =
this.context.runtime.executeAsync<GetCustomerClientResponse>(getCurrentCustomerCli
entRequest);
                } else {
                    result = Promise.resolve({ canceled: false, data: new
GetCustomerClientResponse(null) });
                }

                return result;
            })
            .then((getCurrentCustomerClientResponse:
ClientEntities.ICancelableDataResult<GetCustomerClientResponse>):

Promise<ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>> =>
{

                let currentCustomer: ProxyEntities.Customer =
getCurrentCustomerClientResponse.data.result;

                let result:
Promise<ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>>;
                if (!ObjectExtensions.isNullOrUndefined(currentCart)
             && !ObjectExtensions.isNullOrUndefined(currentCustomer)) {

             let yesButton: ClientEntities.Dialogs.IDialogResultButton = {
                 id: PreEndTransactionTrigger.DIALOG_YES_BUTTON_ID,
                    label: "Yes", // "Yes"
                 result: PreEndTransactionTrigger.DIALOG_RESULT_YES
             };
             let noButton: ClientEntities.Dialogs.IDialogResultButton = {
                 id: PreEndTransactionTrigger.DIALOG_NO_BUTTON_ID,
                    label: "No", // "No"
                 result: PreEndTransactionTrigger.DIALOG_RESULT_NO
             };
                    let showMessageDialogClientRequestOptions:
ClientEntities.Dialogs.IMessageDialogOptions = {
                 title: "Save attribute - Sample",
                 subTitle: StringExtensions.EMPTY,
```

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

```
                    message: "Save attribute ?",
                    button1: yesButton,
                    button2: noButton
                };
                    let showMessageDialogClientRequest:
ShowMessageDialogClientRequest<ShowMessageDialogClientResponse> =
                        new
ShowMessageDialogClientRequest(showMessageDialogClientRequestOptions);

                    result =
this.context.runtime.executeAsync<ShowMessageDialogClientResponse>(showMessageDial
ogClientRequest);
                } else {
                    result = Promise.resolve({ canceled: false, data: new
ShowMessageDialogClientResponse(null) });
                }

                return result;
            })
            .then((showMessageDialogClientResponse:
ClientEntities.ICancelableDataResult<ShowMessageDialogClientResponse>):

Promise<ClientEntities.ICancelableDataResult<SaveAttributesOnCartClientResponse>>
=> {

         // Save the attribute value depending on the dialog result.
                let messageDialogResult:
ClientEntities.Dialogs.IMessageDialogResult =
showMessageDialogClientResponse.data.result;

                let result:
Promise<ClientEntities.ICancelableDataResult<SaveAttributesOnCartClientResponse>>;
                if (!ObjectExtensions.isNullOrUndefined(messageDialogResult)) {

                    let attributeValue: ProxyEntities.AttributeTextValue = new
ProxyEntities.AttributeTextValueClass();
                    attributeValue.Name =
PreEndTransactionTrigger.CART_ATTRIBUTE_NAME;
                    attributeValue.TextValue = messageDialogResult.dialogResult
=== PreEndTransactionTrigger.DIALOG_RESULT_YES ?
                        PreEndTransactionTrigger.CART_ATTRIBUTE_VALUE_TRUE :
PreEndTransactionTrigger.CART_ATTRIBUTE_VALUE_FALSE;
                    let attributeValues: ProxyEntities.AttributeValueBase[] =
[attributeValue];
                    let saveAttributesOnCartRequest:
SaveAttributesOnCartClientRequest<SaveAttributesOnCartClientResponse> =
                        new SaveAttributesOnCartClientRequest(attributeValues);
                    result =
this.context.runtime.executeAsync(saveAttributesOnCartRequest);
                } else {
```

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

```
                    result = Promise.resolve({ canceled: false, data: new
SaveAttributesOnCartClientResponse(null) });
                }

                return result;
            });
}
}
```

10. Create a new json file and under the POSAPIExtension folder and name it as manifest.json.
11. In the manifest.json file, copy and paste the below code, delete the default generated code before copying the below code:

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

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

```
{
  "extensionPackages": [
    {
      "baseUrl": "SampleExtensions2"
    },
    {
      "baseUrl": "POSAPIExtension"
    }
  ]
}
```

Note: The extension.json file should always contain two extensions folder names so don't remove the SampleExtensions folder name.

13. 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

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

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.
Note: Please comment both SampleExtensions2 and POSAPIExtension.

```
"exclude": [
    "AuditEventExtensionSample",
    "B2BSample",
    "CustomerSearchWithAttributesSample",
    "FiscalRegisterSample",
    "PaymentSample",
    "PromotionsSample",
    "SalesTransactionSignatureSample",
     //"SampleExtensions2",
    "SampleExtensions",
    "StoreHoursSample",
    "SuspendTransactionReceiptSample",
    "CustomControlExtensions"
     //"POSAPIExtension"
        ],
```

14. Select the POS.Extensions project and click Show all Files in the solution explorer tab
15. Right click and include the SampleExtensions2 folder in the project.
16. Compile and rebuild the project.

How to test your extension:

1. Press F5 and deploy the POS to test your customization.
2. Once the POS is launched, login to POS and add any item to transaction.
3. Add any customer to a transaction.
4. Click the pay button and commit the transaction.
5. POS should show a dialog box asking whether to save the attribute or not.