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 new POS Operation:

This topic explains how to create a new POS operation and add it to the POS Welcome screen. 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.

Retail POS operation executes the business logic for the Retail POS. Operation can execute multiple steps or workflows. All standard Retail operation supports pre and post trigger.

Each operation should implement the following:

1. **Operation request** – Operation request extends from the ExtensionOperationRequestBase and it contains all the inputs for the operation to execute
2. **Operation response** – Operation response extends from the Response and it contains all the response based on the operation execution.
3. **Operation factory** – Operation handler links the operation button click with operation handler.
4. **Operation handler** – Operation handler extends from ExtensionOperationRequestHandlerBase and it contains core logic for the operation. All the business logic should be returned in the handler and it should return the operation response after execution of the operation.

Scenario: Lets create a new operation to do simplified EOD of processing. In this operation, we will call the standard Tender removal, safe drop, tender declaration and Close shift in a sequence.

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 EODSample.
4. Under EODSample create new folder called Operations. In the Operations add a new folder called EndOfDay

   Create the operation request class:
5. In the EndOfDay folder, add a new ts (typescript) file and name it has EndOfDayOperationRequest.ts
6. Add the below import statement to import the relevant entities and context.

```
import { ExtensionOperationRequestBase } from "PosApi/Create/Operations";
import EndOfDayOperationResponse from "./EndOfDayOperationResponse";
```

7. Create a new class called EndOfDayOperationRequest and extend it from ExtensionOperationRequestBase.

```
export default class EndOfDayOperationRequest<TResponse extends
EndOfDayOperationResponse> extends ExtensionOperationRequestBase<TResponse> {
    constructor(correlationId: string) {
        super(5001, correlationId);
    }
       }
```

Note: In the Super method, we are initializing the operation id as 5001, you can use any operation id starting from 4001. 0 - 4000 is reserved for internal Retail POS operations and no two operations should have the same operation id.

Create the operation response class:

8. In the EndOfDay folder, add a new ts (typescript) file and name it has EndOfDayOperationResponse.ts
9. Add the below import statement to import the relevant entities and context.

```
import { Response } from "PosApi/Create/RequestHandlers";
```

10. Create a new class called `EndOfDayOperationResponse` and extend it from Response class.

```
export default class EndOfDayOperationResponse extends Response { }
```

Create the operation handler class:

11. In the EndOfDay folder, add a new ts (typescript) file and name it has EndOfDayOperationRequestHandler.ts
12. Add the below import statement to import the relevant entities and context.

```
import { ExtensionOperationRequestType, ExtensionOperationRequestHandlerBase }
from "PosApi/Create/Operations";
import { CloseShiftOperationRequest, CloseShiftOperationResponse } from
"PosApi/Consume/Shifts";
import { SafeDropOperationRequest, SafeDropOperationResponse } from
"PosApi/Consume/StoreOperations";
import { TenderDeclarationOperationRequest, TenderDeclarationOperationResponse }
from "PosApi/Consume/StoreOperations";
import { TenderRemovalOperationRequest, TenderRemovalOperationResponse } from
"PosApi/Consume/StoreOperations";
import EndOfDayOperationResponse from "./EndOfDayOperationResponse";
import EndOfDayOperationRequest from "./EndOfDayOperationRequest";
import { ClientEntities } from "PosApi/Entities";
```

13. Create a new class called `EndOfDayOperationRequestHandler` and extend it from `ExtensionOperationRequestHandlerBase` class.

```
export default class EndOfDayOperationRequestHandler<TResponse extends
EndOfDayOperationResponse> extends ExtensionOperationRequestHandlerBase<TResponse>
{}
```

14. Each handler should implement two methods:
    a. Supported request
    b. Execute Async
15. Add the supported request type in the class:

```
public supportedRequestType(): ExtensionOperationRequestType<TResponse> {
return EndOfDayOperationRequest;
    }
```

16. Implement the `executeAsync method:`

```
public executeAsync(printRequest: EndOfDayOperationRequest<TResponse>):
Promise<ClientEntities.ICancelableDataResult<TResponse>> {
```

```
this.context.logger.logInformational("Log message from
PrintOperationRequestHandler executeAsync().",
this.context.logger.getNewCorrelationId());

        // Tender Removal
        let tenderRemovalRequest:
TenderRemovalOperationRequest<TenderRemovalOperationResponse> =
            new
TenderRemovalOperationRequest(this.context.logger.getNewCorrelationId());
        return
this.context.runtime.executeAsync(tenderRemovalRequest).then((result:
ClientEntities.ICancelableDataResult<TenderRemovalOperationResponse>)
            :
Promise<Commerce.Client.Entities.ICancelableDataResult<SafeDropOperationResponse>>
=> {

            // Safe Drop
            if (!result.canceled) {
                let safeDropRequest:
SafeDropOperationRequest<SafeDropOperationResponse> =
             new
SafeDropOperationRequest(this.context.logger.getNewCorrelationId());

         return this.context.runtime.executeAsync(safeDropRequest);
     } else {
         return Promise.resolve({
             canceled: true,
             data: null
         });
     }
        }).then((result:
Commerce.Client.Entities.ICancelableDataResult<SafeDropOperationResponse>)
            :
Promise<ClientEntities.ICancelableDataResult<TenderDeclarationOperationResponse>>
=> {

            // Tender Declaration
            if (!result.canceled) {
                let tenderDeclarationRequest:
TenderDeclarationOperationRequest<TenderDeclarationOperationResponse> =
                    new
TenderDeclarationOperationRequest(this.context.logger.getNewCorrelationId());

                return
this.context.runtime.executeAsync(tenderDeclarationRequest);
            } else {
                return Promise.resolve({
                    canceled: true,
                    data: null
                });
            }
        }).then((result:
ClientEntities.ICancelableDataResult<TenderDeclarationOperationResponse>)
```

```
            :
Promise<ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>> => {

            // Close Shift
            if (!result.canceled) {
                return new Promise(
                    (resolve: (value?:
ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>) => void,
reject: (reason?: any) => void) => {
                    // A delay of ten seconds is added here as a work-around for
issues with printing a second receipt to the windows driver
                    // printer before the first dialog is closed. A ten second
delay gives the user a chance to close the first dialog before
                    // the issue occurs.
                    setTimeout(() => { resolve(null); }, 10000);
                }).then(() => {
                    let closeShiftOperationRequest:
CloseShiftOperationRequest<CloseShiftOperationResponse> =
                        new
CloseShiftOperationRequest(this.context.logger.getNewCorrelationId());
                    return
this.context.runtime.executeAsync(closeShiftOperationRequest);
                });
            } else {
                return Promise.resolve({
                    canceled: true,
                    data: null
                });
            }
        }).then((result:
ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>)
            : ClientEntities.ICancelableDataResult<EndOfDayOperationResponse> => {

            return
<ClientEntities.ICancelableDataResult<EndOfDayOperationResponse>>{
                canceled: result.canceled,
                data: result.canceled ? null : new EndOfDayOperationResponse()
            };
        });
           }
```

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 { ExtensionOperationRequestType, ExtensionOperationRequestHandlerBase } from
"PosApi/Create/Operations";
import { CloseShiftOperationRequest, CloseShiftOperationResponse } from
"PosApi/Consume/Shifts";
import { SafeDropOperationRequest, SafeDropOperationResponse } from
"PosApi/Consume/StoreOperations";
import { TenderDeclarationOperationRequest, TenderDeclarationOperationResponse } from
"PosApi/Consume/StoreOperations";
import { TenderRemovalOperationRequest, TenderRemovalOperationResponse } from
"PosApi/Consume/StoreOperations";
import EndOfDayOperationResponse from "./EndOfDayOperationResponse";
import EndOfDayOperationRequest from "./EndOfDayOperationRequest";
import { ClientEntities } from "PosApi/Entities";

/**
 * (Sample) Request handler for the EndOfDayOperationRequest class.
 */

export default class EndOfDayOperationRequestHandler<TResponse extends
EndOfDayOperationResponse> extends ExtensionOperationRequestHandlerBase<TResponse> {

    /**
     * Gets the supported request type.
     * @return {RequestType<TResponse>} The supported request type.
     */
    public supportedRequestType(): ExtensionOperationRequestType<TResponse> {
        return EndOfDayOperationRequest;
    }

    /**
     * Executes the request handler asynchronously.
     * @param {EndOfDayOperationRequest<TResponse>} request The request.
     * @return {Promise<ICancelableDataResult<TResponse>>} The cancelable async result
containing the response.
     */
    public executeAsync(printRequest: EndOfDayOperationRequest<TResponse>):
Promise<ClientEntities.ICancelableDataResult<TResponse>> {

        this.context.logger.logInformational("Log message from
PrintOperationRequestHandler executeAsync().",
this.context.logger.getNewCorrelationId());

        // Tender Removal
        let tenderRemovalRequest:
TenderRemovalOperationRequest<TenderRemovalOperationResponse> =
            new
TenderRemovalOperationRequest(this.context.logger.getNewCorrelationId());
        return this.context.runtime.executeAsync(tenderRemovalRequest).then((result:
ClientEntities.ICancelableDataResult<TenderRemovalOperationResponse>)
            :
Promise<Commerce.Client.Entities.ICancelableDataResult<SafeDropOperationResponse>> =>
{
```

```
            // Safe Drop
            if (!result.canceled) {
                let safeDropRequest:
SafeDropOperationRequest<SafeDropOperationResponse> =
                    new
SafeDropOperationRequest(this.context.logger.getNewCorrelationId());

                return this.context.runtime.executeAsync(safeDropRequest);
            } else {
                return Promise.resolve({
                    canceled: true,
                        data: null
                    });
            }
        }).then((result:
Commerce.Client.Entities.ICancelableDataResult<SafeDropOperationResponse>)
            :
Promise<ClientEntities.ICancelableDataResult<TenderDeclarationOperationResponse>> => {

            // Tender Declaration
            if (!result.canceled) {
                let tenderDeclarationRequest:
TenderDeclarationOperationRequest<TenderDeclarationOperationResponse> =
                    new
TenderDeclarationOperationRequest(this.context.logger.getNewCorrelationId());

                return this.context.runtime.executeAsync(tenderDeclarationRequest);
            } else {
                return Promise.resolve({
                    canceled: true,
                    data: null
                });
            }
        }).then((result:
ClientEntities.ICancelableDataResult<TenderDeclarationOperationResponse>)
            :
Promise<ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>> => {

            // Close Shift
            if (!result.canceled) {
                return new Promise(
                    (resolve: (value?:
ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>) => void, reject:
(reason?: any) => void) => {
                    // A delay of ten seconds is added here as a work-around for
issues with printing a second receipt to the windows driver
                    // printer before the first dialog is closed. A ten second delay
gives the user a chance to close the first dialog before
                    // the issue occurs.
                    setTimeout(() => { resolve(null); }, 10000);
                }).then(() => {
                    let closeShiftOperationRequest:
CloseShiftOperationRequest<CloseShiftOperationResponse> =
                        new
CloseShiftOperationRequest(this.context.logger.getNewCorrelationId());
```

```
                    return
this.context.runtime.executeAsync(closeShiftOperationRequest);
                });
            } else {
                return Promise.resolve({
                    canceled: true,
                    data: null
                });
            }
        }).then((result:
ClientEntities.ICancelableDataResult<CloseShiftOperationResponse>)
            : ClientEntities.ICancelableDataResult<EndOfDayOperationResponse> => {

            return <ClientEntities.ICancelableDataResult<EndOfDayOperationResponse>>{
                   canceled: result.canceled,
                data: result.canceled ? null : new EndOfDayOperationResponse()
            };
        });
    }
    }
```

Create the operation factory class:

17. In the EndOfDay folder, add a new ts (typescript) file and name it has EndOfDayOperationRequestFactory.ts
18. Add the below import statement to import the relevant entities and context.

```
import EndOfDayOperationResponse from "./EndOfDayOperationResponse";
import EndOfDayOperationRequest from "./EndOfDayOperationRequest";
import { ExtensionOperationRequestFactoryFunctionType, IOperationContext } from
"PosApi/Create/Operations";
import { ClientEntities } from "PosApi/Entities";
```

19. Let's add a function to link the operation handler and the operation button.

```
let getOperationRequest:
ExtensionOperationRequestFactoryFunctionType<EndOfDayOperationResponse> =
    /**
     * Gets an instance of EndOfDayOperationRequest.
     * @param {number} operationId The operation Id.
     * @param {string[]} actionParameters The action parameters.
     * @param {string} correlationId A telemetry correlation ID, used to group
events logged from this request together with the calling context.
     * @return {EndOfDayOperationRequest<TResponse>} Instance of
EndOfDayOperationRequest.
     */
    function (
        context: IOperationContext,
        operationId: number,
        actionParameters: string[],
        correlationId: string
    ):
Promise<ClientEntities.ICancelableDataResult<EndOfDayOperationRequest<EndOfDayOper
ationResponse>>> {

        let operationRequest: EndOfDayOperationRequest<EndOfDayOperationResponse>
= new EndOfDayOperationRequest<EndOfDayOperationResponse>(correlationId);
        return
Promise.resolve(<ClientEntities.ICancelableDataResult<EndOfDayOperationRequest<End
OfDayOperationResponse>>>{
            canceled: false,
            data: operationRequest
        });
    };

       export default getOperationRequest;
```

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 EndOfDayOperationResponse from "./EndOfDayOperationResponse";
import EndOfDayOperationRequest from "./EndOfDayOperationRequest";
import { ExtensionOperationRequestFactoryFunctionType, IOperationContext } from
"PosApi/Create/Operations";
import { ClientEntities } from "PosApi/Entities";

let getOperationRequest:
ExtensionOperationRequestFactoryFunctionType<EndOfDayOperationResponse> =
    /**
     * Gets an instance of EndOfDayOperationRequest.
     * @param {number} operationId The operation Id.
     * @param {string[]} actionParameters The action parameters.
     * @param {string} correlationId A telemetry correlation ID, used to group
events logged from this request together with the calling context.
     * @return {EndOfDayOperationRequest<TResponse>} Instance of
EndOfDayOperationRequest.
     */
    function (
        context: IOperationContext,
        operationId: number,
        actionParameters: string[],
        correlationId: string
    ):
Promise<ClientEntities.ICancelableDataResult<EndOfDayOperationRequest<EndOfDayOper
ationResponse>>> {

        let operationRequest: EndOfDayOperationRequest<EndOfDayOperationResponse>
= new EndOfDayOperationRequest<EndOfDayOperationResponse>(correlationId);
        return
Promise.resolve(<ClientEntities.ICancelableDataResult<EndOfDayOperationRequest<End
OfDayOperationResponse>>>{
            canceled: false,
            data: operationRequest
        });
    };

export default getOperationRequest;
```

20. Create a new json file and under the EODSample folder and name it as manifest.json.
21. In the manifest.json file, copy and paste the below code:

```
{
 "$schema": "../manifestSchema.json",
 "name": "Pos_Extensibility_Samples",
 "publisher": "Microsoft",
 "version": "7.2.0",
 "minimumPosVersion": "7.2.0.0",
 "components": {
   "create": {
      "operations": [
     {
     "operationId": "5001",
     "operationRequestFactoryPath":
"Operations/EndOfDay/EndOfDayOperationRequestFactory",
     "operationRequestHandlerPath":
"Operations/EndOfDay/EndOfDayOperationRequestHandler"
    }
   ]
  }
 }
}
```

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

```
{
 "extensionPackages": [
  {
   "baseUrl": "SampleExtensions2"
  },
  {
   "baseUrl": "POSAPIExtension"
  },
  {
   "baseUrl": "CustomColumnExtensions"
  },
  {
   "baseUrl": "EODSample"
  }

 ]
   }
```

23. 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.

```
"extends": "../tsconfigs/tsmodulesconfig",

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

24. Compile and rebuild the project.

Add Custom operation button to the POS layout in HQ:

25. Go to Retail and commerce > Channel setup > POS setup > POS > Operations in Dynamics 365 for Retail.
26. Create a new operation named "EOD" with ID "5001".
27. Go to Retail and commerce > Channel setup > POS setup > POS > Button grids.
28. Filter for "F2W5M".
29. Click on button "Designer" and follow the instructions to install the designer. If prompted enter the dynamics 365 for Retail username and password.
30. Right-click on the designer area and select Add new row and in the new button right click and select "Button properties".
31. Select Action = "EOD”. Click OK and close the Designer.
32. Go to Retail and commerce > Retail IT > Distribution schedule, select "1090" and click "Run now".

How to validate your extension:

33. Press F5 and deploy the POS to test your customization.
34. On the welcome screen click the new EOD operation button and follow the end of day scenario..