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

Add custom columns to POS transaction grid:

This topic explains how to add new custom column in the POS transaction page using the screen layout designer. 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.

You can add more information to the Retail POS transaction page by using custom column feature. Custom column can be added to the transaction page by using the screen layout designer, from the screen layout designer you can add the custom column to the POS transaction (receipt grid). You can adjust width and position of the columns by using the designer. We added 10 custom columns in the layout for extensions scenarios, from the designer you drag up to ten custom columns, these custom columns are already added part of the designer metadata, you need to add the column to your layout and the run the distribution job and POS will show the column in the transaction grid.

Scenario/business problem

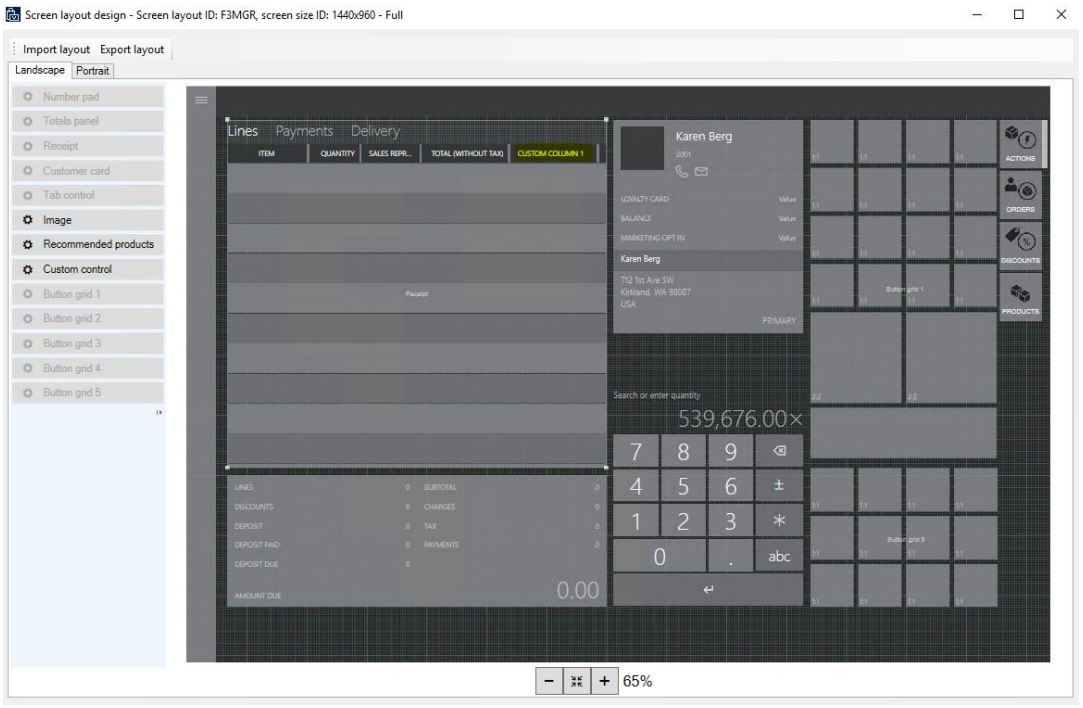
Let's add custom column in POS transaction to show the cart line number.

Lab actions

Add new custom control:

1. Login to Dynamics 365 for Retail.
2. Navigate to Retail > Channel setup > POS setup > POS > Screen layouts (Search for Screen layout in the search bar)
3. Select the F3MGR screen layout ID and click the Designer button in the action bar.
4. Follow the instructions if prompted to install and enter the AAD credentials to launch the designer.
5. Select the 1440x960 – Full layout from the layout sizes and click the Layout designer button.
6. If prompted click Open and follow the instruction to install the designer tool.
7. After installing it will ask for AAD credentials, provide the details to launch the designer.
8. In the designer right click the transaction grid (receipt grid) and select customize.
9. In the customization – Receipt windows, select the Lines in the pivot panel drop down.
10. In the Available columns window select the CUSTOM COLUMN 1 and click the **> (arrow)** button to move the column to the Selected columns.
11. Click OK to save and close the window.
12. Please adjust the column width in the transaction grid using the screen layout designer to make sure the column is visible.

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



13. Click the X button in the designer to close the designer.
14. When prompted to Save changes, click Yes. If you click No the changes will not be saved.
15. Navigate to Retail > Retail IT > Distribution schedule
16. Select the Registers (1090) job and click Run now.

Add business logic to custom control:

17. Open visual studio 2015 in administrator mode.
18. Open ModernPOS solution from …\RetailSDK\POS
19. Under the POS.Extensions project create a new folder called CustomColumnExtensions.
20. Under CustomColumnExtensions, create new folder called Cart.
21. Under Cart, create new folder called LinesGrid.
22. In the LinesGrid folder, add a new ts (typescript) file and name it has CustomColumn1Configuration.ts
23. Add the below import statement to import the relevant entities and context.

```
import {
    ICustomLinesGridColumnContext,
 CustomLinesGridColumnBase
} from "PosApi/Extend/Views/CartView";
import { CustomGridColumnAlignment } from "PosApi/Extend/Views/CustomGridColumns";
import { ProxyEntities } from "PosApi/Entities";
```

24. Create a new class called LinesCustomGridColumn1 and extend it from CustomLinesGridColumnBase.

```
export default class LinesCustomGridColumn1 extends CustomLinesGridColumnBase {}
```

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

25. Inside the class declare private variable to capture the selected tender lines

```
private _selectedTenderLines: ProxyEntities.TenderLine[];
```

26. Create a class constructor method to initialize the context.

```
constructor(context: ICustomLinesGridColumnContext) {
        super(context);
           }
```

27. Add the below methods for the columns title and alignment

```
public title(): string {
        return "Line number";
    }

public alignment(): CustomGridColumnAlignment {
 return CustomGridColumnAlignment.Right;
    }
```

28. Add the column compute value method, which returns the line number

```
public computeValue(cartLine: ProxyEntities.CartLine): string {
       return cartLine.LineNumber.toString();
   }
```

The overall class should look like below:

```
/**
 * 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 {
    ICustomLinesGridColumnContext,
    CustomLinesGridColumnBase
} from "PosApi/Extend/Views/CartView";
import { CustomGridColumnAlignment } from "PosApi/Extend/Views/CustomGridColumns";
import { ProxyEntities } from "PosApi/Entities";

export default class LinesCustomGridColumn1 extends CustomLinesGridColumnBase {

    constructor(context: ICustomLinesGridColumnContext) {
        super(context);
    }

    public title(): string {
        return "Line number";
    }

    public computeValue(cartLine: ProxyEntities.CartLine): string {
```

```
            return cartLine.LineNumber.toString();
        }

        public alignment(): CustomGridColumnAlignment {
            return CustomGridColumnAlignment.Right;
        }
           }
```

29. Create a new json file and under the CustomColumnExtensions folder and name it as manifest.json.
30. 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_Samples",
  "publisher": "Microsoft",
  "version": "7.2.0",
  "minimumPosVersion": "7.2.0.0",
  "components": {
    "extend": {
      "views": {
        "CartView": {
          "linesGrid": {
            "customColumn1": { "modulePath":
"Cart/LinesGrid/CustomColumn1Configuration" }
          }
        }
      }
    }
  }
}
```

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

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

  ]
}
```

32. 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",
    //"SampleExtensions2",
    "SampleExtensions",
    "StoreHoursSample",
    "SuspendTransactionReceiptSample"
    //"POSAPIExtension",
    //"CustomColumnExtensions"
        ],
```

33. Compile and rebuild the project.

Validate the customization:

34. Login to MPOS using 000160 as operator id and 123 as password.
35. Click the current transaction button on the welcome screen
36. Add any item (0005) to transaction.
37. The custom column should display the line number.