Produmex WMS Customization Guide

1. Introduction

Product Customization Overview

Three key layers contribute to the overall functionality and user experience of an application. These layers play crucial roles in extending the capabilities of an application, adapting it to specific user requirements, and enhancing the overall user experience.

- **Addon Configuration Layer:** flexible and modular approach to enhance the standard functionality possibilities. It includes sections as organizational structure (aka as OS), OS Settings, item master data, business master data and default forms.
- **WMS Application Layer:** focuses on the functionalities that we will execute on the device and its behavior. It includes sections as thin client workflow, thin client parameter set and extension parameters configuration.
- **Customization Layer:** extends the functionality of the standard. It includes concepts as .cs scripts, data base management, hook scripts and customization framework of the Thin Client or subflows. Find all the information about how to **Customization Framework on Mobile Client.**

Related to subflows, they are complex structures with complex objects, functions, relations and interdependency with other part of the codes. Understanding and responsibly modify the workflows could be a difficult and time costly activity, therefore from the product department we do not support modifications on subflows.

Helpful Tips and Resources

Click the link below to visit our Article site, where you will find examples and useful information. We are continuously adding new articles featuring the most common customizations.

Produmex WMS Articles: Customization examples (more subsections are available)

The following content is NOT only to developers but to consultants with strong coding knowledge, for more information check the link below:
Customization examples (more subsections are available)

1.1. Required Skills

For advanced scripting in Produmex WMS you will need a strong knowledge about the following programing languages.

Programing languages as required skills:

- **C# - C-Sharp**
  1. **Pltform:** Primarily used with the .NET framework, but also supports cross-platform development with .NET Core.
  2. **Object-Oriented:** C# is an object-oriented programming language, which means it focuses on objects and data rather than actions and logic.
  3. **Type-Safe:** It ensures that code is safe from type errors, which helps in preventing bugs.
  4. **Versatile:** C# can be used for a wide range of applications, including web, mobile, desktop, and game development.
- **SQL**
  1. **Purpose:** SQL is a standard language for managing and manipulating relational databases.
  2. **Data Manipulation:** SQL allows you to insert, update, delete, and retrieve data from a database.
  3. **Data Definition:** You can create and modify database structures like tables, indexes, and views.
  4. **Data Control:** SQL provides commands to control access to data and ensure data integrity.

1.2. Scripting

It is important to lay down general basics about scripting. Generally speaking scripting is a powerful tool for developers and IT professionals, enabling them to automate tasks, enhance functionality, and create dynamic applications.

Definition: Scripting refers to writing a series of commands that are executed by a certain runtime environment. These commands are typically used to automate tasks that would otherwise be performed manually. Here are some key points about scripting:

- **Interpreted Language:** Scripting languages are usually interpreted rather than compiled. This means the code is executed line-by-line by an interpreter.
- **Automation:** Scripts are often used to automate repetitive tasks, such as file manipulation, data processing, and system administration.
- **Integration:** Scripting languages can integrate with other software applications to extend their functionality.

The benefits of scripting:

- **Ease of Use:** Scripting languages are generally easier to learn and use compared to compiled languages.
- **Flexibility:** They allow for quick changes and iterations.
- **Efficiency:** Automate repetitive tasks, saving time and reducing errors.

2. Hookflow Script

Hookflow script is used for inserting custom logic at a certain point in the flow

There are *input* and *output* parameters defined in the Hookflow class.
The value from the parameter can be loaded by the *Get()* method.
Set value in the parameter can be done by the Set() methd: BackRequested.Set(true);
It is not possible to define additional input or output parameters.

Customization Articles for WMS scripting:
How to make custom bin location list in Put Away
How to make custom bin location list in AdHoc move

There are two classes in every HookFlow script that are pre defined in the *Execute()* method.
It is the **Session** and the **ISboProviderService** classes.

References:

```
using Produmex.Foundation.SlimScreen;
using Produmex.Foundation.Wwf.Sbo.LocalServices;
```

Remove the comment before the variables in case you would like to use them!

```
Session session = GetScopeParameter("Session") as Session;
ISboProviderService sboProviderService = GetScopeParameter("
<WwfService>ISboService") as ISboProviderService;
```

2.1. Database Connection

Necessary classes:

Class	Reference
PmxDbConnection	`Produmex.Foundation.Data.Sbo;`

You can get the connection from **sboProviderService**.

Example - creating a Picklist provider with connection

```
sboProviderService.InvokeMethodWithDbConnection<object>(false, false,
null, null, delegate (PmxDbConnection conn, object[] parameters)
{

PmxPickListProvider plProv = new PmxPickListProvider(conn);
PmxPickList pickList = plProv.GetBO( WaveKey.Get() );
return null;
});
```

2.2. Screens

Customization Articles for WMS scripting:
How to display product image after selecting the item in Picking and Ad-Hoc picking

Screen can be generated by the *ShowScreen* method of the *session* object.
There are different types that we can use.

Necessary classes:

Class	Message

Reference	`using Produmex.Foundation.Messages;`
	`using Produmex.Foundation.Wwf.Sbo.LocalServices;`
	`using Produmex.Foundation.SlimScreen;`

2.2.1. Message Screen type

Parameters

Name	Type	Description
MessageKey	String	Set your message
ShowButton	Bool	-

Example:

```
Message msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
this.DefaultCultureInfo, BuildParamCollection(
```

```
      "MessageKey", "YOUR MESSAGE" + DLoc,
      "ShowButton", true
));
msg = WaitForMessage();
```

Result:

2.2.2. Image screen type

Parameters collection

Name	Type	Description
TitleKey	String	Title of the screen
ImagePath	String	Full path of the picture
MessageKey	String	Message under the screen
ShowButton	Bool	-

Example:

```
Message msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wImageScreen),
            this.DefaultCultureInfo, BuildParamCollection(
                "TitleKey", "Picture of the product",
                "MessageKey", "message under the picture",
                "ImagePath", "<PATH OF THE IMAGE>",
                "ShowButton", true
            ));
            msg = WaitForMessage();
```

Result:

2.2.3. Enter String Value type

You can capture additional text information on this screen. The captured data can be get from the message object.
The value can be used in the Hookflow for further processing, or if the Hookflow script has an output parameter, then we can put the captured value into the output parameter.

Parameters

Name	Type	Description
InitialErrorKey	String	n.a. in custom usage
TitleKey	String	Title of the screen
Information	String	Additional information on the screen
Parameters	Object of sting	n.a. in custom usage
AllowToGoBack	Bool	-
ForceDataEntry	Bool	-
AllowMultiLine	Bool	-
MinimumNumberOfCharacters	Int	Minimum number of characters that must be typed

Example:
The entered text will be used on a message screen.

```
string initialErrorKey = null;
        string FreeText = "";

        Message msg = null;
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Inter
faces.IEnterStringValueScreen),
                DefaultCultureInfo.Get(), BuildParamCollection(
                        "InitialErrorKey", initialErrorKey,
                        "TitleKey", "Title of the screen",
                        "Information", "Information text",
                        "Parameters", new object[] { "" },
                        "AllowToGoBack", true,
                        "ForceDataEntry", true,
                        "AllowMultiLine", true,
                        "MinimumNumberOfCharacters", 5
                        ),
                    WorkflowId,
nameof(PickingScript_Screens.EnterStringValueScreen1));
        msg = WaitForMessage();
        if (msg.Name.EndsWith(".StringEntered"))
        {
            FreeText = ExtractParameter<string>(msg.Parameters,
"stringValue");
        }

        msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
        this.DefaultCultureInfo, BuildParamCollection(
            "MessageKey", "Entered text: " + FreeText,
            "ShowButton", true
        ));
        msg = WaitForMessage();
```

Result:

2.2.4. Select Product Screen type

You can create an item list in a DataSet object to select an item from a list. The captured data can be get from the message object.
The value can be used in the Hookflow for further processing, or if the Hookflow script has an output parameter, then we can put the captured value into the output parameter.

Parameters

Name	Type	Description
InitialErrorKey	String	n.a. in custom usage
TitleKey	String	Title of the screen
Information	String	Additional information on the screen
Parameters	Object of sting	n.a. in custom usage
AllowToGoBack	Bool	-
ForceDataEntry	Bool	-
AllowMultiLine	Bool	-
MinimumNumberOfCharacters	Int	Minimum number of characters that must be typed

Example:
The selected item will be used on a message screen.

```
string initialErrorKey = null;
        string FreeText = "";
        Message msg = null;
        DataSet dsItems = null;

        string query = "SELECT DISTINCT PMX_OITMANAGED_BY_PMX.ItemCode
AS ProductCode, PMX_OITMANAGED_BY_PMX.U_PMX_CUDE AS ProductDescription,
PMX_OITMANAGED_BY_PMX.CodeBars AS GTIN,
PMX_OITMANAGED_BY_PMX.U_PMX_HBBD, PMX_OITMANAGED_BY_PMX.U_PMX_PILR,
PMX_OITMANAGED_BY_PMX.ManBtchNum, PMX_OITMANAGED_BY_PMX.U_PMX_LOUN,
PMX_OITMANAGED_BY_PMX.NumInBuy, PMX_OITMANAGED_BY_PMX.BuyUnitMsr,
PMX_OITMANAGED_BY_PMX.InvntryUom, PMX_OITMANAGED_BY_PMX.CodeBars AS
CodeBars, PMX_OITMANAGED_BY_PMX.ItemName FROM PMX_OITMANAGED_BY_PMX WITH
(NOLOCK)  WHERE PMX_OITMANAGED_BY_PMX.InvntItem = 'Y' AND
PMX_OITMANAGED_BY_PMX.InvntItem = 'Y'  AND  NOT (
PMX_OITMANAGED_BY_PMX.frozenFor = 'Y' AND ( (
PMX_OITMANAGED_BY_PMX.frozenFrom IS NULL OR CURRENT_TIMESTAMP >=
PMX_OITMANAGED_BY_PMX.frozenFrom ) AND ( PMX_OITMANAGED_BY_PMX.frozenTo
IS NULL OR CURRENT_TIMESTAMP < DATEADD( day, 1,
PMX_OITMANAGED_BY_PMX.frozenTo ) ) ) ) ORDER BY ProductDescription";
```

```
        dsItems = sboProviderService.RunView(false, null, null, query);
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Inter
faces.ISelectProductScreen),
            DefaultCultureInfo.Get(), BuildParamCollection(
            "InitialErrorKey", initialErrorKey,
            "TitleKey", "Title of the screen",
                "ProductDS", dsItems
                ),
    WorkflowId,
        nameof(ChecksScript_Screens.SelectProductScreen1));
    msg = WaitForMessage();

        if (msg.Name.EndsWith(".ProductSelected"))
        {
            FreeText = ExtractParameter<string>( msg.Parameters,
"itemCode" );
        }

        msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
        this.DefaultCultureInfo, BuildParamCollection(
            "MessageKey", "Entered text: " + FreeText,
            "ShowButton", true
        ));
        msg = WaitForMessage();
```

Result:

2.2.5. Yes/No question Screen type

Parameters

Name,Type,Description TitleKey,String,Name of the screen MessageKey,String,question string

Example:

```
Message msg = null;
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Interface
s.IDecisionScreen),
DefaultCultureInfo.Get(), BuildParamCollection(
"TitleKey", "Title of the screen",
"MessageKey", "Do you want to continue?"),
WorkflowId,
```

```
nameof(ReceptionScript_Screens.DecisionScreen22));
msg = WaitForMessage();

if (msg.Name.EndsWith(".Yes"))
{
// goto Step_ClearDataBeforeNextItem;
}
if (msg.Name.EndsWith(".No"))
{
BackRequested.Set(true);
}
```

2.1. Database Connection

Necessary classes:

Class	Reference
PmxDbConnection	`Produmex.Foundation.Data.Sbo;`

You can get the connection from **sboProviderService**.

Example - creating a Picklist provider with connection

```
sboProviderService.InvokeMethodWithDbConnection<object>(false, false,
null, null, delegate (PmxDbConnection conn, object[] parameters)
{

PmxPickListProvider plProv = new PmxPickListProvider(conn);
PmxPickList pickList = plProv.GetBO( WaveKey.Get() );
return null;
});
```

2.2. Screens

Customization Articles for WMS scripting:
How to display product image after selecting the item in Picking and Ad-Hoc picking

Screen can be generated by the *ShowScreen* method of the *session* object.

There are different types that we can use.

Necessary classes:

Class	Message

Reference	`using Produmex.Foundation.Messages;`
	`using Produmex.Foundation.Wwf.Sbo.LocalServices;`
	`using Produmex.Foundation.SlimScreen;`

2.2.1. Message Screen type

Parameters

Name	Type	Description
MessageKey	String	Set your message
ShowButton	Bool	-

Example:

```
Message msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
this.DefaultCultureInfo, BuildParamCollection(
     "MessageKey", "YOUR MESSAGE" + DLoc,
     "ShowButton", true
));
msg = WaitForMessage();
```

Result:

2.2.2. Image screen type

Parameters collection

Name	Type	Description
TitleKey	String	Title of the screen
ImagePath	String	Full path of the picture
MessageKey	String	Message under the screen
ShowButton	Bool	-

Example:

```
Message msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
```

```
wImageScreen),
            this.DefaultCultureInfo, BuildParamCollection(
                "TitleKey", "Picture of the product",
                "MessageKey", "message under the picture",
                "ImagePath", "<PATH OF THE IMAGE>",
                "ShowButton", true
            ));
            msg = WaitForMessage();
```

Result:

2.2.3. Enter String Value type

You can capture additional text information on this screen. The captured data can be get from the message object.
The value can be used in the Hookflow for further processing, or if the Hookflow script has an output parameter, then we can put the captured value into the output parameter.

Parameters

Name	Type	Description
InitialErrorKey	String	n.a. in custom usage
TitleKey	String	Title of the screen
Information	String	Additional information on the screen
Parameters	Object of sting	n.a. in custom usage
AllowToGoBack	Bool	-
ForceDataEntry	Bool	-
AllowMultiLine	Bool	-
MinimumNumberOfCharacters	Int	Minimum number of characters that must be typed

Example:
The entered text will be used on a message screen.

```
string initialErrorKey = null;
        string FreeText = "";

        Message msg = null;
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Inter
faces.IEnterStringValueScreen),
                DefaultCultureInfo.Get(), BuildParamCollection(
                        "InitialErrorKey", initialErrorKey,
                        "TitleKey", "Title of the screen",
                        "Information", "Information text",
                        "Parameters", new object[] { "" },
                        "AllowToGoBack", true,
                        "ForceDataEntry", true,
```

```
                        "AllowMultiLine", true,
                        "MinimumNumberOfCharacters", 5
                        ),
                    WorkflowId,
nameof(PickingScript_Screens.EnterStringValueScreen1));
        msg = WaitForMessage();
        if (msg.Name.EndsWith(".StringEntered"))
        {
            FreeText = ExtractParameter<string>(msg.Parameters,
"stringValue");
        }

        msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
        this.DefaultCultureInfo, BuildParamCollection(
            "MessageKey", "Entered text: " + FreeText,
            "ShowButton", true
        ));
        msg = WaitForMessage();
```

Result:

2.2.4. Select Product Screen type

You can create an item list in a DataSet object to select an item from a list. The captured data can be get from the message object.
The value can be used in the Hookflow for further processing, or if the Hookflow script has an output parameter, then we can put the captured value into the output parameter.

Parameters

Name	Type	Description
InitialErrorKey	String	n.a. in custom usage
TitleKey	String	Title of the screen
Information	String	Additional information on the screen
Parameters	Object of sting	n.a. in custom usage
AllowToGoBack	Bool	-
ForceDataEntry	Bool	-
AllowMultiLine	Bool	-
MinimumNumberOfCharacters	Int	Minimum number of characters that must be typed

Example:
The selected item will be used on a message screen.

```
string initialErrorKey = null;
        string FreeText = "";
        Message msg = null;
        DataSet dsItems = null;

        string query = "SELECT DISTINCT PMX_OITMANAGED_BY_PMX.ItemCode
AS ProductCode, PMX_OITMANAGED_BY_PMX.U_PMX_CUDE AS ProductDescription,
PMX_OITMANAGED_BY_PMX.CodeBars AS GTIN,
PMX_OITMANAGED_BY_PMX.U_PMX_HBBD, PMX_OITMANAGED_BY_PMX.U_PMX_PILR,
PMX_OITMANAGED_BY_PMX.ManBtchNum, PMX_OITMANAGED_BY_PMX.U_PMX_LOUN,
PMX_OITMANAGED_BY_PMX.NumInBuy, PMX_OITMANAGED_BY_PMX.BuyUnitMsr,
PMX_OITMANAGED_BY_PMX.InvntryUom, PMX_OITMANAGED_BY_PMX.CodeBars AS
CodeBars, PMX_OITMANAGED_BY_PMX.ItemName FROM PMX_OITMANAGED_BY_PMX WITH
(NOLOCK)  WHERE PMX_OITMANAGED_BY_PMX.InvntItem = 'Y' AND
PMX_OITMANAGED_BY_PMX.InvntItem = 'Y'  AND  NOT (
PMX_OITMANAGED_BY_PMX.frozenFor = 'Y' AND ( (
PMX_OITMANAGED_BY_PMX.frozenFrom IS NULL OR CURRENT_TIMESTAMP >=
PMX_OITMANAGED_BY_PMX.frozenFrom ) AND ( PMX_OITMANAGED_BY_PMX.frozenTo
IS NULL OR CURRENT_TIMESTAMP < DATEADD( day, 1,
PMX_OITMANAGED_BY_PMX.frozenTo ) ) ) ) ORDER BY ProductDescription";

        dsItems = sboProviderService.RunView(false, null, null, query);
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Inter
faces.ISelectProductScreen),
            DefaultCultureInfo.Get(), BuildParamCollection(
            "InitialErrorKey", initialErrorKey,
            "TitleKey", "Title of the screen",
                "ProductDS", dsItems
                ),
    WorkflowId,
        nameof(ChecksScript_Screens.SelectProductScreen1));
    msg = WaitForMessage();

        if (msg.Name.EndsWith(".ProductSelected"))
        {
            FreeText = ExtractParameter<string>( msg.Parameters,
"itemCode" );
        }

        msg = null;
session.ShowScreen(typeof(Produmex.Foundation.SlimScreen.Interfaces.ISho
wMessageScreen),
        this.DefaultCultureInfo, BuildParamCollection(
            "MessageKey", "Entered text: " + FreeText,
            "ShowButton", true
        ));
        msg = WaitForMessage();
```

Result:

2.2.5. Yes/No question Screen type

Parameters

Name,Type,Description TitleKey,String,Name of the screen MessageKey,String,question string

Example:

```
Message msg = null;
session.ShowCustomizedScreen(typeof(Produmex.Foundation.SlimScreen.Interface
s.IDecisionScreen),
DefaultCultureInfo.Get(), BuildParamCollection(
"TitleKey", "Title of the screen",
"MessageKey", "Do you want to continue?"),
WorkflowId,
nameof(ReceptionScript_Screens.DecisionScreen22));
msg = WaitForMessage();

if (msg.Name.EndsWith(".Yes"))
{
// goto Step_ClearDataBeforeNextItem;
}
if (msg.Name.EndsWith(".No"))
{
BackRequested.Set(true);
}
```

2.3. Example solutions

2.3.1 Get data from LogisticUnits parameter

Use the Get() method to read the parameter:
LogisticUnits.Get()

Example:

```
protected override void Execute()
```

```
{
// Parameters in scope
Session session = GetScopeParameter("Session") as Session;
ISboProviderService sboProviderService =
GetScopeParameter("<WwfService>ISboService") as ISboProviderService;

foreach(LogisticUnitGoodsReceipt LUGR in LogisticUnits.Get()) {
s_log.Error("CHECK DATA IN LOG - SSCC: " + LUGR.SSCC);
foreach (LogisticUnitItemGoodReceipt LIGR in LUGR.ItemsOnLogisticUnit) {
s_log.Error("CHECK DATA IN LOG - ItemCode: " + LIGR.ItemCode);
s_log.Error("CHECK DATA IN LOG - Quantity: " +
LIGR.Quantity.ToString());
foreach (PackagingTypeInfo P in LIGR.FullListOfPackagingTypes){
s_log.Error("CHECK DATA IN LOG - Uom: " + P.PackagingTypeName);
s_log.Error("CHECK DATA IN LOG - Quantity: " + P.Quantity.ToString());
}
}
}
}
```

3. Standalone Script

Standalone Script is used for starting a logic individually from Produmex WMS by the WMS robot tool. It can be scheduled in the Windows Scheduler. This can be for example a very complex replenishment order generation.

3.1. Database Connection

Necessary classes:

Class	Reference
TransactionScope	`using System.Transactions;`
PmxDbConnection	`using Produmex.Foundation.Data.Sbo;`

Steps:

- **1. Define the connection string:**

Copy your connection string text from any config file of the Produmex WMS tools or Fat Client application.

```
private static string CONNECTION_STRING = “”;
```

- **2. Start a transaction:**

```
using (TransactionScope scope =
PmxDbConnection.GetNewTransactionScope())
```

- **3. Create the connection:**

```
using (PmxDbConnectionDirect conn =
PmxDbConnectionMgr.GetDirectConnection(SboConnectionString.ParseStringTo
Object(CONNECTION_STRING)))
```

Example:

```
using ( TransactionScope scope =
PmxDbConnection.GetNewTransactionScope())
{
      using (PmxDbConnectionDirect conn =
PmxDbConnectionMgr.GetDirectConnection(SboConnectionString.ParseStringTo
Object(CONNECTION_STRING)))
{
conn.Open();
                    Console.WriteLine("Connection is open");

                     string query = @"SELECT TOP 1 DocEntry FROM
PMX_PLHE WHERE DocStatus = 'O' ORDER BY DocEntry ";

                       using (ISboRecordset rs1 =
SboRecordsetHelper.RunQuery(s_log, query, conn))
                        {
                        while (!rs1.EoF)
                         {
                    … Do Something …
                               rs1.MoveNext();
                        }
                     }
                     }
                    scope.Complete();   //Complete transaction
}
```

4. PMX Classes

You can find the list of methods and fields of the Produmex WMS classes in this section.

4.1. General

4.1.1. SboRecordsetHelper

Fields	-	-
Methods	`ISboRecordset RunQuery(ILog log, string query, PmxDbConnection dbConn)`	Provide the result of the query into ISboRecordset class

4.1.2. ISboRecordset

Fields	EoF	Bool
Methods	`void MoveFirst()`	-
	`void MoveLast()`	-
	`void MoveNext()`	-
	`void MovePrevious()`	-
	`void RedoOriginalQuery()`	-
	`GetTypedValue<string>("<Col_Name>")` `GetTypedValue<int>("Col_Name")` `GetTypedValue<double>("Col_Name")`	Read data from the recordset

4.1.3. PmxItemAllConnectionsProvider

Initialization	`new PmxItemAllConnectionsProvider (conn);`
Methods	`PmxItemInfo GetCachedItemInfo (string itemCode)`
	`PmxItemInfo GetItemInfo (string itemCode)`

4.1.4. PmxItemInfo

Fields	private string ItemCode;
	private bool IsLogisticCarrier;
	private bool IsLogisticUnit;
	private bool IsReturnableItem;
	private string Description;
	private bool HasBestBeforeDate;
	private bool IsInventoryItem;
	private bool HasBatchnumber;
	private string CodeBars;
	private int? ItemLabelReportKey;
	private int NumberOfCopiesItemLabel;
	private bool NeedsReasonForPurchaseReturn;
	private bool NeedsReasonForSalesReturn;
	private string QualityStatusCodeProduction;
	private string QuarantinedQualityStatusCodeReception;
	private string ReleasedQualityStatusCodeReception;
	private string QualityStatusCodeReceptionController;
	private string QualityStatusCodeSalesReturn;
	private bool ScanBaseComponent;
	private PmxItemInfo m_baseItemInfo;
	private int ShelfLifeInDays;
	private string ExpiryDefinitionCodeForProduction;
	private string ExpiryDefinitionCodeForReception;
	private string InventoryUom;
	private string m_purchaseUom;
	private string SalesUom;
	private double NumberOfItemsPerPurchaseUnit;
	private double NumberOfItemsPerSalesUnit;
	private string SalesBarcode;
	private string PurchaseBarcode;
	private bool PrintItemLabel;
	private string Uom2;
	private double DefaultQuantityUom2;
	private bool CorrectStockUom;
	private bool CorrectStockUom2;
	private bool UseUom2;
	private PmxUomToUse UomToUseForPurchase;
	private PmxUomToUse UomToUseForInventoryTransitions;
	private PmxUomToUse UomToUseForSales;
	private int UomDecimals;
	private int Uom2Decimals;
	private bool HasSecondBatchNumber;
	private string PictureName;
	private string PackingImage;
	private string PackingRemarks;
	private string VendorItemDescription;
	private string CustomItemDescription;
	private bool HasNoValue;
	private string LowestSellablePackagingType;
	private int ShelfLifeInDaysReception;
	private string SerialNumberFormat;
	private bool CreateSsccOnReception;
	private Collection<PmxItemPackagingTypeInfo> ListOfPackagingTypes = new Collection<PmxItemPackagingTypeInfo>();
	private PmxDictionary<string, double> BarcodesAndUomQuantities = new PmxDictionary<string, double>();
	private PmxDictionary<string, double> PurchaseBarcodesAndUomQuantities = new PmxDictionary<string, double>();
	private PmxDictionary<string, double> SalesBarcodesAndUomQuantities = new PmxDictionary<string, double>();
	private PmxDictionary<string, string> DefaultWarehouseLocationOrZone = new PmxDictionary<string, string>();

Methods	-

4.1.5. PmxItemTransactionalInfoProvider

Initialization	`new PmxItemTransactionalInfoProvider (conn);`	-
Methods	`void ChangeBBDOnBatch (int itriKey, DateTime newBBD, bool changeExpDate = false)`	Change BBD of a batch number itriKey = PMX_ITRI.InternalKey
	`void ChangeInternalBatchNumber (int itriKey, string newBatchNumber2)`	Change 2nd Batch number of a batch number itriKey = PMX_ITRI.InternalKey

4.1.6. PmxLogisticUnitIDProvider

Initialization	`new PmxLogisticUnitIDProvider(conn);`	-
Methods	`int GenerateNewLogisticUnit (string supplierPalletNumber)`	Generates a new SSCC, the result of the method is PMX_LUID.InternalKey
	`bool CheckIsLuidsInInventory (Collection<int> luids)`	-
	`bool CheckIsSSCCMasterLogisticUnit (string sscc)`	-
	`int GetLUIDBySSCC (string sscc)`	PMX_LUID.InternalKey

4.1.7. PmxOseBin

Fields	`private string Code`
	`public string Name`
	`public bool IsActive`
	`public bool IsPickLocation`
	`public int Sequence`
	`public bool CanBeLinedUp`
	`public double? MaximumQuantity`
	`public int? MaximumLogisticUnits`
	`public bool AllowCountDuringCycleCount`
	`public bool AllowCountDuringOtherOperation`
	`public bool NeedsToBeCounted`
	`public int? LockedBy`
	`public int CountAfterNumberOfDays`
	`public int CountAfterNumberOfOperations`

Methods	-

4.1.8. PmxOseBinProvider

Initialization	`new PmxOseBinProvider (conn);`
Methods	`PmxOseBin GetBO(string key)`

4.1.9. LogisticUnitGoodsReceipt

```
public class LogisticUnitGoodsReceipt
```

```
{
AllowReceptionWithoutLUID =
logisticUnitGoodsReceipt.AllowReceptionWithoutLUID,
IsMoveToLockedStockLocation =
logisticUnitGoodsReceipt.IsMoveToLockedStockLocation,
ItemsOnLogisticUnit =
Mapper.LocalCollectionOfLogisticUnitItemGoodReceipts(logisticUnitGoodsReceip
t.ItemsOnLogisticUnit),
LogisticUnitId = logisticUnitGoodsReceipt.LogisticUnitId,
MasterLUID = logisticUnitGoodsReceipt.MasterLUID,
Reason = logisticUnitGoodsReceipt.Reason,
SpecificLocationCode = logisticUnitGoodsReceipt.SpecificLocationCode,
SSCC = logisticUnitGoodsReceipt.SSCC,
SupplierPalletNumber = logisticUnitGoodsReceipt.SupplierPalletNumber,
UnitPrice = logisticUnitGoodsReceipt.UnitPrice,
}
```

4.1.10. LogisticUnitItemGoodReceipt

```
public class LogisticUnitItemGoodReceipt
{
AutoSelectPO = logisticUnitItemGoodReceipt.AutoSelectPO,
BatchNumber1 = logisticUnitItemGoodReceipt.BatchNumber1,
BatchNumber2 = logisticUnitItemGoodReceipt.BatchNumber2,
BeasItemVersion = logisticUnitItemGoodReceipt.BeasItemVersion,
BestBeforeDate = logisticUnitItemGoodReceipt.BestBeforeDate,
FullListOfPackagingTypes =
Mapper.LocalCollectionOfPackagingTypeInfos(logisticUnitItemGoodReceipt.FullL
istOfPackagingTypes),
IsLogisticCarrier = logisticUnitItemGoodReceipt.IsLogisticCarrier,
IsLogisticCarrierForLogisticUnit =
logisticUnitItemGoodReceipt.IsLogisticCarrierForLogisticUnit,
ItemCode = logisticUnitItemGoodReceipt.ItemCode,
ListOfBatchAttributes =
Mapper.LocalCollectionOfItemTransactionalBatchAttributeInfos(logisticUnitIte
mGoodReceipt.ListOfBatchAttributes),
ListOfPackagingTypes =
Mapper.LocalCollectionOfItemTransactionalPackagingTypeInfos(logisticUnitItem
GoodReceipt.ListOfPackagingTypes),
ListOfReasonsByZoneType =
Mapper.LocalDictionaryOfReasonInfos(logisticUnitItemGoodReceipt.ListOfReason
sByZoneType),
ListOfSerialNumbers = logisticUnitItemGoodReceipt.ListOfSerialNumbers,
OverrideLocationCode = logisticUnitItemGoodReceipt.OverrideLocationCode,
PurchaseDocRef =
Mapper.LocalDocumentRef(logisticUnitItemGoodReceipt.PurchaseDocRef),
PurchaseDocumentLineNum =
logisticUnitItemGoodReceipt.PurchaseDocumentLineNum,
QuantitiesForUom2 = logisticUnitItemGoodReceipt.QuantitiesForUom2,
Quantity = logisticUnitItemGoodReceipt.Quantity,
```

```
QuantityPerUom = logisticUnitItemGoodReceipt.QuantityPerUom,
QuantityUom2 = logisticUnitItemGoodReceipt.QuantityUom2,
ReasonInfo = logisticUnitItemGoodReceipt.ReasonInfo,
UnitPrice = logisticUnitItemGoodReceipt.UnitPrice,
Uom2 = logisticUnitItemGoodReceipt.Uom2,
Usage = logisticUnitItemGoodReceipt.Usage,
}
```

4.1.11. PackagingTypeInfo

```
public class PackagingTypeInfo
{
private string m_packagingTypeCode;
private string m_packagingTypeName; // Uom
private double m_quantity;
private double? m_initialQuantity;
private double m_quantityPerPack = 1;
private Collection<string> m_barcode;
private bool m_showOnScreen;
private int m_numberOfDecimals = 0;
private bool m_hideDuringEnteringQuantity;
}
```

4.2. Move

Customization Articles for WMS scripting:
How to change the Quality Status by scripting
How to remove the SSCC for certain items, and put them on the location directly

4.2.1. PmxMoveProvider types

Initialization	`NEW PmxMoveProvider(conn);`	-
Methods	`PmxMoveProvider GetNewBO()`	Generate a new Business Object into memory
	`string AddBO (PmxMove bo)`	Create the object in database
	`PmxMoveLine GetNewAddedLine (PmxMove document)`	Add a new line for the item

4.2.2. PmxMove types

No need to do any configuration on this object.

Example usage:

```
PmxMoveProvider moveProv = new PmxMoveProvider(conn);
PmxMove move
= moveProv.GetNewBO();
PmxMoveLine moveLine = moveProv.GetNewAddedLine(
move
);
moveLine.ItemCode = ...;
moveLine.Quantity = ...;
...;
moveProv.AddBO(
move
, true);
```

4.2.3. PmxMoveLine types

Fields
private string SourceStorageLocationCode; private int? SourceLogisticUnitIdentificationKey; private string DestinationStorageLocationCode; private int? DestinationLogisticUnitIdentificationKey; private string SourceQualityStatusCode; private string DestinationQualityStatusCode; private bool isLogisticCarrier; private int? ItemTransactionalInfoKey; private string ReasonCode; private string ReasonFreeText; private string ReasonLocationCode;

4.2.4. Enumeration types

```
public enum PmxMoveOrderType
    {
        Move = 1,
        PutAway = 2,
        Replenish = 4,
        WarehouseTransfer = 8,
        PutAwayProduction = 12
     }
```

```
public enum PmxMoveOrderStatus
    {
        NothingMoved = 1,
        Closed = 2,
        PartiallyMoved = 4
     }
```

```
public enum PmxMoveInOneTime
    {
        Invalid = 0,
        CannotBeMovedInOneTime = 1,
        CanBeInOneTime = 2,
        MustBeMovedInOneTime = 4
     }
```

```
public enum PmxMoveOrderStockLevel
    {
        Detail = 1,
        Item = 2,
        MasterLuid = 4
     }
```

4.2.5. PmxMoveOrderProvider

Initialization	`new PmxMoveOrderProvider(conn);`	-
Methods	`PmxMoveOrder GetNewBO()`	Generate a new Business Object into memory
	`string AddBO (PmxMoveOrder bo)`	Create the object in database
	`void UpdateBO (PmxMoveOrder bo, bool onlyUpdateHeader, bool baseDocumentIsClosing)`	Updates the Business object in database
	`void CloseDocument (PmxMoveOrder document)`	Closing Move Order document
	`PmxMoveOrderLine GetNewAddedLine (PmxMoveOrder document)`	Add a new line to the document

4.2.6. PmxMoveOrder

Fields	`private PmxMoveOrderStatus MoveOrderStatus;`
	`private DateTime DueDate;`
	`private int Priority;`
	`private PmxMoveOrderType MoveOrderType;`
	`private PmxMoveInOneTime MoveLogUnitIn1Time;`
	`private int? LockedBy;`
	`private string FromPmxWhsCode;`
	`private string ToPmxWhsCode;`
	`private string Remarks;`

Methods	-

4.2.7. PmxMoveOrderLine

Fields	private PmxMoveOrderStatus MoveOrderLineStatus; private string SourceStorageLocationCode; private int? SourceLogisticUnitIdentificationKey; private string DestinationStorageLocationCode; private int? DestinationLogisticUnitIdentificationKey; private string QualityStatusCode; private int? ItemTransactionalInfoKey; private PmxMoveOrderStockLevel StockLevel; private string WaBoxCode;
Methods	-

4.3. Move Order

Customization Articles for WMS scripting:
How to generate Move Order by scripting

4.4. Picklist Proposal

4.4.1. Enumeration types

```
public enum PmxPickListProposalStockStatus
    {
        None,
        Partially,
        All
    }
```

```
public enum PmxPickObjectType
    {
        Sales,
        WhsTransfer,
        Production,
        WhsTransferProd
    }
```

```
public enum PmxInventoryLockingLevel
    {
        ItemNoLocking = 0,
        ItemQuality = 1,
        ItemBatchNumberBestBeforeDate = 2,
        ItemLUID = 4,
        ItemDetail = 8
    }
```

4.4.2. PmxPickListProposalProvider

Initialization	`new PmxPickListProposalProvider (conn);`	-
Methods	`PmxPickListProposal GetNewBO()`	Generate a new Business Object into memory
	`PmxPickListProposal GetBO(int key)`	-
	`string AddBO (PmxPickListProposal bo)`	Create the object in database
	`void UpdateBO (PmxPickListProposal bo, bool onlyUpdateHeader, bool baseDocumentIsClosing)`	Updates the Business object in database
	`void CloseDocument (PmxPickListProposal document)`	Closing Move Order document
	`PmxMoveOrderLine GetNewAddedLine (PmxPickListProposal document)`	Add a new line to the document

4.4.3. PmxPickListProposal

Fields	`private string CardCode;`
	`private string CardName;`
	`private string ShipToCode;`
	`private string ShipToaddress;`
	`private string DestinationStorageLocation;`
	`private PmxPickListProposalStockStatus FullStockStatus;`
	`private PmxPickListProposalStockStatus NotExpiredStockStatus;`
	`private DateTime DueDate;`
	`private string PickPackRemarks;`
	`private string Remarks;`
	`private int? RouteLineDocEntry;`
	`private int? RouteLineLineNum;`
	`private bool IsCustomerCollect;`
	`private string PickListType;`
	`private BusinessObjectProperty<int?> ShippingID;`
	`private string MoveToWarehouse;`
	`private string MoveToLocationCode;`
	`private PmxPickObjectType PickObjType;`

Methods	-

4.4.4. PmxPickListProposalLine

Fields	`private int? ItemTransactionalInfoKey;`
	`private int? LogisticUnitIdentificationKey;`
	`private string QualityStatusCode;`
	`private PmxPickListProposalStockStatus FullStockStatus;`
	`private PmxPickListProposalStockStatus NotExpiredStockStatus;`
	`private double PickListQuantity;`
	`private PmxInventoryLockingLevel InvLockLevel;`
	`private bool ForceBatch;`
	`private bool IsSampleOrder;`
	`public double? QuantityForTempLock;`

Methods	-

4.5. Picklist

4.5.1. Enumeration types

```
 public enum PmxPickListStatus
    {
        NotReady = 1,
        Closed = 2,
        PartiallyReady = 4,
        Ready = 8,
        PartiallyPicked = 0x10,
        Picked = 0x20,
        PartiallyDelivered = 0x40,
        PartiallyPacked = 0x80,
        Packed = 0x100,
        ForcedClosed = 0x200,
        PartiallyShipped = 0x400,
        Shipped = 0x800
    }
```

```
public enum PmxPickObjectType
    {
        Sales,
        WhsTransfer,
        Production,
        WhsTransferProd
    }
```

```
public enum PmxInventoryLockingLevel
    {
        ItemNoLocking = 0,
        ItemQuality = 1,
        ItemBatchNumberBestBeforeDate = 2,
        ItemLUID = 4,
        ItemDetail = 8
    }
```

4.5.2. PmxPickListProvider

Initialization	`new PmxPickListProposalProvider (conn);`	-

Methods	`PmxPickListProposal GetNewBO()`	Generate a new Business Object into memory
	`PmxPickListProposal GetBO(int key)`	-
	`string AddBO (PmxPickListProposal bo)`	Create the object in database
	`void UpdateBO (PmxPickListProposal bo, bool onlyUpdateHeader, bool baseDocumentIsClosing)`	Updates the Business object in database
	`void CloseDocument (PmxPickListProposal document)`	Closing Move Order document
	`PmxMoveOrderLine GetNewAddedLine (PmxPickListProposal document)`	Add a new line to the document

4.5.3. PmxPickList

Fields	`private PmxPickListStatus PickListStatus;`
	`private string CardCode;`
	`private string ShipToAddress;`
	`private string ShipToCode;`
	`private string DestStorLocCode;`
	`private int? PickListProposalEntry;`
	`private BusinessObjectProperty<int> Priority;`
	`private DateTime DueDate;`
	`private int? LockBy;`
	`private string CardName;`
	`private int? RouteKey;`
	`private string ReasonCodeNotFullShipping;`
	`private string ReasonFreeTextNotFullShipping;`
	`private int? WaveKey;`
	`private string PickPackRemarks;`
	`private bool IsCustomerCollect;`
	`private bool IsPrinted;`
	`private string PickListType;`
	`private DateTime? LastPrintDateTime;`
	`private int? PackLockBy;`
	`private string MoveToWarehouse;`
	`private string MoveToLocationCode;`
	`private PmxPickObjectType PickObjType;`

Methods	-

4.5.4. PmxPickListLine

Fields	`private PmxPickListStatus PickListLineStatus;`
	`private int? ItemTransactionalInfoKey;`
	`private string StorageLocationCode;`
	`private int? LogisticUnitIdentificationKey;`
	`private string QualityStatusCode;`
	`private int Sequence;`
	`private double QuantityPicked;`
	`private double QuantityPacked;`
	`private double OriginalQuantity;`
	`private double? QuantityPickedUom2;`
	`private double? QuantityPackedUom2;`
	`private string ReasonCodeNotFullPicking;`
	`private PmxInventoryLockingLevel InvLockLevel;`
	`private bool ForceBatch;`
	`private bool IsSampleOrder;`

Methods	-

4.6. Print

Customization Articles for WMS scripting:
How to trigger a print event from a script

4.6.1. PmxReportProvider

Initialization	`new PmxReportProvider (conn);`	-
Methods	`PmxReport GetBO(int key)`	The code of the report from the OSE

4.6.2. PmxOsePrinterProvider

Initialization	`new PmxOsePrinterProvider (conn);`	-
Methods	`PmxOsePrinter GetNewBO()`	Not supported
	`PmxOsePrinter GetBO(string key)`	The code of the printer from the OSE

The method provides the closest printer to a location.
Page size of the printer must be configured in the call. DeviceID can be null.

```
PmxOsePrinter GetPrinterForLocation( string CurrentLocationCode, string DeviceID, string PageSizeCode)
```

4.6.3. ReportPrinterDevice

Initialization	`new ReportPrinterDevice (conn);`	-

Methods	`void PrintReport`
	`(`
	`PmxReport report,`
	`PmxOsePrinter printer,`
	`CultureInfo cultureInfo,`
	`int numberOfCopies,`
	`DataSet ds,`
	`Collection<ReportParameter> reportParameters)`
	`--- CR parameters from PMX layouts`
		Create a new object for the printing

4.6.4. Example

```
PmxReportProvider reportProvider = new PmxReportProvider(conn);
PmxReport report = reportProvider.GetBO(6); // report code FROM OSE

PmxOsePrinterProvider printerProvider = new PmxOsePrinterProvider(conn);
PmxOsePrinter printer = printerProvider.GetNewBO();
printer = printerProvider.GetBO("PRINTER"); // printer code from OSE

// create the report parameter structure
Collection<ReportParameter> reportParameters = new
Collection<ReportParameter>();
reportParameters.Add(new ReportParameter("@luid", Luids[i]));

ReportPrinterDevice device = new ReportPrinterDevice(conn);
device.PrintReport(report, printer, null, 1, null, reportParameters);
```

4.7 Cycle Count

4.7.1 PmxCycleCountProvider

- Initialization
  - new PmxCycleCountProvider(conn);
- Methods
  - PerformCountDuringCycleCount(Collection<CycleCountInventoryItemInfo> listOfItemsToUpdate, string locationCode, string deviceID)
    - Performs the count during cycle count.

```
<param name="listOfItemsToUpdate">The list of items to
update.</param> <param name="locationCode">The location
code.</param>
```

4.7.2 CycleCountInventoryItemInfo

- Initialization
  - new CycleCountInventoryItemInfo();

- Class to hold the data for cycle count inventory items

4.7.3 Example

```
ReasonInfo reasonInfo = new ReasonInfo();
Collection<CycleCountInventoryItemInfo> listOfItemsToCount = new
Collection<CycleCountInventoryItemInfo>();
PmxCycleCountProvider prov = new PmxCycleCountProvider(conn);
CycleCountInventoryItemInfo invItemInfo = new CycleCountInventoryItemInfo();
PmxItemAllConnectionsProvider provItem = new
PmxItemAllConnectionsProvider(conn);

PmxItemInfo itemInfo = provItem.GetCachedItemInfo("I001");
invItemInfo.ItemInfo = itemInfo;
invItemInfo.ItemCode = "I001";
invItemInfo.ItemDescription = "I001 - I001";
invItemInfo.CurrentStock = 10;
invItemInfo.Difference = -2; // change the stock to 8
invItemInfo.DifferenceUom2 = null;
invItemInfo.QualityStatusCode = "RELEASED";
invItemInfo.ReasonInfo = reasonInfo;
invItemInfo.BatchNumber = null;
invItemInfo.SecondBatchNumber = null;
invItemInfo.BestBeforeDate = null;
invItemInfo.SSCC = null;
invItemInfo.LocationCode = "003";
invItemInfo.Uom = null;
invItemInfo.Uom2 = null;

listOfItemsToCount.Add(invItemInfo);
prov.PerformCountDuringCycleCount(listOfItemsToCount, "003", null);
```

Customization Framework on Mobile Client

Overview

From product version 2023.06, users can start the scanner application of the **Mobile Client in customization mode** and you can customize all the workflows available. From product version 2024.06, users can now create more complex customized workflows in the customization mode by creating user queries and adjusting & filtering the options in the **Customization Manager.**

Users have the possibility to customize the buttons and the screens of the flow while different customization for user groups and users can be defined in a way that is optimal for the warehouse.

Videos on the customization mode are available:

- **Produmex WMS 2021.03 Demo: UI Framework**

- **Produmex WMS UI Framework**

Overview about the customization UIs:

Customization mode

The name of the parameter is /cust. When you start the Mobile Client in customization mode, a customization icon (a cog sign) is displayed on the top-right corner of the screen. If you click the icon, it becomes red and the customization mode is active.

To customize the flow, proceed as follows:

1. Click the customization icon.
2. If you want to customize the screen, you can click anywhere on the screen. If you want to customize a button, click on the given button to be customized.
3. Customize your screen on the displayed Customization form (see customization options below) and click Save.

Note: In order to see the changes the user needs to restart the Mobile Client with disabled/switched off customization mode.

Customization options

1. Customization options on the Customization Tab

User Group: If you select a specific user group, the customization applies to the users in the given user group.

User: By default, no user is selected. Instead of user groups, you can define a specific user to whom the customization applies. Enable the User option and use the drop-down menu to select a user.

Default Button: The Default Button section is active if you customize a button.

- If you select the Default Button option, the user does not need to tap the button when working with the flow because the system automatically proceeds with the button. Only one button can be set as the default button on a screen.
- If you use the Default Button option, you can also set a screen timeout in seconds. In this case the system displays the button for the user for the defined interval. Within this interval the user can tap another button or the flow proceeds with the default button.

Visible: By default, the visible option is enabled.

- If you disable the option, the screen is not displayed for the user during the flow.
- If you disable the option, the button is not displayed for the user even if it is set as a default button.

Note: Hiding buttons overrides customizations. 'User' settings override 'User Group' settings and specific 'User Group' settings override 'All Users' settings.

Example: The customization applies to the Inventory user group. The Order button is set as the default button on the Select a Filter screen. The screen is displayed for the user for two seconds and if the user doesn't tap another button, the system proceeds with the Order button.

Example: The GS1/EAN Barcode button is not visible for any group on the Select a Filter screen.

2. Customization options on the Events Tab


Load Event: In this field you can customize an existing event, write the preferable name in the field than push the save button. What we do here is selecting any of the events that are already in use and modifying it's action.

Load Event Name: In this field you can modify an existing event by giving a unique name for your new customized event, after saving the unique name open the Query Manager in B1 where you can add your query to the name. For example create a *"sales_return"* name in the Load Event field and open it up in the Query Manager for furthermore customization.

Load Event List: Under the Load Event field there are a list of the previous events and actions that you have already been through.

Manage: Pushing the manage button will show the customization manager screen.

Customization Manager

The **Customization Manager** screen is a helpful UI to manage your customizations in a visual way. On this screen you can find all of your user queries in a simple table. In the **Customization Manager** you can not change the database informations, you only have the option to filter/activate/inactivate/delete your added queries.

1. All Components

In the **All Components** part you can find a tree structure, the purpose of this navigation tree is to easily manage the rows where you added a new queries. The “golden arrow” shows the selected row in the tree structure. If you select the **All components** row than in the table on the right side will show every grid, every lines that are found below it in the system.

The main components in the tree structure:

- Mainflows
- Subflows

2. Filter

The **Filter** section contain all the filter options to search for a specific query.

Screen: By default, no screen is selected. In the screen from dropdown menu you can select the preferred screen.

User Group: If you select a specific user group, the customization applies to the users in the given user group.

Options:

- All Users
- Finance
- Inventory
- Purchase
- Sales

User: By default, no user is selected. Instead of user groups, you can define a specific user to whom the customization applies. Enable the User option and use the drop-down menu to select a user.

Show: This options is a dropdown menu where you can choose between several options to search a specific group of queries e.g. if you would like to check on the all of your inactive queries.

Options:

- Everything
- All Active
- All Inactive
- Active Visible

- Active Invisible

Customization: By clicking the options inside of the Customization aggregation you can easily set a quick filter and search for a group of queries.

Options:

- Screen
- Controls
- User Quiers

Full Workflow: Clicking on the Full Workflow will extend the Workflow column with extra information about the path of the flows.

Reset Filters: With this button you can clear the filters that you previously set.

Customization Examples

1. Limitation:
On the **Events tab** you can see your previous steps/actions listed under the **Load Event** field. In that list you can clearly follow all your steps since you opened the mobile client in customization mode. Be aware you will not find those kind of actions listed when you selected a value from a list by a manual click, for example you chose a product by clicking it's name from the list instead of scanning the item barcode.

The selected values will only appear in the action list when you manually entered the value (customercode, location etc.) into the field or scanned that value.

2. Limitation:
If a user query are no longer used there is a specific procedure to remove the query from the system. First you have to delete the query from the **Customization Manager**, after that action open the **Query Manager** in SBO and remove the unused query from that table.

Example 1. - Default customer for sales return

Insert the <*customercode*> to the input field, and after you added the query to the **Query Manager**, the query automatically will select the predefined customer then waits 1 second and clicks the forward button.

Load Event Name: sales_return

MSSQL:

```
SELECT '<customercode>' as "txtCustomerCode", 'btnForward' as
"DefaultButton", 1 as "DefaultButtonClickTimeout"
```

HANA:

```
SELECT '<customercode>' as "txtCustomerCode", 'btnForward' as
"DefaultButton", 1 as "DefaultButtonClickTimeout" FROM DUMMY
```

Implementation: Add the query in the Query Manager in SBO.

Example 2. - Validations of the default value

Validation of the input quantity value on the reception flow, in this example we are showing as default quantity the minor between still to receive and default quantity logistic unit.

Load Event Name: default_quantity

Implementation: Add the query in the Query Manager in SBO.

MSSQL:

```
SELECT
CASE
  WHEN CAST(LEFT('$[lblQuantity]', CHARINDEX(' ', '$[lblQuantity]')-1) AS
INT) < U_PMX_DQLU THEN LEFT('$[lblQuantity]', CHARINDEX(' ',
'$[lblQuantity]')-1)
  ELSE U_PMX_DQLU
END AS "edtCounter0"
FROM "OITM" WHERE U_PMX_CUDE = '$[lblItem]'
```

HANA:

```
SELECT
CASE
WHEN CAST(LEFT('$[lblQuantity]', LOCATE( '$[lblQuantity]',' ')-1) AS
INTEGER) < "U_PMX_DQLU" THEN LEFT('$[lblQuantity]', LOCATE(
'$[lblQuantity]',' ')-1)
ELSE "U_PMX_DQLU"
END AS "edtCounter0"
FROM "OITM" WHERE "U_PMX_CUDE" = '$[lblItem]'
```

Example 3. - Finding a Pick List connected to a default customer

In this example if the query finds a Pick List that is connected to the <*customercode*> then the query will select the first Pick List then clicking on the forward button, if the query will not find a Pick List then nothing will happen.

Load Event Name: finding_picklist_connected_default_customer

Implementation: Add the query in the Query Manager in SBO.

MSSQL & HANA:

```
SELECT TOP 1 "DocEntry" AS "txtPickList",
CASE WHEN "PMX_PLHE"."DocEntry" IS NOT NULL THEN 'btnForward'
ELSE ''
END AS "DefaultButton",
CASE WHEN "PMX_PLHE"."DocEntry" IS NOT NULL THEN '1'
ELSE ''
END AS "DefaultButtonClickTimeout"
FROM "PMX_PLHE" WHERE "CardCode" = '<customercode>'
ORDER BY "DocEntry" ASC
```

Example 4. - Sorting and/or Truncating in "Select from a List" Screen Types

On **“Select from a List”** screen types, users can apply custom logic to truncate and/or sort the displayed data.

In this example, we use a custom query to filter specific items during the purchase order process. Only those specific items will be displayed on the **“Select a Product”** screen in the customized purchase order.

Next Event Name: PurchaseProductListFilter

Implementation: Add the query in the Query Manager in SBO.

MSSQL & HANA:

```
SELECT
  'ProductCode in (' +  STRING_AGG(CONCAT('''', "ItemCode", ''''), ', ') +
')' as "RowFilter",
  'ProductCode desc' as "Sort"
FROM
  OITM
WHERE
  "ItemCode" like 'I%'
```

Result:
In SBO, open the Query Manager to check the results after executing the query. The values in the RowFilter column will be filtered in the dataset and sorted by the Sort column during the customized process.

We can define filters and sortings with the ListColumns values.

ListColumns from the Dataset that can be used:

```
-- $[ListColumns]=ProductCode, ProductDescription, GTIN, U_PMX_HBBD,
U_PMX_PILR, ManBtchNum, U_PMX_LOUN, NumInBuy, BuyUnitMsr, InvntryUom,
CodeBars, ItemName
```

Example 5. - Validating Inserted Values on "Forward" Click

Use a query within the Customization Framework solution to validate inserted values as you move forward in the flow.

In this example, our warehouse has a predefined location fixed for a specific item. During an AdHoc Move, this item can only be moved to its fixed location and cannot be relocated elsewhere. To ensure this, we must validate both the item and the location, that validation can be customized with a query. If an attempt is made to move the item to a non-fixed location, the system will warn us that the **“Location is not fixed for this item!”**.

In the OSE, set the location to fixed for the specific item.

Next Event Name: AHMoveValidateEnteredDestinationLocation

Implementation: Add the query in the Query Manager in SBO

MSSQL:

```
SELECT
    CASE
        WHEN '$[Prev1.lblItem]' = ''
        THEN ''
        ELSE
            CASE
                WHEN
                    (
```

```
                        select COUNT(*) from PMX_SLIT left join
PMX_OITM_MANAGED_BY_PMX on PMX_SLIT."ItemCode" =
PMX_OITM_MANAGED_BY_PMX."ItemCode"
where PMX_SLIT."ItemCode" + ' - ' + PMX_OITM_MANAGED_BY_PMX."ItemCode" =
'$[Prev1.lblItem]' and "ParentCode" = '$[txtStringValue]'
                    ) = 0
                THEN 'Location is not fixed for this item!'
                ELSE ''
        END
    END AS "ValidationError"
```

HANA:

```
SELECT
    CASE
        WHEN '$[Prev1.lblItem]' = ''
        THEN ''
        ELSE
            CASE
                WHEN
                    (
                        select COUNT(*) from PMX_SLIT left join
PMX_OITM_MANAGED_BY_PMX on PMX_SLIT."ItemCode" =
PMX_OITM_MANAGED_BY_PMX."ItemCode"
where PMX_SLIT."ItemCode" + ' - ' + PMX_OITM_MANAGED_BY_PMX."ItemCode" =
'$[Prev1.lblItem]' and "ParentCode" = '$[txtStringValue]'
                    ) = 0
                THEN 'Location is not fixed for this item!'
                ELSE ''
        END
    END AS "ValidationError"
FROM DUMMY
```

Show item UDFs during Mobile flows

Overview

Users can manage item **UDFs in the User-Defined Fields – Management** screen. When reaching the **Select Product** screen in a supported flow with predefined UDFs, an info bar with a button appears at the bottom. Tapping the button opens a scrollable screen showing all UDF values for the selected item.

The customized item UDFs work in the following flows:

- Reception
- Put Away
- Cycle Count
- Ad hoc Moves

- Standard Picking
- Ad hoc Picking and Undo Picking

When the user opens the **Item Master Data** window, they can navigate to **Produmex > Inventory**, where the UDF field table is located. In the first column, where several item UDFs can be added, and the checkboxes in the following columns allow the user to define which flows the UDF applies to.

When UDFs are enabled for an item and flow, their values are automatically sent to WMS. Users can create custom UDFs in **Tools > Customization Tools > User-Defined Field – Management.** Navigate to **Master Data > Items > Items**, then locate the relevant tables and **add the required custom one.**

> **Note:** Keep in mind the UDFs are standard SAP features!

> **Real life scenario**
>
> These UDFs help users track important details—such as allergens, hazard classes, temperature requirements, and pallet rules—throughout warehouse operations. With a double-click on the info icon, users can open the UDF field separately to check if an item has multiple UDFs. To close the window and return to the process flow, users just need to double-click the info icon again. If only one UDF is defined for an item, the info icon will not appear on the Mobile Client. However, the additional window can still be accessed by double-clicking.
>  

Example use cases

- **Mobile Client UI:** The customer wants to filter the result of a list of
  - Locations
- **Mobile Client UI:** The customer wants to add extra information to a column that appears on the *UIsubflow* ☹ not supported
- **Stock manipulation**
  - removing SSCC
  - changing Quality Status
  - move to a specific location
- **Changing Picklist**

    - Status
    - Set picked qty from WAS
- **Generate WMS document**
    - Move order
- **Printing a report**

Example Scripts

- **Auto shipping in picking process with base document Inventory Transfer Request (AfterPickListPackedHookScript):**
  How to configure auto shipping in picking process with base document Inventory Transfer Request
- **Custom bin location list in Put Away (PutAwayDestinationLocationHookScript):**
  How to make custom bin location list in Put Away
- **Custom bin location list in AdHoc move (SelectLocationForAdHocMovesHookScript):**
  How to make custom bin location list in AdHoc move
- **Changing the status of the Picklist to Ready:**
  How to change the status of the Picklist to Ready by scripting
- **Removing the SSCC for certain items:**
  How to remove the SSCC for certain items, and put them on the location directly

Customization possibilities

This section presents a table outlining the available customization options. Each line may include a default value, which represents a built-in system behavior. This default functionality can be overridden with your own customization if needed. In cases there is no default value was provided, you have the flexibility to define your own default behavior, which the system will then use.

For example:

Go to **OSE/Extension Parameters** → Select the desired customizable property.
Open documents screen controller and the parameter label is **View name (Container)**.

Property Name	Parameter Label	Default Value
Freight charges controller	View name	PMX_FREIGHT_CHARGES
Inventory Controller	View name - Detail	PMX_INVENTORY_REPORT_DETAIL

Move controller	The Custom View Name	
Move controller	The Order By Clause of the View	“BaseType” DESC, “BaseEntry”, “BaseLine”, “LockingKey”
Open documents screen controller	View name (Container)	PMX_OPEN_DOCUMENT_REPORT_CONTAINER
Open documents screen controller	View name (Move order)	PMX_OPEN_DOCUMENT_REPORT_MOVE_ORDER
Open documents screen controller	View name (Pick list)	PMX_OPEN_DOCUMENT_REPORT_PICKLIST
Open documents screen controller	View name (Pmx sales shipping)	PMX_OPEN_DOCUMENT_REPORT_PMX_SALES_SHIPPING
Open documents screen controller	View name (Proposal)	PMX_OPEN_DOCUMENT_REPORT_PICKLIST_PROPOSAL
Open documents screen controller	View name (Route)	PMX_OPEN_DOCUMENT_REPORT_ROUTE
Open documents screen controller	View name (Weigh order)	PMX_OPEN_DOCUMENT_REPORT_WEIGH_ORDER
Open Sales Orders Controller	Order by	“DocDueDate” ASC, “CardCode” ASC
Open Sales Orders Controller	View name	PMX_OPEN_SALES_ORDERS_WITH_STOCK_STATUS
Packing Controller	Finished LUID screen: View name	
Pick list robot	Group into waves - View name	PMX_PICK_LIST_ROBOT_GROUP_INTO_WAVES
Pick list robot	Order by	“DocEntry”
Pick list robot	Order by	“DocDueDate”, “DocEntry”
Pick list robot	View name	PMX_PICK_LIST_ROBOT_CREATE_PICK_LISTS
Pick list robot	View name	PMX_PICK_LIST_ROBOT_CREATE_PROPOSALS
Picklist controller	Custom wave description fields for scanner	
Picklist controller	Function name to get the location sequence	PMX_FN_GetFirstSequenceForLocationsForItemForAdHocPicking

Picklist controller	Function name to get the location sequence	PMX_FN_GetAllLocationsForItemForAdHocPicking
Picklist controller	JOIN-sql for custom wave description for scanner	
Picklist controller	Pick items order by - Stored procedure name	PMX_SP_PickItemsOrderByLineNum
Picklist controller	Pick list - Order by	PMX_PLHE.“Priority”,“PMX_PLHE”.“DueDate”,“PMX_PLHE”.“DocEntry”
Picklist controller	Select wave - View name	
Picklist controller	View name to get the products	
Picklist controller	Wave - Order by	PMX_PREPARE_CARTS_WAVE.“Priority”,“PMX_PREPARE_CARTS_WAVE”.“WaveKey”
Picklist controller	Wave - View name	PMX_PREPARE_CARTS_WAVE
Picklist Proposal generator	Stock order by custom code	
Picklist proposal manager screen controller	View name (Inv. transfer info)	PMX_PICKLIST_PROPOSAL_MANAGER_TRANSFER
Picklist proposal manager screen controller	View name (Production info)	PMX_PICKLIST_PROPOSAL_MANAGER_PRODUCTION
Picklist proposal manager screen controller	View name (Sales order info)	
Production controller	Integration Produmex Manufacturing: SP name for getting items to consume	PMX_SP_GetProductionComponentsToConsume
Production controller	Integration Produmex Manufacturing: SP name for removing locks	PMX_SP_RemoveLocksForProductionOrderLine
Purchase delivery generator	Purchase line remarks view name	

Purchase delivery generator	Remarks view name	
Receive from WHS controller	Line remarks view name	
Receive from WHS controller	Remarks view name	
Replenishment generator	Orders view name	PMX_REPLENISHMENT_ORDERS_LIST
Replenishment generator	Orders view order by	“Priority” ASC, “DueDate” ASC, “DocEntry” ASC, “LineID” ASC
Route Controller	Order by - Open pick list (proposals)	“DocType”, “DocEntry”
Route Controller	Order by - Open routes	“DocEntry”
Route Controller	Order by - Route detail header	
Route Controller	View name - Open pick list (proposals)	PMX_ROUTE_PLANNING_OPEN_PICK_LIST_PROPOSALS
Route Controller	View name - Open routes	PMX_ROUTE_PLANNING_OPEN_ROUTES
Route Controller	View name - Route detail header	
Route Controller	View name - Route details	PMX_ROUTE_PLANNING_DETAILS
Sales delivery note generator	Grouping filter columns	
Sales delivery note generator	Join clause for grouping filter	
Sales delivery note generator	Order by for grouping filter	
Stock allocation screen controller	View name (Customer info)	PMX_STOCK_ALLOCATION_SCREEN_CUSTOMER
Stock allocation screen controller	View name (Sales order info)	PMX_STOCK_ALLOCATION_SCREEN_SALES_DOCUMENT

Scripting Support

Scripting is not supported by our standard support tickets!
Support on scripting requires the purchase of **Premium service**.

> **Premium Service** - If you wish to receive assistance on scripting cases, you inquiry will be then classified as Premium Service and will require the involvement of our delivery team in at least one remote session.

Modifying workflows can cause serious disruption of processes and even data corruption. Extreme Caution is advised! **It is recommended that only experienced WMS Consultants attempt to modify these workflows.**

Boyum IT cannot be held responsible for issues resulting from externally modified workflows.