Customization Example: Micro-Vertical Solution for Serial Numbered Appliance Trading Organizations

This document describes how the Free Goods Receipt PO process in Produmex Scan was customized to fit a typical client and form a part of the micro-vertical solution for serial numbered appliance trader and similar companies. The customizations for each screen are described in detail.

Free Goods Receipt PO

In Produmex Scan, the Free Goods Receipt PO consist of 3 screens: a main screen used to start creating and preparing a new Goods Receipt document, a screen showing item lines and quantities and a detail screen on which the receiving bin location and serial numbers can be specified.

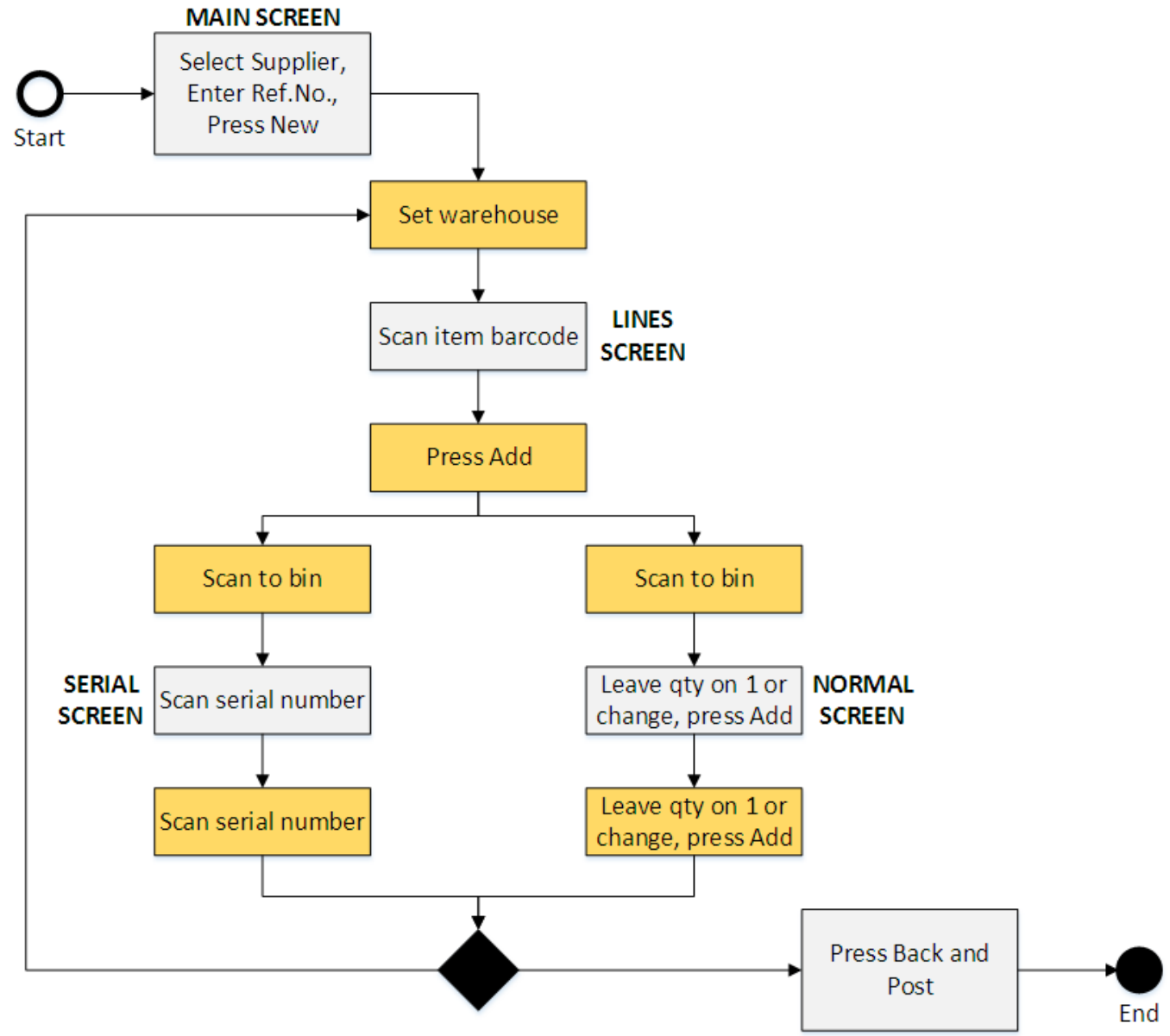
In standard Produmex Scan, these screens are optimized for receiving items sequentially, that is, start receiving ITEM1, enter to location, quantities or serial numbers, then start receiving ITEM2, enter location, quantities or serial numbers. This is a good solution in many cases when there are a couple of different items being received and they are grouped.

However, in this micro vertical solution the typical customer wants to receive many different items, a kind of assortment, which are not grouped by item code, maybe only by similar items. These items are also mostly serial numbered, which means the standard Produmex Scan way of receiving requires a lot of extra actions on the mobile user interface. The goal here was to allow the warehouse worker to use the bar code scanner button 99% of the time when doing the receiving. In practice this means: he will take the next piece of item (boxed), scan the item code bar code, then scan the serial number bar code, and then move on to the next piece of item (boxed).

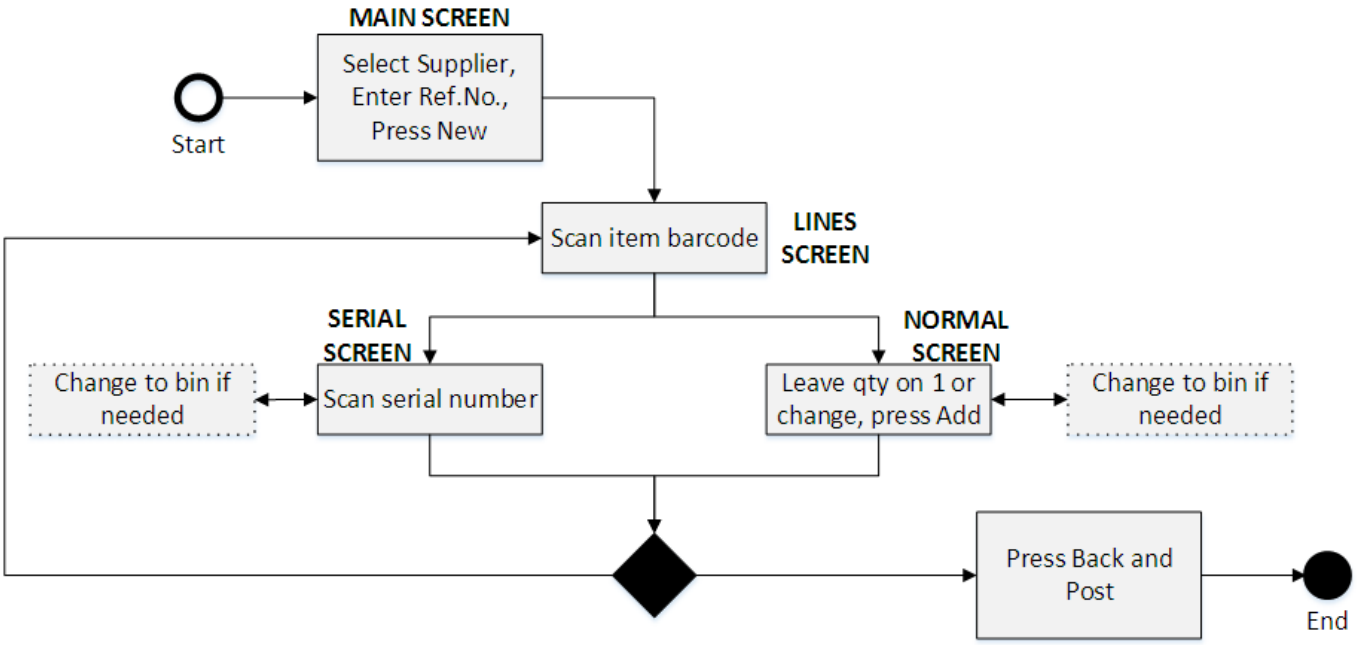
During receipt, the items are moved to a temporary bin location, which is an empty pallet labelled with a bin code. When a pallet is full the warehouse worker can change the bin location to the next temporary bin location, the next pallet's location. In a later step, these pallets are moved to shelves with a forklift and the Produmex Scan Mass Transfer function, which moves all the stocks on one bin to another.

Comparison of the Original and New Receiving

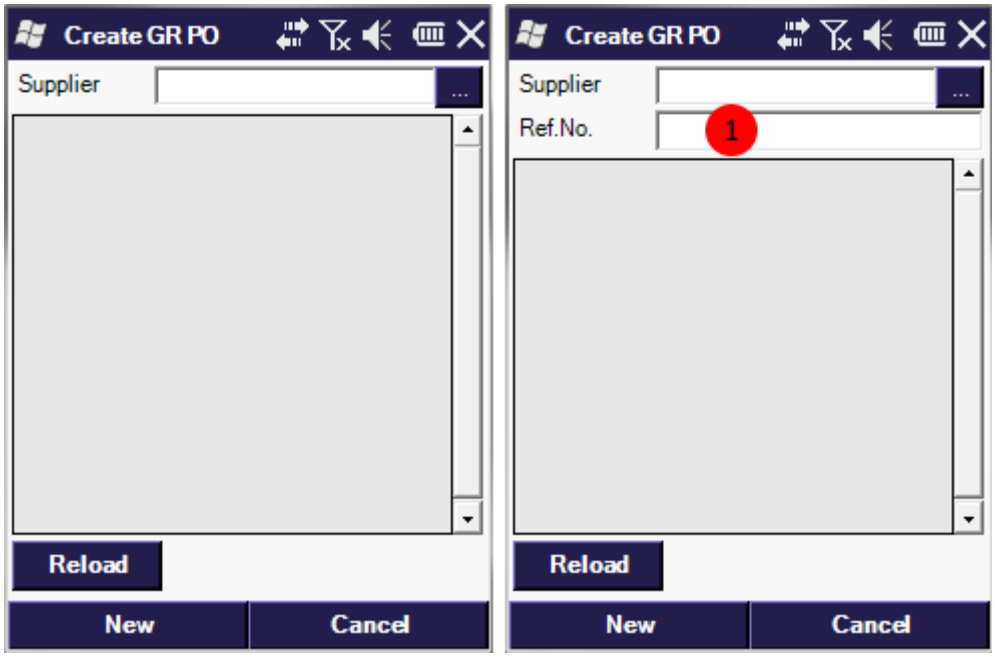
Below you can see the original process for receiving.
The extra steps that this micro-vertical solution eliminates have been colored.
If we look at just the main action (from Set warehouse to just before Finish), receiving 10 serial items takes 10 x 5 = 50 actions this way. It needs 3 bar code scans (item, to bin, serial number) and 2 button presses (Add, Done) per item piece.



The following picture shows you the new process with these steps eliminated.
The core steps of receiving 10 items means 10 x 2 = 20 steps, plus changing the destination palette (to bin) occasionally. Even better, these actions are all bar code scans (item code, serial number), only 2 per item.
This shows how we optimized the process. Please also note, that there are other types of companies who might receive only a few different serial numbered items at once, for them, the original Produmex Scan process of scanning one item code then scanning a lot of serial numbers is much more effective.



Main Screen

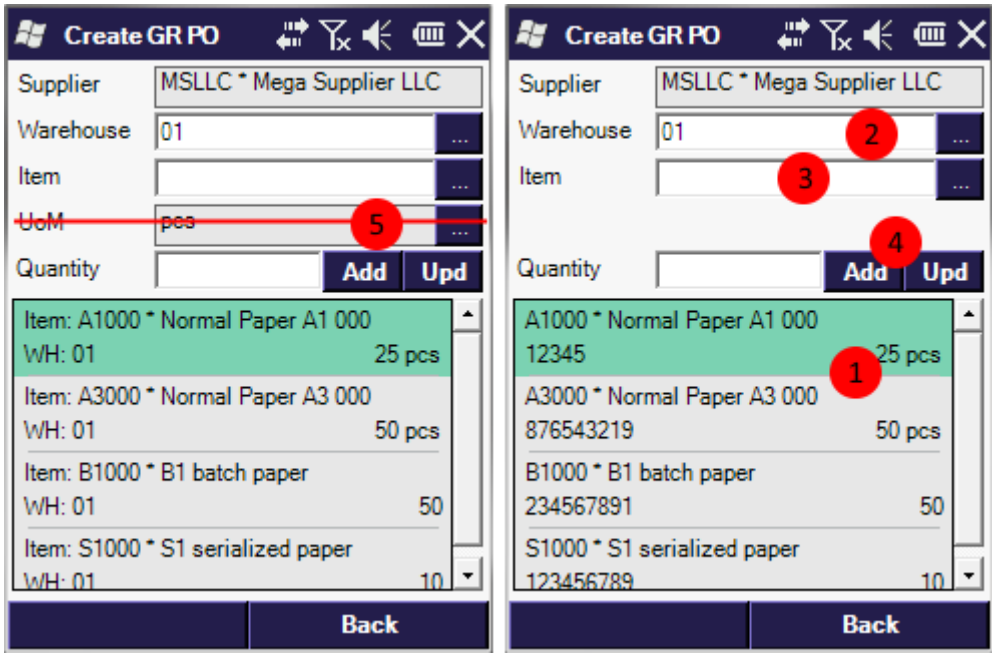


Changes:

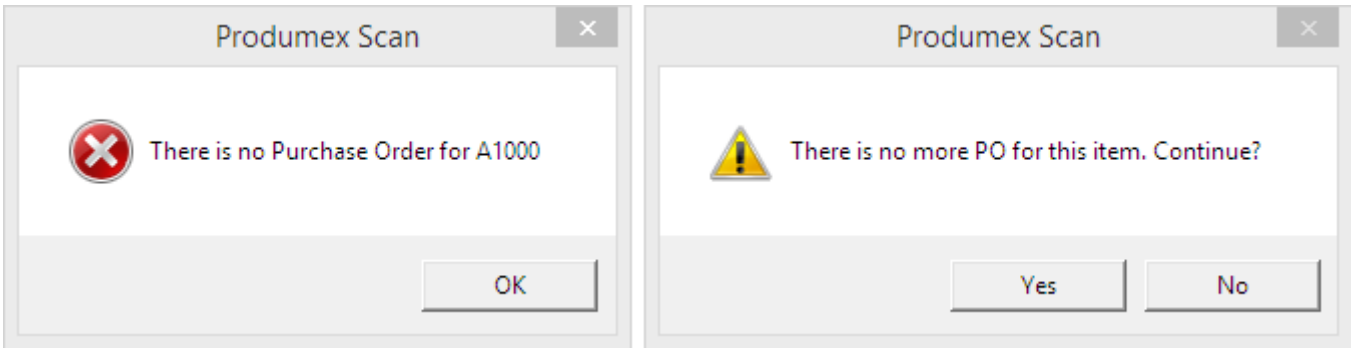
1. Added Ref. No. field which will be copied to the new Delivery document. The user queries and custom fields for the above custom logic:

Field Name	Label	Screen
BO_NumAtCard	Ref.No.	CreateGoodsReceiptPOScreen

Item Lines Screen



Changes:
1. List items are changed to show Item Code + Item Name without prefix and to show bar code (EAN) in the second line instead of warehouse.
2. The warehouse is filled by the last selected warehouse for the employee by default and the value is saved if it has changed.
3. After the item has been scanned, it is checked if that item has any open Purchase Orders from the same supplier. If the item was valid and there are no problems, after scanning item code or bar code the screen automatically opens the next (bin locations) screen. No need to press the Add button explicitly.
4. The Add button checks if there is still open quantity for the given item from the same supplier. It gives a warning if there are no more open quantities.
5. UoM field is hidden because it's not used.



The user queries and custom fields for the above custom logic:

1.
User query: *BXMobileWH9_CreateGoodsReceiptPOLinesScreen_DataRepeater_InternalDataLoad*

```
-- Fill UIItem with item code and name
-- and UIWarehouse with OITM.CodeBars in list
SELECT SUBSTRING (S.splitdata,7,50) AS [DataRepeater.UIItem], C.CodeBars AS
[DataRepeater.UIWarehouse]
FROM dbo.SplitStringForDataRepeater($[DataRepeater.UIItem]) S, OITM C
```

```
WHERE C.ItemCode=SUBSTRING(S.splitdata,7,PATINDEX('%*%',S.splitdata)-8)
```

2.
Create a new user table 'SCANEMPWH' with the following fields:

- To Bin Location (ToWH)
- Employee ID (empID)

User query: *BXMobileWH9_CreateGoodsReceiptPOLinesScreen_TextWarehouse_validate_after*

```
-- Save Warehouse value to remember it for next time
IF (SELECT ISNULL([@SCANEMPWH].U_ToWH,'')
FROM [@SCANEMPWH] WHERE U_empID = $[Employee.EmployeeID]) <>
$[TextWarehouse]
UPDATE [@SCANEMPWH] SET [U_ToWH]=$[TextWarehouse] WHERE U_empID =
$[Employee.EmployeeID]
```

3.
User query: *BXMobileWH9_CreateGoodsReceiptPOLinesScreen_TextItem_validate*

```
-- Check Item and see if it has open quantity in Purchase orders from this
supplier
-- if not, show error message
DECLARE @itemcode nvarchar (MAX)
SET @itemcode = $[TextItem]

IF NOT EXISTS SELECT (T0.ItemCode FROM OITM T0 WHERE T0.ItemCode=@itemcode)
SELECT @itemcode=T1.ItemCode FROM OBCD T1 WHERE T1.BCdCode=@itemcode

IF @itemcode<>'' AND (SELECT ISNULL (SUM(T1.[OpenQTX]),0) FROM OPOR T0
INNER JOIN POR1 T1 ON T0.[DocEntry] = T1.[DocEntry]
WHERE T1.[ItemCode] = @itemcode AND T0.[CardCode] =
LEFT($[TextSupplier],PATINDEX('%*%',$[TextSupplier])-1))=0
SELECT 'There is no Purchase Order for '+@itemcode AS 'Message$', 'E' AS
'MessageType$'
ELSE SELECT '' AS dummy
```

User query: *BXMobileWH9_CreateGoodsReceiptPOLinesScreen_TextItem_validate_after*

```
-- If item is filled then set quantity to empty and press Add automatically
IF $[TextItem]<>'' SELECT '' AS TextQuantity, 'ButtonAdd' AS Click$
```

4.
User query: *BXMobileWH9_CreateGoodsReceiptPOLinesScreen_ButtonAdd_click*

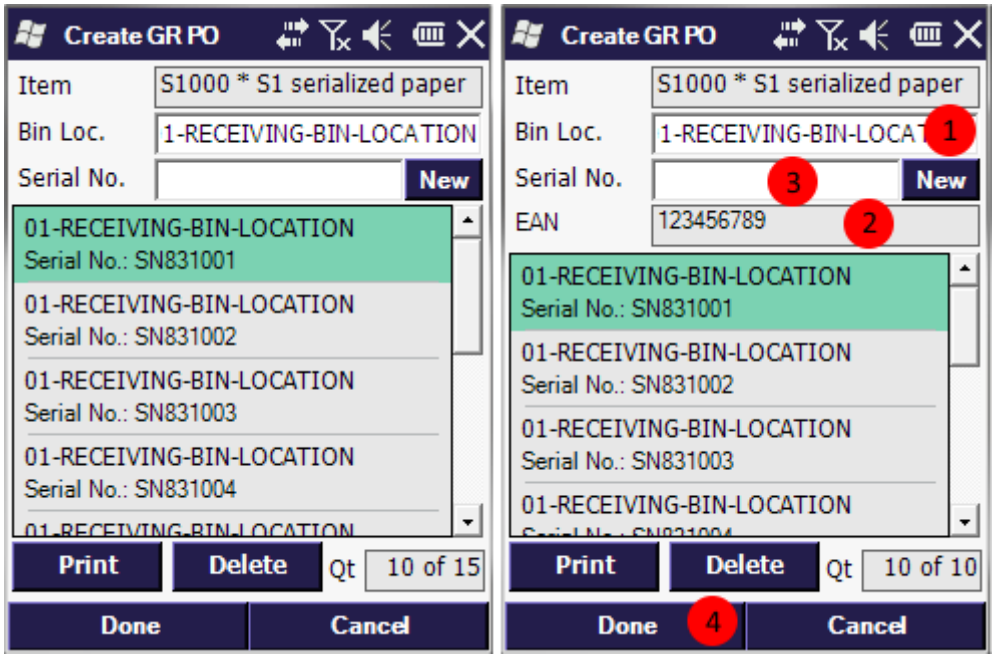
```
-- When add button is pressed check if there is open quantity
-- for that item in Purchase Orders
DECLARE @itemcode nvarchar(MAX)
SET @itemcode = $[TextItem]
IF NOT EXISTS (SELECT T0.ItemCode FROM OITM T0 WHERE T0.ItemCode=@itemcode)
SELECT @itemcode=T1.ItemCode FROM OBVD T1 WHERE T1.BCdCode=@itemcode
```

```
IF (SELECT SUM(CAST(LEFT(S.splitdata2,PATINDEX('% %',Splitdata2)-1) AS
DECIMAL (1,5)))
FROM
dbo.SplitStringForDataRepeater2($[DataRepeater.UIItem],$[DataRepeater.UIQuan
tity]) S
WHERE
@itemcode=SUBSTRING(S.splitdata,1,PATINDEX('%*%',S.splitdatda)-1))>=(SELECT
ISNULL (SUM(T1.[OpenQty]),0) FROM OPOR
T0.[CardCode]=LEFT($[TextSupplier],PATINDEX('%*%',$[TextSupplier])-1))
SELECT 'There is no more PO for this item. Continue?' AS 'Message$', 'YM' AS
'MessageType$'
```

5.
To hide the UoM item on the screen, add a record to the Customization Fields(BXPCUSTFD) user table:

FieldName	Visible	Screen
TextUoM	NO	CreateGoodsReceiptPOLinesScreen

Serial Numbers Screen



Changes:
1. Save the last bin for next time and load it with last value
2. New field: EAN (Item's bar code) added and loaded with data
3., 4. After serial number has been scanned, automatically press the Done button to go to previous screen

The user queries and custom fields needed for these are:

1.
Create a new user table 'SCANEMPBL' with the following fields:

- To Bin Location (ToBL)
- Employee ID (empID)

User query:
BXMobileWH9_CreateGoodsReceiptPOQuantitiesSerialScreen_TextBinLocation_validate_after

```
-- Save Bin Location from screen to employee so it can be loaded next time
IF (SELECT ISNULL([@SCANEMPBL].U_ToBL,'')
FROM [@SCANEMPBL] WHERE U_empID = $[Employee.EmployeeID]) <>
$[TextBinLocation]
UPDATE [@SCANEMPBL] SET [U_ToBL]=$[TextBinLocation] WHERE U_empID =
$[Employee.EmployeeID]
```

1., 2.
User query: *BXMobileWH9_CreateGoodsReceiptPOQuantitiesSerialScreen_Load*

```
-- On screen load fill bin location from saved value
-- and fill EAN field from OITM.CodeBars
SELECT
(SELECT [@SCANEMPBL].U_ToBL FROM [@SCANEMPBL] WHERE U_empID =
$[Employee.EmployeeID]) AS 'TextBinLocation',
(SELECT CodeBars FROM OITM
WHERE ItemCode=SUBSTRING($[TextItem],1,PATINDEX('%*%',$[TextItem])-2) ) AS
'EanCode'
```

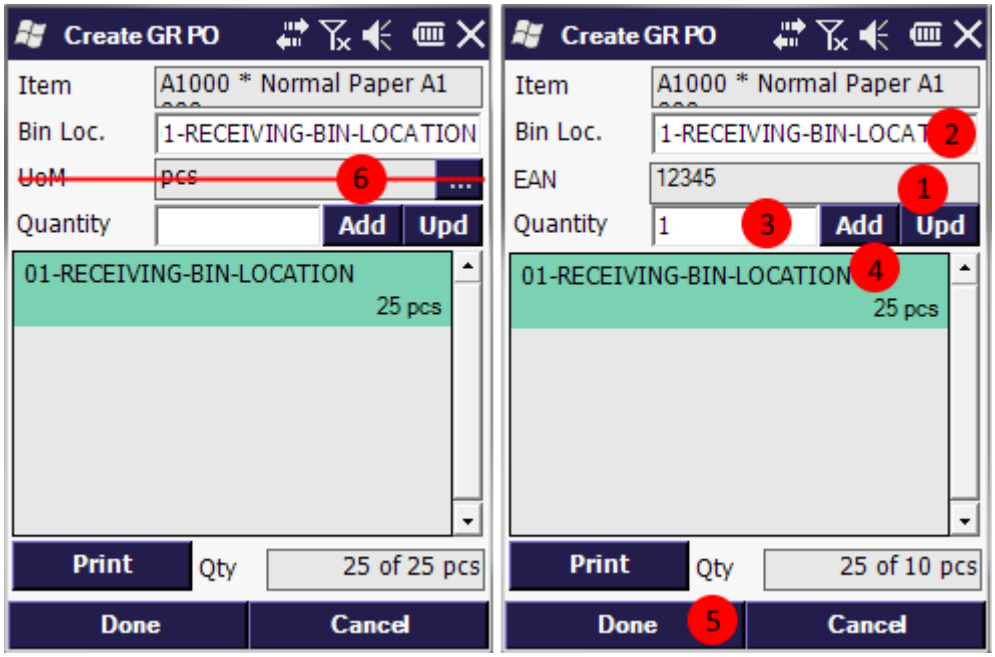
2.
To show the new EAN field on the screen, add a record to the Customization Fields(BXPCUSTFD) user table:

FieldName	Label	ReadOnly	Screen
EanCode	EAN	YES	CreateGoodsReceiptPOQuantitiesSerialScreen

3., 4.
User query: *BXMoblieWH9_CreateGoodsReceiptPOQuantitiesSerialScreen_TextSerial_validate_after*

```
-- After the serial number has been scanned,
-- press OK button to go back to previous screen
IF $[TextSerial]<>'' SELECT 'OK' AS Click$
```

3.1.5. Regular Items Screen



Changes:
1. New field: EAN (Item's bar code) added and loaded with data
2. Save the last bin for next time and load it with last value
3. Set quantity to 1 by default and focus on quantity
4., 5. When the Add button is pressed automatically press Done to go back to previous screen
6. UoM field is hidden because it's not used

The user queries and custom fields needed for these are:

1., 2., 3.
Create a new user table 'SCANEMPBL' with the following fields:

- To Bin Location (ToBL)
- Employee ID (empID)

User query: *BXMobileWH9_CreateGoodsReceiptPOQuantitiesNormalScreen_Load*

```
-- When screen is loaded fill bin location with last value
-- fill EAN with bar code from OITM, set quantit to 1 and
-- focus on Quantity field
SELECT
(SELECT CodeBars FROM OITM WHERE
ItemCode=SUBSTRING($[TextItem],1,PATINDEX('%*%',$[TextItem])-2) ) AS
'EanCode',
(SELECT [@SCANEMPBL].U_ToBL FROM [@SCANEMPBL] WHERE U_empID =
$[Employee.EmployeeID]) AS 'TextBinLocation',
'1' AS 'TextQuantity',
'TextQuantity' AS Click$
```

2.
User query:
BXMobileWH9_CreateGoodsReceiptPOQuantitiesNormalScreen_TextBinLocation_validate_after

```
-- Save bin location for next time
IF (SELECT ISNULL([@SCANEMPBL].U_ToBL,'')
FROM [@SCANEMPBL] WHERE U_empID = $[Employee.EmployeeID]) <>
$[TextBinLocation]
UPDATE [@SCANEMPBL] SET [U_ToBL]=$[TextBinLocation] WHERE U_empID =
$[Employee.EmployeeID]
```

4., 5.
User query: *BXMobileWH9_CreateGoodsReceiptPOQuantitiesNormalScreen_ButtonAdd_click_after*

```
-- When Add button is pressed, automatically press OK to close screen
SELECT 'OK' AS Click$
```

2., 6.
To show the new EAN field on the screen and hide UoM, add record to the Customization Fields(BXPCUSTFD) user table:

FieldName	Label	ReadOnly	Visible	Screen
EanCode	EAN	YES	YES	CreateGoodsReceiptPOQuantitiesNormalScreen
TextUoM			NO	CreateGoodsReceiptPOQuantitiesNormalScreen

Helper SQL procedures

Produmex Scan List splitting 1

```
CREATE FUNCTION [dbo]. [SplitStringForDataRepeater] ( @string NVARCHAR
(MAX))
RETURNS @output TABLE (splitdata NVARCHAR (MAX))
BEGIN
    DECLARE @START INT, @END INT
    SELECT @START = 1, @END = CHARINDEX ('##', @string)
    WHILE @START < LEN (@string) + 1 BEGIN
        IF @END = 0
            SET @END = LEN (@string) + 2
        INSERT INTO @output (splitdata)
        VALUES (SUBSTRING(@string, @START, @END - @START))
        SET @START = @END + 2
        SET @END = CHARINDEX('##', @string, @START)
    END
    RETURN
END
```

Produmex Scan List splitting 2

```
CREATE FUNCTION [dbo]. [SplitStringForDataRepeater2] ( @string NVARCHAR
(MAX), @string2 NVARCHAR (MAX))
RETURNS @output TABLE (splitdata NVARCHAR (MAX), splitdata2 NVARCHAR (MAX))
```

```
BEGIN
    DECLARE @START INT, @END INT @start2 INT, @end2 INT
    SELECT @START = 1, @END = CHARINDEX ('##', @string) , @start2 = 1, @end2
= CHARINDEX ('##', @string2)
    WHILE @START < LEN (@string) + 1 BEGIN
        IF @END = 0
            BEGIN
                SET @END = LEN (@string) + 2
                SET @end2 = LEN (@string) + 2
                END
        INSERT INTO @output (splitdata, splitdata2)
        VALUES (SUBSTRING(@string, @START, @END - @START)
,SUBSTRING(@string2, @start2, @end2 - @start2))
        SET @START = @END + 2
        SET @END = CHARINDEX('##', @string, @START)
        SET @start2 = @end2 + 2
        SET @END = CHARINDEX('##', @string2, @start2)
    END
    RETURN
END
```