

Strategies in BX Mobile Warehouse

BX Mobile Warehouse supports automatic strategies, which are recommendations for select warehouse workers to do stock movements. In practice this can mean that when new goods arrive, the truck is unloaded into a loading area, bin location, and after that, the automatic recommendations instruct the warehouse workers where to move the unloaded stocks. Replenishment or refilling recommendations can be used to make sure that the frequently used, easily reachable bin locations always have adequate and enough stock, and if not, then stock movements are recommended. BX Mobile Warehouse (server) creates the recommendations as Stock Transfer Draft documents, which can be processed on the mobile device and the result is a created Stock Transfer document.

1. Incoming Strategy

The incoming strategy is a user query that is run periodically by the BX Mobile Warehouse server component. The output of the query will create Stock Transfer - Draft documents if moving stock is needed.

1.1. Configuration

1.1.1. Settings

Incoming strategies UQ name (default: bxmblewh9_strategy_incoming)

Incoming strategies frequency (default: 300 seconds = 5 minutes)

1.1.2. Query input, output

By default, the user query must be named 'bxmblewh9_strategy_incoming', but this can be changed in the configuration.

Input: the user query takes no input

Output: the user query must return a table with the recommended movement results.

The result table columns are:

- ItemCode
- BatchNumber (only applicable for batch items, otherwise leave empty)
- SerialNumber (only applicable for serial items, otherwise leave empty)
- Quantity (in inventory UoM)
- SourceBin (where to move from)
- DestinationBin (where to move to)
- GroupID
- Remarks

The GroupID result column can be used to group created output documents: for each different groupID, a different Stock Transfer Draft document will be created.

1.2. Logging, monitoring

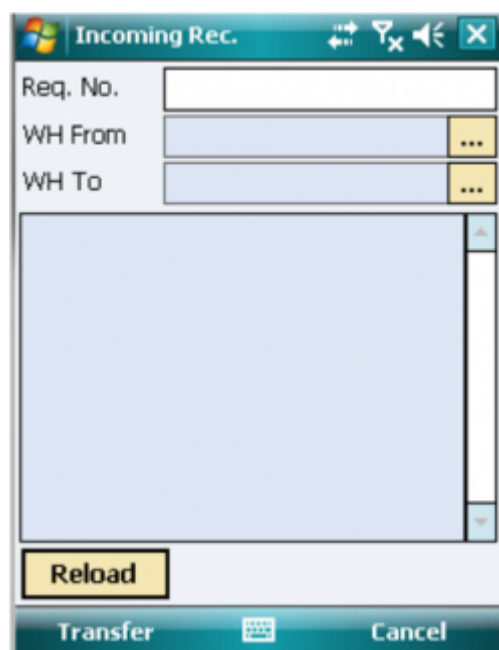
1.2.1. Mobile interface

Incoming strategy recommendation results transfers can be accessed from the Mobile WAREHOUSE main menu, with the Incoming Recommendations icon.

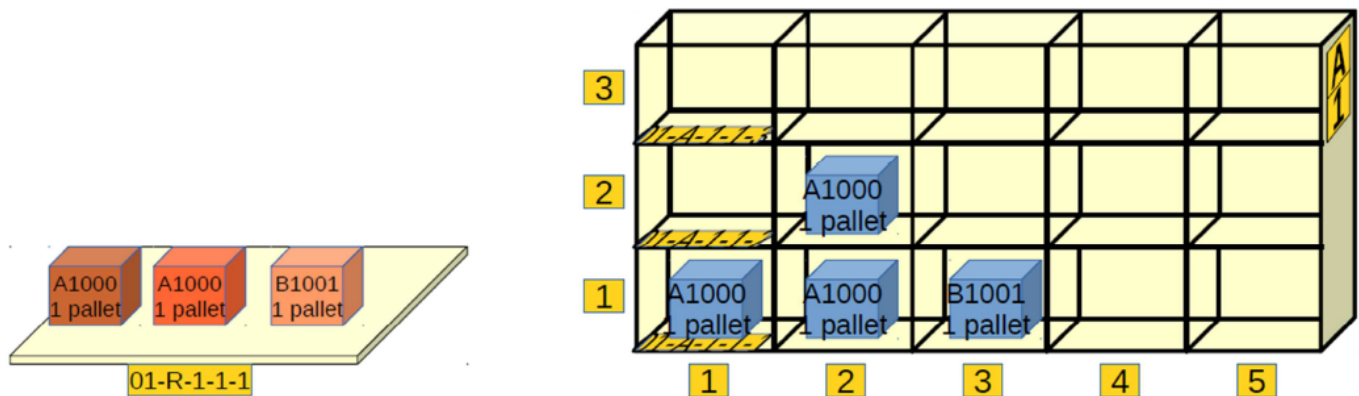


On the incoming recommendations screen, you will see all the Stock Transfer Draft documents, where the Transaction Type user-defined field is Incoming. It is possible to filter by from Warehouse or To warehouse.

After selecting the appropriate document, press the Transfer button to start working on it.



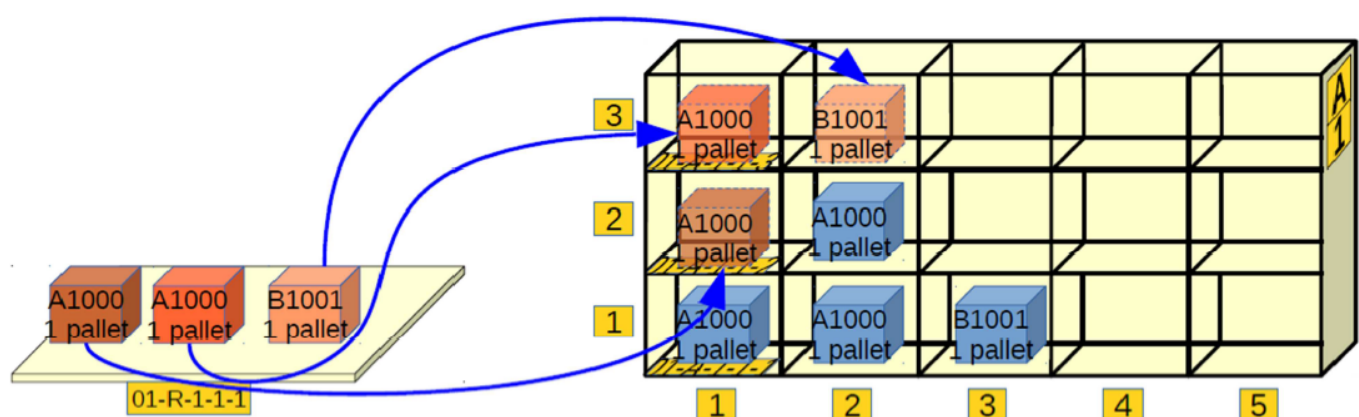
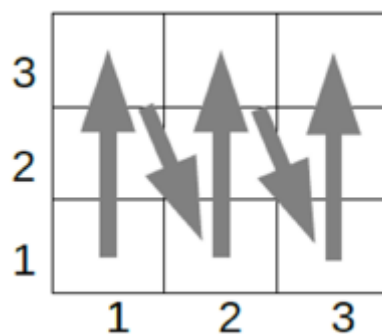
1.2.2. Example incoming strategy



In this example, we have an incoming location (01-R-1-1-1) where incoming trucks unload the goods. Our incoming strategy will check if there is any stock on this location, and automatically recommend moving it to empty locations in the warehouse shelves (locations 01-A-1-*-*).

The algorithm which is implemented in an SQL user query:

1. Check if incoming locations (01-R-1-1-1) have stock
2. Look at the stock, see if it has a recommendation already
3. If not, then find a locations to move to - the next empty location in 01-A-1-*-*. Order of looking for empty locations is by shelf column then level (see figure below).
4. Return results to create recommendations for moving the incoming the item



In this example scenario on the picture above, the algorithm will return the following recommendations:

Item	Batch	From bin	To bin	Quantity
A1000		01-R-1-1-1	01-A-1-1-2	1 (pallet)*
A1000		01-R-1-1-1	01-A-1-1-3	1 (pallet)*

Item	Batch	From bin	To bin	Quantity
B1001	B12345	01-R-1-1-1	01-A-1-2-3	

**Please note: Quantity is always in inventory units, pallet units are only for illustration purposes here.*

2. Replenishment Strategy

2.1. Configuration

2.1.1. Settings

Replenishment strategies UQ name (default: bxmobilewh9_strategy_replenishment)

Replenishment strategies frequency (default: 300 seconds = 5 minutes)

2.1.2. Query input, output

By default, the user query must be named 'bxmobilewh9_strategy_replenishment', but this can be changed in the configuration.

Input: the user query takes no input

Output: the user query must return a table with the recommended movement results.

The result table columns are:

- ItemCode
- BatchNumber (only applicable for batch items, otherwise leave empty)
- SerialNumber (only applicable for serial items, otherwise leave empty)
- Quantity (in inventory UoM)
- SourceBin (where to move from)
- DestinationBin (where to move to)
- GroupID
- Remarks

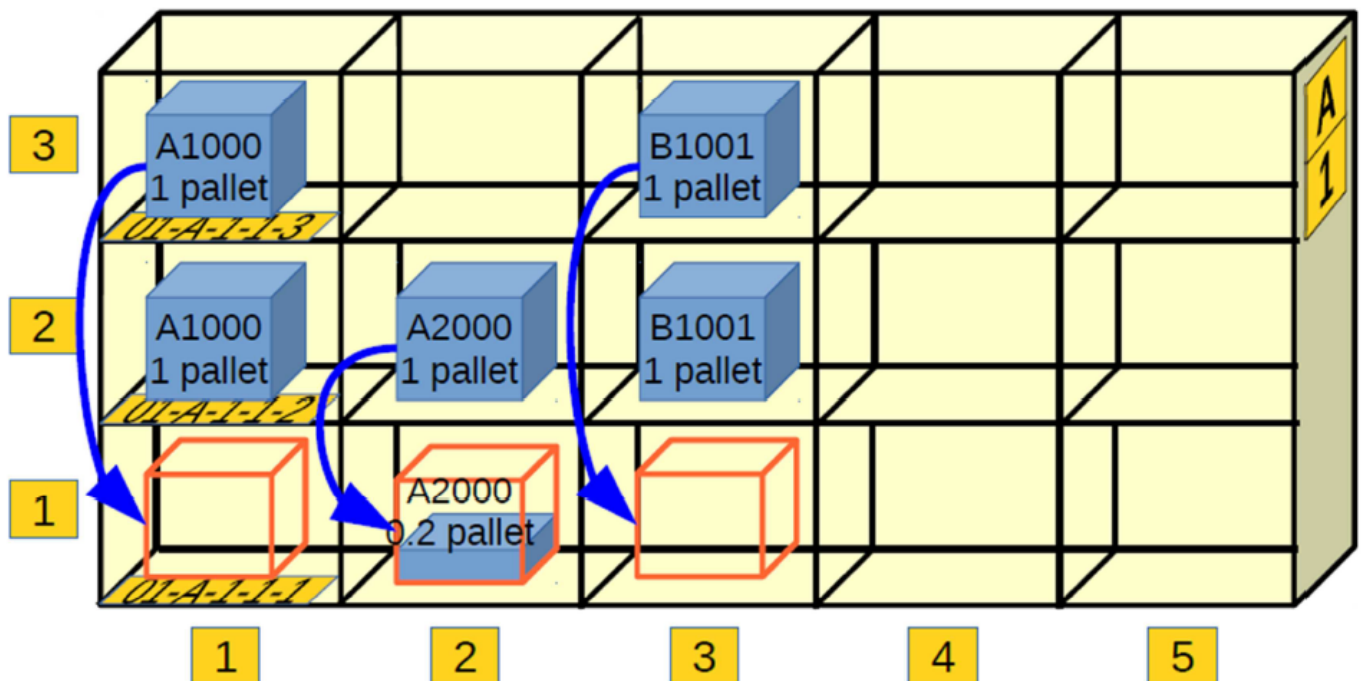
The GroupID result column can be used to group created output documents: for each different groupID, a different Stock Transfer Draft document will be created.

2.1.3. Logging, monitoring

2.1.4. Mobile interface

See the section 'Incoming Strategy - Mobile interface'. The work flow is the same, but you have to select the 'Replenishment recommendations' icon on the main screen.

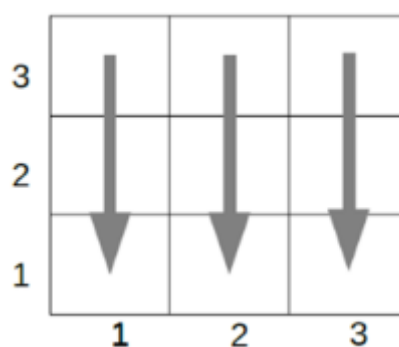
2.1.5. Example replenishment strategy

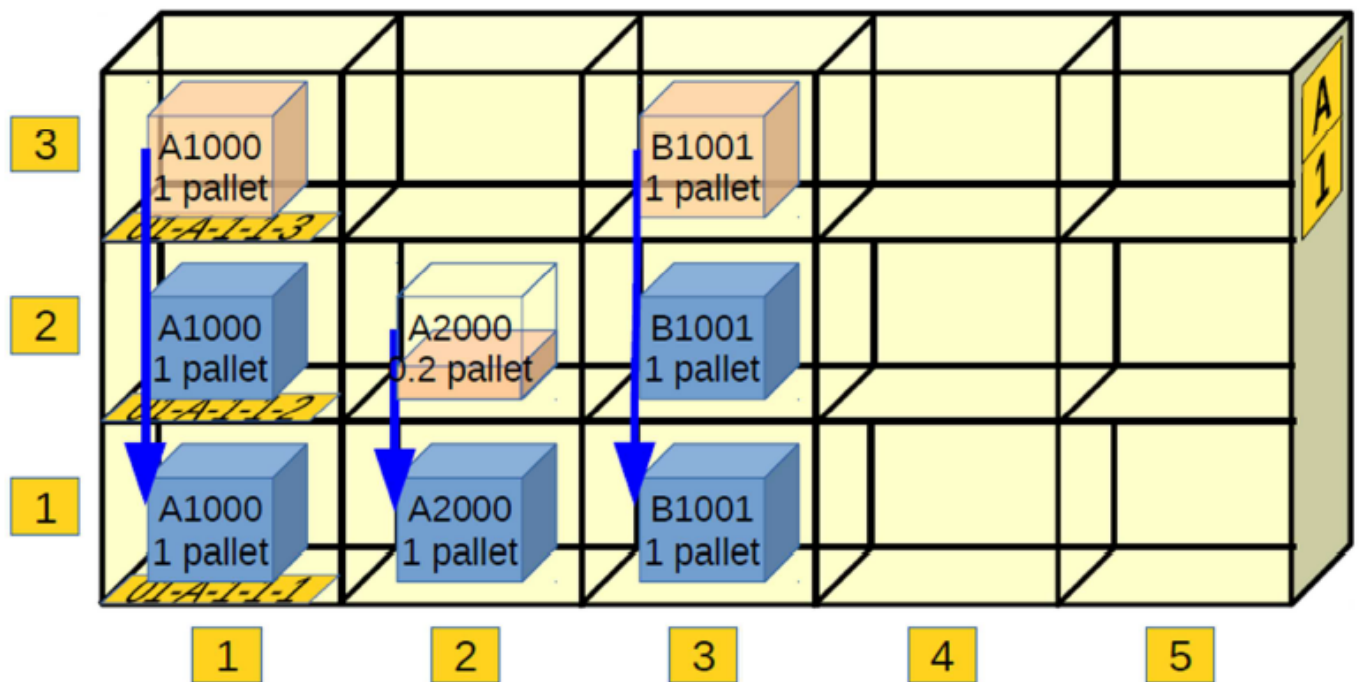


Replenishment strategy can be used to refill easily-reached locations in a warehouse system. In this example warehouse, the locations on the first level (01-A-1-*-1) are reachable by every warehouse worker, but higher levels (01-A-1-*-2, 3) are only reachable by fork lifts.

The refill algorithm, which is implemented in SQL query:

1. Check first level locations (01-A-1-*-1), see if it's empty or it's below minimum level (25% of pallet).
2. Look for refill sources in the same column, but higher levels. If found, recommend moving items from upper levels to lower levels





In this example scenario on the picture above, the algorithm will return the following recommendations:

Item	Batch	From bin	To bin	Quantity
A1000		01-A-1-1-3	01-A-1-1-2	1 (pallet)*
A2000		01-A-1-2-3	01-A-1-1-3	0.8 (pallet)*
B1001	B12345	01-A-1-3-3	01-A-1-2-3	1 (pallet)*

**Please note: Quantity is always in inventory units, pallet units are only for illustration purposes here.*

3. Appendix

3.1. Example Incoming Strategy SQL

```

-----
--
-- Incoming recommendations query
-- Algorithm:
-- 1. Look for stock on RecBinLocation (eg. 01-Q), only non-serial, non-
batch items
-- 2. Subtract/ignore stock quantity already recommended for moving
-- 3. Recommend moving remaining quantity to locations defined by
OutLocationFilter
-- eg. 01-F-%, look for empty locations
-- Only put one PurchaseUoM (eg. pallet) quantity on one out location
-- split moving to multiple out locations if needed.
-----
--

```

```
--
DECLARE @RecBinLocation nvarchar(MAX)
DECLARE @OutLocationFilter nvarchar(MAX)
DECLARE @ItemCode nvarchar(MAX)
DECLARE @Quantity DECIMAL
DECLARE @SourceBin nvarchar(MAX)
DECLARE @DestinationBin nvarchar(MAX)
DECLARE @PurchaseUomQuantity nvarchar(MAX)
DECLARE @QuantityPartial DECIMAL
DECLARE @ExcludeBinList nvarchar(MAX)
DECLARE @QuantityAlready nvarchar(MAX)
SET @RecBinLocation = '01-Q'
SET @OutLocationFilter = '01-F-%'
SET @ExcludeBinList = ''
DECLARE @RESULT TABLE (
ItemCode nvarchar(MAX),
BatchNumber nvarchar(MAX),
SerialNumber nvarchar(MAX),
Quantity DECIMAL,
SourceLocation nvarchar(MAX),
DestinationLocation nvarchar(MAX),
GroupID nvarchar(MAX),
Remarks nvarchar(MAX)
)
DECLARE curs CURSOR FOR
SELECT OIBQ.ItemCode, OIBQ.OnHandQty, OBIN.BinCode FROM OBIN, OIBQ, OITM
WHERE OBIN.BinCode = @RecBinLocation AND OIBQ.BinAbs = OBIN.AbsEntry AND
OITM.ItemCode = OIBQ.ItemCode AND OIBQ.OnHandQty > 0
AND OITM.ManBtchNum = 'N' AND OITM.ManSerNum = 'N'
OPEN curs
FETCH NEXT FROM curs INTO @ItemCode, @Quantity, @SourceBin
WHILE @@FETCH_STATUS = 0
BEGIN
SET @PurchaseUomQuantity = NULL
SELECT @PurchaseUomQuantity = BaseQty FROM UGP1, OITM WHERE UGP1.UgpEntry =
OITM.UgpEntry AND UGP1.UomEntry = OITM.PUomEntry AND OITM.ItemCode =
@ItemCode
SET @QuantityAlready = NULL
SELECT @QuantityAlready = SUM(FromQuantity) FROM
XXX_INVTRANSFER_DRAFT_LINESBIN
WHERE ItemCode = @ItemCode AND FromBin = @SourceBin
IF @QuantityAlready IS NOT NULL BEGIN SET @Quantity = @Quantity -
@QuantityAlready END

WHILE @Quantity > 0 BEGIN
SET @QuantityPartial = @Quantity
IF @PurchaseUomQuantity IS NOT NULL AND @QuantityPartial >
@PurchaseUomQuantity BEGIN SET @QuantityPartial = @PurchaseUomQuantity END
SET @DestinationBin = NULL
SELECT TOP 1 @DestinationBin = BinCode
FROM OBIN LEFT OUTER JOIN OIBQ ON (OBIN.AbsEntry = OIBQ.AbsEntry)
```

```
WHERE OBIN.BinCode LIKE @OutLocationFilter
AND charindex(OBIN.BinCode, @ExcludeBinList) <= 0
AND OBIN.BinCode NOT IN (SELECT ToBin FROM XXX_INVTRANSFER_DRAFT_LINESBIN
WHERE ToBin LIKE @OutLocationFilter)
GROUP BY BinCode HAVING SUM(OnHandQty) = 0 OR SUM(OnHandQty) IS NULL
ORDER BY BinCode
-- if no more places, DestinationBin will be empty
INSERT INTO @RESULT (ItemCode, Quantity, SourceBin, DestinationBin)
VALUES (@ItemCode, @QuantityPartial, @SourceBin, @DestinationBin)
SET @Quantity = @Quantity - @QuantityPartial
SET @ExcludeBinList = @ExcludeBinList + ' ' + @DestinationBin
END
FETCH NEXT FROM curs INTO @ItemCode, @Quantity, @SourceBin
END
CLOSE curs
DEALLOCATE curs
SELECT * FROM @RESULT
```

3.2. Helper view

```
CREATE VIEW XXX_INVTRANSFER_DRAFT_LINESBIN
AS
SELECT DRF1.DocEntry, DRF1.ItemCode, DRF1.LineNum
, fromBin.BinCode AS FromBin, fromBinLine.Quantity AS FromQuantity,
toBin.BinCode
AS ToBin, toBinLine.Quantity AS ToQuantity FROM ODRF, DRF1
LEFT OUTER JOIN DRF19 fromBinLine ON (fromBinLine.ObjType = 67 AND
fromBinLine.DocEntry = DRF1.DocEntry AND fromBinLine.LineNum = DRF1.LineNum
AND
fromBinLine.BinActTyp = 2)
LEFT OUTER JOIN DRF19 toBinLine ON (toBinLine.ObjType = 67 AND
toBinLine.DocEntry =
DRF1.DocEntry AND toBinLine.LineNum = DRF1.LineNum AND toBinLine.BinActTyp =
1)
LEFT OUTER JOIN OBIN fromBin ON (fromBin.AbsEntry = fromBinLine.BinAbs)
LEFT OUTER JOIN OBIN toBin ON (toBin.AbsEntry = toBinLine.BinAbs)
WHERE
ODRF.ObjType = 67 AND DRF1.ObjType = 67 AND toBinLine.ObjType = 67
AND ODRF.DocEntry = DRF1.DocEntry
```

3.3. Example Replenishment Strategy SQL

```
-----
-----
--
-- Replenishment recommendations query
-- Algorithm:
-- 1. See what stocks are on floor (*-1) locations and on upper (*-2, *-3,
```



```

ec) locations
-- for the same itemcode. (Only normal items are considered, and which have
purchase
-- uom groups defined) Only look in bin locations 01-F-* (ScanArea0
-- 2. If floor quantity <= 50% of pallet quantity, look for upper locations
-- Also consider Inventory Transfer Draft documents already recorded
-- Recommend moving quantities from upper locations until floor quantity
reaches
-- 100% of pallet quantity if possible
-----
-----
--
DECLARE @ScanArea nvarchar(MAX)
DECLARE @FloorLocation4 nvarchar(MAX)
SET @ScanArea = '01-F-%'
SET @FloorLocation4 = '1'

DECLARE @FloorBin nvarchar(MAX)
DECLARE @ItemCode nvarchar(MAX)
DECLARE @FloorQuantity DECIMAL
DECLARE @UpperBin nvarchar(MAX)
DECLARE @UpperQuantity DECIMAL
DECLARE @PalletQty DECIMAL
DECLARE @QuantityPartial DECIMAL
DECLARE @LastBin nvarchar(MAX)
DECLARE @LastBinFloorQuantity DECIMAL
DECLARE @QuantityNeeded DECIMAL
DECLARE @QuantityAlready DECIMAL
DECLARE @RESULT TABLE (
ItemCode nvarchar(MAX),
BatchNumber nvarchar(MAX),
SerialNumber nvarchar(MAX),
Quantity DECIMAL,
SourceLocation nvarchar(MAX),
DestinationLocation nvarchar(MAX),
GroupID nvarchar(MAX),
Remarks nvarchar(MAX)
)
DECLARE curs CURSOR FOR
SELECT
binupper.FloorBinCode2 AS FloorBin, binupper.ItemCode AS ItemCode,
binfloor.OnHandQty AS FloorQuantity,
binupper.BinCode AS UpperBin, binupper.OnHandQty AS UpperQuantity,
palletQuantities.BaseQty AS PalletQty
FROM
(
SELECT OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' +
OBIN.SL3Code
AS BinPrefix,
OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' + OBIN.SL3Code
+ '-' +

```

```
@FloorLocation4 AS FloorBinCode2,
OBIN.BinCode, OIBQ.ItemCode, OIBQ.OnHandQty FROM OIBQ, OBIN WHERE
OBIN.BinCode LIKE
@ScanArea AND OBIN.SL4Code <> @FloorLocation4 AND OBIN.AbsEntry =
OIBQ.BinAbs
) binupper
LEFT OUTER JOIN
(
SELECT OBIN.WhsCode + '-' + OBIN.SL1Code + '-' + OBIN.SL2Code + '-' +
OBIN.SL3Code
AS BinPrefix, OBIN.BinCode, OIBQ.ItemCode, OIBQ.OnHandQty
FROM OIBQ, OBIN WHERE OBIN.BinCode LIKE @ScanArea AND OBIN.SL4Code =
@FloorLocation4 AND OBIN.AbsEntry = OIBQ.BinAbs
) binfloor ON (binfloor.ItemCode = binupper.ItemCode AND binfloor.BinPrefix
=
binupper.binprefix)
JOIN (
SELECT ItemCode, BaseQty FROM UGP1, OITM WHERE UGP1.UgpEntry = OITM.UgpEntry
AND
UGP1.UomEntry = OITM.PUomEntry
) palletQuantities ON (binupper.ItemCode = palletQuantities.ItemCode)
WHERE binfloor.OnHandQty IS NULL OR binfloor.OnHandQty <=
palletQuantities.BaseQty
/2 AND binupper.OnHandQty > 0
ORDER BY UpperBin
SET @LastBin = ''
SET @LastBinFloorQuantity = 0
OPEN curs
FETCH NEXT FROM curs INTO @FloorBin, @ItemCode, @FloorQuantity, @UpperBin,
@UpperQuantity, @PalletQty
WHILE @@FETCH_STATUS = 0
BEGIN
IF @FloorQuantity IS NULL BEGIN SET @FloorQuantity = 0 END
IF @LastBin <> @FloorBin BEGIN
SET @LastBin = @FloorBin
SET @QuantityAlready = NULL
SELECT @QuantityAlready = SUM(ToQuantity) FROM
XXX_INVTRANSFER_DRAFT_LINESBIN WHERE ItemCode = @ItemCode AND ToBin =
@FloorBin
IF @QuantityAlready IS NULL BEGIN SET @QuantityAlready = 0 END
SET @LastBinFloorQuantity = @FloorQuantity + @QuantityAlready
END
IF @LastBinFloorQuantity <= @PalletQty / 2 BEGIN
SET @QuantityNeeded = @PalletQty - @LastBinFloorQuantity
SET @QuantityPartial = @QuantityNeeded
IF @QuantityPartial > @UpperQuantity BEGIN SET @QuantityPartial =
@UpperQuantity END
SET @LastBinFloorQuantity = @LastBinFloorQuantity + @QuantityPartial
INSERT INTO @RESULT (ItemCode, Quantity, SourceBin, DestinationBin)
VALUES (@ItemCode, @QuantityPartial, @UpperBin, @FloorBin)
```

```
END
FETCH NEXT FROM curs INTO @FloorBin, @ItemCode, @FloorQuantity, @UpperBin,
@UpperQuantity, @PalletQty
END
CLOSE curs
DEALLOCATE curs
SELECT * FROM @RESULT
```

From:

<http://wiki.produmex.name/> -

Permanent link:

<http://wiki.produmex.name/doku.php?id=implementation:bxmobwh:strategies>



Last update: **2017/03/01 12:27**