# **Job for replenishment priority calculation**

## Summary:

Currently there are multiple places affect replenishment work priority:

1. For work operation code PIARPL, standard config escalate work by every 5 minutes.
2. USR-REPLEN-ESC-BY-SEAL-TIME, this job escalates replenishment work by shipment seal time.
3. USR-UNLOCK-RPL-FOR-MIN-QTY, this job will update priority to 20 for fast move item once job is unlocked.
4. USR-HOT-REPLN-PRIORITY-ESCALATION, this job changes replenishment work to priority 10 when there is cancel pick in 3 minutes.
5. ESC-REPLEN-WORK, this is standard job which will give Trigger replenishment work with same priority as Demand replenishment work when the work is needed by outbound pick and convert TR to DR.
6. When acknowledging assignment system bump replenishment work based on how many quantities needed, also it will change priority to 14 if the location is within first 25 locations of the assignment, or 15 if it’s after 25 locations in the assignment.

## Challenges:

In current situation, it’s difficult to maintain or tweak logic to adjust priority, especially for bumping priority when acknowledge assignment, it’s observed quite often for case like too late when bumping the priority which cause picking short.

## New Approach:

Instead of spreading these logics in multiple places, we are using a single job to handle the priority calculation, other jobs either need to be disabled or changed to remove the ability of changing replenishment work priority.

### New job command name:

**process usr replenishment work priority setup**

this command will be deployed as a job to be executed periodically; this job accept 2 parameters:

**wh\_id:** warehouse ID.

**area\_prefix:** represents for which area this job is calculating replenishment work priority, it can be: P: Produce, C: Chiller, A: ambient.

**Important:** this is relying on the naming convention for arecod created, the first char of arecod will tell if this area belongs to Produce, Chiller or Ambient, so arecod must be started with P, C, or A.

Considering Chiller and Ambient has big number of replenishment locations which could take minutes to finish for each run, a further enhancement is made to support a single area job into multiple jobs which is identified by two parameters:

**Job\_count:** tells how many jobs are needed for the single area.

**Job\_id:** the id of the job that runs for the single area, this id starts from 1.

If we just pass wh\_id, this job will handle above 3 areas at the same time, however as distributed code, there will be 7 jobs to be deployed for above ‘Ambient’, ‘Chiller’ and ‘Produce’ to run for performance improvement, ‘Ambient’ and ‘Chiller’ will have 3 jobs for each, and ‘Produce’ will has one job, and these jobs will be scheduled to run every 30 seconds.

**Logic of the job:**

This job takes some important factors into consideration which related to replenishment work:

**1. pick cancel:** If there is pick short in 3 minutes, this job will give priority 4 for the replenishment work.

**2. Top x assignment fulfillment:** Instead of only looking at ‘ACK’ status assignments and based on that to tell how many quantities is needed, we introduce x number of top assignments needs to be considered, this x is a policy defined value, e.g 3. it means system will take next 3 assignments into calculating how many quantities is needed including ‘ACK’ status assignment on top. and assignment included must be belonging to LOAD which has started picking, so it is not relying on user to ‘ACK’ assignment to trigger the bump with which we expect system should reduce the ‘too late’ scenario. for 3 areas, PRODUCE, CHILLER, and AMBIENT, each one of them will be defined with such a policy so it can be defined with different value, for example, PRODUCE usually picking faster so we can config 4 assignments.

**3. Seal time:** To differentiate replenishment work for shipments with different seal time, we use a rank mechanism to give each replenishment a rank value, first, we get earliest seal time from shipments which need replenishment work. And we take hour from the seal time and put them into order, for example suppose we have below seal times:

07:00, 07:30, 8:00, 8:15, 9:00, 10:00, 11:00

For them we give below rank value:

1, 1, 1, 1, 2, 2, 3

The idea is that for first 2 different hours, we give rank value 1, since hour 07 and 08 are first 2 hour values, so any seal time with 07 or 08 as hour will own rank value 1; for 3 or 4 different hour values we give rank value 2, from 4 to 6 different hour values we give 3, anything beyond 6 different hour values the rank value will be 4. And here the number 2, 4, and 6 are driven by 3 policies as ‘SEAL-TIME-COUNT’, default value is 2, 4 and 6:

[select rtnum1 stc1

from poldat\_view pv

where pv.polcod = 'USR'

and pv.polvar = 'REPLENISHMENT-PRIORITY-SETUP'

and pv.polval = 'SEAL-TIME-COUNT1'

and pv.wh\_id = @wh\_id]

|

[select rtnum1 stc2

from poldat\_view pv

where pv.polcod = 'USR'

and pv.polvar = 'REPLENISHMENT-PRIORITY-SETUP'

and pv.polval = 'SEAL-TIME-COUNT2'

and pv.wh\_id = @wh\_id]

|

[select rtnum1 stc3

from poldat\_view pv

where pv.polcod = 'USR'

and pv.polvar = 'REPLENISHMENT-PRIORITY-SETUP'

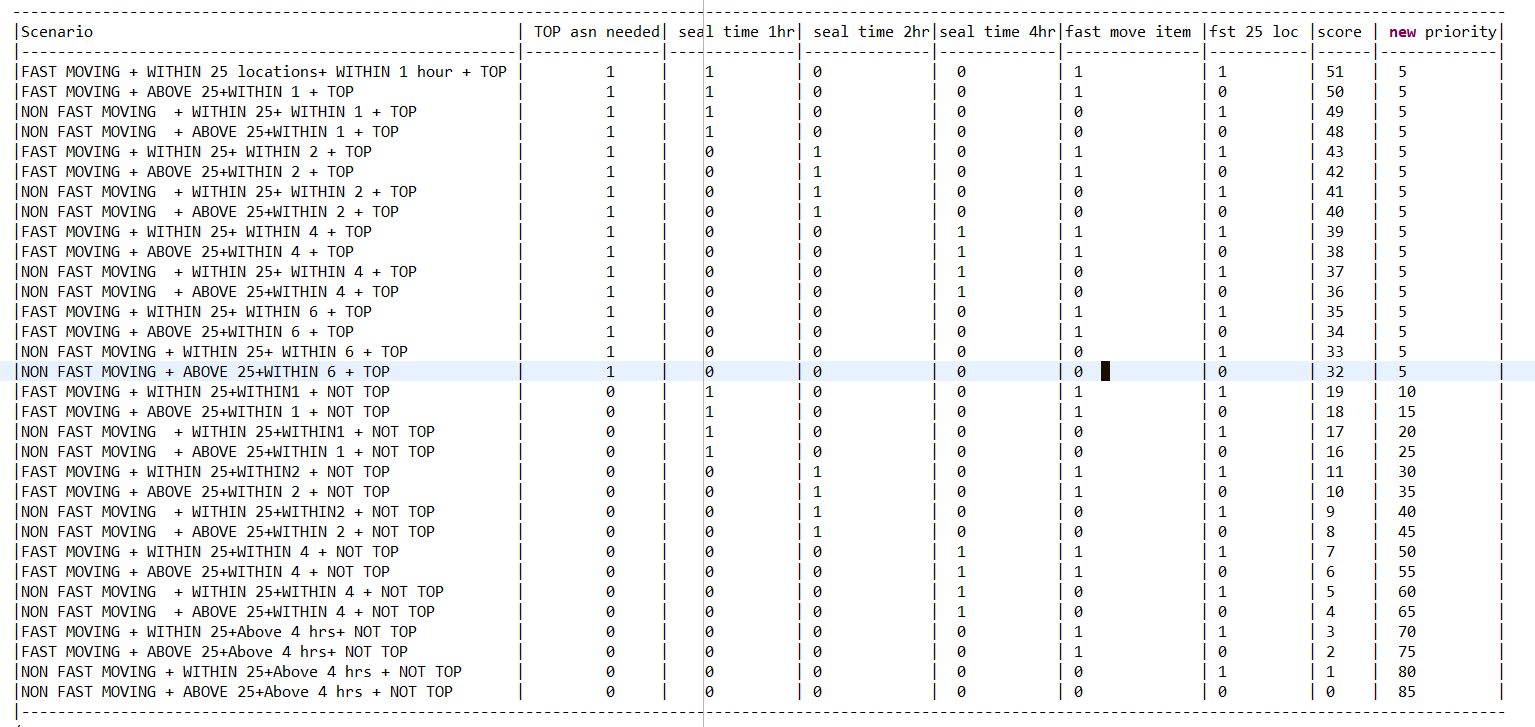
and pv.polval = 'SEAL-TIME-COUNT3'

and pv.wh\_id = @wh\_id]

**4. Fast moving item:** same as current logic, when replenishment minimum value is >= 2 pallets quantity with default ftpcod, then this item is considered as fast-moving item.

**5. Within first 25 loc:** if there is any assignment acknowledged need the replenishment within its 25 locations, then this considered as yes.

Based on above criteria, 6 bits scoring system is introduced as below:



For each scenario above, there will be a score calculated, based on each score, we remap each score to a new priority according to importance:

For example, above score 51 is coming from a binary value 110011 for the 6 criteria’s, it can be converted to 10 based value:

1\*2^5 + 1\*2^4 + 0\*2^3 + 0\*2^2 + 1\*2^1 + 1\*2^0 = 32 + 16 + 2 + 1 = 51

And this 51 is remapped to priority 5, from above chart, for scenario if the replenishment work is needed for top x assignments, they will be mapped with priority 5. This remapping can be adjusted later based on testing result via policy:

[select rtnum2 new\_priority

from poldat\_view pv

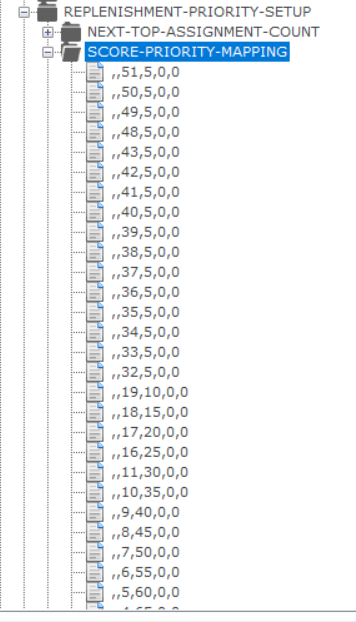
where pv.polcod = 'USR'

and pv.polvar = 'REPLENISHMENT-PRIORITY-SETUP'

and pv.polval = 'SCORE-PRIORITY-MAPPING'

and pv.rtnum1 = @total\_score

and pv.wh\_id = @wh\_id] catch(-1403)



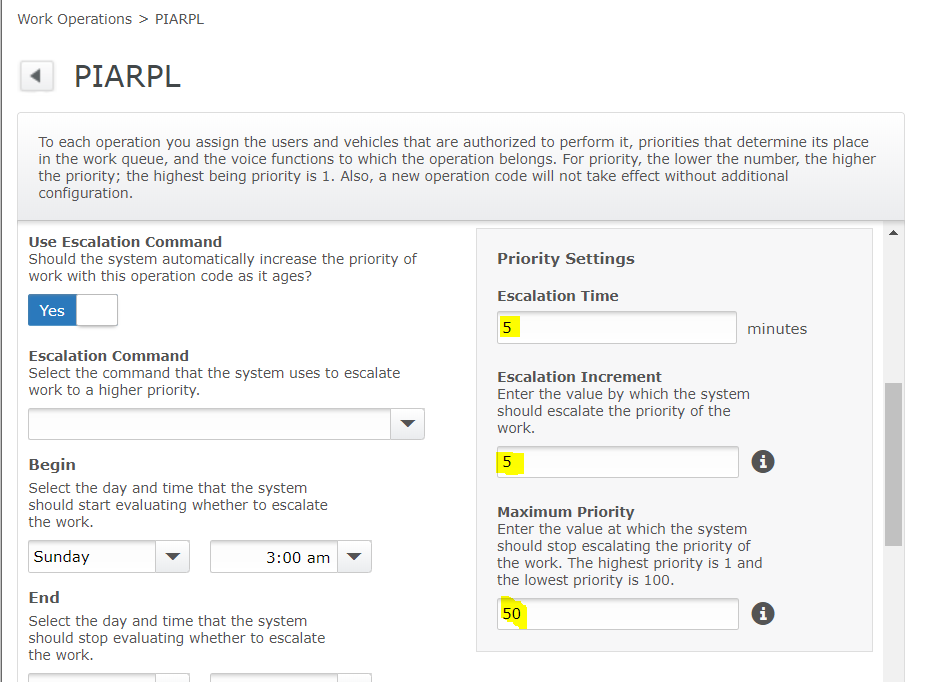
See the csv file for all mapping data, the advantage of using policy to do the mapping is that going forward, we can readjust the mapping according to what we observed, for example, if there are too many priority 5 works in the queue, we can remap some ‘5’ into lower priority, until all priority disturbed in a way we expected.

Note: This job still gives user the ability to manually assign a priority to let the work to be done earlier or later which depends on if policy polval = ‘ALLOW-MANUAL-PRIORITY-CHANGE’ is enabled, to achieve this user need to setup the work on a priority which is not defined in above score-priority mapping, nor the initial priority defined for the replenishment work oprcod.

## Existing Configuration change:

To deploy the new code, below existing changes need to be done:

1. Disable the 5 minutes escalation setup for PIARPL:



Since we are handling priority with a single job, this should not be needed, we can give above 3 values as 0 to disable it.

2. USR-REPLEN-ESC-BY-SEAL-TIME, this job needs to be disabled as it is already considered in the new job.

3. USR-UNLOCK-RPL-FOR-MIN-QTY, this job changes the priority to 20 when unlock, this functionality needs to be considered in new job for simplicity. It can be easily done by looking at fast moving item flag calculated.

4. USR-HOT-REPLN-PRIORITY-ESCALATION, this job changes replenishment work to priority 10 when there is cancel pick in 3 minutes, this job should be disabled as it included in new job.

5. ESC-REPLEN-WORK, this is standard job which will give trigger replenishment work same priority as demand replenishment work when the work is needed by outbound pick and convert TR to DR, this job needs to be disabled, and it will replaced with new USR job ‘USR-PROCESS-REPLEN-CONVERTION’ to convert trigger replenishment work to demand replenishment work.

6. When acknowledging assignment system bumping replenishment work based on how many quantities needed, also it will change priority to 14 if the location is within first 25 location of the assignment, or 15 if it’s after 25 location in the assignment.

We need to disable this functionality by giving 0 to field rtnum1 for below policy:

list policies

where polcod = 'USR'

and polvar = 'BUMP-REPLN'

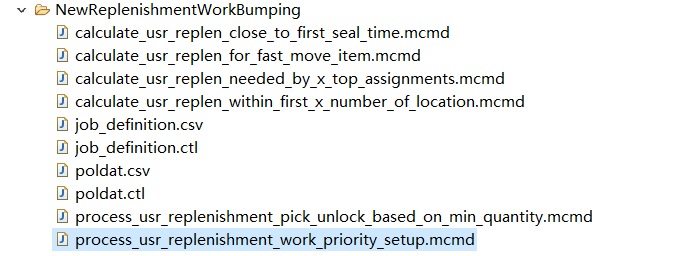
and polval = 'ENABLED'

and wh\_id = @wh\_id **catch**(@?)

7. Change the base priority for PIARPL to 85 as the lowest priority based the new job code is 85.

## Deploying new code

New files included in below screenshot:



1. put mcmd files into usrint folder.

2. mload all provided csv files.

3. run ‘mbuild’.

4 bounce moca server.