

**Version 1.1**

**Date: 2019/02/19**

Abstract

**The objective of this document is to describe the ePart method of stock assurance through effective cyc;e counting**

Stock Cycle Count

*Overview of stock assurance using cycle count method*

**Table of Contents**

Document approval and distribution list 2

1. Introduction 3

2. Audience 3

3. Objectives and overview 4

4. Cycle count modelling overview 5

5. Database entities and relationships 6

6. Programs 7

6.1 MS Windows Executables 7

6.2 SQL Stored Procedures 7

7. Acceptance 8

# Document approval and distribution list

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Name / Title** | **Signature** | **Date** |
| **Document Type / purpose** | | | |
| Prepared by |  |  |  |
| Reviewed by |  |  |  |
| Approved by |  |  |  |

# Introduction

Good corporate governance requires that the organisation has implemented an effective stock on hand assurance strategy.

The Engineparts implementation of cycle counting has proven to be accurate and effective and has negated the need for full wall to wall assurance testing.

# Audience

Management

Warehouse

Financial

# Objectives and overview

The original design objective was based on input received from consultants with extensive knowledge in the automotive industry such as Automakers (Nissan), Ford - South Africa and Mercedes Benz South Africa.

Cycle counting of stock on hand is where all items are counted over a pre-determined time (available resources and volume of work) and items with greater value counted more frequently using a value based (average cost \* average monthly sales) category.

***Review required***:

* All items are counted once per cycle count period regardless of value-based categorisation. This is reflected in the actual source code.
* The categorised priority method is not part of the current implementation as was the original intent.
* There are operational indicators that imply that a categorised method of cycle counting may be too resource intensive and for that reason not implemented as planned
* Should a categorised priority count method be re-considered, this will be easy to implement on the following basis:
  + Stock items are already re-categorised on a regularly scheduled basis.
  + Adjust model generation to include the number of times an item needs to be counted per model period i.e. “A” category items to be counted 6 times in a 3-month cycle or “D” items once in a 3-month cycle given that the months per cycle is variable.
  + Include all events that need to be audited (only picking discrepancies active now) i.e.
    - Binning – supplier related
    - Binning – customer related
    - Stock adjustments

Cycle count models can be generated at any time as ***prototypes*** using a set of parameters. Generated models can be reviewed to see if the prototype model outcome fits the business requirements. A prototype with a good fit is confirmed into production and the daily count sheets become available on demand.

Count sheets are captured into ePart confirming the physical quantity on hand.

All data captured is recorded in an event log for later review. The adjustments, where the actual differs from theoretical, is ***batched*** for approval and signoff. Once signed-off, stock adjustment journals are generated and posted

Any items counted on the day not appearing on the day count sheet are recorded and reflected as counted on future day count schedules. This would typically happen when items are moved from one bin to another bin post the model generation or the item counted naturally shares a bin with an item scheduled for counting.

Any items not counted for the day are re-scheduled for the next day and appear on the next day count sheet.

ePart has a sub system that is initiated at the end of each cycle count interval. This function extracts all items on hand, calculates the value on hand and orders it in descending sequence by value on hand.

Management need to assure that enough human resources are made available to content with the predicted workload.

If the workload is beyond available resources, the cycle count model is changes and recalculated.

On finalisation, the elected model for execution is confirmed and the required cycle count sheets produced daily

***To note:***

Any picking and related deviations detected are automatically injected into the daily cycle count schedule for deviation resolution.

Should the daily cycle count schedule not be completed, the items not counted on the day are passed on to the next day.

To note is that ePart is designed to remain fully operational during the cycle count. The count found is captured and reconciled against the start of day on hand count pus or minus the day movement at the time and this is compared to the actual on hand as counted.

The obvious dilemma is that the cycle count is not capture in real-time consequently there will be time-of-day-based deviations which is operationally catered for by doing the cycle counts off peak production time.

As detailed in the separate document dealing with mobility optimisations and real-time capturing of data, this will be catered for by such an implementation as the quantity counted will be done in real-time and consequently follow the system transaction sequences accurately

# Cycle count modelling overview

Each time a prototype model is generated, it is stored on the system with a unique identifier.

The cycle count events per item are determined against human resource allocated to the process.

The resource resolver will suggest the number of items that need to be counted per working day based on allocated resources.

Reports are available to reflect the profile of a model. Models that are not considered possible to execute on, remain inactive and can be referred to for future reference.

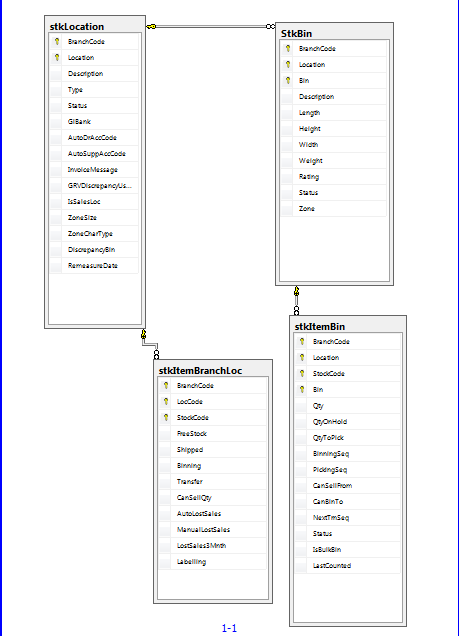
# Stock journals

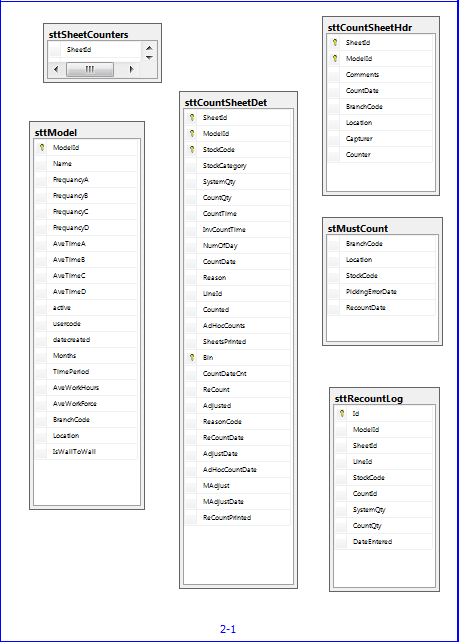
The system automatically generates stock adjustment journal post authorisation which can be tracked using enquiry tools and supporting reports.

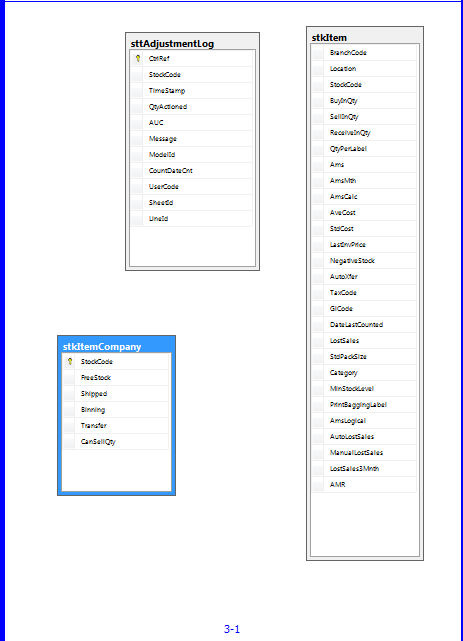
# Database entities and relationships

The following set of diagrams are largely used by the wall to wall stock take procedure whilst some are share by the cycle count process as well.

This is repeated here for completeness sake







# Programs

# MS Windows Executables

|  |  |
| --- | --- |
| **Name** | **Description** |
| stControlReport | Used to control which count sheets were given to which courting team. It also indicates which sheet have not been returned for capture |
| stDiscrepancies | List variances in descending sequence |
| stFinalisation | Call the counting process done and finalise |
| stMaintItemCount | Apply specific adjustments |
| sttAdHocCount | Release ad-hoc counts into the system beyond the standard counts |
| sttCountSheets | Print the manual count sheets |
| sttStockTakeModel | Calculate the cycle count model against a set of parameters to define resource and time |
| sttWallToWallAdjust | Generate stock adjustment journals |
| sttWallToWallCapture | Capture manual counts from sheets returned after count done |
|  |  |

# SQL Stored Procedures

The ePart system requires for the naming conventions to convey the purpose of the stored procedure in which the business rules are declared.

|  |  |
| --- | --- |
| **Name** | **Description** |
| sttAdjust.sql | Apply captured adjustments asper count sheet |
| sttBulkUpdateCountSheet.sql | Generate count sheets with theoretical quantities as per the database |
| sttCreateCountSheet.sql | This is used for the cycle count where count sheets are pre-calculated for every work day |
| sttCreateCountSheetWallToWall.sql | This generated count sheet for wall to wall counting |
| sttCreateModel.sql | Cycle count model generator |
| sttCreateModelData.sql |  |
| sttDeleteModel.sql |  |
| sttDiscrepencyReport.sql |  |
| sttFindByDate.sql |  |
| sttFindStockQty.sql |  |
| sttGetAUC.sql |  |
| sttGetExistingModels.sql |  |
| sttRefreshAdjustments.sql |  |
| sttRefreshCountSheet.sql |  |
| sttRefreshDiscrepancies.sql |  |
| sttRefreshModel.sql |  |
| sttRefreshReasonCodes.sql |  |
| sttRptAdjustmentLog.sql |  |
| sttRptAutoAdjustmentLog.sql |  |
| sttRptCapturedSummary.sql |  |
| sttRptCountDetails.sql |  |
| sttRptCountHistoryForPart.sql |  |
| sttRptCountSheet.sql |  |
| sttRptCountSheetAutoMail.sql |  |
| sttRptCountSheetWallToWall.sql |  |
| sttRptDayCountersNotCaptured.sql |  |
| sttRptDiscrepancyResolveWallToWall.sql |  |
| sttRptDiscrepencyWallToWall.sql |  |
| sttRptModelData.sql |  |
| sttRptModels.sql |  |
| sttRptRecountSummary.sql |  |
| sttRptW2WAdjustmentList.sql |  |
| sttRptWallToWallControlSheet.sql |  |
| sttRptWallToWallProgress.sql |  |
| sttSaveModel.sql |  |
| sttSearchByModelId.sql |  |
| sttSearchByName.sql |  |
| sttSearchByStartDate.sql |  |
| sttUpdateAdHocCount.sql |  |
| sttUpdateCountSheet.sql |  |
| sttUpdateCountSheetAdjust.sql |  |
| sttUpdateCountSheetRecount.sql |  |
|  |  |
|  |  |
|  |  |

# Acceptance

I hereby confirm that I have been fully informed of the document’s content and received training to understand how the detailed instructions are to be applied:

Name …………………………………………………………………………….

Job Title ………………………………………………………………………….

Signed ……………………………………………………………………………

Date ………………………………………………………………………………