

SGASortConveyor - Sorteringsverket


Sorteringsverket är en bana som finns på T2 och snart även på T3 för att putta ner paket i rätt korgar vid utlastningen.

Vi integrerar med SGA via TCP och integrationen i sig kommunicerar med NoMan för uthämtning av data.

 Git:

 <https://github.com/nowastelogistics/Nowaste.Integrations.SGASortConveyor> Connect your Github account

Octopus:

 [Octopus Deploy](#)

Meddelanden

Kommunikationen mellan SGA och vår egen integration sker via ett förbestämt textformat

▼ HREQ / HRES

Ett så kallat heartbeat som kollar att kopplingen fortfarande är aktiv.

SGA skickar en sträng med HREQ och vi svarar med HRES. Detta är enbart aktivt i prod och är inget man behöver tänka på

▼ DREQ/ DRES

Detta är den huvudsakligen kommunikationen som finns i prod idag.

När ett paket kommer fram till skannern på bältet skickar SGA en så kallad DREQ i formatet nedan

```
1 DREQ;DP01;83284;9836378;0.10Kg
2
3 DREQ = typen av meddelande
4 DP01 = Stationen där skanningen utfördes
5 83284 = TransaktionsId som vi skickar tillbaka i DRES
6 9836378 = SSCC på pallen som skannades
7 0.10Kg = Vikten från vågen vid skannern
```

Integrationen kommer svara med en destination vart paketet ska.

Detta görs via tabellen `SORT_CONVEYOR_MATRIX` och svaret kommer i formatet nedan

```
1 DRES;DP01;83284;11
2
3 DRES = typen av meddelande
4 DP01 = Stationen där skanningen utfördes
5 83284 = TransaktionsId från DREQ så SGA vet vilket meddelande vi svarar på
6 11 = Vilken utbana paketet ska puttats ner i.
7 Ibland finns där fler utbanor då skickas det som 1112 där bana 11 är högst prio
8 följt av bana 12
```

▼ EREP

Skickas som del av det nya SGA gränssnittet och meddelar att ett paket har kommit fram till en destination. Vi läser av detta meddelandet och matchar mot ett transaktionsId från en DREQ och loggar till `SORTING_LOG`

```
1 EREP;DP01;83284;11
```

```
2
3 EREP = typen av meddelande
4 DP01 = Stationen där skanningen utfördes
5 83284 = TransaktionsId, kommer matcha med Idet som kom med pallens DREQ
6 11 = Den faktiska lokationen på paketet efter eject
```

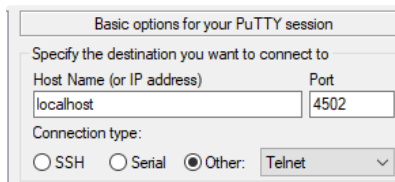
Debug och test 🐛

Lokal debug

Om du ska debugga SGA lokalt börjar du med att starta köra igång SgaSortConveyorService

Sedan öppnar du upp en ftp klient (jag använder putty) och kör följande

```
1 host name: localhost
2 Port: 4502
3 Connection type: Other:Telnet
```



Du kan nu skicka diverse kommandon till SGA integrationen för att träffa breakpoints eller läsa av svaren som kommer i kommandoprompten.

Vill du ta detta ett steg vidare och även debugga Noman så kan du gå in i

```
appsettings.Development.json och ändra BaseAddress från http://mi17-test-effect.everfresh.int/ till
http://localhost:8080 detta kräver att du även har startat Noman i debug
```

Avengers felsökning - Regler för SGA

- För att SSCC ska räknas som ett giltigt SSCC så behöver den innehålla 12 eller fler tecken.
- För att paketet ska behandlas som **B2C** så kontrolleras **CUSTOM.CUSTOM_GROUP = ECOM** or **EHANDEL**. Om det är B2C så måste paketet ha ett giltigt SSCC (Regel ovanför) annars rejectas paketet.
- För att paketet ska behandlas som **B2B** så får kunden inte tillhöra grupperna ECOM eller EHANDEL.
I detta fall så tillåter vi ett ogiltigt SSCC (Under 12 tecken)
- För att "Målet" ska bli korrekt behandlat i logiken så behöver värdet innehålla minst två tecken i sorteringsmatrisen, för satt skriva mål 1 så behöver man skriva 01, exempelvis.
- För att styra dimensionerna så behöver samtliga dimensioner vara ifyllda i Sorteringsmatrisen.
Om man önskar att styra paketen med detta värdet så behöver man utgå från palltypens dimensioner.
- Om paketen redan fått ett mål på transportbandet så kan det dröja innan de kan skickas på bandet igen, om det skulle krävas. Varje timme gör vi en hard flush av minnet och laddar om allt.
 - Varje minut gör den en "look ahead" och ser om där kommer något mer, eller om en transportbokning sker så läser man in det på "fast track"

Test mot Develop

Kommer snart