|  |
| --- |
| DAWA Replikerings klient  Brugervejledning |
| **Versionshistorie:**   |  |  |  |  | | --- | --- | --- | --- | | Version | Forfatter | Dato | Beskrivelse | | 1.00 | JALI | 2019-09-27 | Dokument oprettet | | 1.01 | VZ | 2019-10-18 | Rettelser | |  |  |  |  | |  |  |  |  | |
| Dato: 2019-09-27  Initialer: JALI |

Indholdsfortegnelse

[1 Oprettelse af database 3](#_Toc20488587)

[1.1 SQL Server version 3](#_Toc20488588)

[1.2 General og Options settings 3](#_Toc20488589)

[1.3 Bruger til databasen. 3](#_Toc20488590)

[2 Installation og første udtræk. 5](#_Toc20488591)

[2.1 Kopier til folder 5](#_Toc20488592)

[2.2 Opsætning af replikerings klient 5](#_Toc20488593)

[2.3 Opret tabeller - første gang eller ved database ændringer 5](#_Toc20488594)

[2.4 Lav et initialt udtræk 5](#_Toc20488595)

[2.5 Overvågning 5](#_Toc20488596)

[2.6 SQL-scripts til tabel og kolonne beskrivelse 6](#_Toc20488597)

[3 Programmets settings i produktion 6](#_Toc20488598)

[4 Opsætning af Windows Scheduler 6](#_Toc20488599)

[4.1 Dawa task i Windows Scheduler 6](#_Toc20488600)

[4.2 Dagi task i Windows Scheduler 10](#_Toc20488601)

[5 Udtræk beskrivelse 11](#_Toc20488602)

[5.1 Initialisering af Udtræk 11](#_Toc20488603)

[5.2 Udtræk beskrivelse 11](#_Toc20488604)

[6 Update beskrivelse 12](#_Toc20488605)

[6.1 Initialisering af Update 12](#_Toc20488606)

[6.2 Update beskrivelse 12](#_Toc20488607)

[7 DAGI-udtræk/update beskrivelse 12](#_Toc20488608)

[8 Drift 13](#_Toc20488609)

[8.1 How to handle problems 13](#_Toc20488610)

[8.2 Tabel entitystate – holds current status 13](#_Toc20488611)

[8.3 Fix bad incomming data – FixList.csv 13](#_Toc20488612)

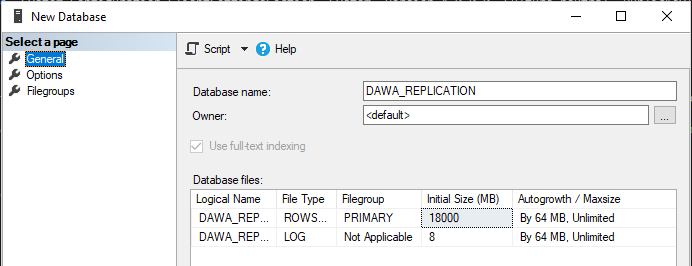
# Oprettelse af database

## SQL Server version

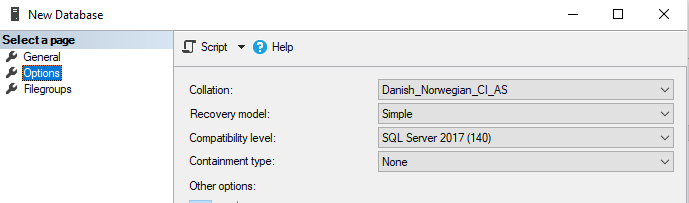
SQL Server version 12 med nyeste security opdates, eller senere version.

## General og Options settings

Opret en database med navnet DAWA\_REPLICATION med Initial Size på 20.000MB:

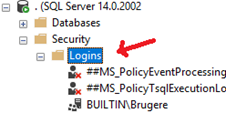


Vælg Collation Danish\_Norwegian\_CI\_AS og Recovery model simple:

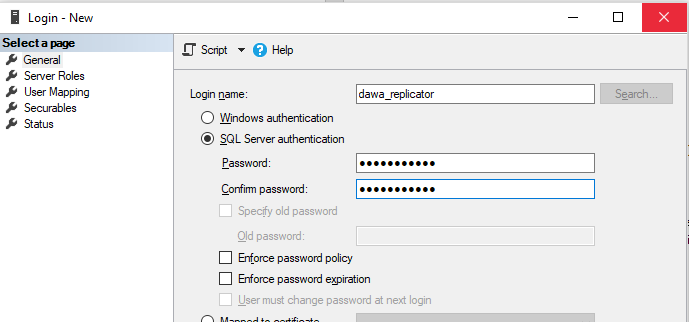


## Bruger til databasen.

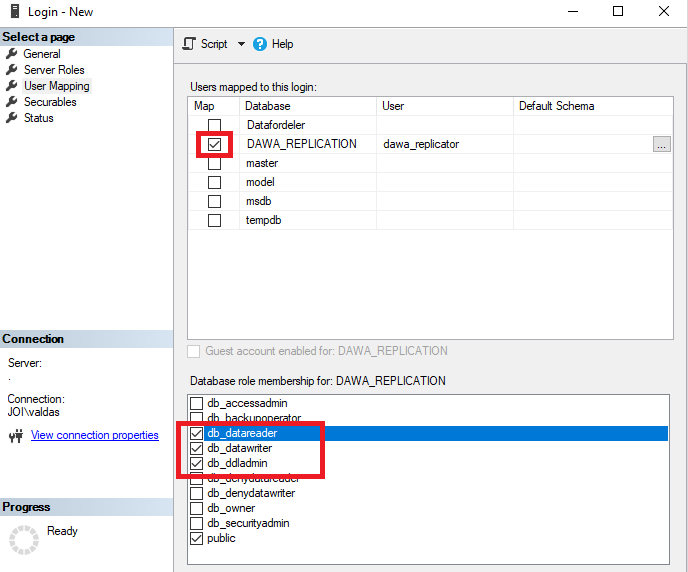
Opret bruger ”dawa\_replicator”, som kan læse fra og skrive til databasen, samt styre DDL niveauet af databasen.



Bemærk: Brugernavn og password benyttes i applikationens .config fil i connectionString.



Rettigheder sættes som i følgende:



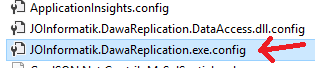
# Installation og første udtræk.

## Kopier til folder

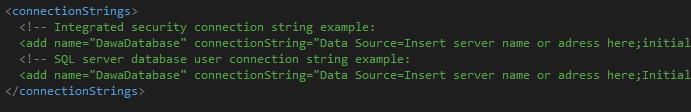
Kopier alle filer til folder Z:\DawaReplication.

## Opsætning af replikerings klient

Opsætning af programmet sker i fil JOInformatik.Dawareplication.exe.config.



I appconfig skal connectionString opdateres mht SQL Server (Data Source), database navn (Initial Catalog) og bruger (User Id og Password)



Gem ændringerne.

## Opret tabeller - første gang eller ved database ændringer

Kør programmet med ” /Update-Database” for at oprette tabeller, eller indføre tabel-ændringer:

Z:> JOInformatik.DawaReplication.exe /Update-Database

## Lav et initialt udtræk

Første gang programmet køres, skal der foretages et ”Udtraek”.

Start programmet med parameteret ”/Udtraek”, eller indtast i konsol vinduet ved start.

Kør programmet:

Z:> JOInformatik.DawaReplication.exe

Et komplet udtræk kan tage nogle timer, og programmet er sat og til at prøve igen med tabeller, der skulle fejle.

For at lave et udtræk af ”dagi\_\_” tabeller, skal progammet køres i Dagi mode:

Z:> JOInformatik.DawaReplication.exe Dagi

## SQL-scripts til tabel og kolonne beskrivelse

Kør medfølgende SQL-scripts, der tilføjer beskrivelser af tabeller og kolonner.

# Programmets settings til vedliholdelse af delta opdateringer

**VIGTIGT:** EntityProcessMode skal altid være ”Update” ved normal drift.

Opsætning af programmet sker i fil JOInformatik.Dawareplication.exe.config.

Når programmet kører i produktion, skal settings være som følgende:

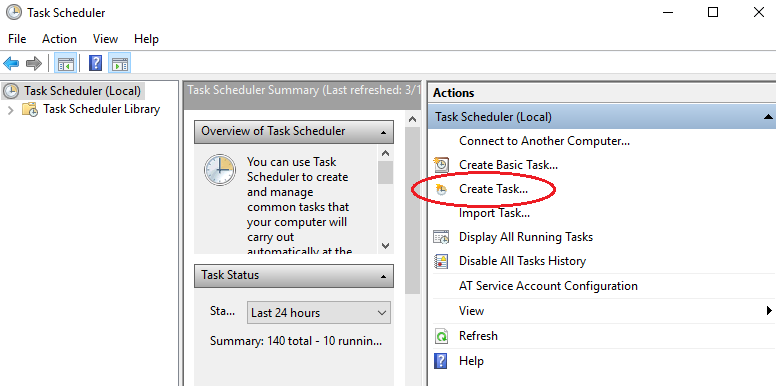
|  |  |  |  |
| --- | --- | --- | --- |
| Navn | Type | Værdi | Beskrivelse |
| DawaApiUri | String | <https://dawa.aws.dk/> |  |
| DawaApiReadTimeout | Int | 600 |  |
| WaitForUserInput | Bool | False | Benyt altid False. Ved True lukker lukke konsolprogrammet ikke. |
| TableInfoFile | String | ProdTableInfoList.csv | Tabeller med ”False”skippes dvs data ikke hentes fra Dawa. |
| TxidOverride | Int | 0 |  |
| EntityProcessMode | String | Update | Forældet, brug konsol parametre. |
| UdtraekRowsMax | Int | 0 | Benyttes kun ved udvikling |
| RetryTimerInMinutes | Int | 5 | Hvor lang tid der ventes i minutter før der prøves igen (ved fejl). |
| RetryCount | Int | 2 | Hvor mange gange der prøves igen ved fejl. |
| DKStedDataStartPos | Int | 96 |  |
| DKStedBulkSize | Int | 2500 | Størrelsen af Bulkinsert er sat til 2500 pga at der er flere tabeller, der indeholder store geometrier, hvilket kan medføre en lukket forbindelse fra DAWA. Skal helst være > 1000 |
| DBCommandTimeout | Int | 120 | Der kan forekomme timeouts ved større mængder data, som fx store geometri felter, så den burde være > 60 |

# Opsætning af Windows Scheduler

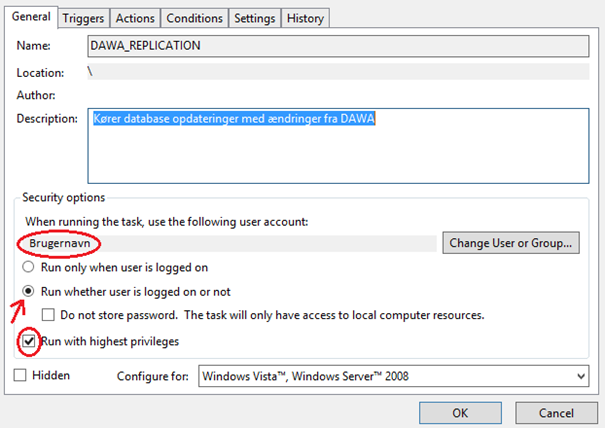
Der skal opsætte både for Dawa og Dagi.

## Dawa task i Windows Scheduler

1. Opret en lokal bruger Svc\_DawaReplication og gør brugeren til lokal administrator. Brugeren skal kunne lave scheduled tasks, afvikle Replikeringsklienten, oprette foldere og filer, køre programmer med net- og databaseadgang.
2. Start med at åbne Task Scheduler.
3. Replikeringsklienten kræver 2 tasks, hvor den første er en task til opdatering af DAWA-tabeller, som sker i løbet af dagen, on den anden er opdatering af DAGI-tabeller, som foregår om natten.
4. I actions menu vælg Create Task.

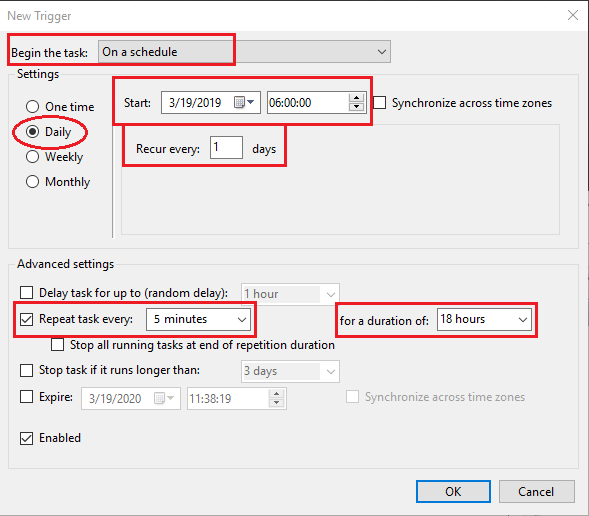


1. Udfyld med navn og passende beskrivelse.
2. Tjek at den rette bruger er valgt, og sæt den til at selvom brugeren er ikke logged på.
3. Vælg den rette Windows version.

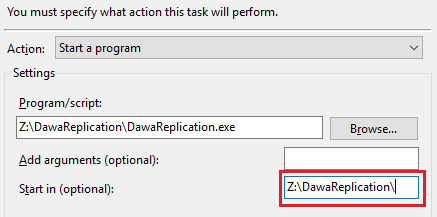


Det er en god idé at have den køre med highest privileges, så den kan skrive og gemme log filer.

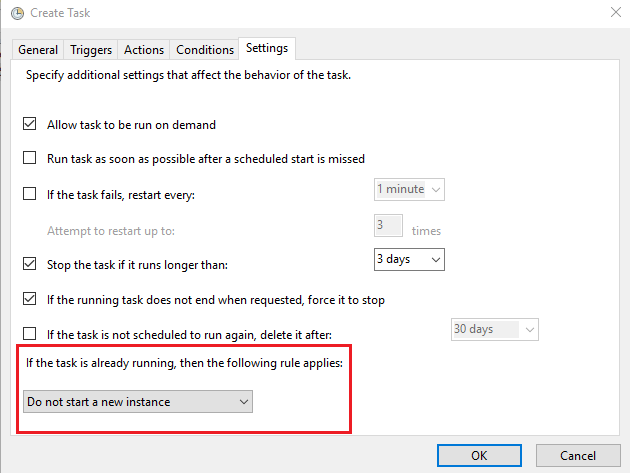
1. Opret en ny trigger, som:
2. Starter dagligt kl. 06:00:00. (Recur every: 1 days)
3. Kører hvert 5. minut i 18 timer.

****

1. Så opretter man en ny action, som skal starte et program med den angivet sti, og skal starte i samme mappe, som programmet ligger i.

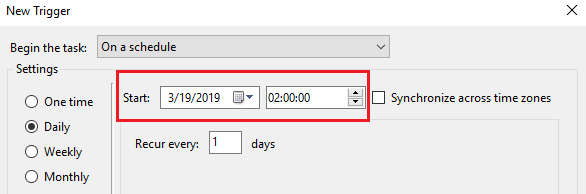
****

1. Herefter er det vigtigt i settings at vælge, at den ikke skal starte en ny instans af programmet, hvis den kører allerede.

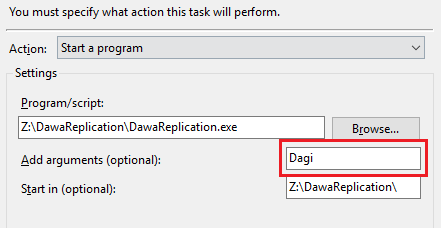
****

## Dagi task i Windows Scheduler

1. For at kunne opdatere DAGI tabeller, skal der oprettes en ny task med forskellige tidspunkter og et parameter som skal angives i starten af programmet.
2. Start med at lave ny task, hvor general, conditions, og settings er samme, som i Dawa task.
3. Lav en ny trigger, hvor programmet skal køre hver dag kl. 02:00:00.



1. Lav derefter en ny action, som peger på programmets placering, og tilføj ”Dagi” som parameter.



# Udtræk beskrivelse

## Initialisering af Udtræk

Første gang programmet køres, vil den tjekke om forbindelse til serveren, med den connection string, der står i App.config. Udtræk regnes som at være den allerførste operation der laves i forbindelse med databasen, derfor vil programmet opdatere databasen med den migration, der følger med.

**VIGTIGT:** Første ting der sker ved udtræk af en tabel, er at vi kalder **TRUNCATE table** for at være helt sikre på, at den ikke ligger noget i tabellen, som ikke burde være der.

## Udtræk beskrivelse

Når man laver et udtræk, så henter programmet sekventielt tabeller fra listen af tabeller i filen ProdTableInfoList.json – som indeholder navne for alle tabeller, samt rækkefølgen i hvilken de skal hentes, og en værdi der fortæller om de skal hentes, eller ej.

Der laves en forespørgsel ved et API kald til DAWA, hvor vi henter den seneste transaktionsId Fra <https://dawa.aws.dk/replikering/senestetransaktion> hvor vi finder ”txid”.

Ved udtræk starter man at finde typen ud fra tabellens navn, hvor man finder den tilhørende tabel C# klasse i DataAccess.

Dawa API kaldes med entitets navn, transaktionsid, som vi havde fundet tidligere, og en ”noformat” parameter. Et eksempel på et API kald:

<https://dawa.aws.dk/replikering/udtraek?entitet=jordstykke&txid=3348912&noformat>

Udtræk API kald leverer et stream af resultater, som vi behandler løbende ved at deserialisere dem fra JSON i forhold til vores C# entitets klasse, hvorefter vi kalder BulkInsert for at indsætte et antal af dem i databasen og tømme hukommelsen.

# Update beskrivelse

## Initialisering af Update

Update – eller delta-opdatering begynder med at hente den seneste transaktions id, ved at lave et kald til DAWA API med url: <https://dawa.aws.dk/replikering/senestetransaktion>, hvor den henter txid.

**INFO:** Update kører med transactions.

## Update beskrivelse

Update fungerer ligesom Udtræk, i og med, at den løber listen af tabeller, der er sat til aktiv i ProdTableInfoList.json filen. For hver aktiv tabel laver vi en forespørgsel hos DAWA om der er sket ændringer ved at kalde deres API. Til dette har vi brug for to ting. Vi har allerede seneste transaktions id fra DAWA – kaldet txidTil, altså den transaktions id som vi skal opdatere vores entitet til. Vi henter dermed transaktions id for entiteten fra databasen, for at finde ude af hvilket txid den er opdateret til, dette er vores txidFra.

Vi laver dermed et kald til DAWA for at se om der er sket ændringer imellem de to txid’er for entiteten.

<https://dawa.aws.dk/replikering/kændelser?entitet=jordstykke&txidfra=3348912&txidtil=3350000&noformat>

Vi får en liste af alle ændringer der er sket mellem disse to txid, dvs. vi får Inserts, Updates, og Deletes. Programmet deserialiserer disse til 3 lister, som vi sorterer, så kun en operation vil blive udført på en række. Vi kalder derefter BulkInsertOrUpdate på listen med rækker, der skal indsættes eller opdateres, og BulkdDelete på dem, der skal slettes.

# DAGI-udtræk/update beskrivelse

Udtræk og opdatering af ”dagi\_” tabeller foregår ligesom den almindelig udtræk, da lige præcis disse tabeller er ikke med i DAWAs replikerings API. Vi kalder fx: <https://dawa.aws.dk/steder?format=geojson&srid=25832&noformat>

For at hente steder med format geojson (for at inkludere geometri), og geometri format SRID 25832.

(NOTE: DAWA replikerings API er opdateret med dagi tabeller d. 03/5 2019, men med færre felter, end hvad vi kan få med Dagi Udtræk. I øjeblikket bliver begge versioner hentet).

Den eneste forskel mellem DAGI-udtræk og DAWA Replikering udtræk er, at DAGI data bliver leveret som et stort geojson objekt, og strukturen er lidt anderledes, derfor skal de behandles en smule anderledes.

Da, der ikke er nogen replikerings api for DAGI, bliver hele objektet hentet hver gang, derfor bliver hele tabellen opdateret hver gang metoden kører, typisk kl. 02 hver nat.

Når vi henter en dagi objekt, bliver den gemt i hukommelsen som et stort geojson variabel, før vi kan behandle hver række for sig selv. Dette skaber problemer med Steder og Stednavne tabellerne, da der er mange, og de indeholder geometri, hvilket gør dem for store til at gemme i hukommelsen. Derfor er vi nødt til at gemme Steder og Stednavne temp filer på harddisken, **åbne dem og læse forbi 96 tegn**, hvilket er headeren på geojson formated, som de leveres i. Derefter kan vi behandle dem en for en, og indsætte løbende, så vi ikke løber tør for hukommelse. Temp filerne bliver slettet når vi er færdige med at behandle data.

For at gøre det ensartet, gør vi det for alle ”dagi\_\_” entiteter.

# Drift

## How to handle problems

**Overview**

* DAWA data: Updated every 5. minute by Windows Scheduler (but not at night)
* DAGI data: Loaded every night (full load) by Windows Scheduler
* DAGI data is in tables prefixed with dagi\_\_Xxxx

## Tabel entitystate – holds current status

This table holds info about last DAWA udtræk and update, and also about last DAGI load.

When error (e.g. in ”Overvågning af webservices”) check tabel entities for any problem:

SELECT \* FROM entitystate WHERE success <> 1

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| entity | varchar | Entity name (aka tabelname) |
| txid | int | Latest transaction id for entity. Set to null if Udtraek failed |
| success | bit | Entity transaction state. Set to null when starting. Updated when finished with success (1) or failure (0) |
| Successtime | datetime | The time success was set to true |
| successtimeChange | datetime | The time there was an actual change (count <> 0) with success |
| Starttime | datetime | Transaction start time |
| Finishtime | datetime | Transaction finishing time. Set to null when starting and updated when finished |
| Count | int | Number of rows processed. Null if transaction didn't succeed |
| Message | varchar | Error message if any. Null if none |

## Fix bad incomming data – FixList.csv

Use file FixList.csv to fix data that cause problem when processing incoming data.

NOTE: Make this feature active in app.config by setting field ActiveFixes to true.

Remember to turn off after 10 minutes.

**Line example with Sogn (see table Sogn):**

Sogn has sometimes null in field geo\_version and the geo\_version databasecolumn is non-null:

Sogn;geo\_version;"geo\_version":null,;"geo\_version":0,;true

**Line format:** TableName;ColumnName;DataValue;FixedDataValue;Active

Semicolon is the separator.

* TableName: Sogn
* ColumnName: geo\_version
* DataValue: "geo\_version":null, (Note: Must include comma!)
* FixedDataValue: "geo\_version":0, (Note: Must include comma!)
* Active: true (Set to false to inactivate the line)