HAIBA

LPR-kontakter, data behandler

Design og Arkitektur

version 1.0

Indhold

1 Formål 3

2 Arkitekturoverblik 4

2.1 Fysisk arkitektur 5

2.2 Logisk Arkitektur 5

2.2.1 Regelhåndtering 5

3 Særlige forhold 7

3.1 Personhenførbare data 7

3.2 Ændring af tidsintervaller til databehandlingen 7

3.3 Ændring/tilføjelse af regler og datamodel 7

4 Fysiske Datamodeller 8

4.1 LPR Database 8

4.2 Indlæggelsesdatabase 8

5 Ændringslog 10

# Formål

Dette dokument giver et overblik over LPR Databehandleren med fokus på design og arkitektur.

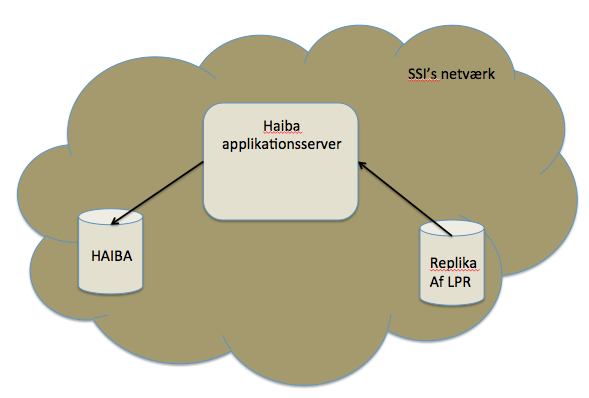
Dokumentet har som formål at give et indblik i det underliggende design, de udstillede snitflader på det overordnede niveau samt særlige forhold vedrørende LPR Databehandleren.

Både udviklere, aftagere samt driftsleverandør kan med fordel læse dette dokument.

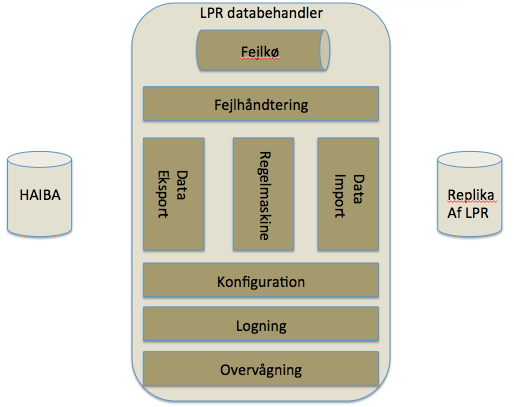
# Arkitekturoverblik

LPR databehandleren sørger for at importere data fra Landspatientregisteret (LPR), for så via nogle forretningsregler igen at eksportere data til HAIBA indlæggelsesdata.

Overordnet set består LPR Databehandleren af en applikation kørende på en JEE applikationsserver (I dette tilfælde Tomcat), som har adgang til to databaser; en med LPR kontakt data hvor der importeres fra, og en med HAIBA indlæggelses data hvor der eksporteres til. LPR databasen er et replika (og subset) af den LPR database sundhedsstyrelsen bruger til afregning af sygehusene.



LPR databehandler applikationen er et standard JEE web arkiv, der er bygget op på følgende måde:



* Data importeren har til ansvar at hente data ind fra LPR databasen, via batches af konfigurerbar størrelse, og placere det i en intern LPR model.

Data importeren er skeduleret til at køre med faste intervaller

* Regelmaskinen sørger for udførelse af de beskrevne forretningsregler i den rækkefølge som står i dette dokument, data bliver i dette step konverteret til en intern HAIBA model.
* Data eksporteren sørger for at skrive det behandlede data ned i HAIBA databasen ud fra den interne HAIBA model
* Konfigurationsmodulet sørger for opsætning af LPR databehandleren ud fra den eksterne konfiguration (eksempelvis batchstørrelser, tidsintervaller m.m.)
* Logningsmodulet sørger for applikationslog til debug og applikationsfejlhåndtering - ikke fejl i forbindelse med databehandling, dette sørger fejlhåndteringsmodulet for.
* Overvågningsmodulet sørger for at udstille en html side, hvor man kan se, eller maskinelt overvåge, LPR databehandlerens tilstand.
* Fejlhåndteringsmodulet har til ansvar at rapportere datafejl, enten fordi input format ikke kan bruges, eller hvis noget data falder uden for gældende regelsæt og derved ikke kan processeres. Fejlhåndteringsmodulet skriver til en fejlkø (Fil og databasetabel), hvor statistikere eller klinisk personale kan se hvad det er for noget data hvor behandlingen er fejlet, og hvad årsagen til fejlen er.

For at se hvordan LPR databehandleren er udviklet med hensyn til pakkestrukturer og klasser for de enkelte moduler henvises til udvikler guiden.

## Fysisk arkitektur

<TODO, beskriv den fysiske arkitektur (war filer, afhængigheder, konfiguration og overordnede principper såsom JNDI til databasehåndtering) - lav tegning af den fysiske arkitektur

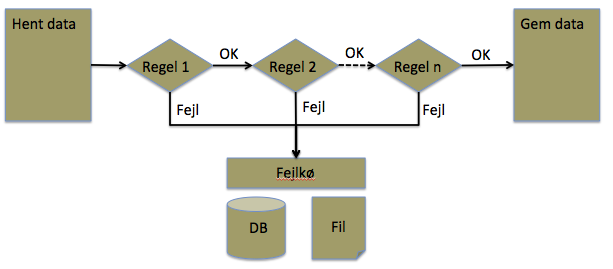
## Logisk Arkitektur

<TODO beskriv den logiske arkitektur, og hvordan hvert underelement er designet - understøt dette med tegninger>

### Regelhåndtering

LPR Databehandleren sender LPR data gennem en række forretningsregler, som beskrevet i løsningsspecifikationen (afsnit 2.4).

Regelhåndteringen er meget simpel, da reglerne skal udføres i en bestemt rækkefølge, og de alle bliver afbrudt på samme måde i tilfælde af fejl i databehandlingen.

Som det ses ud fra figuren bliver data hentet ind og sendt til den 1. regel, går det godt sende data til 2. regel o.s.v. I alle tilfælde hvis der sker en fejl i databehandlingen vil flowet blive afbrudt og fejl detaljer vil blive sendt til fejlhåndteringsmodulet som sørger for at skrive det til en log-fil, og alternativt en tabel i databasen, hvis dette er konfigureret.

For at implementere dette flow blev Spring integration overvejet, Spring integration er godt til at holde styr på løst koblede komponenter og hvilket beskedflow der skal være mellem dem, samt håndtering af eksterne triggere såsom services, hændelser og lignede der kan igangsætte flowet.

Dog er der i denne prototype ikke behov for hverken avanceret routing, eksterne triggere (der "polles" efter data i LPR) eller afkobling, derfor er det mere optimalt at bygge flowet som enkeltstående klasser, der sørger for at behandle data og sende det videre til næste forretningsregel. Dette vil umiddelbart også gøre fejlfinding nemmere da man kan koble en debugger til og følge dataflowet gennem denne, i stedet for at følge beskeder/hændelser som Spring Integration understøtter.

Skulle der efter prototype fasen (der slutter 1/4-2013) vise sig at være et mere avanceret behov for routing, afkobling eller eksterne trigger, kan forretningslogikken fra ovenstående regler genbruges da de er lavet som enkeltstående klasser, det som skal til er at Spring Integration konfigurationen sættes op.

# Særlige forhold

## Personhenførbare data

Data fra LPR indeholder personnumre, og derfor personhenførbare. Dette personnummer bæres igennem regelbehandling og efterfølgende gemt i HAIBA indlæggelsesdatabasen.

Personnumre vil aldrig blive skrevet i logfiler, alt data der logges er referencenumre fra LPR (Record nummer), som gør at personer med relevant viden og rettigheder kan slå op i LPR og se data og derved sammenholde det med fejlbeskeden.

## Ændring af tidsintervaller til databehandlingen

<TODO beskriv særlige forhold, såsom personhenførbare data, betydning af ændringer i datamode, regler m.v.>

## Ændring/tilføjelse af regler og datamodel

Skal der laves ændringer eller tilføjelser til forretningsreglerne, skal disse kodes og testes først. Der vil blive lavet konfigurationsparametre hvor det giver mening (såsom justering af tidsintervaller o.l.), men disse kan ikke ændre på det grundlæggende som forretningsreglen implementerer.

Ligeledes hvis der er ændringer til datamodellen, så skal dette også kodes, da den interne model i LPR databehandleren er bundet hårdt op på datamodellen, det er et bevidst valg at der ikke er indført et konfigurerbart afkoblingslag mellem databasen og koden, f.eks. via Hibernate, da dette erfaringsmæssigt giver et stort overhead, og indfører en kompleksitet der projektet mere omkostningstungt, både under udvikling, men også under senere drift.

Dog er der lavet en række views som sammenholder indlæggelsesdata med data fra FGR (Fælles Grunddata), i sådanne tilfælde kan det være muligt man kan nøjes med at justere viewet.

# Fysiske Datamodeller

LPR Databasen indeholder 3 tabeller med data, hvor T\_ADM er den overordnede tabel med de administrative kontakt data.

HAIBA indlæggelsesdatabasen er de behandlede data, hvor kontakter er gjort til indlæggelsesforløb, Indlaeggelsesforloeb tabellen er den overordnede tabel med de behandlede forløb. HAIBA databasen indeholder desuden en række views, som er sat op for at gøre det nemmere for den efterfølgende databehandling via eksempelvis SAS.

Detaljer omkring feltværdier, datatyper og lignende kan ses i SQL skemaerne som ligger på følgende URL: <https://github.com/trifork/HAIBA-LPRimporter/tree/master/database>

## LPR Database

|  |  |
| --- | --- |
| Tabel | Beskrivelse |
| T\_ADM | Indeholder data for den administrative del af en kontakt, heriblandt personnummer, indlæggelses og udskrivningstidspunkt. |
| T\_DIAG | Indeholder diagnose typer og koder |
| T\_PROCEDURER | Indeholder data om undersøgelser og operationer, bl.a. tidspunkter og koder |

## Indlæggelsesdatabase

|  |  |
| --- | --- |
| Tabel | Beskrivelse |
| Indlaeggelser | Indeholder processeret data fra T\_ADM tabellen, dette er beriget med metadata såsom navne på sygehuse, afdelinger m.v. |
| Diagnoser | Indeholder processeret data fra T\_DIAG tabellen, ligeledes beriget med metadata såsom navne på diagnoser. |
| Procedurer | Indeholder processeret data fra T\_PROCEDURER tabellen, ligeledes beriget med metadata såsom navne på operationer og undersøgelser. |
| Indlaeggelsesforloeb | Referencetabel, der viser sammenhængen mellem de enkelte indlæggelser i.h.t. forretningsreglerne. |
| LPR\_reference | Referencetabel, der viser sammenhængen tilbage til kontakterne i LPR T\_ADM tabellen. Bl.a. hvis der er kontakter der ikke er gyldige i f.t. forretningsreglerne. |
| Regelfejlbeskeder | Tabel til de fejlbeskeder der kommer fra behandlingen af LPR data, data er en kopi af de data fejlhåndteringsmodulet logger. |

Views:

Der er lavet en række views, som vil gøre det lettere for business intelligence programmer som SAS at hente data, disse er lavet som views for ikke at have data kopieret rundt flere gange i tabellerne.

|  |  |
| --- | --- |
| View | Beskrivelse |
| IndlaeggelsesForloebsOversigt | Indeholder et overblik of et indlæggelsesforløb med første indlæggelsesdata og sidste udskrivningsdata samt personens CPR nummer. |
| <TODO, tilføj disse når dokument kommer fra Kenn> |  |

# Ændringslog

Kilden til dette dokument kan findes på:

[https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Design og Arkitektur.docx](https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Design%20og%20Arkitektur.docx)

| Version | Dato | Ændring | Ansvarlig |
| --- | --- | --- | --- |
| 1.0 | 2013-01-03 | Initielt dokument | Trifork  Kjeld Froberg |
|  |  |  |  |