HAIBA

LPR-kontakter, data behandler

Drifts og installationsvejledning

version 1.0

Indholdsfortegnelse

1 Formål 3

2 HAIBA LPR-databehandler 4

2.1 LPR-databehandler komponenter 4

2.1.1 LPR-databehandler 4

2.1.2 LPR database 4

2.1.3 Indlæggelses database 4

2.1.4 Applikationsserver 5

3 Installation 6

3.1 Krav til driftsmiljø 6

3.1.1 Krav til applikationsservere 6

3.1.2 Krav til operativsystem 6

3.1.3 Krav til database 6

3.1.4 Krav til hardware 6

3.2 Installation af LPR Databehandler 6

3.2.1 Installation af Tomcat Applikationsserver 6

3.2.2 Konfiguration af tomcat adgang til database 7

3.2.3 Konfiguration af LPR databehandler 8

3.2.4 Installation af MS SQLServer 8

3.2.5 Installation af LPR database 8

3.2.6 Installation af Indlæggelsesdatabase 9

3.2.7 Installation af LPR Datehandler WAR fil 9

3.3 Opdatering til nye versioner 9

4 Daglig Drift 10

4.1 LPR Databehandler 10

4.1.1 Fremgangsmåde for indlæsning af nye data 10

4.1.2 Konfiguration af LPR Databehandler 10

4.2 Databaser 13

4.2.1 Skemafiler databasen 13

4.2.2 Databaseopsætning 13

5 Backup 14

6 Overvågning 15

6.1 Statusside 15

6.2 Logning 15

6.2.1 Fejlsøgning 15

7 Fejlhåndtering fra forretningsregler 17

7.1 Fejlbeskrivelse 17

7.1.1 Logfiler og logtabel 18

8 Ændringslog 19

# Formål

Vejledning til drift, installation og konfiguration af LPR Databehandler.

Dokumentet er målrettet systemadministratorer og driftspersoner, som skal kunne håndtere driftsmæssige aspekter af komponenten.

Drifts og installationsvejledningen indeholder information om komponentens version, standard placering af logfiler og konfigurationsfiler, eksterne afhængigheder, og evt. krav til genstart af applikationen hvis komponenten bliver ikke-responsiv. Ligeledes indeholder vejledningen en installationsvejledning, som bruges ved inintiel deployment, samt afsnit til efterfølgende delvis installation, eksempelvis når der sker opdateringer.

Kendte fejlkoder som skrives i logfiler er dokumenteret, så disse evt. kan overvåges, og tillige danne baggrund for fejlsøgning. En generel læsevejledning til logfiler vedlægges.

Det er angivet hvorledes komponenten bedst lader sig overvåge, dvs. en generisk beskrivelse af overvågningen, der ikke er værktøjsafhængig.

Evt. specielle krav til backup beskrives, ligesom procedure ved reetablering af komponenten ud fra backup er beskrevet.

# HAIBA LPR-databehandler

Dette dokument omfatter driften af HAIBA LPR-databehandleren

Listen herunder beskriver hver del komponent med type, evt. status URL og filnavne. Status URL’en kan løbende polles for at checke komponentens status.

## LPR-databehandler komponenter

### LPR-databehandler

LPR Databehandleren ligger i sin egen WAR fil, den har sin egen overvågnings URL, der enten fortæller om den er operationsdygtig (HTTP 200 OK), eller om der er fejl i den (HTTP 500 ERROR), Overvågningssiden vil give et bud på hvad fejlen er, dog bør man kigge i log-filen for at få alle detaljer med.

* Type: Batch
* Status Url: http://<hostname>:<port>/lpr-importer/status
* Filnavn: lpr-importer-<version ex. 1.0>.war

### LPR database

**Test:**

Databaseserver ligger på host: SRV-SQL-TEST01

Databasen hedder: HAIBA\_LPR\_REPLIKA

Brugernavn: haiba\_app

Password: Indskrives ikke her

**Produktion:**

Databaseserver ligger på host: SQLSERVER02

Databasen hedder: HAIBA\_LPR\_REPLIKA

Brugernavn: haiba\_app

Password: Indskrives ikke her

### Indlæggelses database

**Test:**

Databaseserver ligger på host: SRV-SQL-TEST01

Databasen hedder: HAIBA

Brugernavn: haiba\_app

Password: Indskrives ikke her

**Produktion:**

Databaseserver ligger på host: SQLSERVER02

Databasen hedder: HAIBA

Brugernavn: haiba\_app

Password: Indskrives ikke her

### Applikationsserver

**Test:**

Ligger på serveren: EPIMIBAAPPSRV

Port: 8080

**Produktion:**

Endnu ikke etableret

# Installation

## Krav til driftsmiljø

### Krav til applikationsservere

LPR Databehandleren er udviklet og testet på en TOMCAT 7 version 35 (64 bit).

Komponenterne kræver Oracle Java 6.0 eller højere.

Desuden skal MSSQL eller MySQL JDBC Connector Driver bruges.

### Krav til operativsystem

Der stilles ingen krav til operativsystemet, ud over det åbenlyse krav om at Java er understøttet på operativsystemet. Det forventes at operativsystemet er Microsoft Windows Server 2003 med servicepack 2 installeret

### Krav til database

Komponenten er testet mod MS SQLServer 2008 Og skal som minimum køre mod denne version

### Krav til hardware

Der er nogle minimumskrav for at kunne afvikle komponenten fornuftigt til test formål. Dog skal man forvente at bruge high-end hardware (både cpu, ram, netkort & diske).

Minimumskravene, for fornuftig performance på et test-setup er:

* Intel Core 2 eller lignende CPU
* 8 GB ram
* Nødvendig harddisk plads for at kunne håndtere LPR og indlæggelses databaserne (100+ GB)

## Installation af LPR Databehandler

LPR Databehandleren består af følgende komponenter:

1. Tomcat Applikationsserver
2. MS SQLServer
3. LPR database
4. Indlæggelses database
5. LPR Databehandler war fil

### Installation af Tomcat Applikationsserver

Tomcat kræver Java for at køre. så først skal den installeres.

Hent Java 6 version 38 (64 bit version, da det skal installeres på en Windows Server 2008), på følgende link: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

bemærk det skal være JDK og ikke JRE.

Log ind som administrator og installer Java

Da det er en 64 bit version af windows og der skal køres java på den, mangler der sikkert en version af Visual C runtime biblioteker. Disse kan hentes på følgende link: <http://www.microsoft.com/en-us/download/details.aspx?id=5555> og skal ligeledes installeres.

Hent Tomcat 7 version 35 (64 bit version) på følgende link:

<http://tomcat.apache.org/download-70.cgi>

Log ind som administrator og installer tomcat.

Når Tomcat spørger hvilken port den skal køre på, så se om 8080 er brugt v.h.a "netstat -a" kommandoen, alternativt vælg en anden port (eksempelvis 9090)

Vælg at installere tomcat som en windows service.

Test at tomcat er kørende ved at spørge på <http://localhost:8080> i en browser (eller brug alternativ port hvis 8080 var optaget).

For at Tomcat kan kommunikere med en Microsoft SQL server, skal der installeres en driver til dette. Den kan hentes på følgende link: <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>

Pak installationsprogrammet ud og tag filen sqljdbc4.jar og kopier den til <tomcat installation>/lib Tomcat skal efterfølgende genstartes. Bemærk dette behøves ikke hvis MySQL bruges som database

For at Tomcat kan kommunikere med MySQL, skal der installeres en driver til dette. Den kan hentes på følgende link: <http://dev.mysql.com/downloads/connector/j/>

Tag filen mysql-connector-java-5.1.22-bin.jar og kopier den til <tomcat installation>/lib Tomcat skal efterfølgende genstartes. Bemærk dette behøves ikke hvis Microsoft SQL Server bruges som database

### Konfiguration af tomcat adgang til database

Datasource til LPR og Indlæggelsesdatabaserne:

Indsæt reference til datakilderne i Tomcats context <tomcat installation>/conf/context.xml Dette gøres ved at tilpasse følgende:

Microsoft SQL:

<Resource name="jdbc/LPRDB" auth="Container" type="javax.sql.DataSource" maxActive="5" maxIdle="5" maxWait="10000" username="<bruger>" password="<password>" driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver" url="jdbc:sqlserver://localhost:1433;databaseName=LPR"/>

<Resource name="jdbc/HAIBADB" auth="Container" type="javax.sql.DataSource" maxActive="5" maxIdle="5" maxWait="10000" username="<bruger>" password="<password>" driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver" url=" jdbc:sqlserver://localhost:1433;databaseName=HAIBA"/>

MySQL:

<Resource name="jdbc/LPRDB" auth="Container" type="javax.sql.DataSource" maxActive="5" maxIdle="5" maxWait="10000" username="<bruger>" password="<password>" driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/LPR"/>

<Resource name="jdbc/HAIBADB" auth="Container" type="javax.sql.DataSource" maxActive="5" maxIdle="5" maxWait="10000" username="<bruger>" password="<password>" driverClassName="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost:3306/HAIBA"/>

Indsæt databasernes brugernavn ("username") og adgangskode ("password") i ovenstående ressourcer.

### Konfiguration af LPR databehandler

LPR databehandleren har indbygget nogle default konfigurationsindstillinger, disse kan dog overstyres så de passer ind i det miljø den er installeret på ved hjælp af filen config.properties.

Driftsvejledningen beskriver hvilke parametre der kan stå i config.properties.

* Opret en folder der hedder "shared\_config" under <tomcat installationsdir>
* Åbn filen <tomcat installationsdir>/catalina.properties
  + Find hvor der står "shared loader" og tilføj så der står følgende:

shared.loader=${catalina.base}/shared\_config

* Kopier filen "config.properties" til shared\_config folderen
* Genstart tomcat.

### Installation af MS SQLServer

Microsoft SQL server er præ-installeret på SSI's miljø, så denne vejledning dækker ikke dette.

### Installation af LPR database

**Test:**

Skema til LPR databasen ligger på følgende link:

<https://github.com/trifork/HAIBA-LPRimporter/tree/master/database> der ligger både en version til MySQL og til MSSQL

**Produktion:**

LPR databasen er præinstalleret

### Installation af Indlæggelsesdatabase

**Test og Produktion:**

Skema til HAIBA Indlæggelses databasen ligger på følgende link:

<https://github.com/trifork/HAIBA-LPRimporter/tree/master/database> der ligger både en version til MySQL og til MSSQL

### Installation af LPR Datehandler WAR fil

LPR Databehandler applikationen ligger i et JEE web arkiv (war fil), for at installere den i tomcat serveren skal den kopieres til <Tomcat installationsdir>/webapps -efterfølgende vil tomcat selv sørge for at klargøre (deploye) filen.

<TODO hvor skal war filen ligges hen? - CD, filsystem eller lign?>

## Opdatering til nye versioner

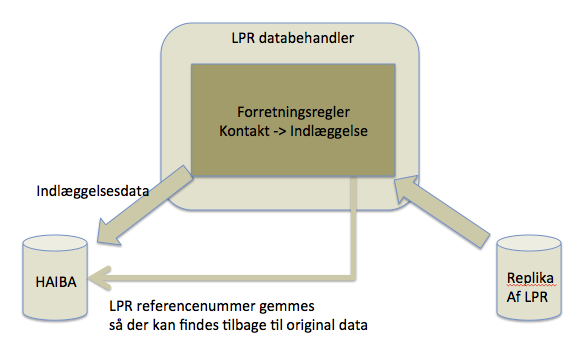
Når nye versioner af LPR Databehandleren udkommer, vil der medfølge release notes som forklarer database-migrering, rollback-procedure, service vinduer mv.

Til installation af første version, og eventuel geninstallation, af LPR Databehandleren henvises til installationsvejledningen.

# Daglig Drift

## LPR Databehandler

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



LPR databehandleren består af en Tomcat applikationsserver indeholdende en war fil som er selve applikationen. Desuden er der eksterne afhængigheder til en LPR database og en HAIBA database

### Fremgangsmåde for indlæsning af nye data

LPR databehandleren står med jævne mellemrum og kigger i LPR databasen for at se om der er kommet nye data, dette interval kan sættes op via konfigurationsfilen. Opdager den nye data vil den indlæse data og køre det igennem forretningsreglerne. Skulle der ske en databehandlingsfejl, vil denne fejl blive skrevet ned i en fejl-kø (fil), som efterfølgende skal behandles af en dataansvarlig.

Bliver data succesfuldt behandlet, vil LPR databasen blive opdateret med et tidsstempel.

### Konfiguration af LPR Databehandler

LPR databehandleren har en default konfigurationsfil (default-config.properties) der er indlejret i war filen, de enkelte konfigurationsindstillinger kan overstyres ved af en miljø specifik konfigurationsfil (config.properties), der ligger i filsystemet på følgende lokation: <Tomcat installationsdir>/shared\_config

**Properties**

|  |  |
| --- | --- |
| **cron.import.job** | Spring scheduler cron konfiguration. Default ”0 0 1 \* \* \*”, hvilket svarer til en kørsel kl 0100 hver morgen. Se eks. http://javahunter.wordpress.com/2011/05/05/cronscheduler-in-spring/ |
| **currentpatient.default.outdate.calculated.interval.from.indate** | Regel 4c) Dato-tid (heldøgn og ambulant)  Grænseværdi for indskrivningsdato. Default 30 (dage) |
| **currentpatient.default.outdate.days.after.indate** | Regel 4c) Dato-tid (heldøgn og ambulant)  Kunstig fremregningsperiode for udskrivningsdato, hvis patienten endnu ikke er udskrevet og indskrivningsdato er udenfor currentpatient.default.outdate.calculated.interval.from.indate. Default er 30 (dage) |
| **currentpatient.default.outdate.hours.after.indate** | Regel 4c) Dato-tid (heldøgn og ambulant)  Kunstig fremregningsperiode for udskrivningsdato, hvis patienten endnu ikke er udskrevet og indskrivningsdato er indenfor currentpatient.default.outdate.calculated.interval.from.indate.  Default 24 (timer) fra indskrivningsdøgnets start |
| **default.contact.in.hour** | Regel 4a) Dato-tid (heldøgn og ambulant)  Indskrivningstime på døgnet, hvis 0  Default 0 |
| **default.contact.outhours** | Regel 4b) Dato-tid (heldøgn og ambulant)  Udskrivningstime på døgnet, hvis 0 og ikke samme dag som indskrivning  Default 12 |
| **default.contact.outhours.added.inhours** | Regel 4b) Dato-tid (heldøgn og ambulant)  Periode tilføjet indskrivningstime til udskrivningstime på døgnet, hvis 0 og samme dag  Default 1 |
| **default.contact.procedure.outhours** | Regel 4d) Dato-tid (heldøgn og ambulant)  Preceduretime på døgnet, hvis 0  Default 12 |
| **disable.database.errorlog** | parameter der fortæller om fejl fra forretningsregler også skal logges i databasen. default værdi er *false,* hvilket betyder at der også logges til databasen, sæt denne til *true* for at slå logningen til databasen fra (der logges dog stadig til fil uanset hvad.) |
| **hours.between.contacts.different.hospital** | Max. antallet af timer mellem LPR kontakter for samme person og fra forskellige hospitaler før det skal betragtes som en indlæggelse. Default værdien er 10 - **Bemærk:** ændres der på denne værdi, bør man genbehandle alt data. |
| **hours.between.contacts.same.hospital** | Max. antallet af timer mellem LPR kontakter for samme person og fra samme hospital før det skal betragtes som en indlæggelse. Default værdien er 4 - **Bemærk:** ændres der på denne værdi, bør man genbehandle alt data. |
| **jdbc.classificationJNDIName** | JNDI navn for datasourcen til klassifikationstabellerne (Klass\_SHAK osv.), der er konfigureret i Tomcat serveren, defaulter til tom streng. |
| **jdbc.classificationtableprefix** | Prefiks til klassifikationstabeller, Klass\_SHAK etc. Defaulter til tom streng. |
| **jdbc.dialect** | Hvilken type database LPR-importeren bruger, værdier er "MySQL" (default) eller "MSSQL" |
| **jdbc.fgrtableprefix** | Prefiks til fgr-importtabeller, Organisation etc. Her hentes sygehusprefix for 3800-sygehuse. Defaulter til ”fgr.” |
| **jdbc.haibaetlprefix** | Prefiks til lpr etl-tabeller (T\_LOG\_SYNC mm.). Defaulter til ”haiba\_etl.” |
| **jdbc.haibaJNDIName** | JNDI navn for datasourcen til HAIBA, der er konfigureret i Tomcat serveren, default: java:comp/env/jdbc/HAIBADB |
| **jdbc.haibareplicatableprefix** | Prefiks til lpr replica-tabeller (T\_ADM mm.) for at kunne styre mssql-schema. Defaulter til ”haiba\_replica.” |
| **jdbc.haibatableprefix** | Prefiks til haiba-tabellerne, eks. Indlaeggelser, Indlaeggelsesforloeb osv. Defaulter til tom streng. |
| **jdbc.lprJNDIName** | JNDI navn for datasourcen til LPR, der er konfigureret i Tomcat serveren, default: java:comp/env/jdbc/LPRDB |
| **jdbc.lprJNDIName\_minipas** | JNDI navn for datasourcen til MINIPAS, der er konfigureret i Tomcat serveren, defaulter til tom streng |
| **jdbc.minipas\_haibaetlprefix** | Prefiks til minipas etl-tabeller (T\_LOG\_SYNC mm.). Defaulter til ”haiba\_etl.” |
| **jdbc.minipas\_haibareplicatableprefix** | Prefiks til minipas replica-tabeller (T\_ADM mm.) for at kunne styre mssql-schema. Defaulter til ”haiba\_replica.” |
| **lpr.cpr.batchsize** | Antal cprnumre hentet i hver regelgennemgang, default 100 |
| **max.days.between.runs** | Antal dage mellem kørsler. Hvis perioden siden sidste kørsel overskrider denne værdi, angives det på statussiden. Default 1 |
| **smtp.auth** | Angiver om authentication skal benyttes på mailserver, default false |
| **smtp.from** | Afsender på emails (skal overholde formen på en emailadresse), default haiba@ssi.dk |
| **smtp.host** | Hostnavn eller ipadresse på mailserver. Default er ”relay.ssi.ad” |
| **smtp.password** | Password til smtpserver, kun nødvendig hvis smtp.auth er true |
| **smtp.port** | Portnummer til smtp på mailserver. Default er 25 |
| **smtp.sendhello** | Angiver om emailmodtagere skal modtage emails om start og slut (inklusive evt. fejl) på kørsel, default false |
| **smtp.starttls** | Angiver om tls skal benyttes på mailserver, default false |
| **smtp.to\_commaseparated** | Kommasepareret liste af emailmodtagere. |
| **smtp.user** | Brugernavn til smtpserver, kun nødvendig hvis smtp.auth er true |

## Databaser

LPR Databehandleren er testet på en MSSQL 2008 v2 og en MySQL 5.5 database, sidstnævnte er brugt til udvikling.

### Skemafiler databasen

SQL skemaer ligger på Github sammen med kildekoden, man skal altid referere til den gældende version af LPR-importeren:

<https://github.com/trifork/HAIBA-LPRimporter/tree/master/database>

Her ligger både skemaer for Microsoft SQLserver og for MySQL

### Databaseopsætning

der er ingen specielle indstillinger til databasen.

# Backup

LPR databehandlere kan reetableres ud fra installations og driftsvejledning i tilfælde af nedbrud.

Man skal dog i tilfælde af ændringer til konfigurationsindstiller sørge for at de bliver noteret ned i dokumenterne således det er nemt at re-etablere systemet.

Data kan re-etableres ved at genbehandle LPR data, men da der er meget data der skal behandles kan dette tage lang tid <TODO, når der er lavet en måling give et estimat hvor lang tid det vil tage> derfor kan det være en god ide at tage backup af HAIBA databasen, sammen med det bør der også tages backup af LPR T\_ADM tabellen, da databehandlere indsætter et tidsstempel for hvornår data er behandlet.

<TODO, henvis til drifstafdelingens procedurer for backup af data.>

# Overvågning

## Statusside

LPR databehandleren har en status-side som periodisk kan kaldes for at tjekke om servicen kører.

Statussiden vil overvåge om komponenten kører, samt om der er forbindelse til databaserne, og så vidt muligt skrive hvad en eventuel fejl er.

Status siden fungerer over HTTP, og har følgende statuskoder:

|  |  |
| --- | --- |
| 200 | Alt er OK. |
| 404 | Siden ikke fundet, er serveren startet? |
| 500 | Der er opstået en fejl, og driften bør undersøge komponentens log for fejlmeddelelser. Kan fejlen ikke opklares simpelt, bør driften kontakte support. |

URL’s for status sider kan findes tidligere i dette dokument.

## Logning

Logfilen er konfigureret via log4j properties filen, der ligger i <Tomcat installationsdir>/appconfig

Log appenderen hedder "HAIBALPR", det er denne appender LPR Databehandleren skriver information til. Log appenderen er sat op som en rullende fil appender, så der er styr på hvor meget diskplads der maksimalt bruges, og den skriver til en fil der hedder "haibalpr.log". hver gang den maksimale filstørrelse nås bliver der således lavet en ny fil " haibalpr.log.1" o.s.v indtil det maksimale antal filer bliver nået, derefter slettes den ældste fil hver gang en ny skrives.

Default opsætning ses her:

log4j.logger.dk.nsi.haiba.lprimporter.importer.ImportExecutor=INFO, HAIBALPR

log4j.additivity.dk.nsi.haiba.lprimporter.importer.ImportExecutor=false

log4j.appender.HAIBALPR =org.apache.log4j.RollingFileAppender

log4j.appender.HAIBALPR.File=haibalpr.log

log4j.appender.HAIBALPR.Append=true

log4j.appender.HAIBALPR.MaxFileSize=50MB

log4j.appender.HAIBALPR.MaxBackupIndex=10

log4j.appender.HAIBALPR.layout=org.apache.log4j.PatternLayout

log4j.appender.HAIBALPR.layout.ConversionPattern=%d [%-2p] %c - %m%n

### Fejlsøgning

Opstår der en fejlsituation i komponenten, skal driften undersøge logfilen for loghændelser på ERROR-niveau. F.eks. i tilfælde af at komponenten ikke kan forbinde til databasen. Visse andre fejl er ikke-kritiske. Det vil sige at komponenten kan forsætte med at fungere. De bliver også logget på ERROR-niveau da disse hændelser bør undersøges. Komponenten vil i så vid udstrækning som muligt forsøge at forsætte på trods af fejl.

# Fejlhåndtering fra forretningsregler

LPR databehandleren sørger for at data fra LPR bliver sendt igennem en række forretningsregler (disse er beskrevet i løsningsbeskrivelsen)m som sørger for at tilrette data så det passer ind i HAIBA indlæggelsesmodellen.

Men der kan være data som på den ene eller anden måde falder uden for disse regler, derfor er der lavet et fejlhåndteringsmodul det kan håndtere dette.

Fejlhåndteringen sørger for at logge disse fejl til både en tabel og til log-filer.

Overvågningsmodulet kigger ikke efter fejl i regelbehandlingen, det er en manuel proces at holde øje med dette.

det er ligeledes en manuel proces at finde ud af hvad der skal ske med det data som fejler i regelbehandlingen, f.eks. om det skyldes en fejl i kode eller en fejl i data.

Hvis der opstår en fejl som følge af databehandlingen, vil alle kontakter tilknyttet det CPR-nummer som fejlen opstod på ikke blive behandlet.

## Fejlbeskrivelse

Tabel layout er:



Log fil layout er:

<Datotidsstempel> [INFO] BusinessRulesErrors - LPR Recordnummer [?], Regel [?], fejlbesked [?], db [?]

Hvor de enkelte felter betyder følgende:

|  |  |
| --- | --- |
| **Felt** | **Beskrivelse** |
| LPR\_dbid | Id for database-kilden til kontakten. 1=offentlige institutioner (oprindelig LPR), 2=MINIPAS |
| LPR recordnummer | Referencen for kontakten i LPR |
| AfbrudtForretningsregel | Forretningsreglen som fejlede på grund af data. |
| Fejlbeskrivelse | Tekstuel beskrivelse af fejlen |
| Fejltidspunkt | Logningstidspunkt for fejlen |

Et eksempel på en fejl kunne være følgende:

2013-01-25 15:29:43,767 [INFO] BusinessRulesErrors - LPR Recordnummer [1234], Regel [LPR dato og tid regel], fejlbesked [Proceduredato findes ikke], db [MINIPAS]

Eksemplet viser at LPR referencenummer 1234 ikke kan behandles af "Dato og Tid" reglen, da der er en tilknyttet procedure hvor proceduredatoen ikke findes.

### Logfiler og logtabel

Logfilen er konfigureret via log4j properties filen, der ligger i <Tomcat installationsdir>/appconfig

Log appenderen hedder "BusinessRulesErrors", det er denne appender LPR Databehandleren skriver fejlene til. Log appenderen er sat op som en rullende fil appender, så der er styr på hvor meget diskplads der maksimalt bruges, og den skriver til en fil der hedder "forretningsregel-fejl.log". hver gang den maksimale filstørrelse nås bliver der således lavet en ny fil " forretningsregel-fejl.log.1" o.s.v indtil det maksimale antal filer bliver nået, derefter slettes den ældste fil hver gang en ny skrives.

Default opsætning ses her:

log4j.logger.BusinessRulesErrors=INFO, ERRORFILE

log4j.appender.ERRORFILE=org.apache.log4j.RollingFileAppender

log4j.appender.ERRORFILE.File=forretningsregel-fejl.log

log4j.appender.ERRORFILE.Append=true

log4j.appender.ERRORFILE.MaxFileSize=100MB

log4j.appender.ERRORFILE.MaxBackupIndex=10

log4j.appender.ERRORFILE.layout=org.apache.log4j.PatternLayout

log4j.appender.ERRORFILE.layout.ConversionPattern=%d [%-2p] %c - %m%n

Logtabellen ligger i HAIBA databasen og hedder RegelFejlbeskeder, der bliver logget det samme i denne tabel som i logfilerne, men der er ikke noget der sikrer at der maksimalt logges eksempelvis 1 GB.

Derimod kan database logning slås til/fra ved at bruge "disable.database.errorlog" parameteren som beskrevet tidligere.

# Ændringslog

Kilden til dette dokument kan findes på:

[https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Drifts og installationsvejledning-1.0.docx](https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Drifts%20og%20installationsvejledning-1.0.docx)

| Version | Dato | Ændring | Ansvarlig |
| --- | --- | --- | --- |
| 1.0 | 2013-01-03 | Initielt Dokument | Trifork  Kjeld Froberg |
| 2 | 2014-02-12 | HAIBA2-ændringer | Trifork  Aksel Schmidt |