SKLTP AGP SAD - Aggregeringsplattform

**Med exempel på aggregerande tjänst för tidbokning**

# Inledning

Aggregerande tjänster beskrivs i T-boken (REV B). Denna SAD syftar till att beskriva ett generellt realiseringsmönster för aggregerande tjänster, enligt T-bokens definition.

I nuvarande version är dokumentet inriktat mot realisering av en aggregerande tjänst för tidbokningar. Men detta dokument har också ett långsiktigt syfte att vara grund för en design av ett generellt mönster för aggregerande tjänster, dvs. när designen är prövad för tidbokning i verkligheten kan designen generaliseras till att vara allmängiltig för alla aggregerande tjänster i form av en aggregeringsplattform (SKLTP AGP) på ett liknande sätt som SKLTP idag stöttar framtagning av virtuella tjänster (se T-boken) med sin virtualiseringsplattform, (SKLTP VP).

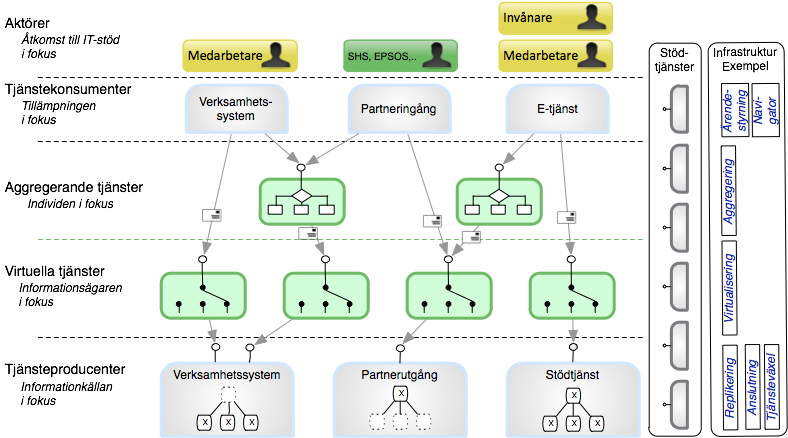


Figure , Översikt, informationssystemvy från T-boken (REV B)

Implementationsnära delar av detta designdokument är baserade på att Mule ESB CE (v3.3.0 eller senare) används samt att tjänsten driftsätts antingen i den nationella tjänsteplattformen (NTjP) eller i en regional tjänsteplattform (RTjP).

**Innehåll**

[Inledning](#h.q2plyoce37s2)

[Översikt](#h.nv5noa7lh2k1)

[Arkitekturella krav](#h.imoqyz81k435)

[Systemsamverkan](#h.1zwqhqvxyycn)

[Användningsfall](#h.g1z6llu75izh)

[Tjänstekonsument anropar aggregerande tjänst](#h.iug1gzxahzw)

[Engagemangsindex notifierar aggregerande tjänst](#h.n7qeu4q1r5ev)

[Automatisk rensning av cache](#h.wd9x5fxwwypv)

[Manuell rensning av cache](#h.2ru5qjru4bqj)

[Logisk vy](#h.hu5a0gm5j81w)

[Implementations vy](#h.lzeujnkfevg)

[Meddelandemappningar](#h.cjuut43azkh1)

[Konsument anropar aggregerande tjänst](#h.9rccxd5kcj2m)

[Mappning AggregatedGetSubjectOfCareSchedule-request till FindContent-request](#h.qprisyrlzazw)

[Mappning FindContent-response till GetSubjectOfCareSchedule-request](#h.qcac4bhbn9ap)

[Mappning GetSubjectOfCareSchedule-response till AggregatedGetSubjectOfCareSchedule-response](#h.76qtbgs4om8k)

[Engagemangsindex uppdaterar aggregerande tjänst](#h.l3a8g15i5l3c)

[Rensning av cache](#h.98on5n2oc9bu)

[Test vy](#h.b6vgruu75xa0)

[Deployment vy](#h.g0fahpx6rxie)

[Referenser](#h.sjth5sjnspb8)

# Översikt

Dokumentet är indelat i följande delar:

1. **Arkitekturella krav**Beskriver de arkitekturella egenskaper som är gemensamma för aggregerande tjänster.
2. **Systemsamverkan**Ger en översiktlig bild av vilka system som samverkar samt dess primära roller i denna samverkan
3. **Användningsfall**  
   Beskriver de användningsfall som har störst påverkan på designen.
4. **Logisk vy**  
   Beskriver designens logiska uppdelning.
5. **Implementationsvy**  
   Beskriver centrala delar av implementationen.
6. **Testvy**  
   Beskriver integrationstester som behövs för att säkerställa den totala lösningens funktionalitet.
7. **Deploymentvy**  
   Beskriver hur lösningen skall driftsättas i QA och produktionsmiljöer.

# 

# 

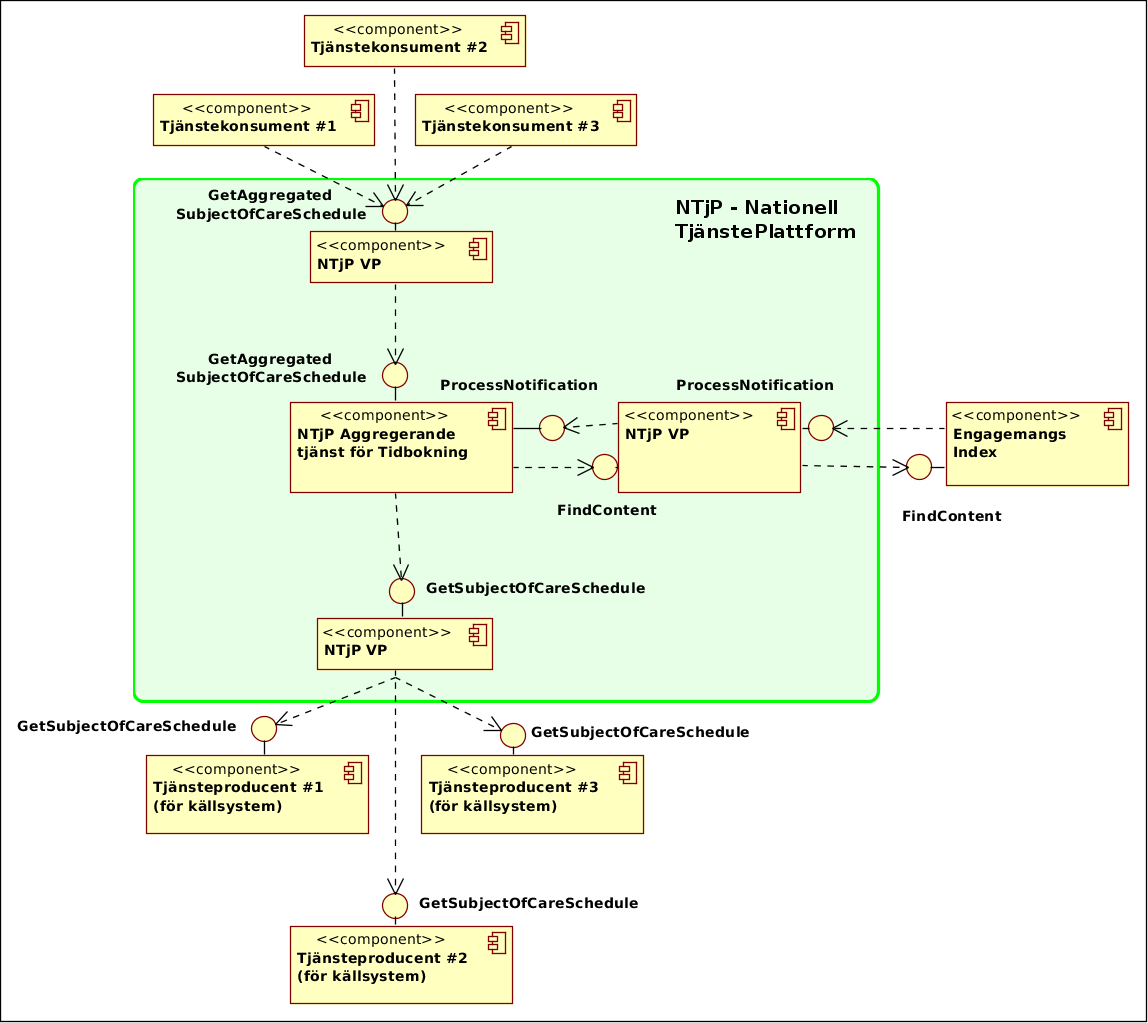
# Arkitekturella krav

Aggregerande tjänster delar en rad egenskaper:

|  |  |
| --- | --- |
| **Egenskap** | **Beskrivning** |
| **1. En per tjänstedomän** | Typfallet är att det finns en aggregerande tjänst per tjänstedomän och att den aggregande tjänstens payload är definierad av ett befintligt tjänstekontrakt i tjänstedomänen. Ett exempel kan vara tjänstedomänen för invånarens tidbokning. Den aggregerande tjänstens resultatmeddelande består av en lista av poster vars struktur definieras av tjänstekontraktet GetSubjectOfCareShedule. Det är också det tjänstekontrakt som den aggregerande tjänsten använder för att hämta data från källsystemen. |
| **2. Mellanlagrar sökresultat** | Aggregerande tjänster ska skydda källsystemen från last och minimera svarstider genom att mellanlagra (t.ex. i primärminnet) sökresultat. Det gör att endast den första sökningen på ett patientid kommer att resultera i att data hämtas från källsystemet. Därefter ligger resultatet i minnet / mellanlager. Mellanlager hålls aktuellt genom att en aggregerande tjänst prenumererar på händelser från engagemangsindex. Vid notifiering från engagemangsindex aktualiseras data för patienter som redan finns i mellanlagret, från de logiska adressater som notifieringen refererade. När en patients data inte efterfrågats eller uppdaterats på 30 dagar tas patienten bort från mellanlagret (cache) i syfte att hålla nere mängden data som mellanlagras.  Om en viss tjänsteproducent inte är tillgänglig när notifiering inkommer, bör den aggregerande tjänsten hantera omförsök periodiskt tills aktuella bokningar kunnat hämtas in till mellanlagret. |
| **3. Söker i Engagemangsindex** | Aggregerande tjänster använder engagemangsindex för att veta vilka logiska adresser som håller data av aktuellt slag om patienten. När data om en patient/invånare efterfrågas och patienten inte finns i mellanlagret, konsulteras EI för information om vilka logiska adressater (vårdmottagningar) som håller information. |
| **4. Nationell omfattning** | Aggregerande tjänster syftar till att sammanställa data från alla källsystemen i Sverige som stödjer tjänstedomänen (genom att ha tjänsteproducenter enligt domänens tjänstekontrakt). Att stödja en tjänstedomän som har en aggregerande tjänst omfattar oftast även krav på att uppdatera engagemangsindex. Det är sådana aggregerande tjänster som kravställs här, även om lösningen också ska kunna tillämpas regionalt (i en regional tjänsteplattform). |
| **5. Eget tjänstekontrakt** | Varje aggregerande tjänst har en egen tjänsteinteraktion (“WSDL”). Visserligen delar det tjänstekontrakt (payload-definition / XML-schema) med ett av de befintliga kontrakten, men den aggregerande tjänsten ska därutöver redovisa status från de olika logiska adresserna enligt beskrivning nedan (se “Redovisning av data om anropet”). Ev. kan det vara samma WSDL, där adresseringen avgör om det är aggergering över alla verksamheter eller hämta från en verksamhet. |
| **6. Patientidentitet är obligatorisk sökparameter** | Aggregerande tjänster kan bara användas för att hämta information om en patient. Även om det finns en cache som täcker flera patienter, får tjänsten inte användas för sammanställningar av data för flera patienter. |
| **7. Samtidig hämtning av data** | Aggregerande tjänster hämtar data från flera logiska adresser (beroende av patientens engagemang). För att ge så bra svarstider som möjligt, måste logiska adressater anropas parallellt. |
| **8. Redovisning av data om anropet** | Tjänstekontraktet redovisar anropsstatus för varje logisk adressat. Anropsstatus anger om data kunnat levereras eller inte. Om anropet gått bra, redovisas om data kom från källan eller från cache. Om datat hämtades från mellanlager redovisas även tidpunkt då datat överfördes från källsystemet till mellanlagret.  Om anropet inte gick bra redovisas felorsak. Felorsaker anges som Timeout eller Serverfel (detaljer från servern rapporteras vid timeout). |
| **9. Behörighetskontroll** | Den aggregerande tjänsten ingår i Tjänsteplattformen. Därför ska den inte ses som en tjänstekonsument med avseende på behörighetskontroll. När den aggregerande tjänsten anropar vitruella tjänster för att hämta data från källsystemen, är det därför HSA-id från den aggregerande tjänstens tjänstekonsument som den virtuella tjänsten ska använda vid behörighetskontroll mot tjänsteadresseringskatalogen. |

# Systemsamverkan

De samverkande systemen beskrivs av följande komponentmodell samt tillhörande text:



* **Tjänstekonsument**  
  Frågar den aggregerande tjänsten efter tidbokningar för ett specifik patientid via tjänstekontraktet **GetAggregatedSubjectOfCareSchedule**.
* **Engengagemangsindex**Förser via en prenumerationstjänst (tjänstekontrakt **ProcessNotification**) den aggregerande tjänsten med information om vilka vårdmottagningar som har tidbokningar för vilka patienter.   
  Tillhandahåller oxå en frågetjänst (tjänstekontrakt **FindContent**) som gör det möjligt för den aggregerande tjänsten att läsa upp index-information om en viss patients samtliga bokningar.
* **Tjänsteproducent (för källsystem)**Tillhandahåller en tjänst, **GetSubjectOfCareSchedule**, för åtkomst av tidbokningar i det specifika källsystemet.
* **NTjP - Virtualiseringplattformen (VP)**Alla informationsutbyten mellan tjänstekonsumenter och tjänsteproducenter sker via virtuella tjänster i den nationella tjänsteplattformen.
* **NTjP - Aggregerande tjänst för Tidbokning**  
  Tillhandahåller aggregerande tjänst för Tidbokning till konsumenter via tjänstekontraktet **GetAggregatedSubjectOfCareSchedule**.  
    
  Utgående från informationen tjänsten får från det nationella engangagemangsindexet så kan tjänsten avgöra vilka vårdmottagningar som har tidbokningar för en patient.   
    
  Med hjälp av NTjP - Virtualiseringstjänst för Tidbokning kan den aggregerande tjänsten hämta patientens bokade tider och kallelser hos var och en av de vårdmottagningar som listats av engagemangsindex. Dessa kan sedan sättas samman till ett samlat svar från den aggregerande tjänsten. Svaret cachas internt i den aggregerande tjänsten under en begränsad tid (30 dagar) för att avlasta källsystemen om tidbokningar för samma patient efterfrågas inom denna tidsperiod. Patientinformation som finns i cachen hålls uppdaterad med hjälp av uppdateringar från prenumerationen i engengagemangsindexet.

# Användningsfall

Tjänsten uppfyller följande användningsfall för att möta de arkitekturella kraven i referens [1]:

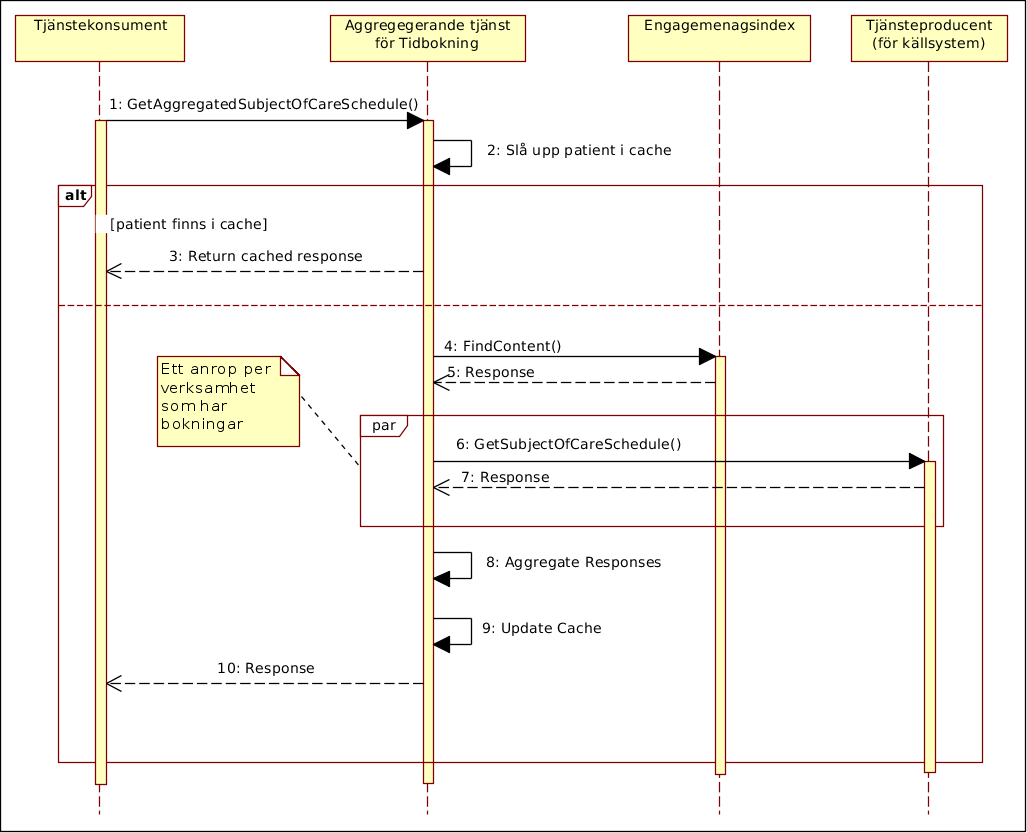
1. Tjänstekonsument anropar aggregerande tjänst
2. Engagemangsindex notifierar aggregerande tjänst
3. Automatisk rensning av cache
4. Manuell rensning av cache

## Tjänstekonsument anropar aggregerande tjänst

Användningsfallet beskrivs av följande löptext och tillhörande sekvensdiagram:

Då en tjänstekonsument anropar den aggregerande tjänsten för tidbokningar så söker den först i cachen på angivet patientId och returnerar informationen från cachen om sådan finns. Annars anropar tjänsten engagemangsindexet för att få redan på var nuvarande bokningar för angivet patientId finns, dvs hos vilka mottagningar. Därefter anropas respektive källsystem parallellt (för att minimera svarstiden) och ett aggregerat svars sätts samman. Det aggregerade svaret sparas i cachen samt returneras till tjänstekonsumenten. Cachen uppdateras också i de fall anrop misslyckats med relevant fel-information.

**Not:** I sekvensdiagrammet nedan är NTjP’s Virtualiseringplattform bortabstraherad i syfte att öka diagrammets läsbarhet men den används i samtliga externa samband, dvs i anrop mellan den aggregerande tjänsten och konsumenter, engagemangsindex och källsystem.



## Engagemangsindex notifierar aggregerande tjänst

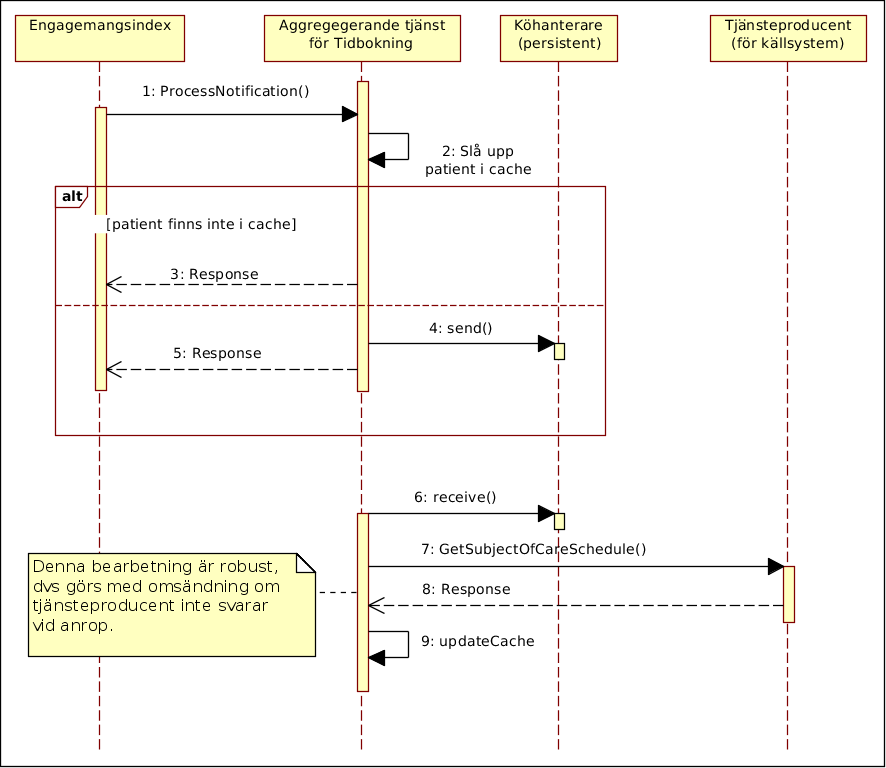
Användningsfallet beskrivs av följande löptext och tillhörande sekvensdiagram:

Då engagemangsindex notifierar den aggregerande tjänsten om uppdaterad eller borttagen bokningsinformation så söker tjänsten först i cachen på angivet patientId. Finns ingen information i cachen för angivet patientId så ignoreras notifieringen. Finns information i cachen för angivet patientId så hämtas bokningar för de mottagningar som anges i notifieringen. För att nå respektive mottagnings källsystem används en virtuell tjänst i NTjP. Cachen uppdateras därefter med denna information eller med fel-information om anropet misslyckats.

Bokningar hämtas asynkront, dvs notifieringen från engagemangsindex bearbetas inte direkt under anropat från engagemangsindex utan läggs på en kö. För att undvika att notifieringar tappas bort pga tillfällig problem att nå källsystem osv så implementeras bearbetningen på ett robust sätt mha persistenta köer och omsändningslogik.

Tjänstekontraktet GetSubjectOfCareSchedule används istället för GetBookingDetails för att hämta bokningar från källsystemet både i fallet uppdaterad och borttagen bokningsinformation. Detta för att dels få en enklare logik för att hålla cachen i synk med källsystemet och dels för att minimera antalet anrop till källsystemet i de fall flera bokningar för en patient är uppdaterade.

**Not:** I sekvensdiagrammet nedan är NTjP’s Virtualiseringplattform bortabstraherad i syfte att öka diagrammets läsbarhet men den används i samtliga externa samband, dvs i anrop mellan den aggregerande tjänsten, engagemangsindex och källsystem.



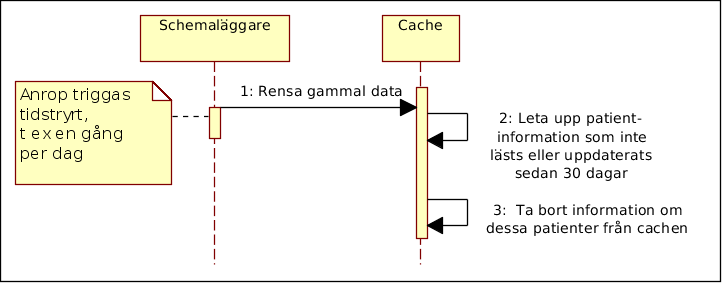
## 

## 

## Automatisk rensning av cache

Användningsfallet beskrivs av följande löptext och tillhörande sekvensdiagram:

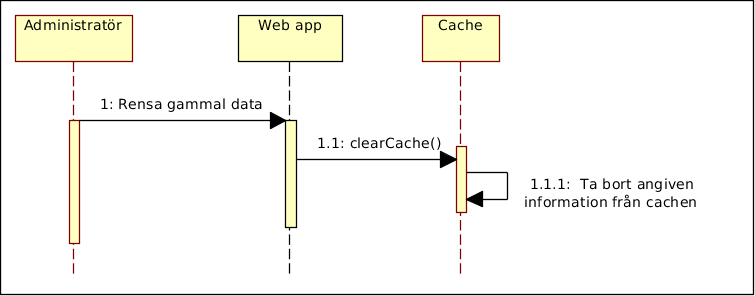
Då information om en patient varken uppdaterats eller lästs under de senaste 30 dagarna så skall tjänsten automatiskt ta bort all information om patienten ur cachen.



## Manuell rensning av cache

Användningsfallet beskrivs av följande löptext och tillhörande sekvensdiagram:

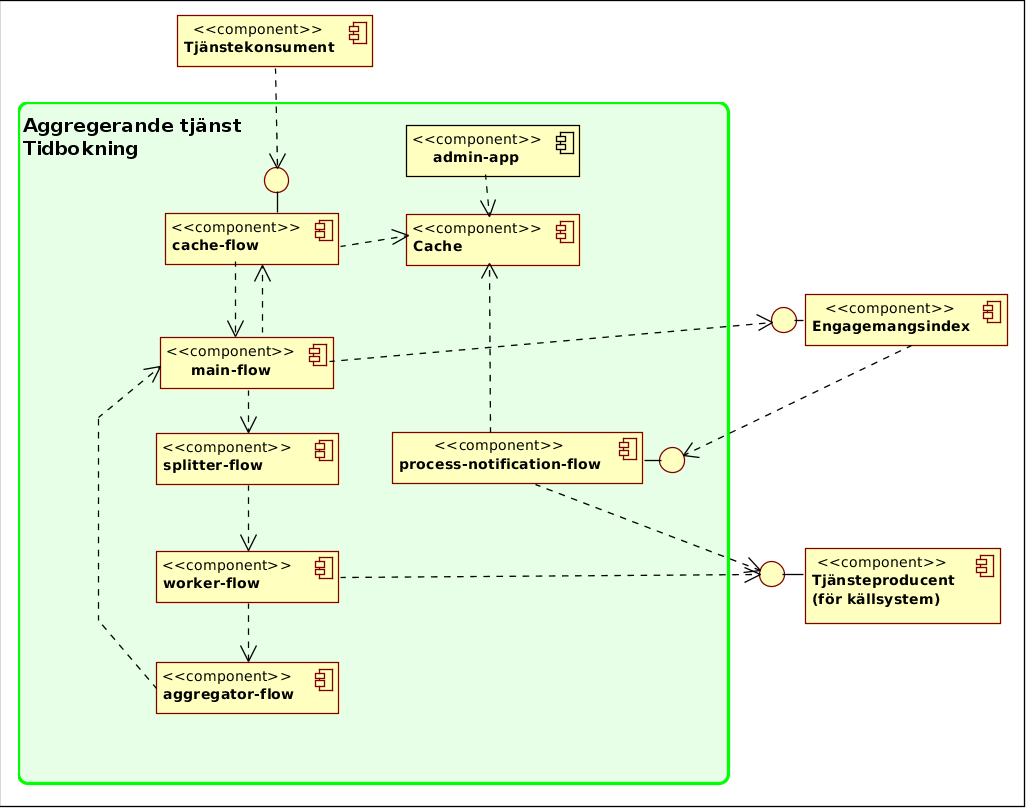
Vid behov kan en administratör manuellt rensa hela eller delar av cachen. Partiell rensning kan göras för ett patientId eller en vårdmottagnings logicalAddress.



# Logisk vy

Den aggregerande tjänsten är uppbyggd av ett antal logiska komponenter. De beskrivs av nedanstående komponentmodell samt efterföljande textuella beskrivning av respektive komponent.

**Not:** I komponentmodellen nedan är NTjP’s Virtualiseringplattform bortabstraherad i syfte att öka bildens läsbarhet men den används i samtliga externa samband, dvs i anrop mellan den aggregerande tjänsten och konsumenter, engagemangsindex och källsystem.

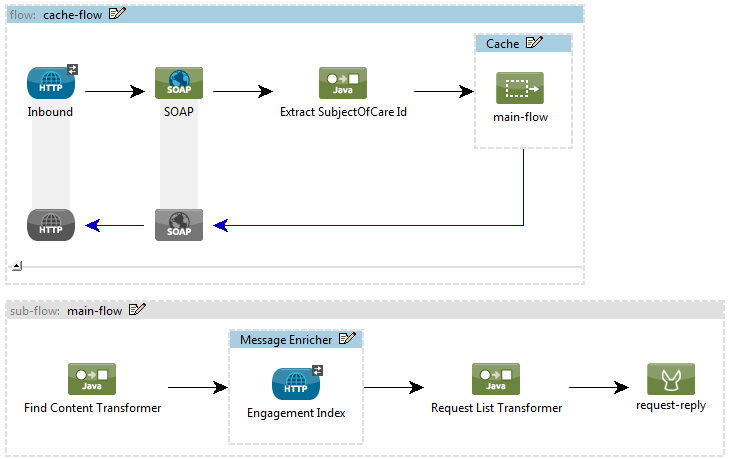


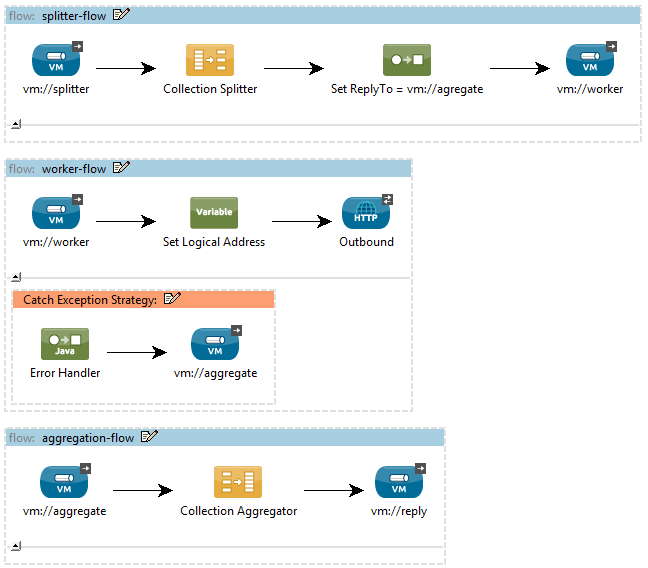
**Komponenter i den aggregerande tjänsten:**

* **Cache-flöde, cache-flow**Synkront Mule flöde som ansvarar för att exponera tidbokningstjänsten.  
  Söker i cache efter information och returnerar den om den finns. Saknas information i cache anropas main-flow.
* **Cache**Mule ESB EE komponent (v3.3 eller senare), delvis specialiserad för att möta specifika behov som den aggregerande tjänsten har. Lagrar bokningsinformation per patient då den efterfrågas första gången. Håller bokningsinformationen per patient i upp till 30 dagar sedan den senast uppdaterats eller efterfrågats.
* **Huvud flöde, main-flow**Synkront Mule flöde som hämtar index-information från engagemangsindex och splitterkomponenten anropas varefter flödet ställer sig och väntar på svar från aggregeringskomponenten innan svar returneras till anropande konsument. Svar från aggregerings-komponenten måste komma inom en konfigurerbar timeout-tid annars returnerar flödet ett timeout-fel tillbaka till konsumenten.
* **Splitterkomponent, splitter-flow**Asynkront Mule flöde som ansvarar för fördela anropen till källsystem på ett antal instanser av workerkomponenten som arbetar parallellt. Flödet ansvarar också för att märka meddelande till workerkomponenterna med ett och samma correlation-id så att aggregeringskomponenten kan gruppera inkommande svar till respektive väntande konsument (blir speciellt viktigt i de fall två eller fler konsumenter samtidigt söker efter patientinformation som inte finns i cachen). På meddelande till workerkomponenterna anges också antalet förväntade svar så att aggregeringskomponenten kan returnera ett svar så fort alla svar kommit in, dvs inte behöva vänta tills timeouten slår till.
* **Workerkomponent, worker-flow**Asynkront Mule flöde som anropar ett källsystem, väntar på svar och förmedlar det vidare till aggregeringskomponenten. Flödet väntar på svar från källsystemet upp till en konfigurerbar timeout-tid. Kommer inget svar inom den tiden skickar den vidare ett timout-svar till aggregeringskomponenten. Uppstår det något fel i samband med anropet till källsystemet så skickar flödet vidare ett svar med felinformation till aggregeringskomponenten.
* **Aggregeringskomponent, aggregator-flow**Samlar in svar från de olika worker-flow instanserna, gruppera svaren per correlationId (satt av splitterkomponenenten) och skickar ett aggregerat svar tillbaka till tidboknings- komponenten när alla svar kommit in eller en konfigurerbar timeout-tid uppnåtts.
* **Notifieringstjänst, process-notification-flow**Synkront Mule flöde som ansvarar för att exponera processNotification-tjänsten och genom den ta emot uppdateringar från engagemangsindex enligt användningsfallet “*Engagemangsindex uppdaterar aggregerande tjänst*”.
* **Administrationsapplikation, admin-app**  
  Gör det möjligt en administratör att rensa hela eller delar av cachen.

# Implementations vy

Mule flödena är implementerade mha Mule Studio och återges nedan som skärmdumpar från verktyget:





## Mekanismer

TODO: Beskriv hantering timeouter och cachning samt konfigurerbara parametrer för detta:

# Default timeout for synchronous services

SERVICE\_TIMEOUT\_MS=4000

AGGREGATE\_TIMEOUT\_MS=4500

AGGREGATED\_SERVICE\_TIMEOUT\_MS=5000

# Cache properties

CACHE\_MAX\_ENTRIES=1000

CACHE\_TTL\_MS=5000

CACHE\_EXPIRATION\_INTERVAL\_MS=1000

## Meddelandemappningar

I detta kapitel beskrivs de mappningar som behöver göras mellan meddelanden i respektive användningsfall.

### Konsument anropar aggregerande tjänst

Tre mappningar behövar göras för detta användningsfall, se tillhörande sekvensdiagram för överblick:

1. Mappning AggregatedGetSubjectOfCareSchedule-request till FindContent-request
2. Mappning FindContent-response till GetSubjectOfCareSchedule-request
3. Mappning GetSubjectOfCareSchedule-response till AggregatedGetSubjectOfCareSchedule-response

De tre mappningarna implementeras av respektive transformeringklass:

1. se.skl.tp.at.tidbokning.engagemangsindex.FindContentRequestTransformer
2. se.skl.tp.at.tidbokning.tidbokning.TidbokningRequestTransformer
3. se.skl.tp.at.tidbokning.tidbokning.TidbokningResponseTransformer

Variabler som används i mappningarna nedan är prefixade med id’n för tillhörande tjänst:

* **at**: Variabel för **a**ggregerings**t**jänsten GetSubjectOfCareAggregatedSchedule
* **ei**: Variebel för **e**ngagemangs**i**ndex tjänsten FindContent
* **ks**: Variable för **k**äll**s**ystemens tjänst GetSubjectOfCareSchedule

#### Mappning AggregatedGetSubjectOfCareSchedule-request till FindContent-request

**GetSubjectOfCareAggregatedSchedule in-parametrar:**

- at.in.logicalAdress = hsa-id för aggr tjänst

- at.in.actor = SUBJECT\_OF\_CARE eller SUBJECT\_OF\_CARE\_AGENT

- at.in.subject\_of\_care = personnummer för patienten i fråga

**FindContent in-parametrar mappas enligt:**

- ei.in.logicalAdress = hsa-id för nationellt EI

- ei.in.registeredResidentIdentification = at.in.subject\_of\_care

- ei.in.serviceDomain = ”riv:crm:scheduling”

**Not**: Övriga element i FindContent requestet utelämnas och antas ge wildcard-funktionelitet, t ex Categorization och MostRecentChange.

### 

### 

#### Mappning FindContent-response till GetSubjectOfCareSchedule-request

**FindContent ut-parametrar:**

- ei.ut.engagement-list, ett element i listan per logical-address som har tidbokningsinfo för patienten i fråga

**Not**: Istället för att anropa GetBookingDetail en gång per bokning så anropas GetSubjectOfCareSchedule en gång per logical-address.

Finns det två eller fler bokningar för en patient på en och samma logical-address så innebär det färre anrop till källsystemet, dvs en prestandavinst och lägre last på källsystemen.

**GetSubjectOfCareSchedule in-parametrar mappas enligt:**

- ks.in.logicalAdress = ei.ut.engagement-list.row.logicalAddress

- ks.in.actor = at.in.actor

- ks.in.healthcare\_facility = ei.ut.engagement-list.row.logicalAddress

- ks.in.subject\_of\_care = ei.ut.engagement-list.row.registeredResidentIdentification

#### Mappning GetSubjectOfCareSchedule-response till AggregatedGetSubjectOfCareSchedule-response

**GetSubjectOfCareSchedule ut-parametrar:**

- ks.ut.timeSlotDetail-list, ett element in listan per bokning för avsedd logical-address för patienten i fråga

Svaren från respektive logical-address läggs samman till ett aggregerat svar.

**GetSubjectOfCareAggregatedSchedule ut-parametrar mappas enligt:**

- ks.ut.timeSlotDetail-list, summan av alla inkomna svar från källsystemen.

### Engagemangsindex uppdaterar aggregerande tjänst

**TBD**

### Rensning av cache

**TBD**

# 

# Test vy

De tre mappningarna testas av respektive unit-tester-klasser:

1. se.skl.tp.at.tidbokning.engagemangsindex.FindContentRequestTransformerTest
2. se.skl.tp.at.tidbokning.tidbokning.TidbokningRequestTransformerTest
3. se.skl.tp.at.tidbokning.tidbokning.TidbokningResponseTransformerTest

Integrationstester skall realiseras såväl som automatiska integrationstester som vara körbara i en test/qa miljö mot skarpa samverkande testsystem.

Följande integrationstester behövs för att säkerställa den totala lösningens funktionalitet.

**TBD**

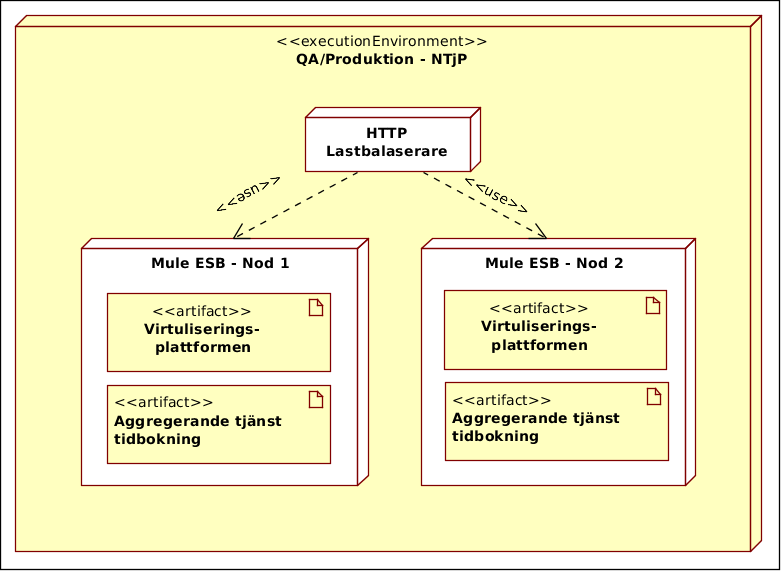
# 

# 

# Deployment vy

Tjänsten skall driftsättas i befintlig infrastruktur för den nationella tjänsteplattformen, NTjP.  
Tjänsten paketeras och deployas som en standard Mule applikation på samma Mule instans som Virtualiseringsplattformen (VP) exekverar på.  
  
**Not:** I dagsläget stöttar VP bara den gamla Mule 2 deploymodellen och inte Mule 3’s nya Mule Applikations koncept, dvs utan möjlighet att deploya och omdeploya enskilda Mule applikationer (t ex nationella tjänster). Detta måste åtgärdas för att denna tjänst skall kunna driftsättas i NTjP’s miljö.

Översiktbil av NTjP’s QA och produktionsmiljö med aggregerande tjänst för tidbokning driftsatt:



# Referenser

1. T-boken (REV B) - <http://www.cehis.se/arkitektur_regelverk/teknisk_arkitektur/>