Dokumentation af Sentinel 5.1

Webbaseret GUI og INTERFACE

|  |  |
| --- | --- |
| Dokument information: | |
|  |  |
| Dokument titel |
| DOKUMENTATION AF SENTINEL 5.1 - WEBBASERET GUI OG INTERFACE |  |
|  |  |
| Forfatter |
| Flemming Bo Hansen |
|  |  |
|  |  |
| Filnavn | Sidst opdateret |
| sentinel\_5.1\_webapi\_dokumentation.docx | 2024-11-04 01:18:00 |

|  |  |  |  |
| --- | --- | --- | --- |
| Dokument historik: | | | |
| Version | Kommentarer | Dato | Ændret af |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 1.0 | Initial version | 2022-03-31 | FBH |
| 2.0 | Rettet anvendelse af certifikat i systemlogind procedure | 2022-05-30 | FBH |
| 5.1 | Tilføjelse af IndsaetForloeb51 og Slet Forloeb51 samt fjernet HsentWebKlientUrl50 | 2024-08-21 | CHET, TRA |

Indholdsfortegnelse:

1. Introduktion 5

1.1. Læsevejledning til systemhuse for fysioterapeuter 5

2. Overordnet systembeskrivelse 6

3. Sentinel-API50 7

3.1. Kort gennemgang af API’ets operationer 7

3.2. Detaljeret beskrivelse af operationer 10

3.3. Eksempel på kald af SentinelAPI’et fra WebKlient via Ajax 34

4. Hvad stilles der til rådighed for udvikling og test 37

5. Hvad skal der til for at systemhuset kan gå i produktion? 38

6. Certificering af praksissystemer 39

# Introduktion

Sentinel benyttes i forbindelse med kvalitetsarbejdet i primærsektoren. Sentinel udbydes både i en windows- og en webbaseret version.

Formålet med dette dokument er at redegøre for det webbaserede API, krav til systemhusene (praksissystemerne), som skal implementere integrationen, samt procedure for, at et praksissystem kan udrulle Sentinel integration i produktion.

Dokument er skrevet til at dække alle faggrupper i primærsektoren. Det betyder, at ikke alle funktionaliteter er relevante for alle faggrupper. F.eks. vil medicin ikke være relevant for kiropraktorer eller fysioterapeuter.

Systemhuset skal certificeres i de dele af Sentinel-API’et, der giver mening for den enkelte faggruppe, samt de funktionaliteter, som journalsystemet understøtter.

Dokumentet indledes med en overordnet beskrivelse af systemet. Derefter følger listen operationer, som API’et indeholder. Først præsenteres alle operationer med en kort beskrivelse efterfulgt af en uddybende beskrivelse af hver operation, hvor hver parameter, der indgår i kaldet, er beskrevet.

Dokumentet inkluderer også et eksempel på hvordan SentinelAPI’et kaldes fra en webklient via Ajax. Dette eksempel er skrevet i JavaScript mens alle andre kodebidder er skrevet i C#.

Den sidste del af dokumentet indeholder informationer om, hvordan systemhusene certificeres, for at få tilladelse til at integrere op imod Sentinel.

## Læsevejledning til systemhuse

Dette dokument er en beskrivelse af det fulde Sentinel API. For hvert enkelt speciale er der aftaler med den faglige organisation hvilke data, og dermed operationer, der er relevant der er relevant for det pågældende speciale. Dette gøres med udgangspunkt i den relaterede overenskomst. Hvilke operationer som skal benyttes af læserens speciale vil fremgå af et regneark DataIntegration\_<speciale>\_<dato>.xlxs.

# Overordnet systembeskrivelse

Sentinel er et kvalitetssikringssystem, der integrerer journalsystemer i sundhedsvæsnets primærsektoren. Dette gøres via en snitflade, som beskrives i detaljer i dette dokument. Ved brug af strukturerede data og events fra praksissystemer foretages indsamling, analyse og rapportering iht. til aftale kvalitetsprojekter.

Sentinel kan på baggrund af analyse af registrerede data og events udløse (trigge) forskellige informationer/meddelelser og formularer (også kaldet popups). Informationer og popups skal præsenteres for brugeren umiddelbart efter at de er trigget.

En databehandleraftale mellem Sentinel, sundhed.dk og klinikkerne skal være underskrevet, før dataudveksling kan finde sted. Dette er en indbygget funktionalitet i Sentinel, for at beskytte brugernes data.

I Windows versionen af Sentinel installeres der en Sentinel-klient, som bl.a. har til formål at håndtere og præsentere meddelelser og popups for brugeren. I den webbaserede version er disse opgaver fordelt anderledes. Hvornår og hvilke informationer og formularer, der skal præsenteres for brugeren, samt præsentationen af formularer, varetages af Sentinel. Mens det er journalsystemets ansvar, at ’aktivere’ præsentationen for brugeren på foranledning af Sentinel.

Bemærk, at det nye API er kompatibelt med det allerede eksisterende activeX baserede interface, så man kan, hvor det er formålstjenesteligt, stadigt installere en Sentinel Klient på en klientmaskine, men benytte de nye webbaserede operationer til kommunikation med Sentinel.

Der webbaserede API mod Sentinel udstilles både i en SOAP baseret - og en JSON baseret version. For at være bagud-kompatibel udstilles API’et både i en engelsk og dansk version. I afsnit 3 gennemgås SentinelAPI, som udgør snitfladen mellem praksissystemer og Sentinel.

SentielAPI’et udstiller operationer til brug for:

* Sikker sessionsoverdragelse (SSO)
* Data- og eventoverførsel
* Brugergrænseflade (GUI) integration

# Sentinel-API50

Alle operationer i Sentinel-API50 udstilles både som SOAP baseret webservice kald, og som JSON baserede servicekald. Alle operationer kan kaldes fra server og/eller klient. Det er derfor op til det enkelte systemhus at afgøre hvilken protokol, der benyttes og hvorfra de enkelte operationer skal foretages. Alle operationer er udstillet både i en engelsk og en dansk version, og man kan frit vælge, hvilken man ønsker at implementere op imod. Grunden til dette overhead er, at der har været ønske om et engelsk interface, men af hensyn til bagud kompatibilitet er det besluttet fortsat at understøtte en dansk version.

Sentinel-API’et bliver hostet hos Sentinel-enheden, sunhed.dk. Der er oprettet et testsite som journal systemerne kan benytte til udvikling og test af integrationen (se afsnit 4 for yderlige information).

Det er et krav at systemhusene skal kunne eftersende data i tilfælde af nedbrud.

## Kort gennemgang af API’ets operationer

Operationer til generering af SSO

* **GenererSystemNoegle50**: Generer en tidsbegrænset engangsnøgle, som krypteret benyttes i et kald til **SystemLogInd50**, for at få en ***systemlogindtoken***. Dette skal benyttes i alle efterfølgende kald til API’et.
* **SystemLogInd50**: Genererer en ***systemlogindtoken***, der kan benyttes i alle øvrige kald i API’et, som system til system integrations-SSO
* **GenererSystemKlientNoegle50**: Benyttes som mellemled i forbindelse med, at en ***systemlogindtoken,*** genereret med **SystemLogind50*,*** skal veksles til en klientbundet ***logindtoken,*** genereret med **SystemKlientLogind50*.***
* **SystemKlientLogInd50**: Dette returnerer en klient-***logindtoken***, som benyttes i ajax-baseret kald fra en webklient.

Operationer til bruger logind

* **LogInd50**: Kaldes når en bruger logger ind i journalsystemet.
* **LogUd50**: Kaldes når en bruger logger ud af journalsystemet.
* **LogIndSession50**: Kan alternativt benyttes i stedet for **LogInd50**, hvis en bruger kan være logget ind i flere sessioner samtidigt fra en og samme klient.
* **LogUdSession50**: Kan alternativt benyttes i stedet for **LogUd50**, hvis en bruger kan være logget ind i flere sessioner samtidigt fra en og samme klient.

Operationer til data- og event-integration

* **OpdaterStamdata50**: Kaldes for at overføre patient stamdata til Sentinel
* **IndsaetAnalyse50**: Kaldes når der oprettes eller importeres patientværdier og laboratorieresultater.
* **IndsaetMedicin50**: Kaldes når der oprettes en recept, der printes en recept eller der udleveres medicin til patienten.
* **IndsaetDiagnose50**: Kaldes når der oprettes en diagnose i journalsystemet, eller der indlæses/importeres en diagnose via henvisninger, epikriser, m.m.
* **IndsaetDiagnose51**: En nyere version af operationen. Den kaldes når der oprettes en diagnose i journalsystemet, eller der indlæses/importeres en diagnose via henvisninger, epikriser, m.m.
* **IndsaetYdelse50**: Kaldes når der registreres ydelser i journalsystemet.
* **IndsaetPost50**: Kaldes når der indlæses eller afsendes elektronisk post, henvisninger, epikriser m.m.
* **IndsaetBehandling50**: Benyttes endnu ikke.
* **IndsaetNotat50**: Benyttes endnu ikke.
* **IndsaetForloeb51**: Angiver at patienten er i et forløb. Alle oplysninger, der høre til dette forløb, bliver knyttet sammen af ***forloebsid***, gennem kald af indsæt operationerne. For nuværende benyttes operationen kun, hvis man bruger forløbsorienteret journalføring i praksissystemet.
* **RegistrerHaendelse50**: Kan benyttes til at overføre nye events, efter aftale med de enkelte specialers behov.
* **AabnerJournal50**: Kaldes når en bruger åbner en journal i praksissystemet. Dette event benyttes bl.a. til at komme med patientspecifikke notifikationer når en bruger arbejder med en given patient.
* **LukkerJournal50**: Kaldes når en bruger lukker en journal i praksissystemet.
* **SletAnalyse50**: Benyttes hvis en patientværdi eller laboratorieværdi slettes i journalsystemet.
* **SletMedicin50**: Benyttes hvis en recept bliver slettet fordi den var fejlagtigt registreret. Der skal ikke foretages sletning blot fordi en recept ikke længere er gyldig.
* **SletDiagnose50**: Benyttes hvis en diagnose slettes i journalsystemet.
* **SletYdelse50**: Benyttes hvis en ydelse slettes i journalsystemet.
* **SletPost50**: Benyttes til at slette fejlagtigt overført elektronisk post.
* **SletBehandling50**: Benyttes endnu ikke.
* **SletNotat50**: Benyttes endnu ikke.
* **SletForloeb51**: Benyttes hvis et forløb slettes i journalsystemet. For nuværende benyttes operationen kun, hvis man bruger forløbsorienteret journalføring i praksissystemet.

Operationer til GUI integration

* **HentMenu50**: Benyttes når brugeren skal interagerer med de to Sentinel-menuer (en med patientkontekst og en med praksiskontekst). Systemhuset skal implementere en adgang til hver menu i journalsystemet. **HentMenu50** kaldes, når brugeren klikker sig ind i en af Sentinel-menuerne.

Hvordan interaktionen med selve menuens menupunkter håndteres uddybes i den mere udførlige beskrivelse af **HentMenu50**.

* **HentAktioner50**: Sentinel kan asynkront komme med popups og meddelelser (her kaldet aktioner). Hvis man kører med en rent webbaseret integration, er det journalsystem-klientens opgave at håndtere disse. Det gøres ved at kalde **HentAktioner50** hvert 3. sek. og håndtere det man evt. får tilbage. Operationen returnerer en liste af meddelelser og en liste af straksaktioner. Strakssktioner skal udføres med det samme. Meddelelser skal vises. De kan indeholde aktioner, der skal udføres når brugeren trykker på meddelelsen. Aktioner kan være.
* **UdfoerAktion50**: Benyttes til at udføre ApiAktioner knyttet til menupunkter, meddelelser og straksaktioner.

Øvrige operationer

* **HentVersion**: Benyttes til at hente informationer om hvilke version af API’et, der er udstillet på nuværende tidspunkt, da Sentinel-API’et altid vil være bagudkompatibel. I version 5.1 vil det stadigvæk være muligt at kalde alle operationer fra version 5.0. Systemhuset kan benytte **HentVersion** til at afgøre hvilken version af API’et, der er i produktion og dermed hvilke operationer, der skal kaldes.
* **LukSentinel50**: Benyttes kun i meget specielle scenarier og efter aftale med Sentinel-enheden.

## Detaljeret beskrivelse af operationer

Alle operationer mod SentinelAPI’et returnerer et resultat nedarvet fra klassen ApiResultat. Hvis ikke andet er angivet, vil operationerne returnerer et ApiResultat.

public class ApiResultat

{

public int StatusKode { get; set; }

public string StatusTekst { get; set; }

}

Hvis statuskoden er negativ, er der sket en fejl i kaldet. Hvilken fejl vil fremgå af *StatusTeksten*. Hvis *StatusKode* er 0, er kaldet udført korrekt. I visse kald eks. **HentAktioner50** kan *StatusKoden* være positiv, hvilket betyder at serveren er ved at beregne relevante aktioner og at man skal prøve igen.

### GenererSystemNoegle50

Kaldes med et systemnavn og systemkodeord, som er aftalt med Sentinel, sundhed.dk. Der generes en tidsbegrænset engangsnøgle, som systemet efterfølgende skal kryptere og benytte i et kald til **SystemLogInd50**, for at få en ***systemlogindtoken***, som skal benyttes i alle efterfølgende kald til API’et. HUSK at benytte denne nøgle i kald til **SystemLogInd50** umiddelbart efter, at den er genereret, da den har en meget kort gyldighedsperiode.

ApiTokenResultat GenererSystemNoegle50(string systemnavn, string systemkodeord);

* ***systemnavn***: Navn på journalsystemet, som aftales mellem systemhuset og Sentinel, sundhed.dk.
* ***systemkodeord***: Et systemkodeord knyttet til systemnavnet, som udleveres af Sentinel, sundhed.dk.

Returnerer:

public class ApiSystemLoginResultat: ApiResultat

{

public string Token { get; set; }

}

* *Token*: En tidsbegrænset krypteret systemnøgle, som efterfølgende skal dekrypteres med et systemhusets private certifikat nøgle. Denne benyttes i kald til **SystemLogind50** for at få genereret en ***systemlogindtoken***, der kan benyttes som SSO i alle efterfølgende kald til API’et.

### SystemLogInd50

Genererer en ***systemlogindtoken***, der kan benyttes i alle øvrige kald i API’et, som system til system integrations-SSO. Læg mærke til, at den af sikkerhedsmæssige årsager ikke må bruges i kald fra webklienter. Der skal i stedet benyttes en klient-***systemlogindtoken*** generet med **SystemKlientLogind50**. **SystemLogInd50** skal kaldes med en krypteret version af en nøgle genereret med **GenererSystemNoegle50**. Sentinel, sundhed.dk skal have tilsendt og registreret public key til det certifikat, som systemhuset benytter til krypteringen, inden det er muligt at kalde denne funktion.

ApiSystemLoginResultat SystemLogInd50(string systemnavn, string encsystemnoegle, string systemversion);

* ***systemnavn***: Se beskrivelse i afsnit 3.2.1.
* ***encsystemnoegle***: En nøgle, som er hentet med kaldet **GenererSystemNoegle50**, derefter dekrypteret med et systemhusets private nøgle til det aftalte certifikat, hvor Sentinel, sundhed.dk har fået tilsendt og registreret en tilhørende public key. Dekrypteringen skal foregå ved følgende procedure:

byte[] encryptedBytes = Convert.FromBase64String(krypteretnoegle);  
byte[] decryptedBytes = privateKey.Decrypt(encryptedBytes);   
string noegle = Encoding.UTF8.GetString(decryptedBytes);

* ***SystemVersion***: Version af praksissystemet, som foretager kaldet. Dette benyttes til fejlsøgning og gør det muligt at afgøre, hvorvidt en fejl er relateret til en bestemt version af journalsystemet.

Returnerer:

public class ApiSystemLoginResultat: ApiResultat

{

public string Token { get; set; }

}

* *Token*: Benyttes som ***systemlogindtoken*** i øvrige kald. Læg mærke til, at denne kun bør benyttes i kald fra serveren, da den ikke er bundet til en bestemt klient. Ved Ajax baserede kald fra Web klienter, skal man benytte **GenererSystemKlientNoegle50** på serveren og efterfølgende benytte den returnerede nøgle til at logge ind på klienten med **SystemKlientLogind50*.*** **SystemKlientLogind50**kan efterfølgende kun benyttes fra den pågældende klientmaskine og for det relaterede ***klientid***.

### GenererSystemKlientNoegle50

Benyttes som mellemled i forbindelse med, at en ***systemlogindtoken,*** genereret med **SystemLogind50*,*** skal veksles til en klientbundet ***systemlogindtoken***, genereret med **SystemKlientLogind50*.*** Dette kald udføres på serveren og den *token*, der bliver returneret, benyttes herefter i webklienten til at generere en ny klient-bundet *token* med **SystemKlientLogind50**. En klient-***systemlogindtoken*** er tidsbegrænset (d.d), og kan kun benyttes fra den browser, maskine og IP, hvor den er genereret, hvilket sikrer imod, at den kan ’stjæles’.

ApiSystemLoginResultat GenererSystemKlientNoegle50(string systemlogindtoken, string klientid);

* ***systemlogindtoken***: En allerede hentet ***systemlogindtoken***.
* ***klientid***: Et unikt id på den klient, som engangsnøglen skal benyttes på/knyttes til.

Returnerer:

public class ApiSystemLoginResultat: ApiResultat

{

public string Token { get; set; }

}

* *Token*: Engangsnøgle, som kan veksles til en systemklientlogintoken i klienten med kald til **SystemKlientLogind50**

### SystemKlientLogInd50

Returnerer en klient-***systemlogindtoken***, som benyttes i ajax-baseret kald fra en webklient. En klient-***systemlogindtoken*** er tidsbegrænset (d.d) og kan kun benyttes fra den browser, maskine og IP, hvor den er genereret, hvilket sikrer imod, at den kan ’stjæles’.

ApiSystemLoginResultat SystemKlientLogInd50(string systemklientnoegle);

* ***systemlogindnoegle***: En tidsbegrænset engangsnøgle udstedt på serveren med **GenererSystemKlientNoegle50** og er bundet til denne klient med ***klientid***.

Returnerer:

public class ApiSystemLoginResultat: ApiResultat

{

public string Token { get; set; }

}

* *Token*: En klient, IP, browser, … bundet ***systemklientlogintoken***, som kan benyttes til alle efterfølgende Ajax kald på klienten.

### LogInd50

Kaldes når en bruger logger ind i journalsystemet.

ApiResultat LogInd50(string systemlogindtoken, string klientid, string ydernr, string initialer);

* ***systemlogindtoken***: En *Token* modtaget fra **SystemLogInd50** eller **SystemKlientLogInd50**.
* ***klientid***: Et globalt unikt id på den klient, som brugeren logger ind på.
* ***ydernr***: Ydernummeret for brugeren. (Hvis en bruger arbejder for flere ydernumre i samme session, skal der foretages et kald til **LogInd50** for hvert ydernummer.)
* ***initialer***: Unik identifikation af brugeren inden for den enkelt klinik.

### LogUd50

Kaldes når en bruger logger ud af journalsystemet.

ApiResultat LogUd50(string systemlogindtoken, string klientid, string ydernr, string initialer);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***ydernr***: Ydernummeret for brugeren. (Hvis en bruger arbejder for flere ydernumre i samme session, skal der foretages et kald til **LogUd50** for hvert ydernummer.)
* ***initialer***: Se beskrivelse i afsnit 3.2.5.

### LogIndSession50

Kan benyttes som alternativ i stedet for **LogInd50**, hvis en bruger kan være logget ind i flere sessioner samtidigt fra en og samme klient.

ApiResultat LogIndSession50(string systemlogindtoken, string klientid, string ydernr, string initialer, string sessionId);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***ydernr***: Ydernummeret for brugeren. (Hvis en bruger arbejder for flere ydernumre i samme session, skal der foretages et kald til **LogIndSession50** for hvert ydernummer.)
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***sessionId***: Et unikt id for den session hvor brugeren logger ind. Samme ***sessionId*** skal senere benyttes i **LogUdSession50** når brugeren logger ud af sessionen.

### LogUdSession50

Kan benyttes som alternativ i stedet for **LogUd50**, hvis en bruger kan være logget ind i flere sessioner samtidigt fra en og samme klient.

ApiResultat LogUdSession50(string systemlogindtoken, string klientid, string ydernr, string initialer, string sessionId);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***ydernr***: Ydernummeret for brugeren. (Hvis en bruger arbejder for flere ydernumre i samme session, skal der foretages et kald til **LogUdSession50** for hvert ydernummer.)
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***sessionId***: Et unikt id for den session hvor brugeren logger ud. Samme ***sessionid*** skal tidligere være benyttet i **LogIndSession50**, da brugeren loggede ind i sessionen.

### OpdaterStamdata50

Benyttes til at overføre stamdata omhandlende klinikkens patienter. Et krav er, at Sentinel altid skal have de nyeste stamdataoplysninger for de patienter, hvis data er registreret i Sentinel.

Man kan initiere integrationen til Sentinel ved at overføre stamdata på alle patienter i klinikken. Derefter holdes stamdata ajour hver gang data opdateres i journalsystemet eller der oprettes nye patienter.

ApiResultat OpdaterStamdata50(string systemlogindtoken, string klientid, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string fornavn, string efternavn, string regionsKode, string emailadresse);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Et globalt unikt id på den klient, som brugeren arbejder på. Kaldes denne operation fra et batchjob uden en bruger, sættes ***klientid***=null.
* ***ydernr***: Ydernummeret for brugeren.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Ydernummeret på patientens egen læge.
* ***egenlaegeinitialer***: Benyttes kun efter aftale med faggruppe.
* ***cpr***: Patientens cpr.
* ***fornavn***: Patientens fornavn og evt. mellemnavne.
* ***efternavn***: Patientens efternavn.
* ***regionskode***: Kode for den region patienten er bosat i (4 cifret).
* ***emailadresse***: Patientens e-mailadresse.

### IndsaetAnalyse50

Kaldes for at overføre information omhandlende patientværdier, laboratorieværdier og andre kodede værdibaserede oplysninger – eks. vægt, højde, blodprocent, m.m. Operationen skal kaldes straks efter, at en patientværdi bliver registreret i journalsystemet. Importerede patientværdier skal ligeledes overføres, straks de registreres i journalsystemet. Information skal overføres hver gang der registreres en patientværdi.

Patientværdier skal bevares historisk, f.eks. skal en ny vejning af en patient give anledning til en ny unik oplysning om vægt og ikke overskrive tidligere overførte vægtoplysninger.

Hvis en patientværdi ændres i journalsystemet, skal man kalde **SletAnalyse50** (se afsnit 3.2.22) efterfulgt af **IndsaetAnalyse50** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historiske patientværdier. (Husk i dette tilfælde at sætte parameteren ***haendelsestidspunkt*** til det tidspunkt, hvor analysen blev foretaget og parameteren ***proevetagningsTidspunkt*** til det tidspunkt, hvor prøven blev taget eller målingen foretaget.)

ApiResultat IndsaetAnalyse50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string kode, string kodeansvarlig, string kodebetydning, string producentKode, string producent, int \_operator, string vaerdi, string enhed, string refFra, string refTil, string refText, string abnorm, string proevetagningsTidspunkt, string kommentarer);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Skal benyttes, hvis oplysningen senere skal rettes eller slettes.
* ***haendelsestidspunkt***: Det tidspunkt, hvor analysen er gennemført. Datotider overføres i xml-format dvs. yyyy-MM-ddTHH:mm:ss.ffff(+|-)zz:zz.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: Patientens cpr.
* ***forloebsid***: Hvis værdien/data er knyttet til et bestemt forløb, angives et unikt id for det pågældende forløb, ellers null.
* ***kode***: Officiel kode for værdien. Der skal benyttes officielle MedCom standarder. Hvis der ikke er nogen officiel kode, kan en anden kode benyttes. Det skal fremgå af parameteren kodeansvarlig, hvem der har defineret koden.
* ***kodeansvarlig***: Id på den myndighed, der har ansvaret for koden.
* ***kodebetydning***: En tekstforklaring af, hvad koden står for. De officielle MedCom standarder skal benyttes, hvis de er defineret for den pågældende kode.
* ***producentkode***: Hvis værdien er produceret på et laboratorie, skal dette være det officielle MedCom id af laboratoriet. Hvis det er en værdi målt i klinikken udfyldes med PNA. Hvis patienten selv har foretaget målingen udfyldes med POT.
* ***producent***: Tekstbaseret identifikation af laboratoriet, klinikken eller patienten alt efter hvor målingen er foretaget.
* ***\_operator***: angiver om den målte værdi er <, = eller > den angivne. -1 svarer til <, 0 svarer til =, og +1 svarer til >. F.eks. hvis en vægt kun kan måle op til 300 kg, men patienten vejer mere end det, kan man ikke angive vægten til andet end >300 kg. Det gøres ved at sætte \_operator = +1 og ***vaerdi*** = 300.
* ***vaerdi***: Den målte værdi (der benyttes punktum som decimal seperator.)
* ***enhed***: Enhed på den målte værdi (kg, cm, m.m.). Værdier skal være angivet i den officielle enhed, der er defineret for koden.
* ***refFra***: Benyttes når der er angivet en nedre grænse for normalområdet af værdien.
* ***refTil***: Benyttes når der er angivet en øvre grænse for normalområdet af værdien.
* ***refText***: En tekstbaseret version af normalområdet.
* ***abnorm***: Udfyldes i hht. MedCom standard med HI, LO eller UN, hvis værdien er konstateret hhv. for høj, for lav, eller unormal. Ellers angives null.
* ***proevetagningsTidspunkt***: Det tidpunkt hvor prøven er taget. Hvis der er tale om en simpel vejning, vil ***proevetagningsTidspunkt*** og ***haendelsestidpunkt*** være ens. Datotider overføres i xml-format dvs. yyyy-MM-ddTHH:mm:ss.ffff(+|-)zz:zz.
* ***kommentarer***:Eventuelle kommentarer.

### IndsaetMedicin50

Kaldes for at overføre information omkring recepter og/eller udleveret medicin. Operationen skal kaldes straks efter, at der er oprettet eller fornyet en recept i FMK, en recept er printet og udleveret til patienten, eller hvis en patient får udleveret medicin.

Hvis en recept ændres i FMK, skal man kalde **SletMedicin50** (se afsnit 3.2.23) efterfulgt af en **IndsaetMedicin50** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historiske receptoplysninger fra FMK. (Husk i dette tilfælde at sætte parameteren ***haendelsestidspunkt*** til det tidspunkt hvor recepten blev aktiveret.)

ApiResultat IndsaetMedicin50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string varenummer, string atckode, string form, string navn, float styrke, string styrkeEnhed, string styrkeTekst, string pakning, string dosKode, string dosKort, string dosLang, int dosPeriodeIDage, string indKode, string indikation, int antalPakn, int antalIte, int IteIntervalIDage);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt hvor medicinen er ordineret. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***:Se beskrivelse i afsnit 3.2.10.
* ***varenummer***: Varenummeret bruges til at identificere den enkelte lægemiddelpakning (lægemiddelnavn, -form, -styrke og pakningsstørrelse).
* ***atckode***: ATC-koden for det pågældende lægemiddel.
* ***form***: Tablet, kapsel, creme ….
* ***navn***: Navnet på produktet. (ikke det aktive stof)
* ***styrke***: Indhold pr. enhed eller koncentrationen af lægemiddellet.
* ***styrkeEnhed***:Enheden hvormed styrken måles.
* ***styrkeTekst***: En tekstbeskrivelse af styrken.
* ***pakning***: Pakningensstørrelse.
* ***dosKort***: Kort/forkortet forklaring/brugsanvisning af, hvordan medicinen skal doseres.
* ***dosLang***: Lang forklaring/brugsanvisning af, hvordan medicinen skal doseres.
* ***dosPeriodeIDage***: antallet af dage medicinen skal tages i.
* ***indKode***: Indikationskoden.
* ***indikation***: Hvad ordineres medicinen imod.
* ***antalPakn***: Antal pakninger, der er ordineret.
* ***antalIte***: Antal gange der kan udleveres på recepten
* ***IteIntervalIDage***: interval mellem udlevering.

### IndsaetDiagnose50

Der findes en nyere version af operationen: **IndsaetDiagnose51**, som bruges i stedet for.

Kaldes for at overføre information omhandlende diagnoser. Operationen skal kaldes straks efter, at der registreres en diagnose i journalsystemet. Importerede diagnoser skal ligeledes overføres, straks de registreres i journalsystemet. Der skal overføres information, hver gang der registreres en diagnose, også hvis diagnoser bliver kopieret mellem kontaktdiagnoser og forløbs-/kroniske diagnose eller tilsvarende begreber.

Hvis en diagnose ændres i journalsystemet, skal man kalde **SletDiagnose50** (se afsnit 3.2.24) efterfulgt af **IndsaetDiagnose50** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historiske diagnoser. (Husk i dette tilfælde at sætte parameteren ***haendelsestidpunkt*** til det tidspunkt hvor diagnosen blev stillet.)

ApiResultat IndsaetDiagnose50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string forloebsrolle, string icpckode, string kodebetydning, string icpckilde, string icd10kode, string icd10kodebetydning, string icd10kilde);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt hvor diagnosen er stillet. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.
* ***forloebsrolle***: Benyttes endnu ikke, men har til formål at kunne tildele diagnoser en bestemt rolle i forhold til et forløb, eks. aktionsdiagnose.
* ***icpckode***: Officiel kode for diagnosen. De officielle MedCom standarder skal benyttes. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***kodebetydning***: Den officielle tekst relateret til icpckoden. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***icpckilde***: Hvis værdien er importeret fra en edifakt eller lign, sættes denne til ydernummeret/sks koden for afsenderen. Hvis diagnosen er stillet i egen praksis udfyldes med ”KLINIK”. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***icd10kode***: Den officielle icd10 kode for denne oplysning. Hvis der i klinikken benyttes ICPC, skal denne oversættes til en passende icd10kode, som indsættes her.
* ***icd10kodebetydning***: En tekstforklaring af hvad icd10 koden betyder. De officielle danske standarder skal benyttes.
* ***icd10kilde***: Hvis værdien er importeret fra en edifakt eller lign, sættes denne til ydernummeret/sks koden for afsenderen. Hvis diagnosen er stillet i egen praksis udfyldes med ”KLINIK”.

### IndsaetDiagnose51

Fra den forrige version af operationen til denne, er parameteren ***privat*** tilføjet.

Kaldes for at overføre information omhandlende diagnoser. Operationen skal kaldes straks efter, at der registreres en diagnose i journalsystemet. Importerede diagnoser skal ligeledes overføres, straks de registreres i journalsystemet. Der skal overføres information, hver gang der registreres en diagnose, også hvis diagnoser bliver kopieret mellem kontaktdiagnoser og forløbs-/kroniske diagnose eller tilsvarende begreber.

Hvis en diagnose ændres i journalsystemet, skal man kalde **SletDiagnose50** (se afsnit 3.2.24) efterfulgt af **IndsaetDiagnose51** med de opdaterede oplysninger.

Man kan initiere integrationen til Sentinel ved at overføre historiske diagnoser. (Husk i dette tilfælde at sætte parameteren ***haendelsestidpunkt*** til det tidspunkt hvor diagnosen blev stillet.)

ApiResultat IndsaetDiagnose51(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string forloebsrolle, bool privat, string icpckode, string kodebetydning, string icpckilde, string icd10kode, string icd10kodebetydning, string icd10kilde);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt hvor diagnosen er stillet. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.
* ***forloebsrolle***: Tildeler diagnoser en bestemt rolle i et forløb. Det kan enten være en aktionsdiagnose eller en bidiagnose.
* ***privat:*** Angiver om diagnosen er stillet i forbindelse med et privat betalt forløb. Hvis det er offentligt betalt forløb, skal denne udfyldes med *false*.
* ***icpckode***: Officiel kode for diagnosen. De officielle MedCom standarder skal benyttes. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***kodebetydning***: Den officielle tekst relateret til icpckoden. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***icpckilde***: Hvis værdien er importeret fra en edifakt eller lign, sættes denne til ydernummeret/sks koden for afsenderen. Hvis diagnosen er stillet i egen praksis udfyldes med ”KLINIK”. Hvis en praksis ikke benytter icpc, sættes denne til null.
* ***icd10kode***: Den officielle icd10 kode for denne oplysning. Hvis der i klinikken benyttes ICPC, skal denne oversættes til en passende icd10kode, som indsættes her.
* ***icd10kodebetydning***: En tekstforklaring af hvad icd10 koden betyder. De officielle danske standarder skal benyttes.
* ***icd10kilde***: Hvis værdien er importeret fra en edifakt eller lign, sættes denne til ydernummeret/sks koden for afsenderen. Hvis diagnosen er stillet i egen praksis udfyldes med ”KLINIK”.

### IndsaetYdelse50

Kaldes for at overføre information omhandlende ydelser, der er registreret relateret til en patient. Operationen skal kaldes straks efter, at ydelsen er registreret i journalsystemet.

Hvis en ydelse ændres i journalsystemet, skal man kalde **SletYdelse50** (se afsnit 3.2.25) efterfulgt af en **IndsaetYdelse50** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historiske ydelsesoplysninger fra journalsystemet. (Husk i dette tilfælde at sætte parameteret ***haendelsestidspunkt*** til det tidspunkt hvor ydelsen blev registreret.)

ApiResultat IndsaetYdelse50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string ydelseskode, string ydelsestekst);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt, hvor ydelsen er opkrævet. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.
* ***ydelseskode***: Den officielle kode for ydelsen, der står i ydelsesregisteret for det givne speciale.
* ***ydelsestekst***: Den officielle tekst relateret til ydelseskoden.

### IndsaetPost50

Kaldes for at overføre information om den elektroniske post, der bruges til kommunikation med det øvrige sundhedsvæsen. Operationen skal kaldes straks efter, at man genererer eller importerer elektronisk post. Man skal som minimum overføre henvisninger og epikriser, men der kan forekomme specialeafhængige krav til at overføre andre kommunikationsstandarder.

Hvis en elekronisk kommunikation ændres i journalsystemet, skal man kalde **SletPost50** (se afsnit 26) efterfulgt af en **IndsaetPost50** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historisk elekronisk kommunikation fra journalsystemet.

ApiResultat IndsaetPost50(string systemlogindtoken, string klientid, string lokalid, string ydernr, string initialer, bool udgaaende, string post, string forloebsid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***udgaaende***: Angives til *true*, hvis post sendes fra klinikken. Angives til *false,* hvis klinikken modtager post.
* ***post***: Den rå tekst fra den elektroniske post (edifakt eller medcom xml), der sendes eller modtages. Som minimum skal alle Henvisninger og Epikriser overføres til Sentinel. Om der skal overføres andet end henvisninger og epikriser, afhænger af individuelle aftaler.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.

### IndsaetBehandling50

Benyttes endnu ikke, da der ikke er aftalt nærmere, hvilke behandlingsregistre, der evt. kan benyttes for de enkelte fagområder.

ApiResultat IndsaetBehandling50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string kode, string kodeansvarlig, string kodebetydning, string kommentarer);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt, hvor behandlingen er udført. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.
* ***kode***: behandlingskoden, fra det valgte behandlingsregister.
* ***kodeansvarlig***: Se beskrivelse i afsnit 3.2.10.
* ***kodebetydning***: En tekst, der forklare hvad koden betyder. Her benyttes den beskrivelse, som er givet i det valgte behandlingsregister.
* ***kommentarer***: En tekst, hvor behandleren kan uddybe eller kommenterer på behandlingen.

### IndsaetNotat50

Benyttes endnu ikke. Der arbejdes i øjeblikket på at lave aftaler omkring overførsel af notater for speciallæger. Kan fremtidigt evt. også blive relevant for andre fagområder.

ApiResultat IndsaetNotat50(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, string forloebsrolle, string notattype, string overskrift, string notat);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt, hvor notatet er skrevet. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.
* ***forloebsid***: Se beskrivelse i afsnit 3.2.10.
* ***forloebsrolle***: Har til formål at kunne tildele notatet en bestemt rolle i forhold til et forløb.
* ***notattype***: Typen af notat.
* ***overskrift***: Notatets overskrift
* ***notat***: Selve teksten/indholdet af notatet.

### IndsaetForloeb51

**IndsaetForloeb51** er endnu ikke obligatoriske, men skal benyttes hvis man kører forløbsorienterede journalføring i praksissystemet. Forløb oprettet i journalsystemet, skal overføres med **IndsaetForloeb51**. Øvrige oplysninger som overføres til Sentinel og som i journalsystemet er knyhttet til et forløb, skal markeres med relationen i feltet ***ForloebsId***.

Hvis et forløb ændres i journalsystemet, skal man kalde **SletForloeb51** (se afsnit 3.2.29) efterfulgt af en **IndsaetForloeb51** med de opdaterede oplysninger.

Man kan initiere integrationen med Sentinel ved at overføre historisk forløb fra journalsystemet.

ApiResultat IndsaetForloeb51(string systemlogindtoken, string klientid, string lokalid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string forloebsid, bool privat, string sluttidspunkt);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Se beskrivelse i afsnit 3.2.10.
* ***haendelsestidspunkt***: Det tidspunkt, hvor forløbet er startet op. Se tidsformat i afsnit 3.2.10.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: Patientens cpr.
* ***forloebsid***: forløbets identifikation. Alle oplysninger/data, der er knyttet til dette forløb skal have angivet dette ***forloebsid*** i deres parameter ***forloebsid***.
* ***privat:*** Se beskrivelse i afsnit 3.2.13.
* ***sluttidspunkt:*** Angiver tidspunktet for forløbets afslutning. I såfald man ikke kender sluttidspunktet angives null.

### RegistrerHaendelse50

Benyttes efter aftale til at overføre forskellige nødvendige hændelser eks. sygesikringskort læst m.m. Der er i øjeblikket ikke aftalt nogen hændelser, som skal overføres via **RegistrerHaendelse50**.

ApiResultat RegistrerHaendelse50(string systemlogindtoken, string klientid, string haendelsestidspunkt, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr, string haendelsesKode, string haendelsesData);

### AabnerJournal50

Operationen skal kaldes hver gang en bruger åbner en patientjournal.

ApiResultat AabnerJournal50(string systemlogindtoken, string klientid, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: Patientens cpr.

### LukkerJournal50

Operationen skal kaldes hver gang en bruger lukker en patientjournal.

ApiResultat LukkerJournal50(string systemlogindtoken, string klientid, string ydernr, string initialer, string egenlaegeydernr, string egenlaegeinitialer, string cpr);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***egenlaegeydernr***: Se beskrivelse i afsnit 3.2.9.
* ***egenlaegeinitialer***: Se beskrivelse i afsnit 3.2.9.
* ***cpr***: patientens cpr.

### SletAnalyse50

Hvis en patientværdi slettes i journalsystemet, eksempelvis fordi den var fejlagtigt registreret, skal man kalde **SletAnalyse50**.

ApiResultat SletAnalyse50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetAnalyse50**.

### SletMedicin50

Hvis en recept slettes i FMK, eksempelvis fordi den var fejlagtigt registreret, skal man kalde **SletMedicin50**. Man skal ikke foretage sletning, hvis en recept seponeres, medmindre patienten ikke har haft en chance for at få præparatet udleveret.

ApiResultat SletMedicin50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetMedicin50**.

### SletDiagnose50

Hvis en diagnose slettes i journalsystemet, skal man kalde **SletDiagnose50**.

ApiResultat SletDiagnose50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetDiagnose50** eller **IndsaetDiagnose51**.

### SletYdelse50

Hvis en ydelse slettes i journalsystemet, eksempelvis fordi den var fejlagtigt registreret, skal man kalde **SletYdelse50**.

ApiResultat SletYdelse50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetYdelse50**.

### SletPost50

Hvis en elekronisk kommunikation slettes i journalsystemet, eksempelvis fordi den var fejlagtigt produceret, skal man kalde **SletPost50**.

ApiResultat SletPost50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetPost50**.

### SletBehandling50

Benyttes endnu ikke. Se forklaring i afsnit 3.2.16.

ApiResultat SletBehandling50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetBehandling50**.

### SletNotat50

Benyttes endnu ikke. Se forklaring i afsnit 3.2.17.

ApiResultat SletNotat50(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetNotat50**.

### SletForloeb51

Hvis en ydelse slettes i journalsystemet, eksempelvis fordi den var fejlagtigt registreret, skal man kalde **SletForloeb51**.

**SletForloeb51** er endnu ikke obligatoriske, men skal benyttes hvis man kører forløbsorienterede journalføring i praksissystemet.

ApiResultat SletForloeb51(string systemlogindtoken, string klientid, string lokalid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.
* ***lokalid***: Journalsystemets eget globalt unikke id på denne information (record). Samme id, som tidligere er benyttet i overførslen af denne record ved brug af**IndsaetForloeb51**.

### HentMenu50

Benyttes til dynamisk at hente en Sentinel menu, med de klinik- og patientopgaver, som en bruger kan, bør og skal udføre i forbindelse med administration af Sentinel.

Praksissystemet skal implementere to Sentinel-relaterede knapper/menupunkter, som brugeren kan klikke på, når denne ønsker at udføre forskellige Sentinel-opgaver.

De to knapper skal hhv. hente patientrelaterede og praksisrelaterede Sentinel-opgaver.

Når en bruger trykker på en af menuerne, kaldes **HentMenu50** og journalsystemet skal præsentere brugeren for de returnerede menupunkter.

Hvis brugeren efterfølgende klikker på en af de returnerede menupunkter, skal journalsystemet udføre den tilknyttede aktion. En tilknyttet aktion er enten en WebAktion eller en ApiAktion. Hvis der er tilknyttet en WebAktion, skal et browservindue med den angivne url åbnes. Hvis der er tilknyttet en ApiAktion, så udføres aktionen ved at kalde API-operationen **UdfoerAktion50** med den pågældende aktion som argument.

Et menupunkt kan indeholde underpunkter. Her gentages processen beskrevet ovenfor.

Menuerne skal indopereres i journalsystemet, så de ligner resten af systemet og holder systemets egne konventioner for visuel visning af prioritet.

ApiMenuResultat HentMenu50(string systemlogindtoken, string klientid, string ydernr, string initialer, string cpr);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***ydernr***: Se beskrivelse i afsnit 3.2.9.
* ***initialer***: Se beskrivelse i afsnit 3.2.5.
* ***cpr***: patientens cpr, hvis man ønsker at hente de patientrelaterede menupunkter, ellers angives null.

Returnerer

public class ApiMenuResultat: ApiResultat

{

public List<ApiMenuPunkt> MenuPunkter { get; set; }

}

public class ApiMenuPunkt

{

public Guid Id { get; set; }

public string Tekst { get; set; }

public string Beskrivelse { get; set; }

public List<ApiMenuPunkt> UnderMenuPunkter { get; set; }

public ApiPrioritet Prioritet { get; set; }

public string PrioritetNavn { get;set;}

public ApiAktionType Aktion { get; set; }

}

public class ApiAktionType

{

public Uri WebAktion { get; set; }

public string ApiAktion { get; set; }

}

public enum ApiPrioritet {mulig=0, anbefalet=1, paakraevet=2 }

### HentAktioner50

Operationen kaldes for at undersøge om Sentinel har popups eller meddelelser, som skal præsenteres for brugeren. Operationen kaldes hvert 3. sekund og skal altid kaldes umiddelbart efter, at der er foretaget overførsel af data eller events – dvs. kald til indsættelser, sletninger, åbner journal, lukker journal, registrer hændelse, login og logud. Operationen returnerer, hvornår næste kald skal foretages, hvis der ikke i mellemtiden overføres data.

Operationen returnerer en liste af meddelelser og en liste af straksaktioner.

Straksaktioner skal udføres øjeblikkeligt. Hvis den tilhørende aktion er en WebAktion, skal der åbnes et browservindue med den pågældende url. Hvis aktionen er en ApiAktion, skal man straks kalde **UdfoerAktion50** med den pågældende aktion som argument.

Meddelelser skal præsenteres for brugeren. Hvis brugeren klikker på meddelelsen, skal den tilhørende aktion (hvis der er en tilknyttet) udføres. Hvis den tilhørende aktion er en WebAktion, skal der åbnes et browservindue med den pågældende url. Hvis det er en ApiAktion, skal man straks kalde **UdfoerAktion50** med den pågældende aktion som argument.

Meddelelser har en meddelelsestype. Den kan have udfaldsrummet: information=0, vigtiginformation=1, advarsel=2, fejl=3. Man bør vælge en præsentation (evt. farve), som afspejler tilsvarende beskedtyper i journalsystemet.

Meddelelser har en overskrift og en tekst, der skal præsenteres for brugeren.

En meddelelse skal fjernes, så snart en bruger klikker på den og den evt. tilhørende aktion udføres. Meddelelser har en property *SynligSek*, som angiver hvor lang tid der bør gå, før meddelelsen forsvinder, hvis brugeren ikke klikker på den. Hvis *SynligSek*=0 bør meddelelsen kun fjernes når brugeren aktivt klikker på den.

Meddelelser har et unikt Id. Man bør aldrig vise samme meddelelse (samme id) flere gange samtidigt.

Hvis man ikke mener at kunne implementere meddelelser i praksissystemet, kan man angive i kaldet, at meddelelser skal autohåndteres. Hvis dette angives, bliver meddelelser omformet til strakshandlinger. Udførslen af strakshandlingen resulterer i, at der åbnes et nyt browservindue, hvor meddelelserne bliver præsenteret og håndteret for brugeren.

ApiAktionerResultat HentAktioner50(string systemlogindtoken, string klientid, bool autohaandtermeddelelser);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***Autohaandtermeddelelser:*** Bør altid være false.

Returnerer

public class ApiAktionerResultat:ApiResultat

{

public ApiStatus Status { get; set; }

public string StatusNavn { get; set;}

public int NaesteKald { get; set; }

public List<ApiMeddelelse> Meddelelser { get; set; }

public List<ApiAktionType> StraksAktioner { get; set; }

}

public class ApiMeddelelse

{

public Guid Id { get; set; }

public ApiMeddelelsesType MeddelelsesType { get; set; }

public string MeddelelsesTypeNavn { get; set; }

public int SynligSek { get; set; }

public string Overskrift { get; set; }

public string Tekst { get; set; }

public ApiAktionType Aktion { get; set; }

}

public class ApiAktionType

{

public Uri WebAktion { get; set; }

public string ApiAktion { get; set; }

}

public enum ApiStatus { fejl = -1, ok = 0, behandlerdata = 1 }

public enum

ApiMeddelelsesType { information=0, vigtiginformation=1, advarsel=2, fejl=3 };

### UdfoerAktion50

Benyttes til at få udført ApiAktioner, som kan være tilknyttet menupunkter, meddelelser og strakshandlinger.

ApiResultat UdfoerAktion50(string systemlogindtoken, string klientid, string Aktion);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.5.
* ***Aktion***: Den ApiAktion, der skal udføres. ApiAktioner kan være tilknyttet menupunkter, meddelelser og strakshandlinger. De skal udføres, hvis der er tale om en straksaktion, eller hvis brugeren klikker på meddelelsen eller menupunktet.

### HentVersion

Benyttes til at hente information om, hvilken version af API’et, der er udstillet. Sentinel-API’et vil altid være bagudkompatibel, dvs. i version 5.1 vil det stadigvæk være muligt at kalde alle operationer fra version 5.0.

Der kan dog være introduceret nye eller opdaterede operationer, som systemhusene, efter aftale, skal benytte fremover.

Alle operationer har et nummer. Hvis der evt. kommer en opdatering til **Indsaetdiagnose** i version 5.1, så vil denne operation få navnet **Indsaetdiagnose51** og **Indsaetdiagnose50** vil fortsat være en del af API’et.

**HentVersion** gør det muligt for journalsystemet at afgøre, om en ny operation er udstillet eller om man evt. fortsat skal kalde en tidligere version. Dette gør at systemhuset kan implementerer nye operationer, så snart en ny snitflade er aftalt. I en overgangsperiode kan de kalde ’gamle’ operationer, hvis systemhuset får lavet en opdatering til eks. **IndsaetDiagnose51** inden Sentinel får sat det i produktion. Det er derfor vigtigt, at praksissystemet kan falde tilbage på den tidligere implementering.

ApiVersionResultat HentVersion();

Returnerer:

public class ApiVersionResultat:ApiResultat

{

public int Primaer { get; set; }

public int Sekundaer { get; set; }

public int Byg { get; set; }

}

### LukSentinel50

Lukker en evt. kørende Sentinel Klient. Benyttes kun i meget specielle tilfælde, og udelukkende efter aftale med Sentinel, sundhed.dk.

ApiResultat LukSentinel50(string systemlogindtoken, string klientid);

* ***systemlogindtoken***: Se beskrivelse i afsnit 3.2.5.
* ***klientid***: Se beskrivelse i afsnit 3.2.9.

## Eksempel på kald af SentinelAPI’et fra WebKlient via Ajax

require(['jquery'], function($) {

//FØRST OPRETTES EN SentinelAPI KLASSE, MED ALLE OPERATIONERNE

//DER BRUGES PROTOTYPE BASERET KLASSER, AF HENSYN TIL KOMBABILITET.

function SentinelAPI(uri) {

if (!uri)

uri = 'https://web.sentinel-test.dk/Services/SentinelJSonAPI.svc/';

this.uri = uri;

}

//PROTOTYPE

//GENERISK AJAX KALD SOM BENYTTES I ALLE DE ØVRIGE OPERATIONER.

SentinelAPI.prototype.AjaxCall = function (operation, data) {

var dfd = jQuery.Deferred();

try {

$.ajax({

type: "POST",

contentType: "application/json; charset=utf-8",

url: this.uri + operation,

data: (data)?JSON.stringify(data):null,

processData: false,

dataType: "json",

success: function (response) {

try {

var result = eval(response.d);

dfd.resolve(result);

} catch (serror) { dfd.reject({ jqXHR: null, textStatus: "ukendt fejl", errorThrown: serror }); }

},

error: function (a, b, c) {

dfd.reject({ jqXHR: a, textStatus: b, errorThrown: c });

}

});

} catch (error) { dfd.reject({ jqXHR: null, textStatus: "ukendt fejl", errorThrown: error }); }

return dfd.promise();

};

SentinelAPI.prototype.HentVersion = function () {

return this.AjaxCall('HentVersion');

}

SentinelAPI.prototype.SystemLogInd50 = function (systemnavn, encsystemnoegle, systemversion){

return this.AjaxCall('SystemLogInd50', {

'systemnavn': systemnavn, 'encsystemnoegle': encsystemnoegle, 'systemversion': systemversion

});

}

SentinelAPI.prototype.LogInd50 = function (systemlogindtoken, klientid, ydernr, initialer) {

return this.AjaxCall('LogInd50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'ydernr': ydernr, 'initialer': initialer

});

}

SentinelAPI.prototype.LogUd50 = function (systemlogindtoken, klientid, ydernr, initialer) {

return this.AjaxCall('LogUd50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'ydernr': ydernr, 'initialer': initialer

});

}

SentinelAPI.prototype.LogIndSession50 = function (systemlogindtoken, klientid, ydernr, initialer, sessionId) {

return this.AjaxCall('LogIndSession50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'ydernr': ydernr, 'initialer': initialer, 'sessionId': sessionId

});

}

SentinelAPI.prototype.LogUdSession50 = function (systemlogindtoken, klientid, ydernr, initialer, sessionId) {

return this.AjaxCall('LogUdSession50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'ydernr': ydernr, 'initialer': initialer, 'sessionId': sessionId

});

}

SentinelAPI.prototype.OpdaterStamdata50 = function (systemlogindtoken, klientid, ydernr, initialer, egenlaegeydernr, egenlaegeinitialer, cpr, fornavn, efternavn, regionsKode, emailadresse) {

return this.AjaxCall('OpdaterStamdata50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'ydernr': ydernr, 'initialer': initialer, 'egenlaegeydernr': egenlaegeydernr, 'egenlaegeinitialer': egenlaegeinitialer, 'cpr': cpr, 'fornavn': fornavn, 'efternavn': efternavn, 'regionsKode': regionsKode, 'emailadresse': emailadresse

});

}

SentinelAPI.prototype.IndsaetAnalyse50 = function (systemlogindtoken, klientid, lokalid, haendelsestidspunkt, ydernr, initialer, egenlaegeydernr, egenlaegeinitialer, cpr, kode, kodeansvarlig, kodebetydning, producentKode, producent, \_operator, vaerdi, enhed, refFra, refTil, refText, abnorm, proevetagningsTidspunkt, kommentarer) {

return this.AjaxCall('IndsaetAnalyse50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'lokalid': lokalid, 'haendelsestidspunkt': haendelsestidspunkt, 'ydernr': ydernr, 'initialer': initialer, 'egenlaegeydernr': egenlaegeydernr, 'egenlaegeinitialer': egenlaegeinitialer, 'cpr': cpr, 'kode': kode, 'kodeansvarlig': kodeansvarlig, 'kodebetydning': kodebetydning, 'producentKode': producentKode, 'producent': producent, '\_operator': \_operator, 'vaerdi': vaerdi, 'enhed': enhed, 'refFra': refFra, 'refTil': refTil, 'refText': refText, 'abnorm': abnorm, 'proevetagningsTidspunkt': proevetagningsTidspunkt, 'kommentarer': kommentarer

});

}

SentinelAPI.prototype.IndsaetDiagnose50 = function (systemlogindtoken, klientid, lokalid, haendelsestidspunkt, ydernr, initialer, egenlaegeydernr, egenlaegeinitialer, cpr, icpckode, kodebetydning, icpckilde, icd10kode, icd10kodebetydning, icd10kilde) {

return this.AjaxCall('IndsaetDiagnose50', {

'systemlogindtoken': systemlogindtoken, 'klientid': klientid, 'lokalid': lokalid, 'haendelsestidspunkt': haendelsestidspunkt, 'ydernr': ydernr, 'initialer': initialer, 'egenlaegeydernr': egenlaegeydernr, 'egenlaegeinitialer': egenlaegeinitialer, 'cpr': cpr, 'icpckode': icpckode, 'kodebetydning': kodebetydning, 'icpckilde': icpckilde, 'icd10kode': icd10kode, 'icd10kodebetydning': icd10kodebetydning, 'icd10kilde': icd10kilde

});

…

//OPRETTER SENTINELAPI OBJEKT

var api = new SentinelAPI();

$(function(){

//TESTER LIDT KALD EFTER AT SIDEN ER LOADED (resultatet sendes til en alert for at se resultatet)

api.IndsaetDiagnose50($("#systemlogindtoken").val(), $("#klientid").val(), "DID\_35454326654365435432", "2020-02-10T10:11:27", "123456", "fbh", "234567", "dontknow", "1001172021", 'T90', 'Diabetes', 'klinik', 'ax193', 'Diabetes', 'klinik')

.done(function (result) { alert(JSON.stringify(result)); })

.fail(function (error) { alert(JSON.stringify(error)); });}

api.HentMenu50($("#systemlogindtoken").val(), $("#klientid").val(), "123456", "fbh","1001172021")

.done(function (result) { alert(JSON.stringify(result)); })

.fail(function (error) { alert(JSON.stringify(error)); });

…

});

});

# Hvad stilles der til rådighed for udvikling og test

Der bliver oprettet et Test site med Sentinel-API’et på adresserne:

* <https://web.sentinel-test.dk/Services/SentinelJSonAPI.svc>
* <https://web.sentinel-test.dk/Services/SentinelSoapAPI.svc>
* <https://web.sentinel-test.dk/Services/SentinelJSonEnAPI.svc>
* <https://web.sentinel-test.dk/Services/SentinelSoapEnAPI.svc>

for hhv:

* det danske JSON baserede API
* det danske SOAP baserede API
* det engelske JSON baserede API
* det engelsek SOAP baserede API

Man kan få adgang til en WSDL på de SOAP baserede API addresser.

For at få adgang til at benytte disse, skal der udveksles information mellem systemhuset og Sentinel, sundhed.dk:

1. Systemhuset skal komme med et navn, som de ønsker at benytte til deres system.
2. Sentinel, sundhed.dk sender en systemnogle til systemhuset. Systemnøglen genereres på baggrund af det aftalte systemnavn.
3. Systemhuset skal sende en public key til et certifikat, som de ønsker at benytte i forbindelse med SSO kald.
4. Sentinel, sundhed.dk registrerer certifikatet, og der vil derefter være åbent for, at systemet kan kalde de udstillede API’er.

For at kalde operationerne i SentinelAPI’et skal praksissystemet generere en ***systemlogintoken***.Til dette skal ovennævnte systemnavn, systemnøgle og certifikatets public key benyttes. Husk at en ***systemlogintoken*** kun virker samme dag, som den er udstedt! Følgende pseudokode viser, hvordan en ***systemlogindtoken*** kan genereres:

var noegleresultat = api.GenererSystemNoegle50(sytemnavn,systempassword);

CheckApiResult(noegleresultat);

var encsystemnoegle = noegleresultat.Token;

var systemnoegle = Decrypt(encsystemnoegle, sytemhusetscertifikat);

var systemlogindresult=api.SystemLogind50(systemnavn, systemnoegle,”2.90.32”);

CheckApiResult(systemlogindresult);

var systemlogintoken = systemlogindresult.Token;

public static string Decrypt(string encryptedText, X509Certificate2 cert)

{

RSACryptoServiceProvider privateKey = (RSACryptoServiceProvider)(cert.PrivateKey);

byte[] encryptedBytes = Convert.FromBase64String(encryptedText);

byte[] decryptedBytes = privateKey.Decrypt(encryptedBytes,false);

string decryptedText = Encoding.UTF8.GetString(decryptedBytes);

return decryptedText;

}

public static string CheckApiResult(ApiResultat result)

{

if (noegleresultat.StatusKode != 0)

throw new Exception($”Fejl I kald til Api {noegleresultat.StatusKode}”);

}

Har man spørgsmål i forbindelse med implementering, er man velkommen til at kontakte Flemming Bo Hansen og/eller Søren Rathje Jensen fra Sentinel, sundhed.dk på [fbh@sundhed.dk](mailto:fbh@sundhed.dk) og/eller [srj@sundhed.dk](mailto:srj@sundhed.dk).

# Hvad skal der til for at systemhuset kan gå i produktion?

Systemhusene og Sentinel, sundhed.dk indgår en samarbejdsaftale om integrationen af Sentinel forud for certificeringen.

Systemhuset skal implementere integrationen iht. krav og beskrivelse præsenteret i dette dokument.

Der bliver afholdt en Workshop, hvor integrationen bliver gennemgået i detaljer. På workshoppen vil Sentinel, sundhed.dk være til rådighed med hjælp og support.

Godkendelsen af systemet foregår ved en certificering (certificeringsprocessen gennemgås i afsnit 6).

# Certificering af praksissystemer

Systemhuset skal inden certificeringen sikre sig, at journalsystemet opfylder de regler, som er beskrevet i dette dokuments affsnit 3. Det er vigtigt, at systemhuset sikrer sig, at regler for integrationen er opfyldt, uanset hvordan en bruger udfører en arbejdsgang i journalsystemet.

Når systemhuset har sikret sig, at integrationen opfylder alle regler, aftales et tidspunkt, for certificering – kontakt Søren Rathje Jensen [srj@sundhed.dk](mailto:srj@sundhed.dk)

Certificeringen foregår som onlinemøde af nogle timers varighed og består af en gennemgang af en række usecases. Systemhusene kender ikke usecasene på forhånd, da formålet er at afdække om implementeringen opfylder krav til integration med Sentinel. Når alle usecases er bestået, er journalsystemet godkendt til integrationen. Systemhusene kan efter nærmere aftale begynde udrulningen af integrationen op mod produktionsmiljøet.