R-RCT

Nationella riktlinjer R-RCT

Bilaga: Utvecklardokumentation

Innehållsförteckning

1. Introduktion

1.1. Syfte

Detta dokuments målgrupp är utvecklare som vill förstå och kunna utveckla kliniska studier med hjälp av R-RCT-ramverket.

2. Domänen ”kliniska prövningar”

2.1. Sites

Sites är de enheter som deltar i studien. De kan vara aktiva eller inaktiva. Om ett site är aktivt kan screening och randomisering göras (se nedan).

2.2. Prövare

En prövare (”investigator” på engelska) är en läkare på ett site som har rätt att rekrytera patienter till aktuell studie.

2.3. Study subject

Study subject är en person som ingår i en studie. Hen är randomiserad att tillhöra en studiegrupp och identifieras med ett subject-Id som är en unik identifierare i studien. Subject-Id är ekvivalent med randomiseringsnumret, så dessa två termer betyder samma sak och används ofta om vartannat.

2.4. Screening och randomisering

Man delar upp processen med att försöka rekrytera en patient i studien i tre steg där nästa steg bara utförs om aktuellt steg får en positiv utgång:

1. "Pre-screening" där registret eller prövaren, ofta informellt, kontrollerar om patienten alls är lämplig för studien (svenskt personnummer? Rätt diagnos?). För denna del finns inget stöd i R-RCT-ramverket.
2. "Screening" där prövaren avgör om patienten uppfyller formella kriterier för deltagande i studien.
3. "Randomisering" eller "inklusion" där prövaren inkluderar patienten i studien. Patienten är därefter ett study subject.

De två sista stegen dokumenteras i Screening log och/eller Subject log (se nedan).

2.5. Screening log

I Screeningloggen dokumenteras samtliga screeningtillfällen i studien. Det är en lista som innehåller:

1. För vem/vid vilket vårdtillfälle screening gjordes (till exempel vid ett visst återbesök för en viss patient).
2. Av vem och när screening gjordes.
3. Om personen inkluderades innehåller loggen ett subject-Id och en studiegrupp.
4. Om personen inte inkluderades innehåller loggen en anledning till detta.
5. Det lagras även något som i koden kallas personidentifierare på den som screenats. Detta har i tidigare versioner av R-RCT-ramverket varit patientens personnummer. **På UCR bedömer vi numera att detta inte är förenligt med datalagringslagarna, och vid screeningtillfället ser vi därför till att registret istället för personnummer bara skickar ett godtyckligt nummer**. Se vidare i API-beskrivningen för screening i detta dokument.

2.6. Subject log (även kallad enrollment log)

I Subject log dokumenteras samtliga inkluderade personer. Denna lista innehåller:

1. Vem som inkluderades (personnummer).
2. Av vem och när inklusionen gjordes.
3. Vid vilket vårdtillfälle inklusionen gjordes (till exempel vid ett visst återbesök).
4. Subject-Id/randomiseringsnummer.
5. Studiegrupp.

2.7. CDISC-standarder

CDIC är en sammanslutning som bland annat definierat ett antal formatstandarder som används allmänt inom kliniska studier. För R-RCT används i synnerhet "Operational Data Model (ODM)-XML":

> *ODM-XML is a vendor-neutral, platform-independent format for exchanging and archiving clinical and translational research data, along with their associated metadata, administrative data, reference data, and audit information.*

För att bygga en studie med R-RCT-ramverket krävs en odm-fil som beskriver studien. Program (till exempel en EDC) som är till för att bygga kliniska studier kan alltid exportera studien som en sådan fil. Det är naturligtvis också möjligt att sätta ihop den för hand.

I odm-standarden beskrivs exempelvis hur studiedata ska namnges/identifieras: Varje variabel som man kan rapportera in data på (till exempel "uppmätt blodtryck vid initialbesöket") identifieras med ett sorts hierarkiskt Id byggt av:

- Namn på aktuell vårdhändelse ("Study Event")
------ Namn på aktuellt formulär som fylls i under vårdhändelsen ("Form")
---------- Namn på grupp av relaterade variabler inom detta formulär ("Item Group")
---------- Namn på variabeln i gruppen av variabler ("Item")

I R-RCT-ramverket har vi utifrån detta valt att alltid namnge variabler med en Id-sträng uppbyggd av det ovan. Exempelvis identifierar strängen

```
"INIT::1::BASIC::1::UNGROUPED::1::WEIGHT"
```

en variabel "WEIGHT" som samlas in vid det initiala patientbesöket "INIT" i ett formulär som heter "BASIC", i variabelgruppen "UNGROUPED".

Kanske vägs patienten även vid ett återbesök, i så fall kanske variabeln istället heter
`"VISIT2:1::BASIC::1::UNGROUPED::1::WEIGHT"`

3. Kodgenomgång

3.1. Använd teknik

Modulerna byggs med Maven som kompilerar med Java 8. Spring 3, konfigurerad med xml samt annoteringar, används för att knyta ihop all kod och tillsammans med Hibernate för access mot databaser. För REST-gränssnitten används RestEasy. I databasväg har MySql och HSQLDB använts, men andra databaser bör inte vara något problem. Själva databaserna sätts upp med Liquibase. I några fall skickas och hanteras commands och events, för detta används Axon. Det är också Axon-events som bygger upp en eventsource-databastabell för studiedata.

3.2. Projektstruktur

Programprojektet R-RCT är uppbyggt av mindre moduler där somliga är fristående och andra beroende av varandra. Varje byggd modul (utom demo-modulen) resulterar i en jar-fil. För att bygga en R-RCT skapar man en webbapplikation (*.war) som använder och konfigurerar dessa jar-filer. Man bygger alltså en specifik webbapplikation för varje studie, vi brukar kalla denna för ”studieappen”. Konfigurationen innebär i stort att konfigurera springbönor, sätta upp accesskontroll, peka ut liquibase-skript och liknande. Dessutom innehåller konfigurationen studiens odm-fil, där studiens struktur definieras. Som ett exempel på hur en studieapp kan byggas finns modulen clinicaltrial-demo, som precis som namnet antyder är en minimal demostudie.

3.2.1. Moduler och deras ansvar

Nedan är en uppräkning av en delmängd av de ingående modulerna. Mer detaljerad information kan finnas i varje moduls README.md-fil.

Hantering av själva studien och dess siter

Modulen **clinicaltrial-studymanagement** innehåller serverkod som tar hand om de anrop som kommer genom Study Management API. En javaklient för att anropa detta API byggs av modulen **clinicaltrial-studymanagement-client**.

Hantering av screening och randomisering

Modulen **clinicaltrial-subjectmanagement** innehåller serverkod som tar hand om de anrop som kommer genom Screening Log API och Subject Log API. Javaklienter för att anropa dessa API byggs av modulen **clinicaltrial-subjectmanagement-client**.

Lagring och uppläsning av studiedata för en deltagare i studien

Modulen **clinicaltrial-studydata** innehåller serverkod som tar hand om de anrop som kommer genom Study Data API. En javaklient för att anropa dessa API byggs av modulen **clinicaltrial- studydata-client**.

Rapporter

Modulen **clinicaltrial-report** innehåller serverkod som tar hand om de anrop som kommer genom Report API. En javaklient för att anropa dessa API byggs av modulen **clinicaltrial-report-client**.

Export av studiedata

Modulen **clinicaltrial-studydataexport** innehåller serverkod som tar hand om de anrop som kommer genom Study Data Export API.

3.3. Generell kodstruktur

3.3.1. Lager

I allmänhet kommer ett inkommande REST-anrop att hanteras så här:

- På serversidan implementeras API-interfacen av klasser som slutar på `Resource`. Ett anrop till en metod som deklareras i `StudyDataAPI`, har sin implementation i `StudyDataResource.`
- Resourceklassen ska bara ha ansvar för omvandling mellan API-objekt och domänobjekt, för att sedan låta en serviceklass sköta logiken. Serviceklasserna är i allmänhet också namngivna så att `StudyDataResource` anropar `StudyDataService`.
- Serviceklasserna har i sin tur tillgång till `Repository`-klasser som i många fall bara är ett Java-Interface som implementeras automatiskt av Spring. `Repository`-klasserna är döpta efter det objekt de hanterar i databasen, så `ScreeningLogRepository` är Interface:et som sköter databasaccess för `ScreeningLog`-objekt.

3.4. Speciella konstruktioner

3.4.1. Axon

Axon Framework (http://www.axonframework.org/) används för att kunna spara all studiedata med *event sourcing*, se till exempel https://msdn.microsoft.com/en-us/library/jj591559.aspx för en genomgång av fördelarna med detta. Kortfattat kan man säga att event sourcing ger oss en perfekt audit log över hur studiedata sparats, vilket kan vara viktigt vid kliniska studier. Dock bygger idén på att alla förändringar av studiedata representeras av Commands och Events, vilket gör att `StudyDataService`-implementationen `DefaultStudyDataService` använder sig av en `AxonService` istället för att spara data med en Repository-klass.

`AxonService` i sin tur använder Axon-interna anrop för att spara händelserna i databasen, det sker under ytan i anropet `commandGateway.sendAndWait(storeInEDCCommand)`.

Hela denna hantering är i nuläget inte gjord på bästa sätt, men slutresultatet överensstämmer med vad vi vill uppnå.

4. Hur gör man för att …

4.1. Lägga till sin egen odm-fil?

Namnet på odm-filen ska stå i studieappens propertyfil. I demoprojektet finns till exempel `demostudy.properties` som pekar ut filen `demo-study-definition-odm.xm.` Denna fil måste sedan ligga någonstans i classpathen.

4.2. Konfigurera randomiseringstjänsten?

I demoprojektet används en dummyimplementation av randomiseringstjänsten som bara returnerar ett slumpmässigt randomiseringsnummer och en av de konfigurerade studiegrupperna. Detta konfigureras i en av springkonfigurationsfilerna med:

```
<bean id="randomizationSource"
class="se.uu.ucr.clinicaltrial.randomization.domain.JvmRandomizationSource"/>
```

Precis som det står i kommentaren till konfigurationen i demoprojektet (`application-context.xml`), ska man för en riktig studie istället peka ut klassen `PregeneratedJpaRandomizationSource`. Se i nämnda kommentar exakt hur konfigurationen ska se ut.

Denna produktionsdugliga randomiseringstjänst utgår från att det finns en randomiseringslista inlagd i tabellen **randomization**. De viktigaste kolumnerna i denna tabell ser ut och fungerar som följer:

RANDOMIZATION_ NUMBER	STUDY_ GROUP	STRATIFICATION_ PARAM0	STRATIFICATION_ PARAM1	TAKEN
SE010000	GROUP_A	SE01		1
SE010000	GROUP_B	SE01		0

Kolumnnamn	Beskrivning
RANDOMIZATION_NUMBER	Det randomiseringsnummer som kommer att tilldelas en patient.
STUDY_GROUP	Den studiegrupp den randomiserade patienten placeras i.
STRATIFICATION_PARAM0	Stratifiering är ett statistiskt begrepp som i praktiken innebär att randomiseringsnumren grupperas. Det är inget som krävs och om det inte används lämnas bara kolumnen tom. Absolut vanligast, som i exemplet ovan, är att stratifiera på site. Det innebär att randomiseringslistorna är gjorda så att studiegrupperna blir jämt fördelade per site. I tabellen måste det då finnas angivet för vilket site som varje randomiseringsnummer gäller. I exemplet gäller alltså båda randomiseringsnumren sitet ”SE01”
STRATIFICATION_PARAM1	Man kan stratifiera på flera parametrar, till exempel först på site, och sedan inom varje site på kön. Denna kolumn används för en eventuell andra stratifieringsparameter.
STRATIFICATION_PARAM[2-5]	Vi stödjer upp till sex stratifieringsparametrar på samma sätt som ovan
TAKEN	Visar om randomiseringen har använts, det vill säga om någon subject blivit tilldelad det. Anges som 0 eller 1.

Om stratifiering används måste man vid varje anrop för att göra en randomisering, skicka med vilka värden för stratifieringsparametrarna som gäller. Exempelvis om stratifiering är gjord på site, så måste man skicka med vilket site patienten tillhör. Hur man gör detta finns beskrivet i API-dokumentationen för randomisering av patient i SubjectLogAPI.

Den sista detaljen som knyter ihop säcken angående stratifiering är att man måste konfigurera vilka namn på stratifieringsparametrar som är tillåtna att skicka i api-anropet ovan. Detta görs i subjectmanagement-config.xml:

```
<randomizationSettings>
   <stratificationParameterDefinition validKey="SITEID" />
</randomizationSettings>
```

4.3. Integrera egen kod, till exempel anrop till en EDC

En `CreateSubjectDataEvent` skickas varje gång ett nytt subject skapas och ett `StoreSubjectDataEvent` skickas varje gång studiedata sparas.

Genom att skriva en egen eventhandler kan man fånga dessa event och exekvera egen kod. Ett exempel på en sådan eventhandler är klassen `ExampleDataEventHandler` som finns i demoprojektet. Förutom att visa hur en Eventhandler kan skrivas, visar också klassen hur man kan skapa CDISC-formaterad data vilket kan vara bra vid integration med en EDC.

Observera att eventhandlerklassen inte måste konfigureras externt, det räcker med att klassen annoteras med:

```
@Service
@Transactional
```

att den scannas av Spring, samt har en metod med signaturen:

```
@EventHandler
public void handle(CreateSubjectDataEvent event)
```

och/eller en metod med signaturen:

```
@EventHandler
public void handle(StoreSubjectDataEvent event)
```

Eftersom klassen är annoterad med `@Transactional` kommer det interna databasskrivandet rullas tillbaka om man i handle-metoderna kastar en exception, vilket är vad man kan förvänta sig.

5. Beskrivning av R-RCT-ramverkets REST-API

5.1. Study Management API

Hanterar själva studien och dess sites. Alla anrop börjar med {url till studiens webbapplikation}`/v4/study.`

Namn	http-metod	Url	Exempelbody	Beskrivning
Get Study	`GET`	`/`	**N/A**	**Returnerar information om aktuell studie**
Ping	`GET`	`/ping`	**N/A**	**Returnerar ”pong”**
Get Site	`GET`	`/sites/{siteId}`	**N/A**	**Returnerar information om sitet. {siteId} är sitets databas-id eller alias.**
Set Site Status	`PUT`	`/sites/{siteid}/enabled/{enabled}`	**N/A**	**Aktiverar eller deaktiverar site. {siteId} är sitets databas-id eller alias. {enabled} kan vara true eller false.**

5.2. Screening Log API

Hanterar screening av patienter. Identifierare som används nedan:

Namn	Beskrivning
`personalIdentityNumber`	**En godtycklig tolvsiffrig siffersträng som inte kan identifiera patienten. I tidigare versioner av R-RCT-ramverket angavs här patientens personnummer, men UCR bedömer numera att detta inte är förenligt med datalagringslagarna. Variabelnamn internt i koden återspeglar fortfarande det gamla synsättet.**
`sex`	**Kön.** `MALE` **eller** `FEMALE`**.**
`dateOfBirth`	**Patientens födelsedag på formatet** `yyyy-MM-dd`**.**
`issuingEntity`	**Id på den entitet som är ansvarig för** `personalIdentityNumber`**. Eftersom studierna i nuläget bara ska fungera i Sverige, ska värdet alltid vara** `SE`**.**
`correlationId`	**Registrets identifierare på det vårdtillfälle för vilken screening gjordes. Används för att knyta samman screeningen med registret, samt för att veta att inte screening görs två gånger vid samma vårdtillfälle.**
`investigatorId`	**Registrets identifierare på prövaren som ansvarar för screeningen.**
`siteId`	**Studiens id för det site som gör screeningen.**
`dateConsidered`	**Datum för screeningen. Exempel på format:** `2017-05-12`
`timeConsidered`	**Tidpunkt för screeningen. Exempel på format:** `10:11:12`
`dateConsideredTimeZoneId`	**Tidszonen för datum och tid ovan. Eftersom studierna i nuläget bara ska fungera i Sverige, ska värdet alltid anges som** `CET`**.**
`reasonNotIncluded`	**Anledning till att randomisering ej kan ske. Ska anges som en av de studiespecifika koder som definieras i** `subject-configuration.xml`
`reasonNotIncludedDetails`	**Eventuellt mer detaljerad anledning till att randomisering ej kan ske. Ska anges som en av de studiespecifika koder som kan definieras i** `subject-configuration.xml`**.**
`comment`	**Frivillig kommentar som sparas tillsammans med screeningen.**

Alla anrop börjar med {url till studiens webbapplikation}`/v4/screeninglog.`

Namn	http-metod	Url	Exempelbody	Beskrivning

Screena Patient	POST	/	<pre>{
			  "subjectIdentity": {
			  "personalIdentityNumber": "66666666",
			  "sex": "MALE",
			  "dateOfBirth": "1912-12-12",
			  "issuingEntity": "SE"
			},
			  "screeningMetadata": {
			  "correlationId": "66666666",
			  "investigatorId": "Dr Agneta",
			  "siteId": "site1",
			  "dateConsidered": "2017-12-31",
			  "timeConsidered": "10:11:12",
			  "dateConsideredTimeZoneId": "CET"
			},
			  "reasonNotIncluded": "NO_INFORMED_CONSENT",
			  "reasonNotIncludedDetail": "",
			  "comment": ""
			}</pre>
				Lägger till en patient till screeningloggen. Patienten räknas därefter som screenad, men inte randomiserad och deltager INTE i studien.

Ping	GET	/ping		**Returnerar ”pong”.**
Get Screening	GET	/correlationid/{correlationid}		**Returnerar information om screeningen med correlationId** {correlationId}.

5.3. Subject Log API

Hanterar randomisering och inklusion av patienter. Observera att en inklusion av en patient automatiskt medför att patienten registreras som screenad också. Det behövs alltså inget explicit anrop för screening vid de tillfällen screeningen leder till inklusion. Identifierare som används nedan:

Namn	Beskrivning
personalIdentityNumber	**Personnummer. 12 siffror utan bindestreck.**
sex	**Kön.** MALE **eller** FEMALE**.**
dateOfBirth	**Patientens födelsedag på formatet** yyyy-MM-dd**.**
issuingEntity	**Id på den entitet som är ansvarig för** personalIdentityNumber**. Eftersom studierna i nuläget bara ska fungera i Sverige, ska värdet alltid vara** SE**.**
correlationId	**Registrets identifierare på det vårdtillfälle för vilken screening gjordes. Används för att knyta samman screeningen med registret, samt för att veta att inte screening görs två gånger vid samma vårdtillfälle.**
investigatorId	**Registrets identifierare på prövaren som ansvarar för screeningen.**
siteId	**Studiens Id för det site som gör screeningen.**
dateConsidered	**Datum för screeningen. Exempel på format:** 2017-05-12
timeConsidered	**Tidpunkt för screeningen. Exempel på format:** 10:11:12
dateConsideredTimeZoneId	**Tidszonen för datum och tid ovan. Eftersom studierna i nuläget bara ska fungera i Sverige, ska värdet alltid vara** CET**.**

stratificationParameter	**En lista av eventuella stratifieringsparametrar som använts i randomiseringslistan, och vilket värde på varje parameter som just denna inklusion har.**
	Exempelvis kan en randomiseringslista vara stratifierad på ”site” vilket innebär att randomiseringslistan är grupperad på vilket site subjectet tillhör. Vid anropet ska man då skicka med dels parameternamnet ”SITEID” och aktuellt värde, till exempel ”site1”:
	`"stratificationParameters": [{`
	`"key": "SITEID",`
	`value": "site1"`
	`}]`
	Att just SITEID **ska användas är en konfigurationsfråga, se konfigurera randomiseringstjänsten**

Alla anrop börjar med {url till studiens webbapplikation}`/v4/subjectlog.`

Namn	http-metod	Url	Exempelbody	Beskrivning

Randomize Patient	POST	/	<pre>{
			  "subjectIdentity": {
			  "personalIdentityNumber": "5555555",
			  "sex": "MALE",
			  "dateOfBirth": "2012-12-12",
			  "issuingEntity": "SE"
			},
			  "screeningMetadata": {
			  "correlationId": "5555555",
			  "investigatorId": "Doktor Kaj",
			  "siteId": "site1",
			  "dateConsidered": "2015-12-31",
			  "timeConsidered": "00:00:00",
			  "dateConsideredTimeZoneId": "CET"
			},
			  "stratificationParameter": [
			  {
			  "key": "{STRATIFICATIONPARAM_1}",
			  "value": "{STRATIFICATIONPARAM_1_VALUE}"
			  }]
			}</pre>
				Lägger till en patient till både subjectloggen och screeningloggen.
				Patienten räknas därefter som en deltagande study subject i studien.
				Det returnerade svaret innehåller både identifieraren subjectId (ekvivalent med randomiseringsnummer), och namnet på den studiegrupp som patienten randomiserats till.

Ping	GET	/ping		**Returnerar ”pong”.**
Get Inclusion Data by SubjectId	GET	/{subjectid}		**Returnerar information om inklusionen utifrån subjektets subjectId/randomiseringsnummer.**
Get Inclusion Data by pin	GET	/subjectidentity/{pin}/{issuingentity}	N/A	**Returnerar information om inklusionen utifrån personnummer. {pin} anges som ett personnummer med tolv siffror utan bindestreck. {issuingentity} anges som SE.**

Namn	http-metod	Url	Exempelbody	Beskrivning
Withdraw Subject	PUT	/{subjectid}	{ ”withdrawalMetadata”: { ”withdrawnBy”: ”Doktor Kaj”, ”reasonForWithdrawal”: ”WITHDRAWN_FROM_REGISTRY”, “dateWithdrawn”: “2015-12-31”, “timeWithdrawn”: “00:00:00”, dateWithdrawnTimeZoneId”: “CET” } }	**Avslutar en subjects medverkan i studien. Efter detta kan inte mer studiedata skickas in för denne. Observera att redan inskickad studiedata inte raderas.**
Is subject withdrawn?	GET	/{subjectid}/withdrawn	**N/A**	**Returnerar TRUE eller FALSE beroende på om angiven subject har avslutat sin medverkan eller inte.**

5.4. Study Data API

Hanterar lagring och uppläsning av studiedata för en deltagare i studien. Identifierare som används nedan:

Namn	Beskrivning
`subjectId`	**Aktuell subject.**
`siteId`	**Studiens Id för det site som aktuell subject tillhör.**
`sourceId`	**Namn eller Id på det som ska sparas som källa, till exempel namnet på registret.**
`reporterId`	**Namn eller Id på prövare som ansvarar för inskickandet av uppgifterna.**

Alla anrop börjar med {url till studiens webbapplikation}`/v4/studydata.`

Namn	http-metod	Url	Exempelbody	Beskrivning
Ping	`GET`	`/ping`		**Returnerar ”pong”**

Save Subject Study Data	`POST`	`/subjects/{subjectid}`	`{`
			`  "siteId": "TEST01",`
			`  "sourceId": "register X",`
			`  "reporterId": "Dr Svensson",`
			`  "item": [`
			`  {`
			`  "id": "INIT::1::BASIC::1::UNGROUPED::1::WEIGHT,`
			`  "value": "85"`
			`  }]`
			`}`
				Lagrar studiedata för {subjectid}. En eller flera variabler med värden kan skickas in i listan ”`item`”.
				Skickas tomma strängen in som ett variabelvärde innebär det att befintligt variabelvärde raderas.

Get Subject Study Data	`GET`	`/subjects/{subjectid}?itemIds={id_1}&itemIds={id_2}`		**Hämtar lagrad studiedata för {subjectid}. Hela query-parameterdelen kan skippas, då returneras alla lagrade variabeler med värden. I annat fall kan eftersökta variabler specificeras som variable1 och variable2 i exemplet.**

Get Study Definition Document	GET	/studydefinition		**Returnerar en Excelfil med en uppräkning av studiens variabler och de Id som ska användas för att lagra och hämta dem. All data har sitt ursprung i studiens odm-fil.**

5.5. Report API

Enklare rapporter. Alla anrop börjar med {url till studiens webbapplikation}`/v4/reports.`

Namn	http-metod	Url	Exempelbody	Beskrivning
Ping	GET	/ping	**N/A**	**Returnerar ”pong”.**
Get Screeninglog	GET	/screeninglog/?siteid=[site-id]	**N/A**	**Returnerar screeningloggen för [site-id].** **Om parametern siteId utelämnas, returneras alla siters screeninglog.** **Kräver en header-parameter enligt följande:** **Accept: application/json** **eller** **Accept: application/xml** **eller** **Accept:** application/vnd.openxmlformats-officedocument.spreadsheetml.sheet **Den sistnämnda header-parametern används för att erhålla screeningloggen som en Excel-fil.**
Get Subjectlog	GET	/subjectlog/?siteid=[site-id]	**N/A**	**Returnerar subjectloggen för [site-id]. Kräver samma header-parameter som Get Screeninglog (se ovan).** **Parametern siteid är obligatorisk. Det ska inte gå att ladda ned subject log för samtliga siter, så det endast är personal på den enskilda kliniken som har rätt att se “sin” subjectlogg.**

5.6. Study Data Export API

Exporterar all studiedata. Alla anrop börjar med {url till studiens webbapplikation}`/v4/studydataexport.`

Namn	http-metod	Url	Exempelbody	Beskrivning
Ping	GET	/ping	**N/A**	**Returnerar ”pong”.**

Export All Study Data	GET	/studydata	**N/A**	**Returnerar all studiedata som en Excelfil.**