Schnittstellenspezifikation der Treuhandstelle

Unabhängige Treuhandstelle der Universitätsmedizin Greifswald

Version 1.4.0 vom 31.01.2017

Hinweis: Work in Progress!
Dieses Dokument stellt keinen finalen Stand dar und wird stetig ergänzt, Bestehendes wird möglichst beibehalten. Die Inhalte dieses Dokumentes und somit der Schnittstelle sind nicht mit einer vollständigen projektspezifischen Implementierung gleichzusetzen. Referenzimplementierungen oder verbindliche Annahmen sollten unbedingt in Abstimmung mit den Autoren dieses Dokumentes erfolgen.

Version	Datum	Bearbeitungsart / Betroffene Abschnitte	Bearbeiter
0.1	18.03.2014	Erstentwurf	Peter Penndorf
0.2	30.05.14	Draft 1 → „requestPSN“ strukturell geändert	Peter Penndorf
0.3	20.08.14	Draft 2 → Funktionen ergänzt	Peter Penndorf
1.0	03.03.15	Version 1.0 final	Peter Penndorf
1.1	06.05.15	Draft → Verbesserungen der Funktionsparameter	Peter Penndorf, Lars Geidel
1.2	10.06.15	Draft → neue Kommunikationsvariante, neue Funktionen	Peter Penndorf, Lars Geidel, Eric Harder, Ronny Schuldt
1.2.1	19.06.15	Draft → kleine Ergänzungen für Funktionen/Messages	Peter Penndorf
1.2.2	02.07.15	Draft → Änderungen an Funktionen (Request PSN, Query Policies)	Peter Penndorf
1.2.3	08.07.15	Draft → Änderungen an Funktionen (Add Patient, Add Consent, Get Consent, Add Patient Identifier) Neue Funktion Add Consent By Patient Umbenennung von „Message“ zu „Notification“	Peter Penndorf
1.2.4	28.07.15	Draft → Minimale Parameteranpassungen	Peter Penndorf
1.2.5	07.08.15	Draft → Änderung von „F_AC“, „F_ACP“	Peter Penndorf

		Nachrichten ergänzt	
1.2.6	21.08.15	Änderung von „F_AC“, F_ACP“ Neu: „F_CREV“, „F_EPM“, „F_RPI“ Umbenennungen	Peter Penndorf
1.2.7	04.09.15	Änderung in „F_RPI-N_TC“, „F_RPI-N_CR“, „F_NCA-N_MC“	Peter Penndorf
1.2.8	30.09.15	Änderung in „F_RPP“, „F_ACP“, „F_RP“, „F_AC“	Peter Penndorf
1.2.9	21.10.15	Änderung in „F_NNC“, „F_NREV“, „F_RPP“, „F_RPI“ „F_NCA“ entfällt	Peter Penndorf
1.2.10	05.11.15	Neue Funktion F_GCS Änderungen farblich hervorgehoben	Eric Harder
1.2.11	11.12.15	Anpassungen bisheriger Funktionen für flexiblere Patientenidentifikation Änderungen farblich hervorgehoben	Peter Penndorf
1.3.0	18.04.16	Unterteilung in Funktionen und Notifications Dokument enthält nur noch bereits implementierte Funktionen Neu: Funktion Edit Patient (F_EP) Neu: Funktion Search Patient (F_SP) Neu: Funktion AddConsent Variante 2 action (F_ACP) Neu: Notification Refusal (F_NREF)	Arne Blumentritt, Ronny Schuldt, Peter Penndorf, Eric Harder

		Neu: Notification New Patient (F_NNP) kleinere Anpassungen	
1.4.0	30.01.2017	Neue Funktion Update IDAT (F_UI) Neue Funktion External Patient Merge (F_EPM) Aktualisierte Funktion F_AP Aktualisierte Funktion F_RPI Aktualisierte Notification F_NNC Änderung in F_SP	Peter Penndorf Eric Harder Arne Blumentritt

Inhaltsverzeichnis

1 Allgemeines

1.1 Ziel

Das Ziel dieses Dokumentes ist eine detaillierte Beschreibung des Konzeptes und der technischen Gegebenheiten der Schnittstelle der Treuhandstelle. Das Konzept stellt zunächst einen generischen Ansatz dar, der durch eine konkrete Implementation technisch umgesetzt werden kann. Die technische Implementation, basierend auf dem generischen Konzept, wird anschließend detailliert beschrieben.

1.2 Erläuterungen

Das Konzept der Schnittstelle ist an das ID-Management „Mainzelliste“ der Universität Mainz angelehnt. Durch die Mainzelliste wurden bereits grundlegende generische Sicherheitsmechanismen, Abläufe und Datenformate entwickelt. Diese können für jede Funktion der Treuhandstelle eingesetzt werden und bieten somit ein solides Grundkonzept der Schnittstelle.

1.3 Kompatibilität mit Mainzelliste

Das Kommunikationskonzept und dessen REST-Implementierung sind mit der Mainzelliste kompatibel. Da aber die Mainzelliste ein reines ID-Management ist und die THS der Universätsmedizin Greifswald weitere Funktionalitäten unterstützt, unterscheiden sich die beiden Schnittstellen im Funktionsumfang. Aber auch bei inhaltlich gleichen Funktionen sind teilweise zusätzliche Parameter notwendig.

2 Konzept

Im Folgenden wird ein allgemeines Konzept erläutert, welches den Ablauf der Kommunikation und die Funktionen der Treuhandstelle in einer abstrakten Form beschreibt.

2.1 Grundlegendes Konzept

Jede Komponente, die mit der Treuhandstelle kommuniziert, muss sich zunächst authentifizieren. Authentifizierte Komponenten sind durch zugeordnete Rollen (siehe Kapitel 2.2) zur Nutzung bestimmter Funktionalitäten autorisiert. Die Nutzung einer Funktionalität ist in einem in Kapitel 2.3 beschriebenen Ablauf integriert.

2.2 Rollen

Rollen definieren Aufgaben die Systemkomponenten / Software oder Anwendungsteile übernehmen können. Mit einer Rolle sind konkrete Kommunikationsabläufe verbunden, in denen Nachrichten für konkrete Funktionen wie z.B. "Patient anlegen" kommuniziert werden. Folgende Rollen wurden definiert THS-Service, Receiver und Provider.

2.2.1 THS-Service

Diese Rolle bildet die Funktionalitäten z.B. „Patient anlegen“ einer Treuhandstelle ab. Die

anderen Rollen können diese Funktionen des THS-Services nutzen.

2.2.2 Receiver

Diese Rolle übernimmt eine Komponente, die Daten vom THS-Service erhalten soll, z.B. ein Pseudonym oder Einwilligungsinformationen. Der Receiver initiiert, durch die Wahl der der Funktion, den gesamten Ablauf mit dem THS-Service und erhält nach dem Funktionsaufruf die angeforderten Daten.

2.2.3 Provider

Da der Receiver die Funktion der Treuhandstelle vorgibt, liefert der Provider dem THS-Service die Daten, die zum erfolgreichen Ausführen der Funktion benötigt werden.

Besonderheit

In der Regel werden Receiver und Provider nicht von der gleichen Komponente eingenommen. Die Pseudonyme des Receivers sind dem Provider nicht bekannt und umgekehrt. Der Vermittler zwischen diesen Rollen ist der THS-Service. Im Standardfall ist auch eine Kommunikation zwischen den Rollen Provider und Receiver implementiert bzw. diese Kommunikation ist der „Auslöser“ für die Interaktion mit dem THS-Service. Die Kommunikation zwischen Provider und Receiver (sekundäre Kommunikation) wird hier nicht näher erläutert sondern nur angedeutet, um einen beispielhaften Überblick zu geben.

2.3 Interaktion der Rollen (Kommunikationsabläufe)

Im Folgenden wird die Interaktion der Rollen näher erläutert. Es existiert ein allgemeingültiger Ablauf, der als Variante 1 definiert ist. Prinzipiell ist für jede Funktion der Treuhandstelle diese Variante möglich. Ist die Variante 1 nicht passend, ist eine Alternative Variante 2 beschrieben, die je nach Funktion ebenfalls möglich sein können.

Begriffserläuterungen

Session → Eine Session ist ein Datenobjekt, welches Informationen zum Receiver beinhaltet. Sie dient als eindeutige Identifikation des Receivers. Ebenso beinhaltet eine Session die Funktionen, die der Receiver angefordert hat. Eine Session hat eine eindeutige ID und besitzt eine Gültigkeit.

Token → Ein Token ist eine Art Ticket und bildet eine Funktion der Treuhandstelle ab. Ein Token ist immer einer Session zugeordnet. Das Token wird durch den Receiver angefordert und durch den Provider eingelöst. Anhand der Session und des Tokens können Receiver und Provider zusammen im THS-Service eindeutig identifiziert und deren Daten verarbeitet werden. Ein Token hat eine eindeutige ID und besitzt eine temporäre Gültigkeit. Es kann, sofern es kein Token für eine Stapelverarbeitungsfunktionalität ist, nur einmal eingelöst werden.

Request → Ein Request ist eine Anfrage zur Datenübertragung zwischen Rollen.

Response → Ein Response ist eine Antwort einer Rolle auf einen Request.

Callback → Der Callback dient der Übermittlung von Daten an den Receiver. Der Receiver definiert eine Callback-URL, an die die Daten vom THS-Service gesendet

werden.

Redirect → Der Redirect ist eine URL des Receivers, auf die der Provider vom THS-Service weitergeleitet werden kann, wenn das Token vom Provider eingelöst wurde.

2.3.1 Variante 1 (KV1)

In folgender Abbildung 1 ist eine vereinfachte Darstellung der Variante 1 erläutert am Beispiel „Senden von MDATs von Provider an Receiver mit Austausch des PSN über die Treuhandstelle“. Diese Variante definiert die Kommunikation mit 3 Rollen. Die folgenden Abbildungen beschränken sich zunächst nur auf die Kommunikation, die auch in der Spezifikation erläutert wird. Dies ist die primäre Kommunikation. Weitere Kommunikation zwischen Receiver und Provider(z.B. der Austausch der Token-ID) wird als sekundäre Kommunikation bezeichnet.

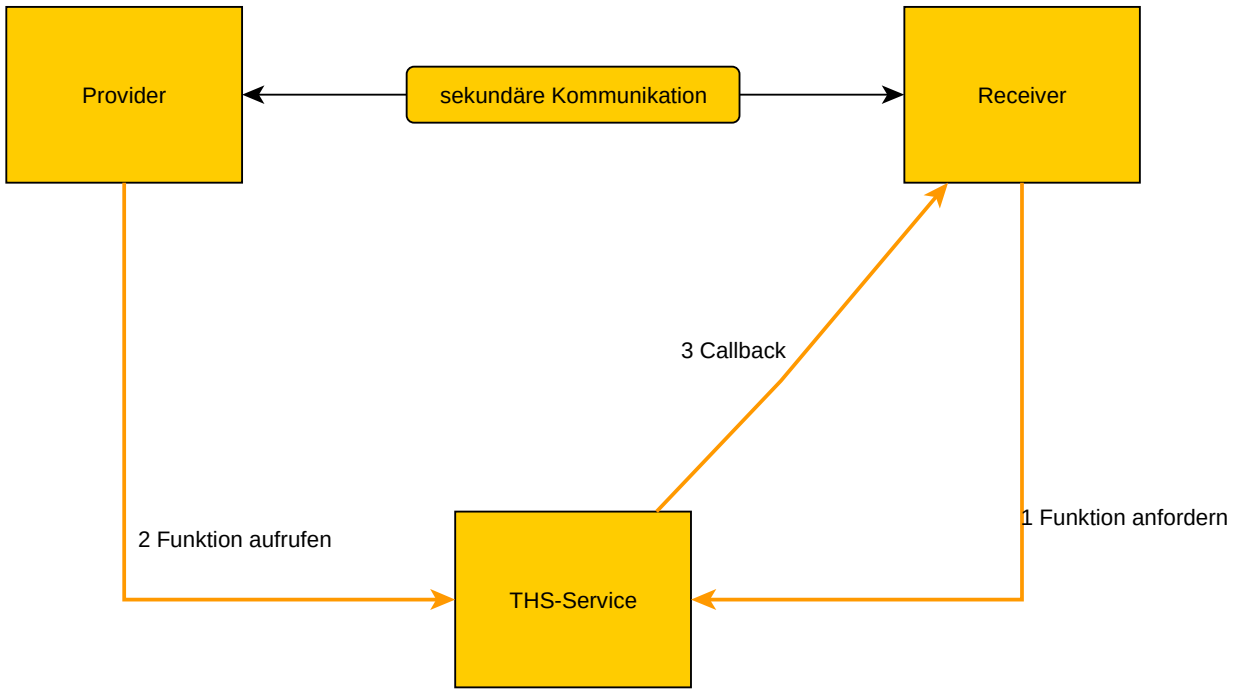

Abbildung 1: vereinfachte primäre Kommunikation zwischen den Rollen in Variante 1

Konventionen für nachfolgende Tabellen und Diagramme

Im Folgenden werden Funktionen und Nachrichten näher beschrieben. Für eine einheitliche und standardisierte Bezeichnung wurden folgende Konventionen festgelegt:

- Funktionen beginnen mit „F“, gefolgt von einem „_“ und dem Kürzel der Funktion, z.B. „F_AP“ für die Funktion „addPatient“.
- Die allgemeingültigen und funktionsunspezifischen Nachrichten im generischen Konzept beginnen mit einem „N“ gefolgt von einem „_“ und dem Kürzel, z.B. „N_TC“ für Nachricht „Token-Call“.

Abbildung 2 zeigt den Ablauf in einem Sequenzdiagramm und folgende Tabelle beschreibt die Nachrichten der primären Kommunikation im Detail.

Schritt	Beschreibung	Nachricht
1	Receiver initiiert eine Session beim THS-Service und übergibt Parameter	N_S (Session)
2	THS-Service gibt die SessionID an den Receiver zurück (Response)	N_SR (Session-Response)
3	Receiver initiiert Token beim THS-Service übergibt die Funktion und funktionsspezifische Parameter	N_T (Token)
4	THS-Service gibt TokenID und weitere Informationen des Tokens an den Receiver zurück (Response)	N_TR (Token-Response)
5	Provider löst das Token ein und übergibt die funktionsspezifischen Daten an den THS-Service	N_TC (Token-Call)
5.1	THS-Service übermittelt die angeforderten Daten an den Receiver zurück (Callback)	N_C (Callback)
5.2	Receiver bestätigt den erfolgreichen Erhalt der Daten (Callback-Response)	N_CR (Callback-Response)
6.a	THS-Service gibt funktionsspezifische Daten an den Provider zurück (Response)	N_TCR (Token-Call-Response)
6.b	THS-Service gibt funktionsspezifische Daten an den Provider zurück (Redirect)	N_R (Redirect)

Die Nachricht 6 unterscheidet 2 Typen. Je nachdem, ob der Provider z.B. ein Browser ist und eine User-Interaktion benötigt (6.b) oder der Provider ein System ist, welches automatisiert die Daten überträgt (6.a). Es wird nur ein Typ je Aufruf verwendet. Nähere Erläuterungen hierzu sind in Kapitel 2.5.2 zu finden.

Variante 1 - Sequenzdiagramm

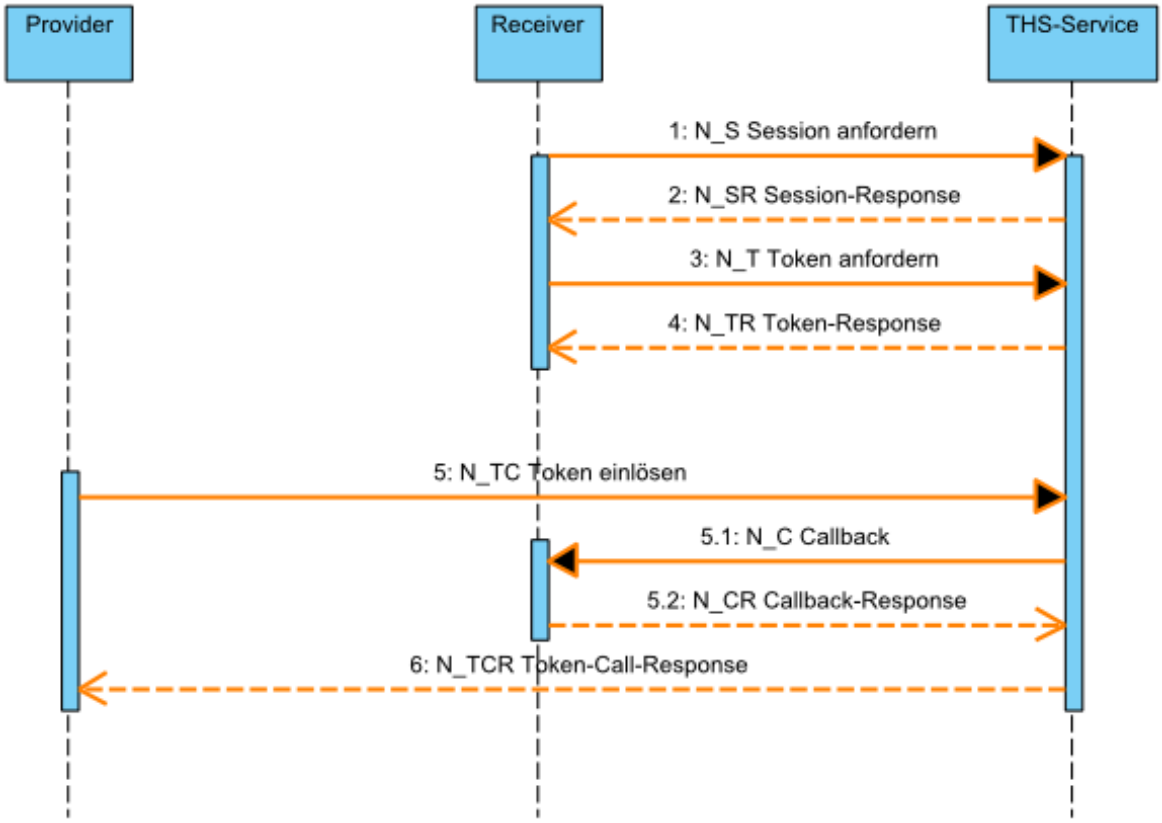

Abbildung 2: Sequenzdiagramm für Variante 1 (nur primäre Kommunikation)

Variante 1 inklusive sekundärer Kommunikation

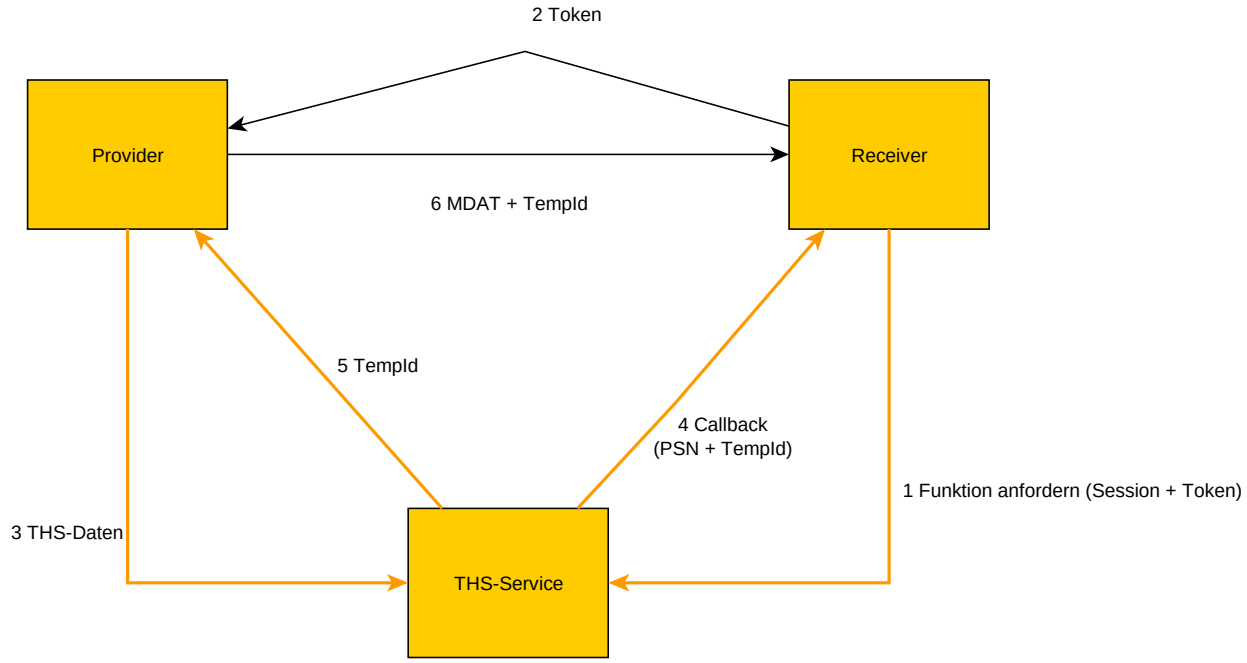

Abbildung 3: vereinfachte Variante 1 mit sekundärer Kommunikation zwischen Provider und Receiver

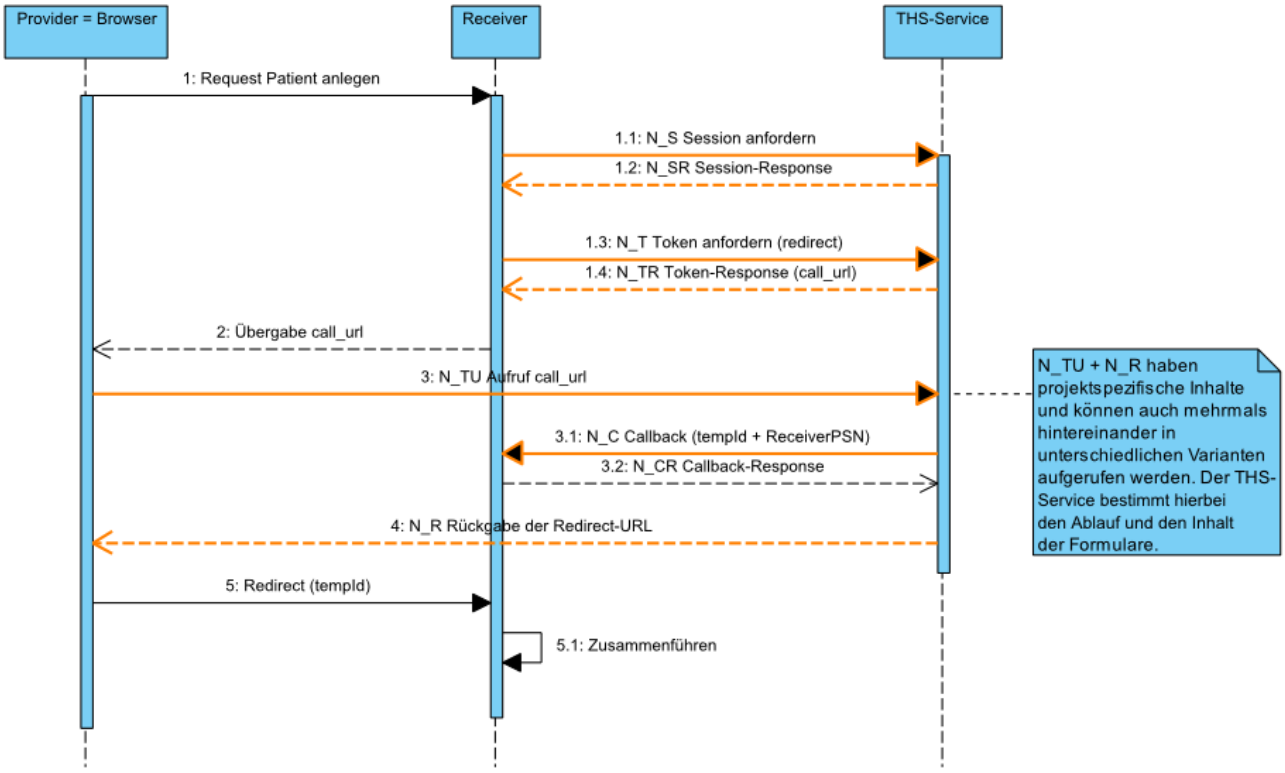

Abbildung 4: Erweitertes Sequenzdiagramm der Variante 1 mit sekundärer Kommunikation zwischen Receiver und Provider. Nur die farbigen Kommunikationsschritte werden in dieser Spezifikation beschrieben.

Fehlerverhalten

Aus Sicht des THS-Services folgt auf einem Request immer ein Response. Tritt im THS-Service ein Fehler auf, wird ein entsprechender Fehlercode als Response zurückgegeben. Eine Ausnahme bildet der Aufruf des Callbacks. Löst der Provider ein Token ein, wird dieses im THS-Service bearbeitet und der Callback wird aufgerufen. Es können bei der Abarbeitung eines Tokens folgende Ereignisse eintreten:

1. Token kann nicht verarbeitet werden,
2. Callback-URL ist nicht erreichbar oder im Callback-Response wurde ein Fehler zurückgegeben.

In beiden Fällen wird im Token-Call-Response an den Provider ein Fehlercode zurückgegeben. Der Receiver würde in keinem der Fälle einen Fehler erhalten, aber auch keine gültigen Daten. In der Regel müsste nun der Provider dem Receiver den Abbruch des Vorgangs mitteilen.

Tritt einer der beschriebenen Fehler auf, wird das Token ungültig und kann nicht wiederverwendet werden.

2.3.2 Alternative Variante 2 „Eine Komponente ist gleichzeitig Provider und Receiver“ (KV2)

Beschreibung

In einigen Anwendungsfällen kommuniziert nur eine Komponente mit dem THS-Service, z.B. bei dem Abfragen von Einwilligungen. In diesem Fall ist die Variante 1 und die Übermittlung der Ergebnisse per Callback nicht zwingend notwendig. Für die einfachere Umsetzung wurde Variante 2 konzipiert. Ist der Receiver auch gleichzeitig Provider kann das Ergebnis der Funktion direkt an den Provider im Response übermittelt werden.

Ablauf

Folgende Abbildung 5 zeigt den Ablauf in einem Sequenzdiagramm.

Schritt	Beschreibung	Nachricht
1	Receiver initiiert eine Session beim THS-Service und übergibt Parameter	N_S (Session)
2	THS-Service gibt die SessionID an den Receiver zurück (Response)	N_SR (Session-Response)
3	Receiver initiiert Token beim THS-Service übergibt die Funktion und funktionsspezifische Parameter	N_T (Token)
4	THS-Service gibt TokenID und weitere Informationen des Tokens an den Receiver zurück (Response)	N_TR (Token-Response)
5	Provider löst das Token ein und übergibt die funktionsspezifischen Daten an den THS-Service	N_TC (Token-Call)
6	THS-Service übermittelt die angeforderten Daten an den Provider (und Receiver) zurück	N_TCR2 (Token-Call-Response Variante 2)

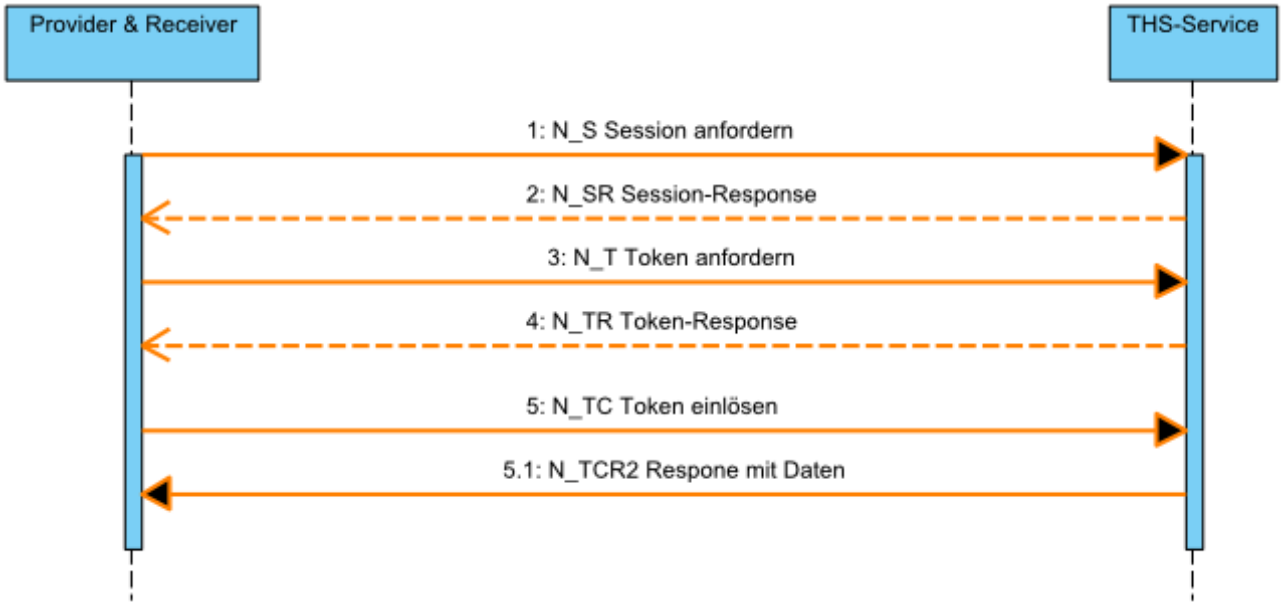

Abbildung 5: Sequenzdiagramm für Variante 2

Fehlerverhalten

Tritt ein Fehler auf, dann wird ein entsprechender Fehlercode im Response

zurückgegeben. Da hierbei weder Callback- noch Redirect-URLs durch den THS-Service genutzt werden, entspricht das Fehlerverhalten der Standardprozedur.

2.3.3 Variante 3 „THS-Service ist gleichzeitig auch Provider“ (KV3)

Für einige Anwendungsfälle ist es notwendig, dass der THS-Service selbst als Provider agiert, weil die THS Informationen z.B. durch andere Systeme erhalten hat und weiter kommunizieren muss. Ein Beispiel wäre die Änderung eines Consents oder gar ein kompletter Widerruf. Hierfür muss der Receiver beim THS-Service registriert/konfiguriert sein und bekommt anschließend die notwendigen Nachrichten (vom Provider) automatisch.

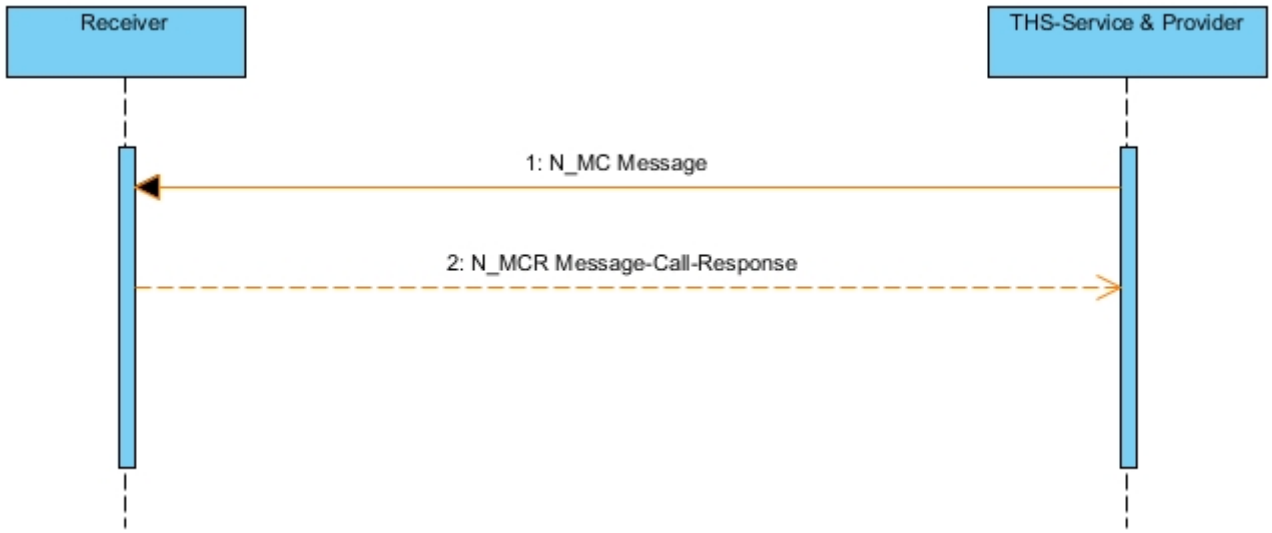

Abbildung 6: Sequenzdiagramm für Variante 3

Schritt	Beschreibung	Nachricht
1	THS-Service schickt eine Nachricht an den Provider.	N_MC (Message-Call)
2	Reciever quittiert die erhaltene Nachricht mit einem Status oder ErrorCode.	N_MCR (Message-Call-Response)

2.4 Notification Service

Die THS muss in bestimmten Anwendungsfällen Informationen an Receiver mitteilen, die sie durch anderen Systeme oder auch Personen erhalten und verarbeitet hat. Hierzu bietet die THS einen Notification Service an, dessen Funktionsweise im folgenden näher beschrieben wird.

2.4.1 Receiver Registrierung

Es ist vorgesehen, dass jeder Receiver, der über den Notification Service Nachrichten erhalten soll, beim THS-Service registriert sein muss. Diese Registrierung erfolgt nicht automatisiert. Die Receiver werden manuell konfiguriert, um zum einen die Vertrauenswürdigkeit sicherzustellen und zum anderen die Implementierung seitens des Receivers auf ein Minimum zu reduzieren.

Folgende Informationen vom Receiver werden für die Registrierung benötigt bzw. konfiguriert:

- eindeutige ID
- Empfangs-URL des Receivers + Aufruf-Methode (z.B. HTTP-POST/PUT)
- Nachrichtenformat
- Nachrichtentypen (siehe Messages in Kapitel 2.5)
- Patienten-Identifier-Typ

2.4.2 Kommunikation

Der Notification Service verwendet den Kommunikationsablauf Variante 3.

2.5 Funktionen

Im Folgenden werden die Funktionen der Treuhandstelle erläutert, die von der Rolle THS-Service bereitgestellt werden können. Welche Funktionen eine Komponente mit der Rolle THS-Service implementiert hat, wird hierdurch nicht festgelegt.

Für jede Funktion ist angegeben, welcher Kommunikationsaublauf (Variante 1, Variante 2 Variante 3) für diese Funktion notwendig ist. Es können auch mehrere Abläufe je Funktion möglich sein.

2.5.1 Begriffserläuterungen und Konventionen

Begriffserläuterungen

tempId → eine eindeutige ID, die der THS-Service vergibt, um eine Zusammenführung von medizinischen Daten zwischen Provider und Receiver zu gewährleisten.

Konventionen

- „|“ → der senkrechte Strich stellt ein logisches „Oder“ dar.
- „[]„ → Parameter in eckigen Klammern sind als ein Eintrag einer Liste zu verstehen.
- „=“ → Parameter, denen ein Wert zugeordnet ist, müssen genau diesen Wert aufweisen.
- „{}“ → Werte in geschweiften Klammern sind Parameter, die gruppiert und einem Objekt zugeordnet sind.
- *kursive* Parameter sind optional
- „“ → 3 Punkte definieren variable Parameter, die z.B. per Konfiguration festgelegt werden können.
- Für mögliche auftretende Fehler ist ein Fehlercode definiert. Fehlercodes beginnen mit „FC“, gefolgt von einer Nummer, z.B. „FC1“. Die Bedeutung der Fehlercodes ist im Anhang näher erläutert.

Eine nähere Beschreibung der Parameter ist im Anhang (Nr. i) zu finden.

2.5.2 Funktionsaufrufe

Es wird bei jedem Funktionsaufruf unterschieden, ob dieser mit oder ohne User-Interaktion erfolgt. Eine User-Interaktion wäre zum Beispiel die Eingabe von Informationen beim Einlösen eines Tokens. Keine User-Interaktion würde bedeuten, ein System wäre der Provider und würde die Informationen über eine Schnittstelle automatisiert bereitstellen. Diese beiden Aufruftypen werden in jeder Funktionsbeschreibung unterschieden und durch die Angabe von „form“ und „action“ gekennzeichnet. Somit ist jede Kombination aus Variante (KV1-2) und Aufruftyp („form“ oder „action“) möglich. Aber nicht jede Funktion biete alle Kombinationen.

2.5.3 Add Patient (F_AP)

2.5.3.1 Variante 1 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_AP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,*	FC1, FC2

		user_title	
F_AP-N_SR	Session-Response	sessionId, uri	
F_AP-N_T	Token anfordern	apiKey, sessionId, type=addPatient, study_id, study_name, *study_shortname,* location_id, location_name, targetIdType, event	FC1, FC2

F_AP-N_TR	Token-Response	tokenId,
		uri,
		call {
		form {
		url,
		method=GET
		}
		}	

F_AP-N_TU	Hierbei werden die Daten über Webseiten, die durch den THS-Service bereitgestellt werden, vom Provider übermittelt.	tokenId, optional apiKey, Consent-Informationen. Dies ist abhängig vom Projekt.	FC1, FC2, FC10
F_AP-N_C	Callback mit Target-PSN	tokenId, targetIdType targetId	FC3, FC10
F_AP-N_CR	Callback-Response mit ErrorCode	*errorCode*	
F_AP-N_R	Redirect-URL für Browser		

2.5.3.2 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_AP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,*	FC1, FC2

		user_lastname, *user_title*	
F_AP-N_SR	Session-Response	sessionId, uri	

F_AP-N_T	Token anfordern	apiKey,
		sessionId,
		type=addPatient,
		study_id,
		study_name,
		study_shortname,
		location_id,
		location_name,
		targetIdType,
		event,
		options {
		resultType=simple |
		detailed
		}
			FC1, FC2

F_AP-N_TR	Token-Response	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_AP-N_TC	Hierbei werden die Daten durch den Provider übermittelt. Welche Parameter vom Patienten Pflicht sind, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.	apiKey,
		tokenId,
		patients [{
		index,
		patient {
		identifier [{
		domain,
		name,
		id
		}],
		patientGroupId,
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
			FC1, FC2, FC3, FC10

		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		},
		consents [{
		template,
		version,
		processType=addConsent | refusal,
		modules [{
		name,
		status=accepted | declined | not_asked | not_choosen | unknown
		}],
		scan {
		fileType=pdf,
		contentType=base64,
		content
		}
		}]
		}]	

F_AP-N_TCR2	Token-Call-Response. Die Scans werden aus	**resultType → simple:** [{	

	Gründen der Performance und des Traffics nicht zurückgegeben.	index,
		targetId,
		tentative=true|false,
		errorCode
		}]
		resultType →
		detailed:
		[{
		index,
		patient {
		identifier [{
		domain,
		name,
		id
		}],
		patientGroupId,
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime	

		}]
		},
		consents [{
		template,
		version,
		processType=addConsent | refusal,
		modules [{
		name,
		status=accepted | declined | not_asked | not_choosen | unknown
		}]
		}]
		targetId,
		tentative,
		errorcode
		}]	

2.5.4 Request PSN (F_RP)

Diese Funktion ermöglicht das Generieren und Abfragen von Pseudonymen. Ein beispielhafter Anwendungsfall könnte das Generieren eines Pseudonyms für System B unter Angabe des Pseudonyms aus System A sein. Ein zweites Beispiel wäre die Generierung von Pseudonymen für Forschungsanfragen aus den Pseudonymen der Forschungsdatenbank.

2.5.4.1 Variante 1 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_RP-N_SR	SessionResponse	sessionId, uri	
F_RP-N_T	Token anfordern	apiKey, sessionId, type=requestPSN,	FC1, FC2

		method=get | getOrCreate | create, study_id, study_name, *study_shortname,* event, reason, callback, targetType	

F_RP-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_RP-N_TC	Daten zur Umwandlung von PSNs	apiKey,
		tokenId,
		
		patients [
		{
		index,
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier
		},
		relatedIdentifier
		[{
		sourceId,
		idType
		}]
		}
		]
			FC1, FC2, FC3, FC10

F_RP-N_C	Callback mit Target-PSNs. Source enthält eine tempID	tokenId, targetType, patients [{ targetId,	FC3, FC10

		tempId, *errorCode*, *relatedIdentifier* {[targetId, idType, tempId, *errorCode*,]} }]	

F_RP-N_CR	Callback-Response mit StatusCode	patients [
		~~patient~~ {
		targetId,
		tempId,
		errorCode,
		relatedIdentifier
		[{
		targetId,
		idType,
		tempId,
		errorCode,
		}]
		}
		]	

F_RP-N_TCR	Gesendete Daten angereichert mit tempIDs	patients [
		{
		index,
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier
		}
		tempId,
		errorCode,
		relatedIdentifier
		[{
		sourceId,	

		idType, tempId, *errorCode*, }] }]	

2.5.4.2 Variante 2 (action):

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_RP-N_SR	SessionResponse	sessionId, url	
F_RP-N_T	Token anfordern	apiKey, tokenId, type=requestPSN, method=get | getOrCreate | create, study_id, study_name, event, reason, targetType	FC1, FC2

F_RP-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_RP-N_TC	Daten zur Umwandlung von PSNs	apiKey, tokenId, patients [{	FC1, FC2, FC3, FC10

		index,
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier
		},
		relatedIdentifier
		[{
		sourceId,
		idType
		}]
		}
		]	

F_RP-N_TCR2	Antwort mit Target-PSNs	targetType,
		patients [
		{
		index,
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier
		},
		targetId,
		errorCode,
		relatedIdentifier
		[{
		sourceId,
		idType,
		targetId,
		errorCode
		}]
		}
		]
			FC3, FC10

2.5.5 Query Policies (F_QP)

Diese Funktion ermöglicht die Abfrage von Einwilligungen (Policies). Wurden zu den Patienten Einwilligungen erfasst, sind diese durch verschiedene Möglichkeiten abfragbar. Ein Beispiel wäre das Abfragen der Policies durch die Transferstelle bei der Herausgabe von Daten für Forschungsanfragen.

2.5.5.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_QP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_QP-N_SR	SessionResponse	sessionId, uri	
F_QP-N_T	Token anfordern	apiKey, sessionId, type=queryPolicies, study_name, study_id, event, reason	FC1, FC2

F_QP-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_QP-N_TC	Daten zur Abfrage von Policies Folgende Kombinationen der Parameter sind zulässig: 1. patients & policies & queryType=policyBased & jeder resultType 2. patients & queryType=eventBased &	apiKey, tokenId, patients [index, patientIdentifier{ domain, name, id, type= patientPSN |	FC1, FC2, FC3, FC10

	resultType=simple (policies dürfen nicht angegeben werden)	localIdentifier | localIdentifierPSN }], *policies* [policyId, policyVersionRange] , options { resultType=simple | detailed, queryType=eventBased | policyBased, unknownStateIsConsideredAsDeclined=true|false }	
F_QP-N_TCR2	Response des THS-Service mit boolschen Werten	**resultType → detailed:** patients [index, patientIdentifier{ domain, name, id, type= patientPSN | localIdentifier | localIdentifierPSN } , policies [policyId, isConsented=true | false]] **resultType → simple:** patients [index, patientIdentifier{ domain, name,	FC3, FC10

		id, type= patientPSN | localIdentifier | localIdentifierPSN } , isConsented=true | false]	

2.5.6 Add Consent (F_AC)

Nur das Anlegen einer Einwilligung zu einem bereits vorhandenem Patienten kann mit dieser Funktion erfolgen. Ebenso sollte hierbei auch ein erneutes Ausfüllen oder das Ausfüllen einer neuen Version einer Einwilligung möglich sein. Es sind verschiedene Varianten denkbar wie z.B. automatisierte Übergabe eines ausgefüllten ICs durch eine externe Anwendung oder die Bereitstellung des ICs als Formular durch die Treuhandstelle. Es wird bei dieser Funktion für die Patientenidentifizierung eine PSN erwartet.

2.5.6.1 Variante 2 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes[1]
F_AC-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_AC-N_SR	Session-Response	sessionId, uri	
F_AC-N_T	Token anfordern	apiKey, sessionId, type=addConsent, study_id, study_name, *study_shortname,* location_id, *location_name,* redirect, event, targetIdType,	FC1, FC2

1 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

		targetId,
		options {
		digitalSignature=true|false,
		processType=addConsent | revocation | refusal | user-defined
		}	

F_AC-N_TR	Token-Response	tokenId,
		uri,
		call {
		form {
		url,
		method=GET
		}
		}	

F_AC-N_TU	Hierbei werden die Daten über Webseiten, die durch den THS-Service bereitgestellt werden, vom Provider übermittelt.	tokenId, optional apiKey, PSN, Consent-Informationen. Dies ist abhängig vom Projekt.	FC1, FC2, FC10
F_AC-N_R	Redirect-URL für Browser		

2.5.7 Add Consent By Patient (F_ACP)

Nur das Anlegen einer Einwilligung zu einem bereits vorhandenem Patienten kann mit dieser Funktion erfolgen. Ebenso sollte hierbei auch ein erneutes Ausfüllen oder das Ausfüllen einer neuen Version einer Einwilligung möglich sein. Es sind verschiedene Varianten denkbar wie z.B. automatisierte Übergabe eines ausgefüllten ICs durch eine externe Anwendung oder die Bereitstellung des ICs als Formular durch die Treuhandstelle. Es werden bei dieser Funktion für die Patientenidentifizierung IDATs erwartet.

2.5.7.1 Variante 2 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes[2]
F_ACP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,*	FC1, FC2

2 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

		user_lastname, *user_title*	
F_ACP-N_SR	Session-Response	sessionId, uri	

F_ACP-N_T	Token anfordern. Welche Parameter vom Patienten Pflicht sind, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.	apiKey,
		sessionId,
		type=addConsentByPatient,
		study_id,
		study_name,
		study_shortname,
		location_id,
		location_name,
		redirect,
		event,
		patient {
		identifier [{
		domain,
		name,
		id
		}],
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
			FC1, FC2

		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		},
		options {
		digitalSignature
		=true|false,
		processType=a
		ddConsent |
		revocation |
		refusal | user-
		defined
		}	

F_ACP-N_TR	Token-Response	tokenId,
		uri,
		call {
		form {
		url,
		method=GET
		}
		}	

F_ACP-N_TU	Hierbei werden die Daten über Webseiten, die durch den THS-Service bereitgestellt werden, vom Provider übermittelt.	tokenId, optional apiKey, Consent-Informationen. Dies ist abhängig vom Projekt.	FC1, FC2, FC10
F_ACP-N_R	Redirect-URL für Browser	processResult=cancele d | completed | error	

2.5.7.2 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_ACP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2

F_ACP-N_SR	Session-Response	sessionId, uri	
F_ACP-N_T	Token anfordern	apiKey, sessionId, type=addConsentByPatient, study_id, *study_name,* *study_shortname,* location_id, *location_name,* event	FC1, FC2

F_ACP-N_TR	Token-Response	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_ACP-N_TC	Daten zum Hinzufügen von Consents. Welche Parameter vom Patienten Pflicht sind, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.	apiKey,
		tokenId,
		patients [
		index,
		patient
		{
		identifier [{
		domain,
		name,
		id
		}],
		firstName,
		lastName, middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,motherTongue,
		nationality,
		race,
			FC1, FC2, FC3, FC10

		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		},
		consents [{
		template,
		version,
		modules [{
		name,
		status=accepted
		| declined |
		not_asked |
		not_choosen |
		unknown
		}],
		scan {
		fileType=pdf,
		contentType=bas
		e64,
		content
		},
		processType=ad
		dConsent |
		refusal
		}]]	

F_ACP-N_TCR2	Antwort ggf. mit ErrorCodes.
	Die Scans werden aus Gründen der Performance und des Traffics nicht zurückgegeben.
		patients [
		index,
		patient {
		identifier [{
		domain,
		name,
		id
			FC3, FC10

		}],
		firstName,
		lastName, middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,mo
		therTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		},
		consents [{
		template,
		version,
		processType=add
		Consent |
		revocation |
		refusal,
		modules [{
		name,
		status=accepted
		| declined |
		not_asked |
		not_choosen |
		unknown	

		}], processType=addConsent | refusal, *errorCode* }], errorCode }]	

2.5.8 Request PSN by Patient (F_RPP)

Diese Funktion ähnelt der Funktion F_RP (Request PSN), nur dass hierbei nicht eine reine Pseudonymisierung statt findet, sondern für einen vorhandenen Patienten (IDATs) eine PSN generiert/abgefragt wird. Dies ist zum Beispiel der Fall, wenn ein patientenführendes System den Patienten bereits über addPatient angelegt hat und über ein anderes Quellsystem für MDATs die Daten für die Forschung bereitgestellt werden sollen, aber diese enthalten noch die IDATs der Patienten und müssen gegen eine PSN ausgetauscht werden.

2.5.8.1 Variante 1 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RPP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_RPP-N_SR	SessionResponse	sessionId, uri	
F_RPP-N_T	Token anfordern	apiKey, sessionId, type=requestPsnByPatient, method=get | getOrCreate | create, study_id, *study_name,* *study_shortname,* event, reason, callback, targetIdType,	FC1, FC2

		options {
		resultType=simple | detailed
		}	

F_RPP-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_RPP-N_TC	Daten zur Umwandlung von PSNs. Welche Parameter vom Patienten Pflicht sind, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.	apiKey,
		patients [{
		index,
		patient {
		identifier [{
		index,
		domain,
		name,
		id
		}],
		relatedIdentifier
		[{
		sourceId,
		index,
		idType
		}],
		firstName,
		lastName, middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
			FC1, FC2, FC3, FC10

		city, country, countryCode,district, email, phone, state, street, zipCode, *municipalityKey,* *originDateTime* }] }]	

F_RPP-N_C	Callback mit Target-PSNs.	tokenId,
		targetIdType,
		patients [{
		tempId,
		targetId,
		identifier [{
		tempId,
		targetId,
		domain,
		name
		}],
		relatedIdentifier
		[{
		tempId,
		idType,
		targetId
		}],
		}]
			FC3, FC10

F_RPP-N_CR	Callback-Response mit StatusCode	patients [{
		tempId,
		identifier [{
		tempId,
		targetId,
		domain,
		name
		}],
		relatedIdentifier
		[{
		tempId,
		idType,	

		targetId }], *errorCode* }]	

F_RPP-N_TCR	Gesendete Daten angereichert mit tempIDs, target enthält eine tempID	**resultType → simple:**
		patients [{
		index,
		tempId,
		identifier [{
		tempId,
		index,
		domain,
		name,
		id
		}],
		relatedIdentifier
		[{
		tempId,
		index,
		sourceId,
		idType
		}],
		errorCode
		}]
		resultType → detailed:
		patients [{
		index,
		patient {
		identifier [{
		tempId,
		index,
		domain,name,
		me,
		id
		}],
		relatedIdentifier
		[{
		tempId,
		index,
		sourceId,
		idType
		}],
		firstName,
		lastName, middleName,
		prefix,	

		suffix, civilStatus, degree, gender, birthdate, birthPlace, mothersMaidenName,moth erTongue, nationality, race, religion, originDateTime, }, contacts [{ city, country, countryCode,district, email, phone, state, street, zipCode, *municipalityKey,* *originDateTime* }] tempId, *errorCode,* }]	

2.5.8.2 Variante 2 (action):

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RPP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_RPP-N_SR	SessionResponse	sessionId, uri	
F_RPP-N_T	Token anfordern	apiKey,	FC1, FC2

		tokenId,
		type=requestPsnByPatient,
		method=get | getOrCreate | create,
		study_id,
		study_name,
		event,
		reason,
		targetIdType,
		options {
		resultType=simple | detailed
		}	

F_RPP-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_RPP-N_TC	Daten zur Umwandlung von PSNs.
	
	identifier → domain: hiermit ist die Domain oder das System von dem die ID generiert wurde gemeint. Welche Parameter vom Patienten Pflicht sind, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.
		apiKey,
		patients [{
		index,
		patient {
		identifier [{
		index,
		domain,
		name,
		id
		}],
		relatedIdentifier
		[{
		sourceId,
		index,
		idType
		}],
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
			FC1, FC2, FC3, FC10

		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		}
		contacts [{
		city,
		country,
		countryCode,district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		
		}]	

F_RPP-N_TCR2	Antwort mit Target-PSNs	**resultType → simple:**
		patients [{
		index,
		targetId,
		identifier [{
		targetId,
		index,
		domain,
		name,
		id
		}],
		relatedIdentifier
		[{
		targetId,
		index,
		sourceId,
		idType
		}],
		tentative,
		errorCode
		}]
		resultType → detailed:
			FC3, FC10

		patients [{
		index,
		patient {
		identifier [{
		domain,
		name,
		index,
		targetId,
		id
		}],
		relatedIdentifier
		[{
		sourceId,
		index,
		targetId,
		idType
		}],
		firstName,
		lastName, middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,moth
		erTongue,
		nationality,
		race,
		religion,
		originDateTime
		},
		contacts [{
		city,
		country,
		countryCode,district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]	

		targetId, *errorCode* }]	

2.5.9 Get Consent Document (F_GCD)

Diese Funktion bietet die Möglichkeit die aktuellen Einwilligungen eines Patienten als Dokumente (menschenlesbar) zu erhalten. Die weitere Verarbeitung oder Anzeige der Dokumente ist anschließend Aufgabe des Receivers. Es sind hierbei mehrere Dokumente möglich, da je Studie z.B. unterschiedliche Einwilligungsdokumente erfasst werden können (Basis-Einwilligung, Bioproben). Die Funktion gibt die zeitlich zuletzt erfassten Dokumente jedes möglichen Templates zurück.

2.5.9.1 Variante 2 (action):

Nachricht	Beschreibung	Parameter	Fehlercodes
F_GCD-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_GCD-N_SR	SessionResponse	sessionId, uri	

F_GCD-N_T	Token anfordern (currentOnly: default = true)	apiKey,
		tokenId,
		type=getConsentDocument,
		study_id,
		study_name,
		event,
		options {
		currentOnly=true|
		false
		}
			FC1, FC2

F_GCD-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST	

		} }	

F_GCD-N_TC	Daten zur Identifizierung des Patienten	apiKey,
		patient {
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier
		}
		}
			FC1, FC2, FC3, FC10

F_GCD-N_TCR2	Antwort mit Target-PSN und Dokument.	patient {
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier |
		localIdentifierPSN
		}
		},
		documents [{
		template,
		version,
		consentDate,
		fileType=pdf,
		contentType=base64,
		content
		}]
			FC3, FC10

2.5.10 Add Patient Identifier (F_APID)

Vor allem im Umfeld eines KAS/KIS werden zu einem Patienten mehrere Identifier z.B. für oder durch Einrichtungen, Geräte oder Abteilungen generiert, die neben der Patienten-ID existieren. Ein anderes Beispiel wären noch die Fallnummern eines Patienten. Hiermit sind keine Pseudonyme gemeint, sondern eindeutige Identifier des Patienten in einer Domäne.

2.5.10.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes

F_APID-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_APID-N_SR	SessionResponse	sessionId, uri	
F_APID-N_T	Token anfordern	apiKey, sessionId, type=addPatientIdentifier, reason, study_id, *study_name,* event	FC1, FC2

F_APID-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_APID-N_TC	Daten zum Hinzufügen von lokalen Identifiern zu einem Patienten	apiKey,
		tokenId,
		patient {
		
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier |
		localIdentifierPSN
		}
		},
		newIdentifier [{
		domain,
		name,
		id,
		creationDate
		}]
			FC1, FC2, FC3, FC10

F_APID-N_TCR2	Antwort ggf. mit ErrorCodes	patient {
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier |
		localIdentifierPSN
		}
		},
		errorCode
			FC3, FC10

2.5.11 Confirm Revocation (F_CREV)

Mit dieser Funktion kann eine Person oder ein System die Umsetzung eines Widerrufs (auch Teilwiderruf) gegenüber der Treuhandstelle bestätigen. Durchaus ist hierbei eine digitale Unterschrift (auch über Signpad) denkbar.

2.5.11.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_CREV-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_CREV-N_SR	SessionResponse	sessionId, uri	
F_CREV-N_T	Token anfordern	apiKey, sessionId, type=confirmRevocation	FC1, FC2

F_CREV-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}	

		}	
F_CREV-N_TC	Angabe der Notification, die bestätigt werden soll. Diese muss vom Typ her ein Widerruf sein.	apiKey, tokenId, notificationId	FC1, FC2, FC3, FC10
F_CREV-N_TCR2	Antwort ggf. mit ErrorCodes	notificationId, *errorCode*	FC3, FC10

2.5.12 Request Patient by Identifier (F_RPI)

Über diese Funktion kann mit Hilfe einer PSN oder eines anderen patientenbezogenen Identifier die Daten eines vorhandenen Patienten (IDATs) abgefragt werden.

2.5.12.1 Variante 1 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RPI-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_RPI-N_SR	SessionResponse	sessionId, uri	
F_RPI-N_T	Token anfordern	apiKey, sessionId, type=requestPatientByIdentifier, study_id, *study_name,* *study_shortname,* event, reason, callback	FC1, FC2
F_RPI-N_TR	TokenResponse	tokenId, uri, call {	

		action {
		url,
		method=POST
		}
		}	

F_RPI-N_TC	Daten zur Umwandlung von PSNs	apiKey,
		patients [{
		index,
		patientIdentifier {
		domain,
		name,
		id,
		type
		}
		}]
			FC1, FC2, FC3, FC10

F_RPI-N_C	Callback mit Patienten. Welche Parameter vom Patienten zurückgegeben werden, hängt von den für das Projekt konfigurierten Matching-Kriterien ab.	tokenId,
		patients [{
		patient {
		
		identifier [{
		domain,
		name,
		targetId,
		tempId
		}],
		relatedIdentifier
		[{
		tempId,
		idType,
		targetId
		}],
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
			FC3, FC10

		nationality, race, religion, originDateTime }, contacts [{ city, country, countryCode,district, email, phone, state, street, zipCode, *municipalityKey,* *originDateTime* }] }, tempId }]	
F_RPI-N_CR	Callback-Response mit ErrorCode	patients [{ tempId, *errorCode* }]	

F_RPI-N_TCR	Provider bekommt seine Daten zurück ggf. mit ErrorCode	patients [{
		index,
		tempId
		patientIdentifier {
		domain,
		name,
		id,
		type
		}
		}]	

2.5.12.2 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_RPI-N_S	Session anfordern	apiKey, user_id, user_name,	FC1, FC2

		user_role, *user_firstname,* *user_lastname,* *user_title*	
F_RPI-N_SR	SessionResponse	sessionId, uri	
F_RPI-N_T	Token anfordern	apiKey, sessionId, type=requestPatientByIdentifier, study_id, *study_name,* *study_shortname,* event, reason, callback	FC1, FC2

F_RPI-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_RPI-N_TC	Daten zur Umwandlung von PSNs	apiKey,
		patients [{
		index,
		patientIdentifier {
		domain,
		name,
		id,
		type
		}
		}]
			FC1, FC2, FC3, FC10

F_RPI-N_TCR2	Provider bekommt seine Daten zurück ggf. mit ErrorCode	patients [{
		index,
		patient {
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,	

		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime
		},
		contacts [{
		city,
		country,
		countryCode,district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}],
		patientIdentifier {
		domain,
		name,
		id,
		type
		}
		}]	

2.5.13 Get Consent Status (F_GCS)

2.5.13.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_GCS-N_S	Session anfordern	apiKey, user_id, user_name, *user_role*,	FC1, FC2

		user_firstname, *user_lastname,* *user_title*	
F_GCS-N_SR	SessionResponse	sessionId, uri	
F_GCS-N_T	Token anfordern	apiKey, sessionId, type=getConsentStatus, study_id, study_name, reason, event	FC1, FC2

F_GCS-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_GCS-N_TC	Daten zum Abfragen des ConsentStatus	apiKey,
		tokenId,
		patients[{
		index,
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier |
		localIdentifierPSN
		}
		}]
			FC1, FC2, FC3, FC10

F_GCS-N_TCR2	Antwort ggf. mit ErrorCodes	patients[{
		index,
		patientIdentifier{
		domain,
		name,
		id,
		type=
		patientPSN |
		localIdentifier |
			FC3, FC10

		localIdentifierPSN }, consentStatusType= noConsent | consent | revocation | refusal, expirationDate, *errorCode* }]	

2.5.14 Edit Patient (F_EP)

Das direkte Editieren eines Patienten muss projektspezifisch möglich sein. Je nach Anforderungen können hier verschiedene Funktionen wie „IDATs ändern“ oder „Consent erstellen“ durchgeführt werden.

2.5.14.1 Variante 2 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes[3]
F_EP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_EP-N_SR	Session-Response	sessionId, uri	
F_EP-N_T	Token anfordern	apiKey, sessionId, type=editPatient, study_id, study_name, *study_shortname,* location_id, location_name, event, idType, idString, reason, redirect	FC1, FC2
F_EP-N_TR	Token-Response	tokenId,	

3 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

		uri,
		call {
		form {
		url,
		method=GET
		}
		}	

F_EP-N_TU	Hierbei werden die Daten über Webseiten, die durch den THS-Service bereitgestellt werden, an den Provider übermittelt.	tokenId, optional apiKey, IDATs, Consent-Informationen. Dies ist abhängig vom Projekt.	FC1, FC2, FC10
F_EP-N_R	Redirect-URL für Browser	[{ tempId }]	

2.5.15 Search Patient (F_SP)

2.5.15.1 Variante 1 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes[4]
F_SP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_SP-N_SR	Session-Response	sessionId, uri	
F_SP-N_T	Token anfordern	apiKey, sessionId, type=searchPatient, study_id, study_name, *study_shortname,* location_id, location_name, event, targetIdType, redirect, callback	FC1, FC2

4 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

F_SP-N_TR	Token-Response	tokenId,
		uri,
		call {
		form {
		url,
		method=GET
		}
		}	

F_SP-N_C	Callback mit Target-PSN	tokenId, targetIdType targetId	FC3, FC10
F_SP-N_CR	Callback-Response mit ErrorCode	*errorCode*	
F_SP-N_R	Redirect-URL für Browser		

2.5.15.2 Variante 2 (form)

Nachricht	Beschreibung	Parameter	Fehlercodes[5]
F_SP-N_S	Session anfordern	apiKey, user_id, user_name, *user_role*, *user_firstname*, *user_lastname*, *user_title*	FC1, FC2
F_SP-N_SR	Session-Response	sessionId, uri	
F_SP-N_T	Token anfordern	apiKey, sessionId, type=searchPatient, study_id, study_name, *study_shortname*, location_id, location_name, event, targetIdType, redirect	FC1, FC2
F_SP-N_TR	Token-Response	tokenId, uri,	

5 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

		call {
		form {
		url,
		method=GET
		}
		}	

F_SP-N_R	Redirect-URL für Browser		

2.5.16 Update IDAT – identifizierende Daten (F_UI)

Diese Funktion ermöglicht das Updaten von identifizierenden Daten von Patienten, falls die THS nicht das patientenführende System ist sondern nur die IDAT verschiedener patientenführender Systeme verwaltet (IHE-PIX).

2.5.16.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes[6]
F_UI-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_UI-N_SR	Session-Response	sessionId, uri	

F_UI-N_T	Token anfordern	apiKey,
		sessionId,
		type=updateIDAT,
		event,
		options {
		resultType=simple |
		detailed
		}
			FC1, FC2

F_UI-N_TR	Token-Response	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

6 Beschreibung und Bedeutung der Fehlercodes sind im Anhang zu finden

F_UI-N_TC	Übergabe der neuen IDATs inklusiver aller Identifier (neue und bereits bekannte). Ein bereits bekannter Identifier erhöht die Wiederkennung der Person in der THS.	apiKey,
		tokenId,
		patients [{
		index,
		patient {
		identifier [{
		domain,
		name,
		id
		}],
		patientGroupId,
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		}]
			FC1, FC2, FC10

F_UI-N_TCR	Token-Call-Response.	**resultType → simple:** [{ index,	

		errorCode
		}]
		resultType → detailed:
		[{
		index,
		patient {
		identifier [{
		domain,
		name,
		id
		}],
		patientGroupId,
		firstName,
		lastName,
		middleName,
		prefix,
		suffix,
		civilStatus,
		degree,
		gender,
		birthdate,
		birthPlace,
		mothersMaidenName,
		motherTongue,
		nationality,
		race,
		religion,
		originDateTime,
		},
		contacts [{
		city,
		country,
		countryCode,
		district,
		email,
		phone,
		state,
		street,
		zipCode,
		municipalityKey,
		originDateTime
		}]
		errorcode
		}]	

2.5.17 External patient merge (F_EPM)

Mit dieser Funktion kann eine Person oder ein System die seiner seitige Zusammenführung von Patienten dem THS-Service mitteilen. Dies ist quasi der umgekehrte Fall zur Notification „Patient zusammenführen“ (F_MPM).

2.5.17.1 Variante 2 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_EPM-N_S	Session anfordern	apiKey, user_id, user_name, *user_role,* *user_firstname,* *user_lastname,* *user_title*	FC1, FC2
F_EPM-N_SR	SessionResponse	sessionId, uri	
F_EPM-N_T	Token anfordern	apiKey, sessionId, type=externalPatientMerge, event	FC1, FC2

F_EPM-N_TR	TokenResponse	tokenId,
		uri,
		call {
		action {
		url,
		method=POST
		}
		}	

F_EPM-N_TC	Daten zum Zusammenführen zweier Patienten	apiKey,
		tokenId,
		masterPatient {
		patientIdentifier{
		domain,
		name,
		id,
		type=
		localIdentifier
		},
		patientGroupId
		},
		slavePatient {
		patientIdentifier{
		domain,
		name,
			FC1, FC2, FC3, FC10

		id, type= localIdentifier }, *patientGroupId* }	

F_EPM-N_TCR2	Antwort ggf. mit ErrorCodes	masterPatient {
		patientIdentifier{
		domain,
		name,
		id,
		type=
		localIdentifier
		},
		patientGroupId
		},
		slavePatient {
		patientIdentifier{
		domain,
		name,
		id,
		type=
		localIdentifier
		},
		patientGroupId
		},
		errorCode
			FC3, FC10

2.6 Notifications

2.6.1 Notification: New consent for study (F_NNC)

Es wurde ein Consent für diesen Patienten hinzugefügt.

2.6.1.1 Variante 3 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_NNC-N_MC	Informationen der THS	apiKey, notificationId, creationDate, notificationType=newConsent, expirationDate, patientSignatureDate,	FC1, FC2, FC3, FC10

		study_id,
		targetId,
		targetIdType,
		identifier [{
		domain,
		name,
		id
		}],
		events [{
		name,
		isConsented=true |
		false
		}],
		policies [{
		name,
		version,
		isConsented=true |
		false
		}]	

F_NNC-N_MCR	Antwort ggf. mit ErrorCode	*errorCode*	FC3, FC10

2.6.2 Notification: Revocation (F_NREV)

Der Patient hat komplett widerrufen (kein Teilwiderruf).

2.6.2.1 Variante 3 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_NREV-N_MC	Informationen der THS	apiKey, notificationId, creationDate, notificationType=revocation, study_id, patientSignatureDateRevocation, patientSignatureDate, expirationDateRevocation, targetId, targetIdType	FC1, FC2, FC3, FC10
F_NREV-N_MCR	Antwort ggf. mit ErrorCode	*errorCode*	FC3, FC10

2.6.3 Notification: Patient merge (F_NPM)

In der THS wurden 2 Patienten zusammengeführt aufgrund eines möglichen Matchings, welches die THS ermittelt hat.

2.6.3.1 Variante 3 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_NPM-N_MC	Informationen der THS. Die masterTargetId und slaveTargetId sind die lokalen Identifier (oder PSNs) des Zielsystems. Der „Slave“ wurde dem „Master“ zugeordnet.	apiKey, notificationId, creationDate, notificationType=patientMerge, masterTargetId, slaveTargetId, targetIdType	FC1, FC2, FC3, FC10
F_NPM-N_MCR	Antwort ggf. mit ErrorCode	*errorCode*	FC3, FC10

2.6.4 Notification: Refusal (F_NREF)

In der THS wurde eine Ablehnung zur Teilnahme an einer Studie angelegt.

2.6.4.1 Variante 3 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_NREF-N_MC	Informationen der THS.	apiKey, notificationId, creationDate, notificationType=refusal, patientSignatureDateRefusal, expiationDateRefusal, study_id, targetId, targetIdType	FC1, FC2, FC3, FC10
F_NREF-N_MCR	Antwort ggf. mit ErrorCode	*errorCode*	FC3, FC10

2.6.5 Notification: New Patient (F_NNP)

In der THS wurde ein neuer Teilnehmer angelegt. Der Empfänger bekommt die für Ihn bestimmte patientID z.B. ein lokaler Identifier oder ein Pseudonym.

2.6.5.1 Variante 3 (action)

Nachricht	Beschreibung	Parameter	Fehlercodes
F_NNP-N_MC	Informationen der THS.	apiKey, notificationId, creationDate, notificationType=newPatient, study_id, targetId, targetIdType	FC1, FC2, FC3, FC10
F_NNP-N_MCR	Antwort ggf. mit ErrorCode	*errorCode*	FC3, FC10

3 REST[7]-Implementation

3.1 Technische Umsetzung

Das Konzept der Schnittstelle sieht vor, dass zunächst durch das externe System eine Session über die REST-Schnittstelle angefordert wird. Das Anfordern von Sessions ist durch eine Authentifizierung und Autorisierung abgesichert. Diese Session ist vergleichbar mit einer HTTP-Session einer Webanwendung. Diese Session autorisiert nun das externe System bestimmte Funktionalitäten der Treuhandstelle zu nutzen. Für jede Funktion muss ein sogenanntes Token angefordert werden. Dieses Token ist der zuvor angeforderten Session zugeordnet und bildet eine Funktion ab. Eine Funktion kann z.B. der Aufruf einer REST-Methode sein oder die Berechtigung zum Anlegen eines Patienten über eine Webseite. Sowohl Token als auch Session haben eine zeitlich begrenzte Gültigkeit. Ein Token anzufordern mit einer ungültigen Session ist ebenso wenig möglich wie die Nutzung einer Funktion mit einem ungültigen Token. Nach dem „Einlösen“ eines Token, d.h. Aufruf einer Funktion, ist in der Regel das Token ungültig und kann nicht wieder verwendet werden. Eine Ausnahme bilden Token, die für einen Batch-Vorgang genutzt werden könnten. Diese sind aber auch auf eine gewisse Anzahl von Aufrufen begrenzt. Erfolgt das Einlösen eines Tokens über eine REST-Methode muss zusätzlich zum Token auch ein JSON[8]-Objekt als Parameter übergeben werden.

3.2 Datenformat

Bei der Nutzung der REST-Schnittstelle sind folgende Datenformate für die Übergabe von Parametern zulässig:

- Parameter in REST-URL z.B. https://ths.de/sessions/{SESSIONID}/tokens
- Parameter im HTTP-Header, z.B. apiKey
- JSON-Objekt

7 REST - „Representational State Transfer“ bezeichnet ein Programmierparadigma für Webanwendungen
8 JSON – Javascript Object Notation

3.3 Fehlercodes

Die je Nachricht angegebenen Fehlercodes werden in WebExceptions und HTTP-Status-Codes umgesetzt. Der Fehlercode ist in der Beschreibung der Exceptions zu finden.

3.4 Nachrichten

Konkrete Nachrichten einer Funktion beginnen mit dem Kürzel der Funktion, gefolgt von einem „-“ und dem Kürzel der Nachricht, z.B. „F_AP-N_TC“. Hiermit ist eine eindeutige Bezeichnung der Nachrichten gewährleistet, aber der Bezug zum generischen Konzept bleibt erhalten.

3.4.1 Requests (an THS-Service)

Nachricht	THS-Service-URL	URL-Parameter	HTTP-Header-Parameter
F_-N_S	/rest/sessions	-	apiKey
F_*-N_T	/rest/sessions/{sessionId}/tokens	sessionId (REST[9])	apiKey
F_*-N_TC	Token-Response-Parameter „call → → action → url“	tokenId (REST)	apiKey
F_*-N_TU	Token-Response-Parameter „call → → form → url“	tokenId (GET[10])	-

3.4.2 Responses (von THS-Service)

Nachricht	URL	HTTP-Status-Codes
F_*-N_SR	Response	201, 400, 401, 415, 500
F_*-N_TR	Response	201, 400, 401, 415, 500
F_*-N_TCR	Response	200, 400, 401, 415, 500
F_*-N_TCR2	Response	200, 400, 401, 415, 500
F_*-N_R	Redirect	
F_*-N_TR	Response	200, 400, 401, 415, 500

3.4.3 Requests (von THS-Service)

Nachricht	URL
F_*-N_C	Callback

9 REST-Parameter sind in der URL als Teil des URL-Pfades anzugeben, z.B. „http://test.de/rest/sessions/sessionId/tokens/“

10 GET-Parameter sind URL-Parameter die nach einem „?“ in der URL angegeben werden, z.B. „http://test.de/?tokenId=asdgfaf“

F_N*-N_MC	URL des konfigurierten Notification-Consumers

3.4.4 Responses (an THS-Service)

Nachricht	URL	HTTP-Status-Codes
F_*-N_CR	Response	200, 400, 401, 415, 500, 503
F_N*-N_MCR	Notification-Response	200, 400, 401, 415, 500, 503

4 Anhang

i. Parameter-Beschreibungen

action (call)	Beschreibt eine Service-Methode, bei der das Token eingelöst werden kann. Die URL und die Methode sind ebenfalls angegeben.	String
call	Beschreibt die Möglichkeit das Token beim THS-Service einzulösen.	Object
callback	Für den Callback wird eine URL erwartet, an die die Treuhandstelle nach Abschluss der Funktion (Anlegen des Patienten) die Rückgabeparameter übermittelt. Momentan ist die Angabe einer Callback-URL optional. Da es sinnvoll ist die Rückgabeparameter zwischen Treuhandstelle und Applikation direkt zu übermitteln, anstatt z.B. eine MPI oder PSN nur dem User anzuzeigen, wird der Callback voraussichtlich ein Pflichfeld.	String
consents	Liste von ausgefüllten Consents.	Object
contacts	Kontakte eines Patienten wie z.B. Anschrift.	Object
data	Beschreibt eine Gruppierung von Daten	Object
errorCode	Wird angegeben, wenn ein Fehler vorliegt.	String
event	Eine Bezeichnung für einen Vorgang, der beschreibt, in welchem Zusammenhang die Funktion in der THS aufgerufen wurde, z.B. „Datenexport LIMS“ oder „Forschungsanfrage“. Diese Angabe ist zudem ein Key-Feld für die THS mit dem interne Funktionalitäten (z.B. Consentüberprüfung) ggf. konfiguriert werden.	String
fields	Beschreibt eine Gruppierung von Felder bzw. Parametern	Object
form (im Objekt „call“)	Beschreibt ein Formular, mit dem das Token eingelöst werden kann. Die URL und die Methode sind ebenfalls angegeben.	String
gender	Angabe des Geschlechts: M → männlich F → weiblich U → unbekannt O → anderes	String

identifier (patient)	Liste von lokalen Identifiern wie z.B. Fallnummer, KIS-ID etc.	Object
idString	ID des Patienten zu dem eine Policy abgefragt werden soll.	String
idType	Beschreibt von welchem Typ der idString ist z.B. kann dies ein System sein oder eine Domain wie „mpi“ oder „zdm“ oder „bdm“.	String
isConsented	Gibt als boolschen Wert an, ob die Policy für den angefragten Patienten erfüllt ist.	boolean
location_id	Eindeutige ID des Standortes, wo der Patient aufgenommen werden soll.	String
location_name	Name des Standortes.	String
method (requestPSN)	Bezeichnet den Vorgang der Pseudonymisierung z.B. Forschungsanfrage, Forschungsdatenbank oder Qualitätssicherung. Ist projektspezifisch	String
module (consents)	Ein Modul ist eine konkrete einzelne Einwilligung, der der Patient zustimmen kann.	String
Parameter	**Beschreibung**	**Datentyp**
patient	Identifizierende Informationen eines Patienten.	Object
policyId	ID der Policy die abgefragt werden soll.	String
policyVersionRange	Intervall der Versionsnummern der Policy. Folgendes Pattern ist erlaubt: (…,) oder […,] oder (…,] oder […,) Runde Klammern bedeuten exklusiv und eckige Klammern bedeuten inklusive. Beispiel: [1.3, 1.5) bedeutet 1.3 <= x < 1.5	String
queryType (queryPolicies → options)	Gibt an, auf welcher Grundlage die Funktion ausgeführt werden soll. Folgende Werte sind möglich: • policyBased → es werden die im Request angegebenen Policies verwendet • eventBased → es werden die in der THS zum Event hinterlegten Policies verwendet. Parameter „resultType“ muss hierbei „simple“ sein.	String
reason	Angabe einer Begründung für Auditzweck.	String

redirect	Für den Redirect wird eine URL erwartet, an die z.B. eine Webapplikation zum Anlegen eines Patienten, den Anwender weiterleitet, dies kann z.B. eine Webapplikation des anfragenden Systems sein. Die Angabe einer Redirect-URL ist optional.	String
resultType (addPatient → options)	Gibt an, wie die Rückgabe des Ergebnisses erfolgen soll. Folgende Werte sind möglich: • simple → beschränkt sich auf den ListenIndex als Identifikator des Patienten • detailed → weitere Informationen wie z.B. PSN oder IDAT werden zusätzlich ausgegeben	String
resultType (queryPolicies → options)	Gibt an, wie die Rückgabe des Ergebnisses erfolgen soll. Folgende Werte sind möglich: • simple → Je Patient wird nur ein boolscher Wert zurückgegeben • detailed → Je Patient werden alle Policies und deren Ergebnis zurückgegeben	String
sessionId	eindeutige ID der Session.	String
sourceId	Bezeichnet die PSN des Quellsystems.	String
sourceIdType	Bezeichnet das Quellsystem bzw. die Quelle woher der idString stammt.	String
status (consents)		String
study_id	Eindeutige ID der Studie, der der anzulegende Patient zugeordnet werden soll.	String
study_name	Name oder Bezeichnung der Studie.	String
study_shortname	Kurzform der Studienbezeichnung.	String
targetId	Bezeichnet die PSN für das Zielsystem.	String
targetIdType	Bezeichnet das Zielsystem oder das Ziel für das eine PSN angefragt wird.	String
tempId	Eine eindeutige ID für diese Transaktion. Wird ebenfalls im Callback angegeben, damit eine Zusammenführung der Informationen in unterschiedlichen Systemen erfolgen kann, ohne den Datenschutz zu verletzen.	String
template (consents)	Ein Template ist ein unausgefüllter Consent,	String

	quasi eine Vorlage zum Ausfüllen. Es besitzt eine Version und beinhaltet Module.	
tentative	Gibt an, ob das Pseudonym vorläufig ist, also eventuell ein Duplikat eines vorher bestehenden Patienten darstellt oder neu generiert wurde. Mögliche Werte: „true“, „false“	boolean
type	Der Type gibt an, welche Funktion des THS-Service bzw. welches Token angefordert wird.	String
uri	URI zum Abrufen der Session/Token-Informationen.	String
url	URL des THS-Service für das Einlösen von Token.	String
user_firstname	Vorname des Users.	String
user_id	Die eindeutige User-ID des anfragenden System z.B. angemeldeter Nutzer einer Anwendung oder ein System-User.	String
user_lastname	Nachname des Users	String
user_name	Der Username z.B. eines angemeldeten Benutzers des anfragenden Systems.	String
user_role	Rolle des Users im anfragenden System.	String
user_title	Titel des Users.	String
version (consents)	Dies ist die Version eines Templates.	String

ii.Fehlercodes

Fehlercode	Kurzbezeichnung	Beschreibung/Bedeutung
FC1	NotAllowed	Bedeutet, dass das anfragende Systeme nicht authentifiziert wurde oder nicht autorisiert ist.
FC2	InvalidParameter	Der übergebende Parameter wurde nicht in der benötigten Struktur übergeben.
FC3	NotAvailable	Wenn ein nicht vorhandener Service oder eine unbekannte URL aufgerufen wurde.
FC10	Embedded ErrorCode	Dieser Fehlercode steht allgemein für einen implementierungs-spezifischen Fehlercode.