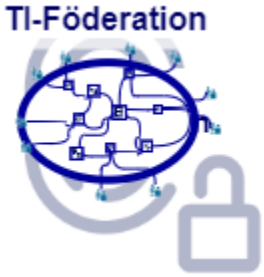
App-App Flow

Die Seite beschreibt den exemplarischen Ablauf der Nutzung einer App auf einem mobilen Endgerät im Kontext föderierter IDPs.



Terminologie

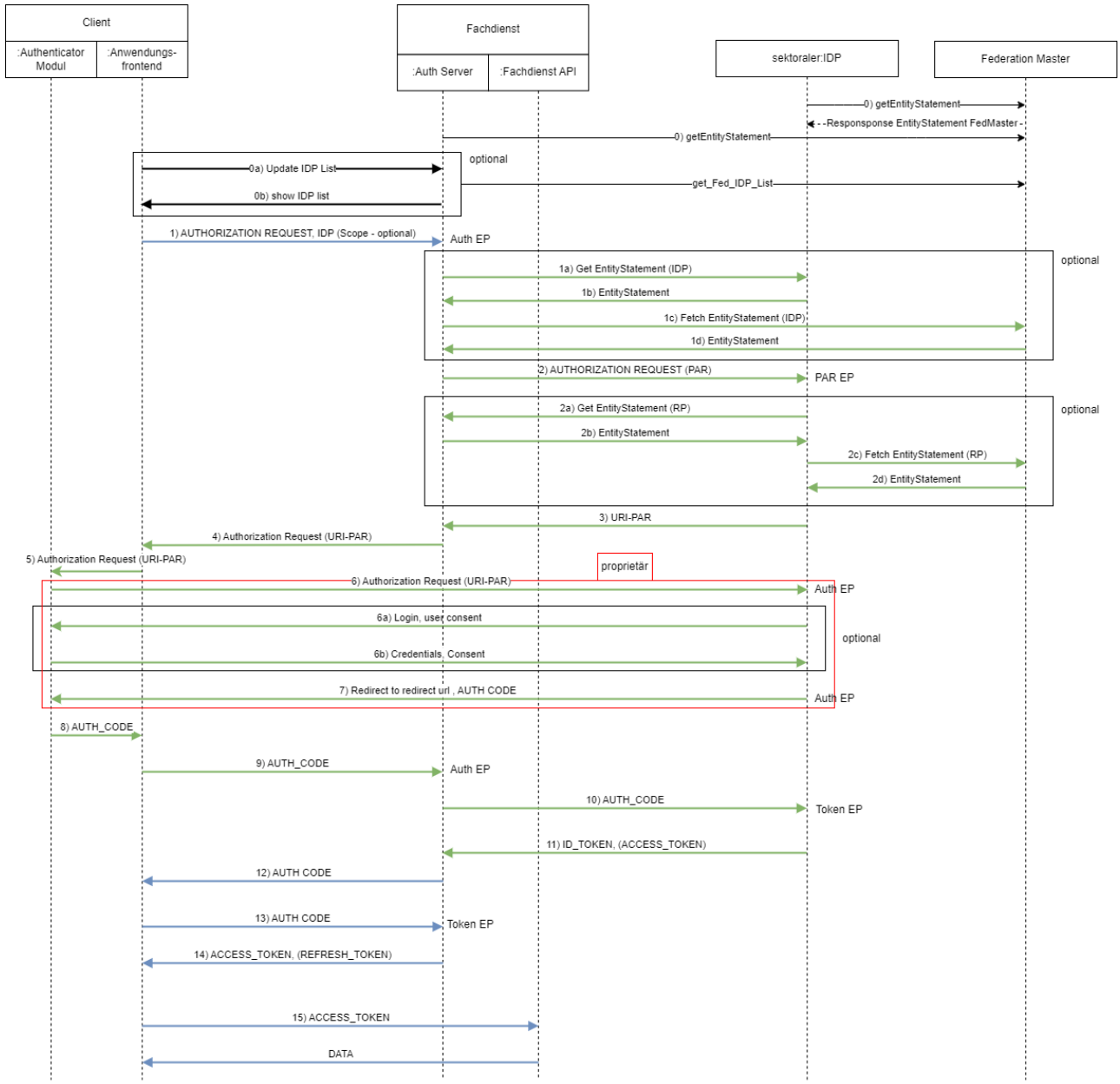
Erläuterungen zur verwendeten Terminologie analog sind zentral für alle Flows hier abgelegt.

Vorbedingungen

- Registrierung des App-Link/Universal-Link für das Frontend auf dem Gerät des Nutzers (auf redirect Adresse des Fachdienst) - oder einreichen über Web.
- Registrierung des App-Link/Universal-Link für das Authenticator-Modul des IDP auf dem Gerät des Nutzers (auf Adresse des IDP) - oder anfragen über Web.
- Aktueller Signaturschlüssel des Federation Master ist bekannt und vertrauenswürdig bei IDP und Fachdienst eingebracht worden.

Flow - OIDC

Flow Diagramm



Legende:

OAuth äußerer Flow | OIDC innerer Flow

Textuelle Beschreibung der Aktivitäten je Transaktion sowie in den Übergängen

Schritt		Funktionen	Beschreibung	Standard
0			Bezug des *Entity Statement* des Federation Master unter Nutzung des bekannten Signaturschlüssels. Das *Entity Statement* wird von einer Entität eines IDP (im föd. Verbund) in Form eines signiertes JSON Web Token (JWT) ausgestellt.	• Entity Statement • Key Rollover for a Trust Anchor

	0-a		Bei Bedarf ruft das Anwendungsfrontend beim Autorisierungsserver die Liste aller in der Föderation registrierten IDPs ab. Die Ermittlung der registrierten IDPs erfolgt über den Federation Master. Beim Federation Master sind zentrale Informationen aus den *Entity Statements* aller registrierten IDP hinterlegt. Die Bereitstellung der Liste kann über zwei Wege erfolgen: a) Der Fachdienst verwendet das OIDC Federation API . Der Fachdienst muss dann aus dem Response die IDP herausfiltern, die für eine Auswahl notwendigen Informationen extrahieren und seinen Anwendungsfrontends zur Verfügung stellen. b) Der Federation Master stellt ein zusätzliches API neben dem Standard-API bereit und liefert hier nur die für eine Auswahl notwendigen Informationen (Name der Organisation/Kasse, Icon, weitere Informationen für Folge-Request zur Ermittlung des vollständigen *Entity Statement*) aller registrierten IDPs. Die Adresse des API kann z.B. als custom-metadata im *Entity Statement* des Federation Master hinterlegt werden.	• Entity Listings Request • OP-Metadata organisation_name • Metadata Erweiterung
	0-b		Der Autorisierungsserver antwortet dem Anwendungsfrontend mit der Liste aller IDPs. Das Anwendungsfrontend zeigt dem Nutzer eine Suchfunktion an, in der er in der Liste seine Kasse per Name und mit Icon auswählen kann.	
1			Das Anwendungsfrontend sendet dem Autorisierungsserver des Fachdienstes einen *Authorization-Request* und eine *code-challenge* sowie den zur Anmeldung gewünschten IDP. Wenn die Wahl des IDP nicht im Anwendungsfrontend getroffen wurde (0a) kann der Autorisierungsserver des Fachdienstes in diesem Schritt einen Auswahldialog anzeigen lassen.	• Authorization Request • PKCE/Code Challenge
	1-a		Falls der Autorisierungsserver des Fachdienstes das *Entity Statement* des IDP noch nicht kennt, lädt er dies herunter. (`/.well-known/openid-federation`)	• Federation Entity Configuration Request
	1-b		Der IDP sendet sein *Entity Statement* an den Autorisierungsserver des anfragenden Fachdienstes zurück.	• Federation Entity Configuration Response • OAuth 2.0 Pushed Authorization Requests
	1-c		Der Autorisierungsserver des Fachdienstes fragt das Teilnehmer *Entity Statement* zum angefragten IDP beim Federation Master an.	• EntityStatement-Request
	1-d		Der Federation Master sendet ein Teilnehmer *Entity Statement* zum angefragten IDP zurück.	• Federation Entity Configuration Response • Validation trust chain

2			Der Autorisierungsserver des Fachdienstes sendet einen *Pushed Authorization Request* (PAR) inkl. *code-challenge* und benötigte r *scopes* an den IDP.	• OAuth 2.0 Pushed Authorization Requests • Authentication Request • PKCE /Code Challenge • Client Authentication
	2-a		Falls der IDP das *Entity Statement* des Autorisierungsserver des anfragenden Fachdienstes noch nicht kennt, lädt er dies herunter. (`/.well-known/openid-federation`)	• Federation Entity Configuration Request
	2-b		Der Autorisierungsserver des Fachdienstes sendet sein *Entity Statement* zurück und der IDP registriert ihn als Client.	• Federation Entity Configuration Response • RP Metadata • Entity Statement • OAuth 2.0 Pushed Authorization Requests
	2-c		Der IDP fragt das Teilnehmer *Entity Statement* zum Autorisierungsserver des Fachdienstes beim Federation Master an.	• EntityStatement-Request
	2-d		Der Federation Master sendet ein Teilnehmer *Entity Statement* zum angefragten Autorisierungsserver des Fachdienstes zurück.	• Federation Entity Configuration Response • Automatic Registration • Validation trust chain • Entity Statement
3			Der IDP sendet eine R*equest-URI* (mit Bezug zum vorherigen *Authorization-Request*) an den Autorisierungsserver des Fachdienstes.	• Request-URI
4			Der Autorisierungsserver des Fachdienstes sendet die R*equest-URI* und *Client ID* an das Anwendungsfrontend des Fachdienstes zur Weiterleitung an die Adresse des Authenticator des IDP.	
5			Anwendungsfrontend des Fachdienstes öffnet das Authenticator-Modul für die eigentliche Authentifizierung des Anwenders (Deep-Link/Universal-Link).	
6			Das Authenticator Modul leitet den *Authentication Request* an den IDP weiter.	

	6-a		Der IDP prüft anhand der URI ob der Request zu einem vorherigen *Authorization-Request* gehört. Der Authorization-Endpunkt des IDP stellt (wenn nötig) entsprechend den angefragten *claims* einen Consent (Zustimmung des Nutzers zur Verarbeitung der angezeigten Daten) zusammen. Der Authorization-Endpunkt des IDP überträgt (wenn nötig) Consent-Abfrage und ggf. für die Authentisierung des Nutzers notwendige Daten zu dem Authenticator-Modul.	
	6-b		Das Authenticator-Modul des IDP fordert den Nutzer (wenn nötig) zur Consent-Zustimmung auf und führt die Authentisierung des Nutzers nach den Verfahren des IDP durch. Das notwendige Vertrauensniveau steht im Request (*acr-claim*). Das Authenticator-Modul des IDP bestätigt dem IDP die erfolgreiche Durchführung der Authentisierung. Der Authorization-Endpunkt des IDP erstellt den *authorization-code (IDP)*.	
7			Der Authorization-Endpunkt des IDP antwortet dem Authenticator Modul mit dem *authorization-code (IDP)* und einem Redirect zum Autorisierungsserver des Fachdienstes.	
8			Das Authenticator Modul des IDP ruft über einen App-Link bzw. Universal-Link entsprechend der Redirect-URL das Anwendungsfrontend des Fachdienstes auf (eigentlich ein Redirect zum Fachdienst aber das Frontend ist auf die Adresse registriert) und übergibt den *authorization-code (IDP).*	
9			Die Anwendungsfrontend des Fachdienstes leitet den *authorization-code (IDP)* an den Autorisierungsserver des Fachdienstes.	
10			Der Autorisierungsserve des Fachdienstesr reicht den *authorization-code (IDP)* und den *code-verifier* beim Token-Endpunkt des IDP ein.	• Authorization Code und Code Verifier • Client Authentication
11			Der Autorisierungsserver des Fachdienstes erhält vom Token-Endpunkt des IDP einen *ID-Token* mit den gewünschten *claims*, und ein *Access-Token* der mit dem öffentlichen Schlüssel aus der Registrierung verschlüsselt ist. Der Autorisierungsserver des Fachdienstes entschlüsselt das *Access-Token*, prüft den Herausgeber *iss*, validiert die Signatur des *ID-Token* gegen den zur *kid* passenden Schlüssel aus den JWKS des IDP und zieht die *claims* (d. h. die Key/Value-Paare im Payload eines Tokens) der authentisierten Identität aus dem *ID-Token*.	
12			Zum weiteren Zugriff erstellt der Autorisierungsserver des Fachdienstes ein *authorization-code (AS)* und sendet diese an das Anwendungsfrontend des Fachdienstes.	
13			Anwendungsfrontend des Fachdienstes übergibt dem Autorisierungsserver des Fachdienstes den *authorization-code (AS)* sowie den *code-verifier* (Token-Endpunkt).	
14			Anwendungsfrontend des Fachdienstes erhält *Access-Token* und *Refresh-Token* mit den notwendigen Daten vom Autorisierungsserver des Fachdienstes.	
15			Das Anwendungsfrontend des Fachdienstes greift auf die Fachdienst API zu und übergibt dabei das *Access-Token*. Nach erfolgreicher Validierung des *Access-Token* gibt die Fachdienst API den Zugriff auf die Fachdaten dieser Identität frei.	

Schnittstellenbeschreibung

(0) Vorbedingung - Abruf der Schlüssel des Federation Master

Dazu wird das selbst signierte *Entity Statement* des Federation Master abgerufen und gegen den vorher bekanntgemachten Signaturschlüssel des Federation Master geprüft.

Response auf GET an die Adresse "http://master0815.de/.well-known/openid-federation"

HTTP 200 mit Content-Type: application/entity-statement+jwt

Folgende Werte müssen im Header des selbst signierten *Entity Statement* des Federation Master auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	-
kid	wie aus *jwks* im Body des Dokumentes	"master0815-1"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des Statement
typ	entity-statement+jwt	-	-

Folgende Werte müssen im Body des selbst signierten *Entity Statement* des Federation Master enthalten sein:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	*iss* anstelle issuer ist hier Spec konform = URL des Federation Master (wird definiert)
sub	URL	"http://master0815.de"	URL des Federation Master (wird definiert) = *iss*

iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645398001	2022-02-21 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1646002800	Beispielhafte Gültigkeit von 7 Tagen
jwks	JWKS Objekt	unter anderem "master0815-1"	Schlüssel für die Signatur des *Entity Statement* Gemäß OpenID Connect Federation werden hier auch Schlüssel für einen Key-Rollover transportiert.
metadata {			Der Block *metadata* enthält eine Reihe von Metadaten, welche z.B. vom Typ des Teilnehmers im Kontext der Föderation abhängt.
federation_entity {			Der Block *federation_entity* enthält die Metadaten für eine Federation Entity.
federation_fetch_endpoint	URL	"http://master0815.de/federation_fetch_endpoint"	Adresse des Endpunktes zum Abrufen einzelner oder aller Statements des Masters über IDPs und Fachdienste
idp_list_endpoint		"http://master0815.de/idp_list.jws"	non Standard Claim - ggf. auch als reine Konfiguration machbar z.B /.well-known/entity_listing
}			Ende des Blocks *federation_entity*
}			Ende des Blocks *metadata*

IDP Liste

Wir folgen der Idee des „IdP Choosers“ - Hier wird vom Nutzer eines Telematik Services erwartet, dass dieser die Institution kennt, welche seine Identität herausgibt. Bei einem Versicherten wäre dies z.B. seine Krankenkasse (bei einem Arzt die zuständige Ärztekammer, usw.).

Begriff	Erläuterung	Beispiel	
Identität	Von Institution gemanagte ID	 IK - Institutionskennzeichen einer gesetzl. Krankenkasse LANR - Lebenslange Arztnummer ...	

Jede Kasse wird als eigener IDP mit eigenen Endpunkten und *Entity Statements* geführt. Ein Dienstleister kann dahinter aber denselben Dienst stehen haben und die Kassen als Mandanten pflegen. Damit bleibt es auch möglich für die Kasse bei fehlender Installation auf einer eigenen Infoseite zu ihren Apps zu verweisen. Kassen geben die Freigabe für ihren Eintrag in der Föderation frei.

Die Liste der Kassen wird aus der Föderation generiert und am Federation Master zum Abruf bereitgestellt. Die Integrität der Liste wird mittels Signatur über einen Schlüssel aus dessen Keyset sichergestellt.

(0 a) Das Anwendungsfrontend fragt die Liste aller IDPs ab, oder der Autorisierungsserver lässt diese Liste selbst im Frontend anzeigen (Webview)

Die Kommunikation zwischen Anwendungsfrontend und Fachdienst ist anwendungsspezifisch und wird hier nicht weiter spezifiziert.

(0 b) Der Autorisierungsserver antwortet dem Anwendungsfrontend mit der Liste aller IDPs

Der Authorization-Server antwortet dem Anwendungsfrontend mit der Liste aller IDPs oder der Authorization-Server lässt diese Liste selbst im Frontend anzeigen.

Diese Liste wird als JWS formatiert und mittels eines Schlüssels des Federation Master signiert.

Das Frontend lässt den Nutzer die Wahl seines IDP (seiner Kasse) treffen oder diese Auswahl erfolgt über eine Webseite des Fachdienstes.

Die notwendigen Informationen können aus den *Entity Statements* gelesen werden. Das signierte JWS der IDP-Liste hat folgende Inhalte

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	*iss* anstelle issuer ist hier Spec konform = URL des Federation Master (wird definiert)
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645398001	2022-02-21 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1646002800	Beispielhafte Gültigkeit von 7 Tagen
idp_entity {			Der Block *idp_entity* enthält die Informationen zu einem registrierten sektoralen IDP in der Liste aller in der Föderation registrierten sektoralen IDPs.
organization_name	String (max. 128 Zeichen)	`"IDP 4711"`	Der Name des IDP zur Anzeige für den Benutzer ist die Definition von *organization_name* im *Entity Statement* des IDP
iss	`URI`	"https://idp4711.de"	issuer Wert des jeweiligen sektoralen Identity Provider (URL) - sollte nach Vorgaben der Föderation der Adresse für die Authentisierung entsprechen
logo_uri	URI	„https://idp4711.de/logo.png“	Parameter *logo_uri* aus dem *Entity Statement* des IDP
user_type_supported	[HCI = Health Care Institution, HP = Health Professional, IP = Insured Person]	"IP"	Parameter *user_type_supported* aus dem *Entity Statement* des IDP
}			Ende des Blocks *idp_entity*

Folgende Werte müssen im Header der vom Federation Master signierten IDP-Liste auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	
kid	wie aus *jwks* im Body des *Entity Statement*	"master0815-1"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des Statement
typ	idp-list+jwt	-	

(1) Authorization Request von Anwendungsfrontend zum Authentication Endpunkt (Auth EP) des Autorisierungsserver des Fachdienstes

Das Anwendungsfrontend sendet ein HTTP-GET an den AS des Fachdienstes.

Die folgenden GET-Parameter werden im query string verwendet:

Name	Werte	Beispiel	Anmerkungen
client_id	VSCHAR	`"eRezeptApp"`	kein ";" und kein "" (definiert gem. Unicode U+253C (9532)), kein Leerzeichen
state	VSCHAR	state_frontend	
redirect_uri	URL	"https://Fachdienst007.de"	Adresse des Fachdienst weil da soll der ACCESS_TOKEN am Ende landen.
code_challenge	Hash über code_verifier	code_challenge_frontend	
code_challenge_method	S256	-	
response_type	`code`	-	
scope	string	"e-rezept"	Anwendungsspezifisch zu definieren kein open-id [RFC6749-section-3.3]

idp_iss	URL	"https://idp4711.de"	• nicht Standard Parameter • *iss* URL des IDP den der Nutzer für die Authentisierung ausgewählt hat. • optional - nötig wenn Auswahl des IDP im Frontend passiert.

(1 a) Falls der Autorisierungsserver des Fachdienstes das *Entity Statement* des IDP noch nicht kennt, lädt er dies herunter

Request:

HTTP-GET

Adresse: "https://idp4711.de/.well-known/openid-federation"

(1 b) Der IDP sendet sein *Entity Statement* zurück

Der Autorisierungsserver verifiziert die Signatur des *Entity Statement* gegen einen Schlüssel aus dem *Entity Statement* des Federation Master über diesen issuer Validation trust chain.

Response:

HTTP 200 mit Content-Type: application/entity-statement+jwt

Folgende Werte müssen im Header des selbst signierten *Entity Statement* des sektoralen IDP auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	
kid	wie aus *jwks* im Body des Dokumentes	"idp4711-3"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des *Entity Statement*
typ	entity-statement+jwt	-	

Folgende Werte müssen im Body selbst signierten *Entity Statement* des sektoralen IDP-Dienstes enthalten sein:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"https://idp4711.de"	*iss* anstelle issuer ist hier Spec konform = URL des IDP (variabel je Mandant /Kasse)
sub	URL	"https://idp4711.de"	URL des IDP (variabel je Mandant/Kasse) = *iss*
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	`1645484401`	2022-02-22 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	`1645570800`	entspricht Gültigkeit von 24 Stunden
jwks	JWKS Objekt	unter anderem "idp4711-3"	Schlüssel für die Signatur des *Entity Statement*
authority_hints	[string]	"http://master0815.de"	*iss* Bezeichnung des Federation Master
metadata {			Der Block *metadata* enthält eine Reihe von Metadaten, welche z.B. vom Typ des Teilnehmers im Kontext der Föderation abhängt.
openid_provider {			Der Block *openid_provider* enthält die Metadaten für einen OpenID Provider.
issuer	URL	"https://idp4711.de"	URL des IDP (variabel je Mandant/Kasse)
signed_jwks_uri	URL	"https://idp4711.de/jws.json"	Ablageort für weitere Schlüssel des IDP etwa die zur Signatur seiner Token Wenn eine signed_jwks_uri im *Entity Statement* angegeben ist müssen diese Schlüssel importiert werden.
organization_name	String		Name des IDP - wird genutzt in der Auswahlliste für den Benutzer (Alternativ name im Feld federation_entity nutzen)
logo_uri	URL	„https://idp4711.de/logo.png“	Attribut ist nicht im Standard, ist nach OpenID Connect Discovery 1.0 - aber in Federation Spec auch für ein OP gelistet
authorization_endpoint	URL	„https://idp4711.de/Auth“	Adresse des IDP-Endpunkt (im Internet)
token_endpoint	URL	„https://idp4711.de/Token“	Adresse des IDP-Endpunkt (im Internet)

pushed_authorization_request_endpoint	URL	„https://idp4711.de/PAR_Auth“	Adresse des IDP-Endpunkt (im Internet) nach RFC9126-section-5
client_registration_types_supported	`[automatic]`	-	
subject_types_supported	[pairwise]	-	
response_types_supported	[code]	-	Weitere Werte sind möglich
scopes_supported	[openid profile email telematik]	-	Weitere Werte sind möglich RFC6749-section-3.3
response_modes_supported	[query]	-	
grant_types_supported	[authorization_code]	-	
require_pushed_authorization_requests	true	-	RFC9126-section-5
token_endpoint_auth_methods_supported	[self_signed_tls_client_auth]		Weitere Werte sind aktuell nicht vorgesehen

request_authentication_methods_supported	<pre>{
	"ar": ["none"],
	"par": ["self_signed_tls_client_auth"]
	}</pre>
		-	

request_object_signing_alg_values_supported	[ES256]	-	
id_token_signing_alg_values_supported	[ES256]	-	Weitere Werte sind möglich.
id_token_encryption_alg_values_supported	[ECDH-ES]	-	Weitere Werte sind möglich.
id_token_encryption_enc_values_supported	[A256GCM]	-	Weitere Werte sind möglich.
user_type_supported	[IP]		IP = Insured Person
}			Ende des Blocks *openid_provider*
federation_entity {			Der Block *federation_entity* enthält die Metadaten für eine Federation Entity.
name	string	"IDP 4711"	Name des IDP - wird genutzt in der Auswahlliste für den Benutzer (alternativ *organization_name* aus Metadata nutzen)
contacts	strings	"support@idp4711.de"	optional
homepage_uri	URL	"https://idp4711.de"	optional
}			Ende des Blocks *federation_entity*
}			Ende des Blocks *metadata*

signed_jwks

Ablageort für weitere Schlüssel des IDP etwa die zur Signatur seiner Token . Wenn eine *signed_jwks_uri* im *Entity Statement* angegeben ist müssen auch diese Schlüssel importiert werden

Folgende Werte müssen im Header des selbst signierten KeySet des sektoralen IDP auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	
kid	wie aus *jwks* im Body des *Entity Statement*	"idp4711-3"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des *Entity Statement*
typ	JWT	-	

Folgende Werte müssen im Body enthalten sein:

Name	Werte	Beispiel	Anmerkungen

keys {			
kty		EC	
kid		idp4711-3	
crv		P-256	
x		qAOdPQROkHfZY1daGofOmSNQWpYK8c9G2m2Rbkpbd4c	
y		G_7fF-T8n2vONKM15Mzj4KR_shvHBxKGjMosF6FdoPY	
use		sig	nach RFC7517-section-4.2
}			
iss	URL	"https://idp4711.de"	= URL des IDP (variabel je Mandant/Kasse)
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645484401	

(1 c) Der Autorisierungsserver des Fachdienstes ruft das *Entity Statement* zum IDP beim Federation Master ab

Request:

HTTP-GET

Adresse: "http://master0815.de/federation_fetch_endpoint"

HTTPS GET Request an den federation_fetch_endpoint aus dem *Entity Statement* des Federation Master mit dem folgenden Parameter:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	issuer des Federation Master - verpflichtender Parameter für unser Szenario aber ohne Relevanz
sub	URL	"https://idp4711.de"	issuer des angefragten sektoralen IDP

(1 d) Der Federation Master sendet sein *Entity Statement* über den angefragten sektoralen IDP zurück

Response:

HTTP 200 mit Content-Type: application/entity-statement+jwt

Folgende Werte müssen im Header des *Entity Statement* des Federation Master über den sektoralen IDP enthalten sein:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	-
kid	wie aus *jwks* im Body des Dokumentes	"master0815-1"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des Statement
typ	entity-statement+jwt	-	-

Folgende Werte müssen im Body des *Entity Statement* des Federation Master über den sektoralen IDP enthalten sein:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	URL des Federation Master
sub	URL	"https://idp4711.de"	URL des angefragten IDP (variabel je Mandant/Kasse)
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645398001	2022-02-21 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645480801	Beispielhafte Gültigkeit von 1 Tag um schneller Sperrungen durchzuführen
jwks	JWKS Objekt	unter anderem "idp4711-3"	Schlüssel für die Signatur des *Entity Statement* des IDP

Als Ergebnis des Schritts (1-d) kennt der Authorization-Server des Fachdienstes die öffentlichen Schlüssel des IDP für Verschlüsselung und Signatur..

(2) Der Autorisierungsserver des Fachdienstes sendet ein (Pushed) Authorization Request an den Authentication Endpunkt (Auth EP) des sektoralen IDP

Der innere Flows startet mit dem Pushed Authorization-Request (RFC9126) des Fachdienst an den sektoralen IDP. Als client_assertion wird self_signed_tls_client_auth verwendet (siehe OIDC Standard OpenID Connect Core 1.0 (section-9)).

Anmerkung: Dies passiert als Folge des Authorization-Request des Anwendungsfrontends.

HTTP-POST

Der Authorization Request des Fachdienst zum sektoralen IDP enthält die folgenden Parameter:

Name	Werte	Beispiel	Anmerkungen
client_id	URL	"https://Fachdienst007.de"	siehe oben: kein ";" und kein "" (definiert gem. Unicode U+253C (9532)), kein Leerzeichen
state	VSCHAR	state_Fachdienst	Das ist ein anderer state als in dem OAUTH Request des Frontend an den Fachdienst
redirect_uri	URL	https://Fachdienst007.de/AS	Adresse des Fachdienst Authorization Server
code_challenge	Hash über code_verifier des Fachdienst	code_challenge_Fachdienst	
code_challenge_method	S256	-	
response_type	`code`	-	
nonce	string	nonce_Fachdienst	Hier nutzen wir auch die Nonce die mit dem ID_TOKEN abgeglichen wird.
scope	string	"urn:telematik:display_name urn:telematik: versicherter openid"	RFC6749-section-3.3
acr_values	"gematik-ehealth-loa-high" oder "gematik-ehealth-loa-substancial"	"gematik-ehealth-loa-high"	

Ein sektoraler Identity Provider welcher den Identitäten für Versicherte verwaltet MUSS mindestens die folgenden Scopes und Claims unterstützen:

Scope	Claim	Wert	Beschreibung
urn:telematik: geburtsdatum	birthdate	string	Die Angaben des Geburtsdatums des Nutzers erfolgt im Format ISO 8601:2004 [ISO86012004] YYYY-MM-DD. Ist das Geburtsdatum nicht bekannt, so wird es (analog einer Festlegung für diesen Fall bei Ausstellung einer eGK) durch diese Regeln erstellt (dabei wird davon ausgegangen, dass das Geburtsjahr immer vorhanden ist): • Ist der Monat aber nicht der Tag des Geburtsdatum bekannt, so wird der 15. des Monat als Geburtsdatum festgelegt (TT.03.1975 ->15.03.1975) • Sind Tag und Monat des Geburtsdatum nicht bekannt, so wird der 15.06. des Jahres als Geburtsdatum festgelegt (TT.MM.1975 ->15.06.1975)
urn:telematik: alter	urn:telematik: claims:alter	string	Alter der Person in Jahren zum Zeitpunkt der Erstellung des Tokens
urn:telematik: display_name	urn:telematik: claims: display_name	string	Analog zu name gemäß [OpenID Connect Core 1.0] Vollständiger Name des Versicherten in anzeigbarer Form inklusive aller Namensbestandteile und ggf. vorhandener Titel oder Namenszusätze.
urn:telematik: family_name	urn:telematik: claims: family_name	string	Nachname des Versicherten
urn:telematik: given_name	urn:telematik: claims: given_name	string	Vorname des Versicherten
urn:telematik: geschlecht	urn:telematik: claims:geschlecht	string	Analog VSDM M =männlich, W = weiblich, X = unbestimmt, D = divers
urn:telematik: email	urn:telematik: claims:email	string	E-Mail Addresse des Versicherten, wenn bekannt.

urn:telematik: versicherter	urn:telematik: claims:profession	string	Für Versicherte 1.2.276.0.76.4.49
	urn:telematik: claims:id	string	Für Versicherte der unveränderliche Anteil der KVNR
	urn:telematik: claims: organization	string	ID oder Name der Attributsbestätigenden Stelle (IK-Nummer der Kasse)

Die angefragten "scopes" werden so mit Werten belegt, wie sie zum Abfragezeitpunkt beim sektoralen IDP (bzw. dessen Quellsystem) vorliegen. Die über den "scope" urn:telematik:email angefragte Adresse ist vor der Verwendung durch den Fachdienst zu verifizieren. Eine Verifikation durch den IDP ist nicht vorgesehen, da diese ohnehin nur eine begrenzte zeitliche Gültigkeit haben kann.

(2 a) Falls der IDP das *Entity Statement* des Autorisierungsserver des Fachdienst noch nicht kennt, lädt er dies herunter.

Request:

HTTP-GET

Adresse: "https://Fachdienst007.de/.well-known/openid-federation"

(2 b) Der Autorisierungsserver des Fachdienst sendet sein *Entity Statement* zurück und der IDP registriert ihn als Client (Automatic Registration)

Der IDP verifiziert die Signatur des *Entity Statement* über einen Dienst gegen einen Schlüssel aus dem *Entity Statement* des Federation Master gemäß den Standards:

- OpenID Connect Federation 1.0 (section-10.1)
- OpenID Connect Federation 1.0 (section-8.2)

Response:

HTTP 200 mit Content-Type: application/entity-statement+jwt

Folgende Claims müssen im Header des selbst signierten *Entity Statement* des Fachdienstes auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	-
kid	wie aus *jwks* im Body des Dokumentes	"Fachdienst007-42"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des Statement
typ	entity-statement+jwt	-	-

Folgende Body-Claims müssen im selbst signierten *Entity Statement* des Fachdienstes enthalten sein:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"https://Fachdienst007.de"	*iss* anstelle issuer ist hier Spec konform = URL des Fachdienst
sub	URL	"https://Fachdienst007.de"	URL des Fachdienst (variabel je Mandant/Kasse) = *iss*
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	`1645484401`	2022-02-22 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	`1645570800`	//Gültigkeit von 24 Stunden
jwks	JWKS Objekt	unter anderem "Fachdienst007-42" "Fachdienst007-69" wenn nicht im *signed_jwks* transportiert	Schlüssel für die Signatur des *Entity Statement*
authority_hints	[string]	"http://master0815.de"	*iss* Bezeichnung des Federation Master
metadata {			Der Block *metadata* enthält eine Reihe von Metadaten, welche z.B. vom Typ des Teilnehmers im Kontext der Föderation abhängt.
openid_relying_party {			Der Block *openid_relying_party* enthält die Metadaten für einen OpenID Relying Party.

signed_jwks_uri	URL	https://Fachdienst007.de/jws.json	enthält Schlüssel für die Signatur des *Entity Statement*, die TLS Client Schlüssel und Zertifikate (x5c, use = sig) und für die Verschlüsselung der ID_TOKEN (use = enc) Wenn eine *signed_jwks_uri* im *Entity Statement* angegeben ist müssen auch diese Schlüssel importiert werden
jwks	Liste von JWKS Objekten	unter anderem "Fachdienst007-69", wenn nicht im *signed_jwks_uri* transportiert	Optional - gemäß OpenID Connect Federation für den Fall das ein Fachdienst *signed_jwks_uri* nicht anbieten kann.
organization_name	String	007 GmbH	Optional: Name der Organisation die hinter dem Fachdienst steht
client_name	String	Fachdienst007	Name des Fachdienstes (redundant zum name in den "Federation Entity" claims)
logo_uri	URL	https://Fachdienst007.de/logo.jpg	Wenn vorhanden zur Darstellung der Anfrage durch den Authenticator/IDP zu verwendet
redirect_uris	[URLs]	https://Fachdienst007.de/client	One of these registered Redirection URI values MUST exactly match the redirect_uri parameter value used in each Authorization Request
response_types	[code]	-	
client_registration_types	`[automatic]`	-	gemäß OpenID Connect Federation
grant_types	[authorization_code]	-	OpenID Connect Registration
require_pushed_authorization_requests	true	-	RFC9126-section-6
token_endpoint_auth_method	self_signed_tls_client_auth	-	
default_acr_values	"gematik-ehealth-loa-high" oder "gematik-ehealth-loa-substancial"	"gematik-ehealth-loa-high"	
id_token_signed_response_alg	ES256	-	Weitere Werte sind möglich.
id_token_encrypted_response_alg	ECDH-ES	-	Weitere Werte sind möglich.
id_token_encrypted_response_enc	A256GCM	-	Weitere Werte sind möglich?
scope	[string]	[urn:telematik:display_name urn:telematik:versicherter openid]	RFC7591-section-2 RFC6749-section-3.3 String mit Space-delimited Scope Values
}			Ende des Blocks *openid_relying_party*
federation_entity {			
name	string	"Fachdienst007"	Optional: Name des Fachdienstes - wird z. B., genutzt in der Consent-Freigabe des Benutzers (redundant zum *client_name*)
contacts	strings	"Support@Fachdienst007.de "	Optional
homepage_uri	URL	"https://Fachdienst007.de"	Optional
}			Ende des Blocks *federation_entity*
}			Ende des Blocks *metadata*

signed_jwks

Ablageort für weitere Schlüssel des Fachdienstes etwa die zur TLS Client Schlüssel und Zertifikate (x5c, use = sig) oder für die Verschlüsselung der ID_Token (use = "enc").

Wenn eine *signed_jwks_uri* im *Entity Statement* angegeben ist müssen auch diese Schlüssel importiert werden.

Folgende Werte müssen im Header des selbst signierten KeySet des Fachdienstes auftauchen:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	-
kid	wie aus *jwks* im Body des Dokumentes	"Fachdienst007-42"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des *Entity Statement*
typ	JWT	-	-

Folgende Werte müssen im Body enthalten sein:

Name	Werte	Beispiel	Anmerkungen
keys {			
kty		EC	
kid		Fachdienst007-42 / Fachdienst007-69	
crv		P-256 / P-256	
x		qAOdPQROkHfZY1daGofOmSNQWpYK8c9G2m2Rbkpbd4c /	
y		G_7fF-T8n2vONKM15Mzj4KR_shvHBxKGjMosF6FdoPY / ...	
use		sig / enc	nach RFC7517-section-4.2 Der Fachdienst listet sowohl sig als auch enc Schlüssel
x5c	base64-encoded (Section 4 of [RFC4648] -- not base64url-encoded) DER [ITU.X690.1994] PKIX certificate	MIIDQjCCAiqgAwIBAgIGATz /FuLiMA0GCSqGSIb3DQEBBQUAMGIxCzAJBgNVBAYTAlVTMQswCQYDVQQ IEwJDTzEPMA0GA1UEBxMGRGVudmVyMRwwGgYDVQQKExNQaW5nI...	selbstsigniertes Zertifikat für die TLS Client Authentisierung des Fachdienst gegenüber dem IDP
}			
iss	URL	"https://Fachdienst007.de"	= URL des Fachdienst
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645484401	

(2 c) Abruf des *Entity Statement* zum Fachdienst beim Federation Master

Request:

HTTP-GET

Adresse: "http://master0815.de/federation_fetch_endpoint"

HTTPS GET Request an den *federation_fetch_endpoint* aus dem *Entity Statement* des Federation Master mit dem folgenden Parameter:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	issuer des Federation Master - Verpflichtender Parameter für unser Szenario aber ohne Relevanz
sub	URL	"https://Fachdienst007.de"	issuer des angefragten Fachdienst

(2 d) Der Federation Master sendet sein *Entity Statement* über den Fachdienst zurück

Response:

HTTP 200 mit Content-Type: application/entity-statement+jwt

Folgende Werte müssen im Header zum *Entity Statement* des Federation Master über den Fachdienst enthalten sein:

Name	Werte	Beispiel	Anmerkungen
alg	ES256	-	-
kid	wie aus *jwks* im Body des Dokumentes	"master0815-1"	Identifier des verwendeten Schlüssels aus dem *jwks* im Body des Statement
typ	entity-statement+jwt	-	-

Folgende Werte müssen im Body des *Entity Statement* des Federation Master über den Fachdienst enthalten sein:

Name	Werte	Beispiel	Anmerkungen
iss	URL	"http://master0815.de"	URL des Federation Master
sub	URL	"https://Fachdienst007.de"	URL des angefragten Fachdienstes
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645398001	2022-02-21 00:00:01
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645480801	Beispielhafte Gültigkeit von 1 Tag für Möglichkeit der Sperrung
jwks	JWKS Objekt	unter anderem "Fachdienst007-42"	Schlüssel für die Signatur des *Entity Statement*

Als Ergebnis des Schritts (2-d) kennt der IDP die öffentlichen Schlüssel des Fachdienstes für Verschlüsselung und Signatur.

(3) Der PAR-Endpunkt (PAR EP) des sektoralen IDP antwortet dem AS des Fachdienst mit einer Request URI

Zuvor verifiziert der IDP das TLS CLientzertifikat gegen einen Schlüssel aus dem *Entity Statement* des Fachdienstes.
Response:

HTTP 201 mit Content-Type: application/json

Name	Werte	Beispiel	Anmerkungen
request_uri	URI	urn:Fachdienst007:bwc4JK-ESC0w8acc191e-Y1LTC2	URI zur späteren Identifikation des Requestes
expires_in	Gültigkeitsdauer der URI	90	nach RFC 6749 - Maximal 90 Sekunden scheint praktikabel

Diese URI wird als redirect an das Anwendungsfrontend gesendet um über das Authenticator Modul den IDP zu erreichen

(4) Der Authorization Server des Fachdienst antwortet dem Frontend mit einem redirect und seiner Request URI

`HTTP-302,`

Mit mindestens den folgenden HTTP-Header Elementen:

- Location

Die Location setzt sich zusammen aus:

<target_url><authorization request des Authorization Server zum sektoralem IDP>

Die target_url entspricht dabei der Adresse des Authorization-Endpunkt des sektoralen IDP entsprechend dem *Entity Statement* welche auf dem Gerät auf das Authenticator Modul weitergeleitet wird.

Der Request des Fachdienst AS zum sektoralen IDP enthält dabei die folgenden Parameter:

Name	Werte	Beispiel	Anmerkungen
client_id	URL	"https://Fachdienst007.de"	Hier muss die URL des Fachdienst eingetragen werden = seine client_id in der Föderation
request_uri	URI	urn:Fachdienst007:bwc4JK-ESC0w8acc191e-Y1LTC2	URI zur späteren Identifikation des Requestes (Rückgabewert des PAR)

(5) Das Anwendungsfrontend sendet den Authentication Request an die URI des IDP und leitet ihn somit an das Authenticator Modul weiter

Das Anwendungsfrontend sendet ein HTTP-GET an den Authorization-Endpunkt des sektoralen IDP.

Die GET-Parameter entsprechen dem Request des Fachdienstes aus Schritt 4:

Das Authenticator Modul des sektoralen IDP fängt diesen Request dadurch das er diese Adresse für App2App Kommunikation im Betriebssystem registriert hat.

(6) Das Authenticator Modul leitet den Authentication Request an den IDP weiter. (proprietär)

Die Schritte zur Nutzer-Authentifizierung und zur Erstellung des AUTHORIZATION_CODE durch den IDP sind anwendungsspezifisch und werden hier nicht weiter spezifiziert.

(7) Der Authorization-Endpunkt des sektoralen IDP antwortet dem Authenticator Modul mit einem Redirect zum Fachdienst. (proprietär)

Beispielsweise

HTTP-302,

Mit mindestens den folgenden HTTP-Header Elementen:

- Location

Die Location setzt sich zusammen aus:

<uri_Fachdienst_AS>?code=<AUTHORIZATION_CODE_IDP>&state=<state_Fachdienst>

Name	Werte	Beispiel	Anmerkungen
uri_Fachdienst_AS	URI	https://Fachdienst007.de/AS	redirect_uri aus der Anfrage in Schritt 2
code	maximal 2000 Zeichen	AUTHORIZATION_CODE_IDP	Authorization_Code des sektoralen Identity Provider
state	VSCHAR	state_Fachdienst	state des Fachdienstes um den Code zu dereferenzieren

(8) Das Authenticator Modul des IDP ruft über einen App-Link bzw. Universal-Link entsprechend der Redirect-URL das Anwendungsfrontend auf und übergibt den "AUTHORIZATION_CODE"

Der App-Link bzw. Universal-Link Aufruf des Authenticator Modul ist anwendungsspezifisch und wird hier nicht weiter spezifiziert.

Das Anwendungsfrontend fängt diesen Request dadurch das er diese Adresse für App2App Kommunikation im Betriebssystem registriert hat.

(9) Das Anwendungsfrontend leitet den "AUTHORIZATION_CODE" an den Autorisierungsserver des Fachdienstes

HTTP-POST (Content-Type: application/x-www-form-urlencoded) nach uri_Fachdienst_AS

Der Request des enthält dabei die folgenden Parameter:

Name	Werte	Beispiel	Anmerkungen
code	maximal 2000 Zeichen	AUTHORIZATION_CODE_IDP	Authorization_Code des sektoralen Identity Provider
state	VSCHAR	state_Fachdienst	state des Fachdienstes um den Code zu dereferenzieren

(10) Der Autorisierungsserver reicht den "AUTHORIZATION_CODE" und den "Code_Verifier" beim Token-Endpunkt des IDP ein.

HTTP POST mit Content-Type: application/x-www-form-urlencoded

Die folgenden Parameter werden im payload verwendet:

Name	Werte	Beispiel	Anmerkungen
grant_type	"authorization_code"	-	
code	<authorization_code des sektoralen IDP base64-kodiert> - maximal 2000 Zeichen	AUTHORIZATION_CODE_IDP	Authorization_Code des sektoralen Identity Provider
code_verifier	<code_verifier des Fachddienstes>	code_verifier_Fachdienst	
client_id	URL	"https://Fachdienst007.de"	URL des Fachdienst = seine Client_ID
redirect_uri	URL	"https://Fachdienst007.de/AS"	

(11) Der Autorisierungsserver erhält vom Token-Endpunkt des IDP einen ID_TOKEN und ACCESS_TOKEN mit den gewünschten Claims, der mit dem öffentlichen Schlüssel aus der Registrierung verschlüsselt ist.

Der Autorisierungsserver des Fachdienstes entschlüsselt den ID_TOKEN und verifiziert anschließend dessen Signatur. Damit endet der innere Flow.

HTTP-200

- Content-Type=application/json
- Cache-Control=no-store
- Pragma=no-cache

Die JSON-Struktur sieht so aus:

{

"access_token": <ACCESS_TOKEN>,

"id_token": <ID_TOKEN>,

"token_type": "Bearer",

"expires_in": 300, (Gültigkeit des ACCESS_TOKEN in Sekunden, RFC6749 section 4.2.2)

}

Der ACCESS_TOKEN wird ignoriert. Hier stellen wir keine Anforderungen.

Der Encryption Header-Claims des ID_TOKEN sieht dabei wie folgt aus:

Name	Werte	Beispiel	Anmerkungen
alg	ECDH-ES	<-	
enc	A256GCM	<-	
kid	wie aus *signed_jwks*	"Fachdienst007-69"	Ein Schlüssel mit der use *enc* aus dem *signed jwks* des Fachdienst
cty	JWT	<-	Ohne einen exp Claim kann hier JWT nach Standard genutzt werden. Der Mehraufwand auch ggf. abgelaufene Token erst entschlüsseln zu müssen steht in keinem Verhältnis zum Aufwand hier NJWT unterstützen zu müssen

Signature Header-Claims des ID_TOKEN sind genau die folgenden:

Name	Werte	Anmerkungen
alg	ES256	P256 wird zugelassen
typ	JWT	
kid	wie aus *jwks* in *Entity Statement* des sektoralen IDP	Für die Signatur des Token verwendeter Schlüssel

Die Body-Claims für den ID_TOKEN des sektoralen IDP sind beispielsweise die folgenden:

Name	Werte	Beispiel	Anmerkungen
iss	URL	https://idp4711.de	Adresse des sektoralen IDP / reicht als Authentizitätsnachweis
sub	Beliebig, aber eindeutig je Nutzer und fest je Fachdienst.	"UserC3PO-666"	Wird als pseudonymer Identifier verwendet und ist einzig relevanter Claim für Dienste die keiner Nutzerdaten erhalten sollen oder wollen.
iat	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645565035	2022-02-22 22:23:55
exp	Alle time Werte in Sekunden seit 1970, RFC 7519 Sect.2,	1645565335	Zeitliche Gültigkeit des Token von 5 Minuten
aud	URL	"https://Fachdienst007.de"	Die client_ID des Fachdienst - dieser hat die Anfrage gestellt.
nonce	String (maximal 512 Zeichen)	nonce_Fachdienst	

acr	"gematik-ehealth-loa-high" oder "gematik-ehealth-loa-substancial"	gematik-ehealth-loa-high	Stärke der durch den IDP durchgeführten Authentisierung des Nutzers
amr	gemäß A_23129	urn:telematik:auth:eID	Details zur durchgeführten Authentisierung des Nutzers auf dem Niveau "gematik-ehealth-loa-high"
urn:telematik:claims:profession	OID	1.2.276.0.76.4.49	Wird immer mit der OID des Versicherten belegt Abhängig von Scope/Claims
urn:telematik:claims:given_name	maximal 64 Zeichen	-	Wird zur Anzeige verwendet und durch Kassen belegt. Möglich ist hier z. B. die Verwendung des Wertes von "givenName" wie im X.509-Zertifikat der eGK (spezifiziert in [gemSpec_PKI_V2] Kap. 5.1.2 in GS-A_4593) Abhängig von Scope/Claims
urn:telematik:claims:organization	maximal 64 Zeichen	-	IK-Nummer der Kasse. Abhängig von Scope/Claims
urn:telematik:claims:id	10 Zeichen (für KVNR)	-	Hier muss die KVNR rein Abhängig von Scope/Claims

(12) Der Autorisierungsserver des Fachdienst erstellt ein AUTHORIZATION_CODE und sendet diesen an das Anwendungsfrontend zum Einreichen beim Token Endpunkt.

Beispielsweise

HTTP-302,

Mit mindestens den folgenden HTTP-Header Elementen:

- Location

Die Location setzt sich zusammen aus:

<https://Fachdienst007.de/Token>?code=<authorization code AS>&state=<state Frontend>

Name	Werte	Beispiel	Anmerkungen
code	maximal 2000 Zeichen	AUTHORIZATION_CODE_AS	Authorization_Code des Fachdienst
state	VSCHAR	state_frontend	state des Frontend um den Code zu dereferenzieren

(13) Anwendungsfrontend übergibt dem Autorisierungsserver den AUTHORIZATION_CODE sowie den Code_Verifier

HTTP POST mit Content-Type: application/x-www-form-urlencoded

Die folgenden Parameter werden im payload verwendet:

Name	Werte	Beispiel	Anmerkungen
grant_type	"authorization_code"	<-	
code	<authorization_code des Fachdienstes base64-kodiert> - maximal 2000 zeichen	AUTHORIZATION_CODE_AS	Authorization_Code des Fachdienst
code_verifier	<code_verifier des Fachdienst>	code_verifier_Frontend	
client_id	VSCHAR	"eRezeptApp"	
redirect_uri	URI	"https://Fachdienst007.de"	

(14) Anwendungsfrontend erhält ACCESS_TOKEN und REFRESH_TOKEN mit den notwendigen Daten vom Autorisierungsserver des Fachdienst

HTTP-200

- Content-Type=application/json

- Cache-Control=no-store
- Pragma=no-cache

Die JSON-Struktur sieht so aus:

{

"access_token": <ACCESS_TOKEN>,

"refresh_token": <REFRESH_TOKEN>,

"token_type": "Bearer",

"scope": "e-rezept",

"expires_in": 300, (Gültigkeit des ACCESS_TOKEN in Sekunden, RFC6749 section 4.2.2)

}

(15) Das Anwendungsfrontend greift auf die Fachdienst API zu und übergibt dabei das ACCESS_TOKEN

Die Kommunikation zwischen Anwendungsfrontend und Fachdienst ist anwendungsspezifisch und wird hier nicht weiter spezifiziert.