

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
17
19
21
23
24
25
26
27
28
29
31

Telematikinfrastuktur 2.0

Spezifikation Zero Trust Access (ZETA)

Version: 2.0.0_CC
Revision: 1658346
Stand: 09.07.2026
Status: zur Abstimmung
freigegeben
Klassifizierung: öffentlich_Entwurf
Referenzierung: gemSpec_ZETA

34

Dokumentinformationen

35 **Gender-Hinweis**

36 Aus Gründen der besseren Lesbarkeit wird in diesem Dokument überwiegend die
 37 männliche Form verwendet. Sämtliche Personenbezeichnungen gelten gleichermaßen für
 38 alle Geschlechter.

39 **Änderungen zur Vorversion**

40 Anpassungen des vorliegenden Dokumentes im Vergleich zur Vorversion können Sie der
 41 nachfolgenden Tabelle entnehmen.

42

43 **Dokumentenhistorie**

Version	Stand	Kap./ Seite	Grund der Änderung, besondere Hinweise	Bearbeitung
1.0.0	28.02.202 5		initiale Erstellung	gematik
1.1.0	06.08.202 5		Einarbeitung ZETA_25_2	gematik
1.2.0	05.12.202 5		Einarbeitung ZETA_25_3	gematik
1.3.0	17.04.202 6		Einarbeitung ZETA_26_1	gematik
2.0.0_CC	09.07.202 6		Einarbeitung ZETA_26_2 Einarbeitung ZETA Stufe 2 u.a. mit detaillierten Abläufen zu mobilen Clients, Ergänzung Notification Management und Client Management, Konkretisierung von HW Attestierung bei stationären Clients	gematik

44

Inhaltsverzeichnis

45	Inhaltsverzeichnis	
<hr/>		
46	1 Einordnung des Dokuments.....	9
47	1.1 Zielsetzung.....	9
48	1.2 Zielgruppe.....	9
49	1.3 Abgrenzungen.....	9
50	1.4 Methodik.....	10
51	1.4.1 Anforderungen.....	10
52	2 Features und Epics.....	11
53	2.1 Client-Registrierung.....	11
54	2.1.1 Wiedererkennung bekannter Clients.....	11
55	2.1.2 Client-Attestation.....	11
56	2.1.3 Device Security Rating.....	12
57	2.2 Policy Enforcement.....	12
58	2.2.1 Zugriffsschutz.....	12
59	2.2.2 HTTP Proxy.....	12
60	2.3 Decide from Policies.....	12
61	2.3.1 Maschinenlesbare Zugriffsregeln.....	12
62	2.3.2 Ein reproduzierbares Ja/Nein.....	12
63	2.3.3 Policies nach Betroffenheit.....	13
64	2.4 Policy-Information und -Administration.....	13
65	2.4.1 Policy-Verwaltung.....	13
66	2.4.2 Monitoring.....	13
67	2.5 Authorization.....	13
68	2.5.1 Autorisierung auf Basis von Policy-Entscheidungen.....	13
69	2.5.2 Client Authentication.....	14
70	3 Einordnung in die TI 2.0.....	15
71	3.1 ZETA als Umsetzung der Zero Trust Architektur.....	16
72	3.2 Kurzbeschreibung der Komponenten.....	17
73	3.3 Kurzbeschreibung der Schnittstellen.....	18
74	4 Technisches Konzept.....	21
75	4.1 ZETA Guard.....	21
76	4.2 Policy Enforcement Point (PEP).....	21
77	4.3 Policy Decision Point (PDP).....	22
78	4.4 ZETA Client.....	23
79	4.5 Policy-Information und -Administration.....	23
80	4.5.1 Policy Information Point (PIP).....	23
81	4.5.2 Policy Administration Point (PAP).....	23
82	4.5.3 PIP und PAP Ausprägung in ZETA.....	24

83 **4.6 Client-Registrierung.....24**

84 **4.7 Monitoring.....24**

85 4.7.1 Security Information and Event Management (SIEM).....24

86 4.7.2 Shared Signals.....25

87 4.7.3 Telemetrie, Monitoring und Logging.....25

88 **4.8 Zusammenspiel mit Identity Provider.....25**

89 **4.9 TI 2.0-Dienst-Backend.....26**

90 **5 Spezifikation.....27**

91 **5.1 Anforderungen an die Sicherheit und den Datenschutz.....27**

92 5.1.1 Übergreifende Anforderungen für Datenschutz und Sicherheit.....27

93 5.1.2 Sicherheits- und Datenschutzanforderungen an Logging und Monitoring.....30

94 5.1.3 Sicherheits- und Datenschutz-Anforderungen an das Security Monitoring.....31

95 5.1.4 Sicherheits- und Datenschutz-Anforderungen an die Protokollierung von

96 Administrationsaktivitäten.....39

97 5.1.5 Sicherheits- und Datenschutz-Anforderungen an die Verarbeitung von Daten

98 mit dem Schutzbedarf "sehr hoch".....40

99 5.1.6 Sicherheits- und Datenschutz Anforderungen an dem ZETA Client in FdVs.....42

100 **5.2 Schlüssel Management und Verwaltung in ZETA.....42**

101 5.2.1 Nomenklatur für Schlüssel-IDs.....42

102 5.2.2 Übersicht der ZETA Guard Schlüssel (Anbieter-Seite).....43

103 5.2.3 ZETA Client Schlüssel (Nutzer-Seite).....46

104 5.2.4 gematik verwaltete Schlüssel (TI).....47

105 **5.3 ZETA Abläufe.....48**

106 5.3.1 Abläufe für stationäre Clients.....49

107 5.3.1.1 *Client Installation und Schlüsselgenerierung.....51*

108 5.3.1.1.1 Schlüsselgenerierung auf Windows und Linux Systemen.....52

109 5.3.1.1.2 Schlüsselgenerierung auf macOS Systemen.....53

110 5.3.1.2 *Client Start mit TPM und ZAS.....54*

111 5.3.1.3 *Service Discovery.....54*

112 5.3.1.4 *Vorbereitung der Client-Registrierung beim ZETA Guard.....56*

113 5.3.1.4.1 ZAS und TPM.....56

114 5.3.1.4.2 Secure Enclave.....57

115 5.3.1.5 *Client-Registrierung.....57*

116 5.3.1.6 *Authentifizierung.....60*

117 5.3.1.6.1 Client Statement mit ZAS und TPM Attestation.....60

118 5.3.1.6.2 Client Statement mit Apple AppAttest.....61

119 5.3.1.6.3 Token Exchange mit Attestation.....63

120 5.3.1.6.4 Token Request mit grant_type=refresh_token.....66

121 5.3.1.7 *Zugriff auf den Resource Server.....68*

122 5.3.1.7.1 Zugriff auf den Resource Server mit ZETA/ASL.....69

123 5.3.1.7.2 Zugriff auf den Resource Server ohne ZETA/ASL.....71

124 5.3.2 Abläufe für mobile Clients.....73

125 5.3.2.1 *Initialisierung und Schlüsselgenerierung.....74*

126 5.3.2.1.1 Android TEE oder Strongbox.....74

127 5.3.2.1.2 Apple Secure Enclave.....76

128 5.3.2.2 *Client Registrierung und Authentifizierung.....79*

129	5.3.2.3 Plattformabhängige Attestierung.....	81
130	5.3.2.3.1 Android.....	81
131	5.3.2.3.2 Apple.....	83
132	5.3.2.3.3 Software Attestierung (Fallback).....	85
133	5.3.2.4 Service Discovery.....	85
134	5.3.2.5 Authentifizierung.....	85
135	5.3.2.5.1 Voraussetzungen.....	86
136	5.3.2.5.2 Übersicht.....	86
137	5.3.2.5.3 Teilablauf (A): Authorization Request mit äußerem und innerem PAR.	87
138	5.3.2.5.4 Teilablauf (B): Nutzerauthentisierung am sektoralen IDP.....	90
139	5.3.2.5.5 Teilablauf (C): Token-Bezug und Ausstellung der ZETA Token.....	91
140	5.3.2.5.6 Authentifizierung für eine andere Resource am gleichen ZETA Guard.	93
141	5.3.3 Authentifizierung ohne Nutzer-Identität.....	94
142	5.3.4 Dienst-zu-Dienst Kommunikation.....	96
143	5.3.4.1 Ausgehende Verbindungen der Dienst-zu-Dienst Kommunikation mit ZG	
144	Client.....	98
145	5.3.4.1.1 Motivation und Abgrenzung.....	98
146	5.3.4.1.2 Architektur.....	98
147	5.3.4.1.3 Konfiguration über die Policy Engine.....	99
148	5.3.4.1.4 Ablauf.....	99
149	5.3.4.1.5 Anforderungen.....	101
150	5.3.5 Token Revocation.....	102
151	5.4 Clientsystem und ZETA Client.....	104
152	5.4.1 Hersteller.....	105
153	5.4.2 Verbindungsaufbau.....	105
154	5.4.2.1 Definition der Endpunkte.....	106
155	5.4.2.2 Adressierung und Gültigkeitsbereich (Audience und Scope).....	106
156	5.4.2.3 Zusammenhang der Token Exchange Komponenten.....	106
157	5.4.2.4 Resultierende Token für den Aufruf von Fachdiensten.....	107
158	5.4.2.5 Versionierung des Token-Ausstellungsverfahrens.....	108
159	5.4.2.6 Anforderungen.....	108
160	5.4.3 Client-Registrierung.....	111
161	5.4.4 Nutzerauthentifizierung.....	112
162	5.4.5 Session Management.....	113
163	5.4.6 Fehlerbehandlung.....	114
164	5.4.6.1 Liste der HTTP-Statuscodes.....	114
165	5.4.6.2 Behandlung von PEP-Fehlern mittels HTTP-Header.....	118
166	5.4.7 ZETA Attestation Service.....	119
167	5.5 Client Management.....	119
168	5.5.1 Client Registrierungsdaten.....	119
169	5.5.2 Client Assertion JWT.....	120
170	5.5.3 Client Statement.....	120
171	5.5.4 Posture Informationen.....	121
172	5.5.4.1 TPM Posture.....	121
173	5.5.4.2 Software Posture.....	121
174	5.5.4.3 Apple Posture.....	122
175	5.5.4.4 Android Posture.....	123
176	5.5.5 Vertrauensmodell, Faktoren und Geltungsbereich.....	124

177	5.5.6 Erstregistrierung (First Use).....	126
178	5.5.7 Folgeregistrierung und E-Mail-Bindung.....	126
179	5.5.8 Schlüssel-Rollover (Proof-of-Possession).....	129
180	5.5.9 Verwaltung von Clients/Geräten.....	130
181	5.5.10 Außerordentliche Löschung (Out-of-Band) und Protokollierung.....	131
182	5.6 ZETA Guard.....	132
183	5.6.1 Klassifizierung der ZETA Guard Komponenten.....	136
184	5.6.1.1 Unveränderliche Kernkomponenten.....	136
185	5.6.1.2 Austauschbare Kernkomponenten.....	137
186	5.6.1.3 Hilfskomponenten und Funktionen.....	137
187	5.6.2 Kommunikation mit Diensten der gematik.....	138
188	5.6.3 Deployment Szenarien.....	138
189	5.6.3.1 Geo-Redundanz.....	138
190	5.6.4 Provisioning Image für den ZETA Guard.....	139
191	5.6.4.1 Struktur und Inhalt.....	139
192	5.6.4.2 Integritätsschutz und Verifikation.....	140
193	5.6.4.3 Verarbeitung durch ZETA Guard Komponenten.....	141
194	5.6.4.4 Lifecycle und Update-Regeln.....	141
195	5.6.5 Laufzeitüberwachung.....	142
196	5.6.5.1 Aktuell eingesetzte Verfahren.....	142
197	5.6.5.1.1 Pod Security Standards (PSS).....	142
198	5.6.5.1.2 Admission Controller.....	143
199	5.6.5.1.3 Network Policies.....	143
200	5.6.5.1.4 Service Mesh.....	143
201	5.6.5.1.5 Ingress und Egress / Gateway.....	144
202	5.6.5.2 Geplante Verfahren.....	144
203	5.6.5.2.1 Pod-Überwachung gemäß Cilium Tetragon.....	144
204	5.6.5.2.2 RBAC-Härtung (Role-Based Access Control).....	145
205	5.7 Policy Enforcement Points.....	145
206	5.7.1 PEP HTTP Proxy.....	145
207	5.7.2 Sicherheits- und Datenschutz-Anforderungen an den PEP.....	151
208	5.8 Policy Decision Point.....	151
209	5.8.1 Policy Engine.....	152
210	5.8.2 PDP Authorization Server.....	152
211	5.8.2.1 PDP Relying Party.....	158
212	5.8.2.2 Claims amr und acr bei Token Request.....	158
213	5.8.2.3 Token-Ausstellung.....	159
214	5.8.2.4 Token Revocation (RFC 7009).....	160
215	5.8.3 PDP Datenbank.....	160
216	5.8.4 Sicherheits- und Datenschutzanforderungen an den PDP.....	161
217	5.9 Policies und Daten.....	162
218	5.10 Telemetriedaten-Service.....	163
219	5.11 HSM Proxy.....	165
220	5.11.1 Sicherheits- und Datenschutzanforderungen an den HSM Proxy.....	165
221	5.12 Notification Service.....	167
222	5.12.1 Ablauf des Push-Versands.....	167
223	5.12.2 Authentisierung.....	169
224	5.12.3 Rollen der Komponenten.....	170
225	5.12.4 Anforderungen.....	170

226	5.13 Betrieb.....	179
227	5.13.1 Anforderungen an Hersteller einer ZETA Komponente.....	179
228	5.13.2 Anforderungen an Hersteller eines TI 2.0-Dienstes.....	179
229	5.13.3 Anforderungen an Anbieter eines TI 2.0-Dienstes.....	179
230	5.13.4 Anforderungen für nahtlose Aktualisierungen.....	180
231	5.13.5 Überwachung des Betriebsstatus.....	181
232	5.13.6 Leistungs-Anforderungen.....	182
233	5.13.7 Betriebliche Schnittstellendefinition.....	184
234	5.13.8 Prozesse zur Inbetriebnahme eines ZETA Guard.....	185
235	5.14 Anforderungen an Dienste der TI.....	186
236	5.15 Sicherheitsleistungen des ZETA Guard für Resource Server.....	186
237	5.16 Weitere Leistungen des ZETA Guard für Resource Server.....	187
238	6 Beispiele und Referenzimplementierungen.....	188
239	7 Anhang A - Verzeichnisse.....	189
240	7.1 Abkürzungen.....	189
241	7.2 Glossar.....	191
242	7.3 Abbildungsverzeichnis.....	193
243	7.4 Tabellenverzeichnis.....	193
244	7.5 Referenzierte Dokumente.....	194
245	7.5.1 Dokumente der gematik.....	194
246	7.5.2 Weitere Referenzen.....	198
247		

1 Einordnung des Dokuments

248

249 Dieses Dokument stellt eine übergreifende Spezifikation dar, ohne einen konkreten Bezug
250 zu einem Produkttypen herzustellen. Anforderungen dieses Dokuments werden
251 Produkttypen, Schnittstellen, Komponenten oder Diensten von konkreten Use Cases bzw.
252 von Fachanwendungen zugewiesen.

253 Die in diesem Dokument beschriebenen Konzepte, Abläufe und Informationsmodelle
254 dienen der Umsetzung der Paradigmen des Zero Trust in der "Telematikinfrastruktur 2.0".

255 Das Zero Trust-Modell ist ein Sicherheitskonzept, das auf dem Prinzip strenger
256 Zugriffskontrollen und dem grundsätzlichen Misstrauen (kein implizites Vertrauen)
257 gegenüber jedem Kommunikationsteilnehmer beruht, selbst denen, die sich bereits
258 innerhalb eines Netzwerkperimeters befinden. Es handelt sich um ein
259 Sicherheitsrahmenwerk, das erfordert, dass alle Nutzer und deren Clients (Gerät und
260 App), sowohl innerhalb als auch außerhalb der Netzwerkperimeter, authentifiziert,
261 autorisiert und kontinuierlich auf ihre Sicherheitskonfiguration und Sicherheitsnachweise
262 überprüft werden, bevor ihnen Zugriff auf Anwendungen und Daten gewährt oder dieser
263 aufrechterhalten wird. Motiviert durch den „Assume Breach“-Ansatz basiert dieses
264 Architekturdesign-Paradigma im Kern auf dem Prinzip der minimalen Rechte aller
265 Entitäten in der Gesamtinfrastruktur.

1.1 Zielsetzung

267 Ziel des Dokuments ist die Sammlung der technischen, betrieblichen und testrelevanten
268 Anforderungen an Clients, Komponenten und Dienste, die Zero Trust-Aspekte beinhalten
269 oder nutzen.

270 Das Ziel des Zero Trust-Ansatzes besteht darin, die IT-Sicherheitslandschaft grundlegend
271 zu transformieren, um den Schutz von Daten, Anwendungen und Systemen vor modernen
272 Bedrohungen und Angreifern zu gewährleisten. Im Gegensatz zu traditionellen
273 Sicherheitsmodellen, die auf dem Konzept eines sicheren Perimeters basieren, setzt Zero
274 Trust auf die Annahme, dass keine Nutzer oder Systeme, unabhängig von ihrem Standort
275 innerhalb oder außerhalb des Netzwerks, von Natur aus vertrauenswürdig sind.

1.2 Zielgruppe

277 Dieses Dokument richtet sich an Architekten und Entwickler von Komponenten, Diensten,
278 Produkttypen, Schnittstellen und Clients für den Datenaustausch im deutschen
279 Gesundheitswesen.

1.3 Abgrenzungen

281 Diesem Dokument ist kein Produkt- oder Anbietertyp zuzuordnen. Anforderungen in
282 diesem Dokument finden Anwendung in Produkt- und Anbietertypen von konkreten
283 Fachanwendungen bzw. Use Cases.

284 **1.4 Methodik**285 **1.4.1 Anforderungen**

286 Anforderungen als Ausdruck normativer Festlegungen werden durch eine eindeutige ID
287 sowie die dem [RFC2119] entsprechenden, in Großbuchstaben geschriebenen deutschen
288 Schlüsselworten MUSS, DARF NICHT, SOLL, SOLL NICHT, KANN gekennzeichnet.

289 Da in dem Beispielsatz „Eine leere Liste DARF NICHT ein Element besitzen.“ die Phrase
290 „DARF NICHT“ semantisch irreführend wäre (wenn nicht ein, dann vielleicht zwei?), wird
291 in diesem Dokument stattdessen „Eine leere Liste DARF KEIN Element besitzen.“
292 verwendet. Die Schlüsselworte werden außerdem um Pronomen in Großbuchstaben
293 ergänzt, wenn dies den Sprachfluss verbessert oder die Semantik verdeutlicht.

294 Anforderungen werden im Dokument wie folgt dargestellt:

295 **<AFO-ID> - <Titel der Afo>**

296 Text / Beschreibung

297 [**<=>**]

298 Dabei umfasst die Anforderung sämtliche zwischen Afo-ID und Textmarke [**<=>**]
299 angeführten Inhalte.

2 Features und Epics

300

301 Der folgende Abschnitt gibt einen groben Überblick über die Features und Epics, die sich
302 in Anwendungen wiederfinden, wenn sie nach dem Paradigma des Zero Trust umgesetzt
303 werden. Diese Epics sind als Enabler zu verstehen, um Fachanwendungen einen sicheren
304 Verbindungsaufbau zwischen Clients und Backenddiensten zu ermöglichen. Es werden
305 keine User Stories formuliert, da für den Verbindungsaufbau keine Nutzerinteraktion
306 angedacht ist.

307 Im Rahmen der Nutzeridentifikation (Authentifizierung) findet eine Verifikation
308 ausgegebener Authentisierungsmerkmale statt, deren Nutzerinteraktion als Teil der
309 Spezifikation des Identity Managements beschrieben sind.

2.1 Client-Registrierung

311 Gemäß des Zero Trust-Ansatzes ist jeder Schnittstellenaufruf potentiell gefährlich, soweit
312 nicht anders festgestellt. Dazu zählt auch das Vertrauen in bekannte bzw. Misstrauen in
313 unbekannte Clients. Clients sind hier als Kombination aus Gerät (z. B. PC oder mobile
314 Device) und App (Client-Software wie FdV oder Primärsystem) definiert. Um Clients
315 wiedererkennbar zu machen, muss eine Registrierung dieser erfolgen. Sind in der
316 Registrierung zusätzliche Sicherheitsmerkmale über den Client und den Aufrufkontext
317 feststellbar, stärken diese das Vertrauen in nachfolgenden Aufrufen fachlicher
318 Schnittstellen.

2.1.1 Wiedererkennung bekannter Clients

320 Die Wiedererkennung bekannter Clients und deren Bindung an identifizierbare Nutzer des
321 Gesundheitswesens muss über eine Registrierung erfolgen. Die Identifikation des Nutzers
322 erfolgt dabei über ein unterstütztes Identifikationsmerkmal (SmartCard oder digitale
323 Identität) und einen selbstgewählten, vom System unterstützten zweiten Faktor (E-Mail,
324 SMS, etc.).

2.1.2 Client-Attestation

326 Die Client-Attestation dient dem Nachweis der Integrität einer Client-Instanz (z. B. einer
327 Frontend-Applikation oder eines Primärsystems) gegenüber einem ZETA Guard. Im
328 Rahmen der Client-Registrierung reicht der Client einen kryptografisch signierten
329 Nachweis ein, der belegt, dass die Anwendung nicht durch unbefugte Dritte modifiziert
330 wurde (Integrität).

331 Dieser Mechanismus ist ein wesentlicher Bestandteil des Zero-Trust-Modells der TI 2.0,
332 um den Missbrauch von Client-Identitäten durch manipulierte Software oder
333 automatisierte Skripte zu verhindern.

334 Die ZETA Client-Attestation erfolgt nach [RFC9334].

335 **2.1.3 Device Security Rating**

336 Zum Einschluss bzw. Ausschluss bestimmter Eigenschaften von Clients, sollen selbige
337 einer automatischen Sicherheitsprüfung unterzogen werden können (Device Security
338 Rating - DSR), soweit es die gegebenen Plattformmechanismen erlauben.

339 **2.2 Policy Enforcement**

340 Für den Zugriff auf personenbezogene und medizinische Daten und zur Sicherstellung der
341 Integrität, Verfügbarkeit, Vertraulichkeit und Authentizität transportierter Daten gelten
342 Regeln. Diese fachlichen, technischen und organisatorischen Regeln gelten bei jedem
343 Zugriff auf Daten, die über eine Schnittstelle zugreifbar gemacht werden.

344 **2.2.1 Zugriffsschutz**

345 Das Policy Enforcement soll als eine Art Gatekeeper bzw. Türsteher den Zugriff auf
346 Schnittstellen von Backendservices durch beliebige Clients durchsetzen. Grundlage ist
347 das Vertrauen in eine Policy-Entscheidung durch eine Komponente zur Auswertung eines
348 Regelwerks.

349 **2.2.2 HTTP Proxy**

350 Der HTTP Proxy stellt sicher, dass nur Requests mit gültigem Access Token sowie
351 bestandenen zusätzlichen Prüfungen an den Resource Server weitergeleitet werden.
352 Welche Prüfungen zusätzlich erfolgen, wird über Attribute im Access Token gesteuert.

353 **2.3 Decide from Policies**

354 Die Menge an Regeln für die Gewähr eines Zugriffs auf Daten oder Schnittstellen speist
355 sich aus gesetzlichen Forderungen bzw. Verboten, Vertragskonstrukten,
356 Sicherheitsmechanismen, Architekturentscheidungen und Informationen aus der
357 "Umgebung" des Betriebs von Clients und Backendservices.

358 **2.3.1 Maschinenlesbare Zugriffsregeln**

359 Die Menge (potentiell) geltender Regeln zur Absicherung des Zugriffs auf Daten und
360 Dienste formt ein Set von Policies. Um im Fall eines Zugriffsversuchs schnell entscheiden
361 zu können, sollen diese Regeln maschinenlesbar definiert sein. Die Regeln sollen
362 zusätzlich menschenlesbar sein, um die Entwicklung und Wartung der Regeln zu
363 vereinfachen.

364 **2.3.2 Ein reproduzierbares Ja/Nein**

365 Die Auswertung eines komplexen Regelwerks liefert bei identischen Eingangsparametern
366 reproduzierbar das identische Ergebnis.

367 **2.3.3 Policies nach Betroffenheit**

368 Regeln beziehen sich auf verschiedene Aspekte einer Zugriffsentscheidung. Es gelten
369 fachliche Regeln, Regeln zur Benutzung von Clients und ebenso technische Regeln sowie
370 solche, die Betriebsumgebung von Backenddiensten betreffend.

371 **2.4 Policy-Information und -Administration**

372 Die Aufgabe des Policy Information Point (PIP) ist es, relevante Attribute und
373 Informationen zur Entscheidungsfindung (Daten) zu liefern, während der Policy
374 Administration Point (PAP) für die Verwaltung und Bereitstellung der Richtlinien (Policies)
375 verantwortlich ist.

376 **2.4.1 Policy-Verwaltung**

377 Eine Policy-Entscheidung kann Eingangsinformation für andere Policies sein, ebenso kann
378 das Ändern von Rahmenbedingungen oder eine Anomalie-Erkennung zur Beeinflussung
379 von Policies führen. Aus diesem Grund führen Beobachtungen über Policy-
380 Entscheidungen zu Informationen über das Gesamtsystem, die als Eingangsdaten für
381 nachfolgende Policy-Entscheidungen herangezogen werden. Daneben ist es erforderlich,
382 Anpassungen am Regelwerk dem System über authentizitäts- und integritätsgeschützte
383 Wege bekannt zu machen.

384 **2.4.2 Monitoring**

385 Durch ein Monitoring von Betriebsparametern und Telemetriedaten wird die
386 Durchsetzung von Policies sowie die Auswirkung möglicher Policy-Änderungen
387 transparent.

388 **2.5 Authorization**

389 Die Autorisierung (Authorization) beschreibt innerhalb von ZETA den Prozess der
390 Prüfung und Erteilung von Zugriffsrechten auf geschützte Ressourcen. Während
391 die *Authentisierung* die Identität eines Nutzers oder Systems feststellt, definiert
392 die *Autorisierung*, welche Operationen dieser Teilnehmer auf welchen Daten oder
393 Diensten ausführen darf.

394 In ZETA Guard bildet die Autorisierung den Kern der Zugriffskontrolle. Ziel ist es,
395 das Prinzip der minimalen Rechtevergabe (Principle of Least Privilege)
396 konsequent umzusetzen.

397 **2.5.1 Autorisierung auf Basis von Policy-Entscheidungen**

398 Die Autorisierung von Zugriffen auf Daten oder Schnittstellen wird bei positiver
399 Entscheidung durch ein Set von Policies gewährt. Die Zugriffsentscheidung und -gewähr
400 bettet sich in eine Verkettung von Informationen und von Aufrufen verschiedener
401 Schnittstellen ein, die dem fachlichen Aufruf einer Schnittstelle bzw. Abruf von Daten
402 voranstehen. Stand der Technik dieses Flows mehrerer Aufrufe und der dabei
403 transportierten Informationen ist der OAuth2-Standard, vgl. [RFC6749 et al.].

404 2.5.2 Client Authentication

405 Menschen und Clients werden anhand sicherer Merkmale authentifiziert, die Identifikation
406 ist nachrangig bzw. in nachgelagerten fachlichen Anwendungsfällen bzw. in fachlichen
407 Zugriffsregeln relevant.

408 Kann ein Mensch oder Client nicht sicher authentifiziert werden oder wird der
409 Authentifizierung zeitlich oder anderweitig nicht vertraut oder passen die Umgebungs-
410 bzw. die den Aufruf begleitenden Parameter nicht zum Vertrauen in die Authentifizierung,
411 wird eine erneute Authentifizierung als erforderlich angesehen ("Step-Up-
412 Authentication").

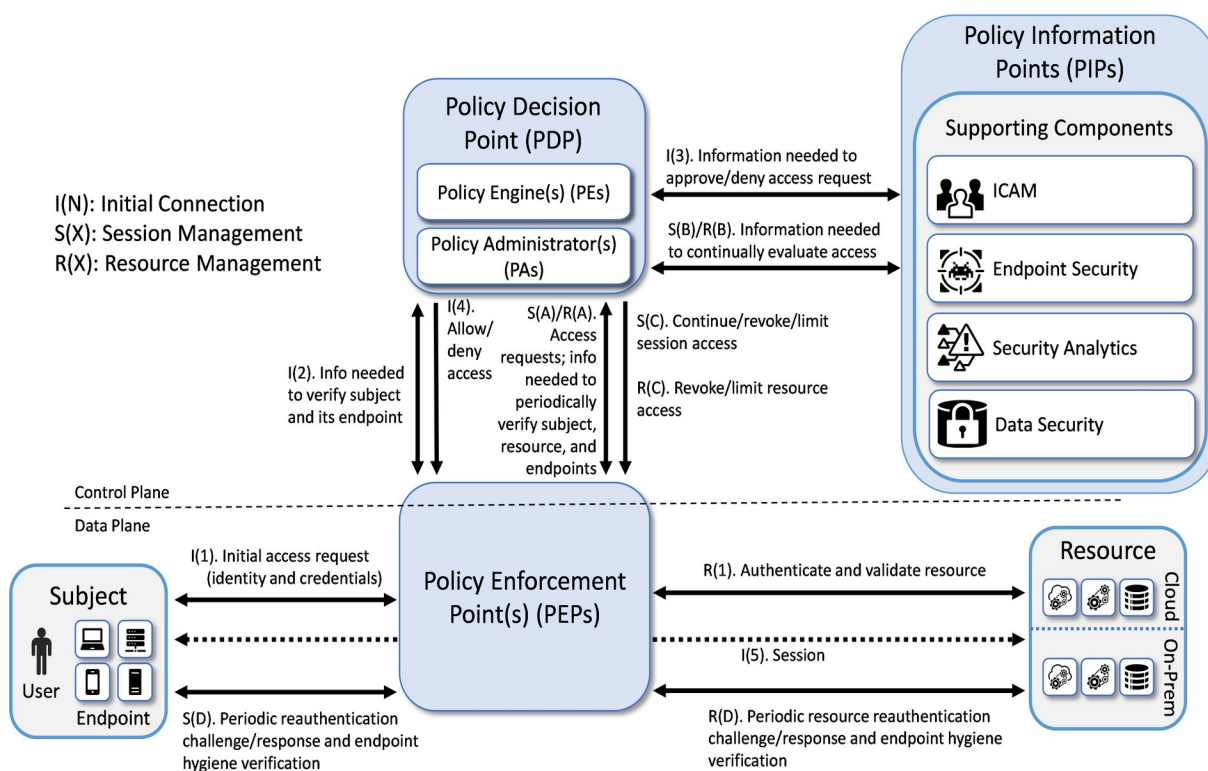
3 Einordnung in die TI 2.0

413

414 Die TI 1.0 bildet eine Infrastruktur, deren Sicherheit auf der sicheren Zugangskontrolle zu
 415 einem geschlossenen zentralen Netzwerk mit Diensten beruht. In der TI 2.0 werden die
 416 Dienste dezentral im Internet angeboten und bedürfen daher eines Schutzes vor
 417 unberechtigtem Zugriff pro Dienst. Dieser Schutz wird nach dem Zero Trust-Paradigma
 418 durch den Policy Enforcement Point und den Policy Decision Point durchgesetzt.

419 Diese übergreifende Spezifikation richtet Anforderungen an Akteure, die sich über das
 420 Internet miteinander vernetzen. Diese Akteure seien im Folgenden einerseits Clients
 421 (Software: Aufrufende einer Schnittstelle, Anfragende an einen Datenabruf oder -zugriff,
 422 wird auf einem bestimmten Client ausgeführt), häufig bedient durch einen Menschen, und
 423 Backendservices (Software: bereitstellende Schnittstelle, Datenbereitstellung etc.) auf der
 424 anderen Seite.

425 Zur Absicherung der Clients und Backendservices werden Anforderungen erhoben, die in
 426 konkreten Softwarekomponenten innerhalb dieser Akteure umzusetzen sind. Die
 427 Separierung der Zero Trust-Mechanismen in unterschiedliche Komponenten folgt der Zero
 428 Trust-NIST-Referenzarchitektur.



429

430

Abbildung 1: NIST Zero Trust-Referenzarchitektur, Quelle [NIST_SP1800-35_FIG1]

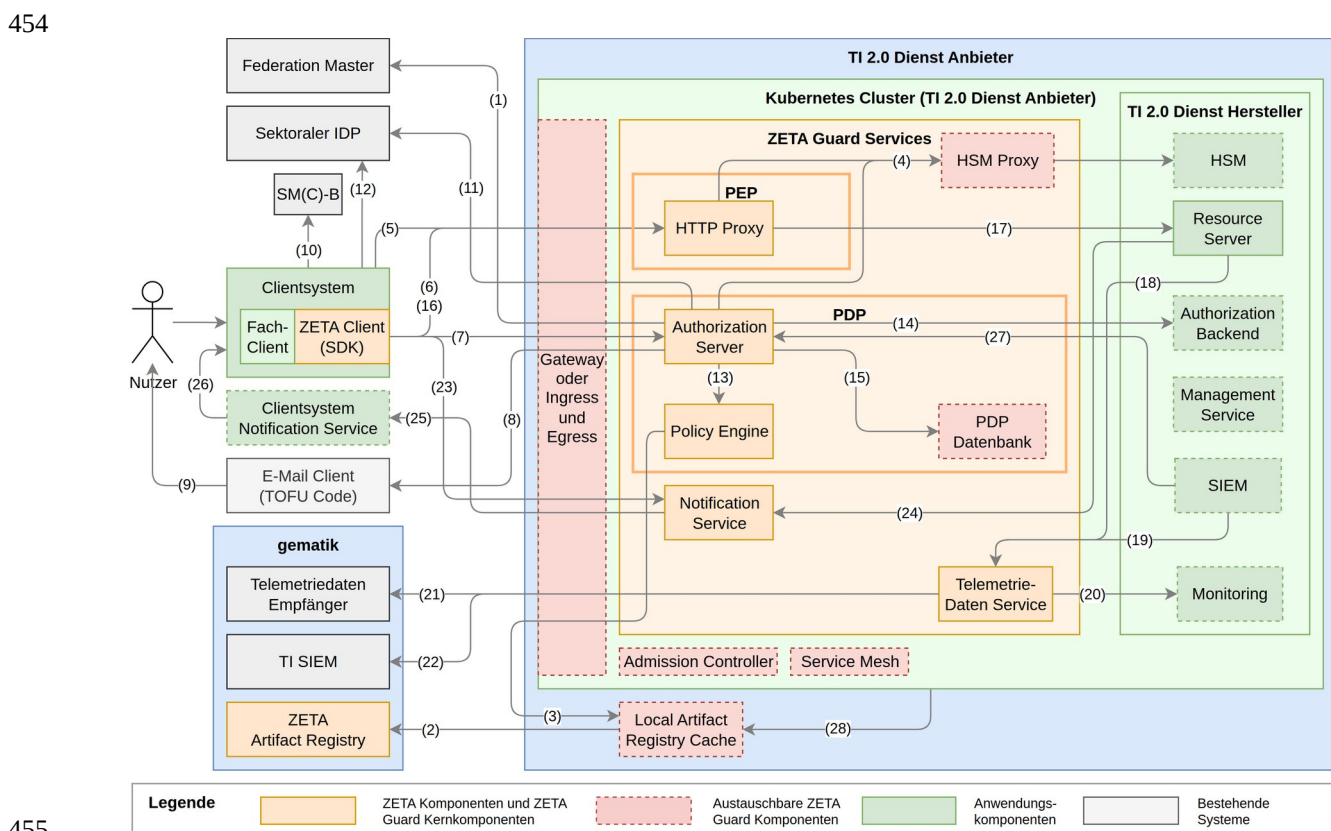
431

432 Im Architekturkonzept der TI 1.0 werden konkrete Umgebungsannahmen zu Consumer
 433 Zonen, Secure Consumer Zonen, Plattformzonen, Personal Zonen usw. getroffen, in
 434 denen kein (Personal Zone) bzw. ein gewisses Sicherheitsniveau (überall sonst)
 435 axiomatisch angenommen wird. Das Zero Trust-Konzept löst sich von der Aufteilung in
 436 verschiedene Zonen, insbesondere, da keine TI-Plattform-Produkttypen in der
 437 Kommunikation zwischen Clients mit Diensten involviert werden.

438 **3.1 ZETA als Umsetzung der Zero Trust Architektur**

439 Im Folgenden ist eine generische Produkttypzerlegung für die Umsetzung der Zero Trust-
 440 Referenzarchitektur einer TI 2.0- Fachanwendung dargestellt. Die Zero Trust Ausprägung
 441 der Telematikinfrastruktur wird Zero Trust Access (ZETA) genannt.

442 In diesem Pattern greift ein Nutzer über einen Clientsystem (inkl. ZETA Client) auf Daten
 443 eines TI 2.0-Dienstes zu. Ein Nutzer wird hier als der Akteur definiert, der einen TI 2.0-
 444 Dienst nutzt. Dabei kann es sich um einen Versicherten oder um einen Mitarbeiter in
 445 einer Organisation des Gesundheitswesens handeln (z. B. LEI, LEO, Krankenkasse). Ein TI
 446 2.0-Dienst setzt sich zusammen aus einem ZETA Guard und mindestens einem Resource
 447 Server. Der ZETA Guard wird als eingebettetes Modul engmaschig mit dem Resource
 448 Server verbunden und vom Anbieter des TI 2.0-Dienstes zwingend mit betrieben. Der
 449 Anbieter implementiert zusätzliche Komponenten, um seine Infrastruktur vor Angriffen
 450 aus dem Internet zu schützen und um die Zugriffe zu optimieren (z. B. Content Delivery
 451 Network, eigener Ingress und Load Balancer, Web Application Firewall). Das folgende Bild
 452 zeigt eine Übersicht der beteiligten Komponenten in der Vernetzung zwischen einem
 453 Clientsystem (links grün) und einem Backendservice (rechts grün: Resource Server).



455 **Abbildung 2: Abb-ZETA-Architektur**

456
 457
 458 Die obige Abbildung zeigt die Einbettung von ZETA bezogenen, logischen Komponenten
 459 (orange) in die Aufrufkette zwischen einem Clientsystem und einem Resource
 460 Server (grün). Darüber hinaus soll durch die roten und gestrichelt umrandeten Komponenten
 461 hervorgehoben werden, dass je nach Einsatzumgebung und Anforderungen an die
 462 Umgebung verschiedene Deployment-Szenarien unterstützt werden. Dargestellt sind
 463 zusätzlich heute bereits vorhandene und genutzte Komponenten und Dienste, die für die
 464 Nutzerauthentifizierung (z. B. eGK und IDP) bzw. die Betriebsüberwachung (z. B. mittels
 465

466 gematik Telemetriedaten Empfänger) in Anwendungsfällen der TI 2.0 weitergenutzt
467 werden können (grau). In diese Abbildung sind diverse Architekturentscheidungen
468 eingearbeitet, die im Kapitel 4 und 5 erläutert bzw. spezifiziert werden.

469 **3.2 Kurzbeschreibung der Komponenten**

470 Zu ZETA gehören ZETA Guard, ZETA Client und die ZETA Artifact Registry. Der ZETA
471 Client implementiert clientseitig die Schnittstellen des ZETA Guard und die Attestation-
472 Funktionen des Clientsystems. Der ZETA Client wird von einem Fach-Client initialisiert
473 und mit fachlichen HTTP Requests aufgerufen. ZETA Client führt alle Schritte aus, um ein
474 Access Token vom ZETA Guard zu erhalten, um dann mit diesem Access Token den
475 Request vom Fach-Client an den ZETA Guard zu senden. Die Response wird an den Fach-
476 Client weitergeleitet. Die ZETA Artifact Registry stellt signierte Images für die ZETA Guard
477 Komponenten sowie signierte Policies und Daten für die ZETA Guard Policy Engine bereit.
478 Zusätzlich werden Provisioning Daten wie aktuelle TSL und TPM Hersteller-
479 Stammzertifikate bereitgestellt.

480 **ZETA Guard besteht aus folgenden Komponenten:**

- 481 • PEP HTTP Proxy: Die zentrale Durchsetzungsinstanz für den Datenverkehr.
- 482 • PDP Authorization Server: Die Instanz zur Ausstellung von Access Token und Session-
483 Management.
- 484 • PDP Policy Engine (OPA): Die Komponente zur Auswertung der Zugriffsregeln.
- 485 • Telemetriedaten-Service: Der OpenTelemetry Collector für sicherheitsrelevante
486 Ereignisse, Traces, Log-Daten und Metriken.
- 487 • Notification Service: Optionale ZETA Guard Komponente zur einheitlichen
488 Bereitstellung von Notification-Funktionen für mobile Clients.
- 489 • PDP Datenbank: Persistenzschicht für Sessions sowie Nutzer- und Client-Daten.
- 490 • HSM Proxy: Schnittstelle zu den ZETA Guard Komponenten und das Hardware-
491 Sicherheitsmodul des Anbieters.
- 492 • Local Artifact Registry Cache: Lokales Repository zur Bereitstellung der Images
493 innerhalb der Anbieter-Umgebung.
- 494 • Gateway oder Ingress und Egress: Netzwerk-Einstiegspunkt für den Kubernetes
495 Cluster und kontrollierter Grenzübergang für den ausgehenden Verkehr.
- 496 • Admission Controller: Hilfskomponente, die Anfragen an den Kubernetes API Server
497 überwacht um Security Policies (wie z. B. die Prüfung der Image-Signatur, bevor ein
498 Image ausgeführt werden kann) durchzusetzen.
- 499 • Service Mesh: Infrastrukturschicht, die die Kommunikation zwischen den einzelnen
500 Microservices innerhalb eines Clusters steuert, sichert und überwacht.

501 **3.3 Kurzbeschreibung der Schnittstellen**

502 (1) Die ZETA Guard Instanz wird beim TI Federation Master registriert. Dadurch wird die
503 Zugehörigkeit des ZETA Guard zur TI (in einer bestimmten Umgebung wie Prod, Ref oder
504 Test) durch Signatur des Federation Masters verifizierbar. In einer Folgeversion wird ZETA
505 Guard im Authorization Server Well-known ein Trustmark vom Federation Master
506 enthalten, das die Zugehörigkeit zur TI nachweist.

- 507 (2) Die gematik stellt eine ZETA Artifact Registry bereit, die Helm Charts, Images für die
508 ZETA Guard Komponenten und Provisioning Images als OCI Container Images bereitstellt.
509 Alle Images haben eine Signatur mit einer gematik Identität. Anbieter von TI 2.0-Diensten
510 betreiben einen lokalen Artifact Registry Cache, der die ZETA OCI Images direkt in der
511 Umgebung des Anbieter verfügbar macht. In einer Folgeversion wird der Local Artifact
512 Registry Cache Teil von ZETA Guard, um die Anbieter von TI 2.0 Diensten zu entlasten.
- 513 (3) Die Policy Engine erhält die Policies und Daten direkt aus der ZETA Artifact Registry.
- 514 (4) Die Schnittstelle des HSM Proxy Richtung ZETA Guard Komponenten ist spezifiziert
515 und ermöglicht die einheitliche Anbindung von HSMs verschiedener Hersteller. Der HSM
516 Proxy enthält aktuell nur Funktionen, die von ZETA Guard Komponenten benötigt werden.
517 Grundsätzlich soll der HSM Proxy auch Komponenten des Resource Servers den
518 einheitlichen Zugriff auf HSMs ermöglichen. Fehlende Funktionen werden nach Bedarf
519 ergänzt.
- 520 (5) Falls eine Mandantentrennung verwendet wird, dann kann der Fachclient durch
521 Abfrage eines Well-known JSON-Dokuments die zu verwendende URL des Resource
522 Servers ermitteln.
- 523 (6) Der ZETA Client bietet dem Fach-Client Operationen zur Initialisierung und zum
524 Versenden von HTTP Requests an. Das heißt, der Fach-Client erstellt einen vollständigen
525 HTTP Request inkl. fachlich benötigter Header und ruft den ZETA Client zur Ausführung
526 des Requests auf. Der ZETA Client ermittelt aus der URL des HTTP Requests im ersten
527 Schritt die Well-known Metadaten des Resource Servers und des Authorization Servers
528 und konfiguriert sich für die Nutzung der ZETA Guard Instanz.
- 529 (7) Aus dem Authorization Server Well-known hat der ZETA Client die Endpunkte für die
530 Client-Registrierung und Autorisierung ermittelt und beginnt den OAuth Flow.
- 531 (8) Bei mobilen Clients erfolgt eine Trust On First Use Bindung des Clients an eine E-Mail-
532 Adresse des Nutzers.
- 533 (9) Der Nutzer muss einen Code aus der Mail vom ZETA Guard im mobilen Clients
534 eingeben, um den Registrierungsprozess des mobilen Clients fortzusetzen.
- 535 (10) Bei Primärsystemen erfolgt die Authentifizierung des Nutzers (hier der Organisation)
536 durch ein SMC-B-signiertes Token per OAuth Token Exchange. Die Signatur kann durch
537 Nutzung eines Konnektors oder TI Gateways aber auch durch Nutzung eines direkt am
538 Primärsystem angeschlossenen Kartenterminals erfolgen.
- 539 (11), (12) Mobile Nutzer werden mit Hilfe des sektoralen IDPs und des OIDC Flows
540 authentifiziert.
- 541 (13) Nachdem die Session-, Client- und Authentifizierungsdaten am Authorization Server
542 validiert wurden, werden diese Daten an die Policy Engine übergeben. Die Policy Engine
543 wendet die Daten auf die Policies und vorgegebenen Wertebereiche an, ermittelt eine
544 Entscheidung und sendet die Entscheidung an den Authorization Server.
- 545 (14) Wenn eine Anwendung eine zusätzliche Authorization benötigt, dann wird der
546 Authorization Server so konfiguriert, dass das Authorization Backend für weitere
547 Autorisierungs-Schritte angefragt wird.
- 548 (15) Der Authorization Server speichert seine Session-, Client- und Userdaten in der PDP
549 Datenbank. Wenn die Policy Engine eine "Allow"-Entscheidung gefällt hat und auch kein
550 Verbot vom Authorization Backend vorliegt.
- 551 (16) Der ZETA Client sendet einen Request Richtung Resource Server. Der HTTP Proxy
552 erlaubt den Zugriff auf Daten des Resource Servers, wenn ein gültiges Access Token im
553 Authorization Header enthalten ist.

554 (17) Nach erfolgreicher Prüfung des Access Token und des DPoP Token (wenn vorhanden
555 auch des PoPP Token) erfolgt die Weiterleitung des Requests zum Resource Server. Falls
556 ZETA/ASL verwendet wird, erfolgt zuerst der ASL-Verbindungsaufbau.

557 (18) Vom Resource Server werden Traces und die Selbstauskunft an den
558 Telemetriedaten-Service gesendet. Der Telemetriedaten-Service empfängt von allen
559 ZETA Guard Komponenten Traces, Logs und Metriken. Die Verbindungspfeile dafür sind
560 nicht dargestellt. Zum Einsatz kommt OpenTelemetry. Alle von OpenTelemetry
561 unterstützten Protokolle können vom Anbieter durch Konfiguration des Telemetriedaten-
562 Service verwendet werden, um Daten an den Telemetriedaten-Service zu senden und
563 vom Telemetriedaten-Service zu empfangen. Empfohlen wird das native OTLP Protokoll
564 über gRPC.

565 (19) Der Telemetriedaten-Service empfängt vom SIEM des TI 2.0-Dienst-Anbieters
566 Security Events.

567 (20) Der Telemetriedaten-Service kann Monitoring-Daten der ZETA Guard Komponenten
568 an das Monitoring des TI 2.0-Dienst-Anbieters senden.

569 (21) Traces, Metriken und Log Events werden an den Telemetriedaten-Empfänger der
570 gematik weitergeleitet.

571 (22) Security Events werden an das TI SIEM der gematik weitergeleitet.

572 (23) Das Clientsystem kann über den ZETA Client eine Push-Konfiguration im ZETA Guard
573 Notification Service einrichten.

574 (24) (25) Wenn Ereignisse eintreten, für die der Resource Server eine Push Nachricht für
575 den Nutzer generiert, dann werden entsprechend der vorhandenen Push-Konfigurationen
576 des Nutzers im Notification Service, Benachrichtigungen an den Client gesendet.

577 (26) Über den Clientsystem Notification Service (und hier nicht dargestellte Notification
578 Services des mobile OS Herstellers) werden Notifications an das Clientsystem gesendet.

579 (27) Mittels Shared Signals kann das Monitoring (oder eine andere Komponente des TI
580 2.0-Dienst-Anbieters) die Session eines Clients beenden und somit eine neue
581 Authentifizierung erzwingen. Aktuell ist die Shared Signals Unterstützung durch SIEM
582 Systeme und Infrastrukturkomponenten noch nicht hinreichend gegeben. Für die Session-
583 Beendigung wird daher eine anwendungsspezifische, serverseitige Auslösung der
584 Session-Termination über die Plug-In-Schnittstelle des Authorization Servers umgesetzt.

585 (28) Der Kubernetes Cluster bezieht seine Deployment-Images aus dem lokalen Artifact
586 Registry Cache sowie die Provisioning Daten für ZETA Guard (TSL, roots.json, TPM-
587 Hersteller Stammzertifikate, Federation Master URL).

588 Der Admission Controller setzt durch, dass nur von der gematik signierte OCI Container
589 Images der ZETA Guard Kernkomponenten ausgeführt werden können.

590 Über ein Service Mesh kann sichergestellt werden, dass nur die Komponenten, die
591 miteinander kommunizieren dürfen, eine Verbindung zueinander aufbauen können.

592 Der optionale Management Service überwacht die Konfiguration der ZETA Guard
593 Komponenten und setzt durch, dass die im Git Repository der ZETA Guard Instanz
594 gespeicherte Konfiguration ausgeführt wird. Dadurch wird ein Configuration-Drift
595 unterbunden.

4 Technisches Konzept

596

597 Im Kapitel zuvor wurden zwei Abbildungen vorgestellt, welche technischen ZETA Guard
598 Komponenten (orange) an der Umsetzung fachlicher Anwendungsfälle von Clients,
599 Komponenten und Backendservices von Fachanwendungen (grün) beteiligt sind. Im
600 Folgenden werden diese technischen Komponenten genauer beschrieben und
601 eingeordnet, welche Rolle sie in einer Architektur nach dem Zero Trust-Paradigma
602 einnehmen.

603 Zero Trust in der TI zeichnet sich über folgende Eigenschaften aus:

- 604 • Registrierung des Clients (Gerät und App) zu einer Identität
- 605 • Attestation der Client-Eigenschaften
- 606 • Bereitstellung einer von Maschinen interpretierbaren Policy durch die gematik
- 607 • Einheitliches Durchsetzen der Policy durch den ZETA Guard
- 608 • Sicherstellung des Sicherheitszustands der gesamten TI, Anbieter übergreifend
- 609 • Telemetrie und Monitoring

4.1 ZETA Guard

611 Der ZETA Guard besteht aus Policy Enforcement Point (PEP), Policy Decision Point (PDP)
612 sowie betriebsunterstützenden Komponenten (Management Service und Telemetriedaten
613 Service). Der PEP besteht aus einem HTTP Proxy. Der PDP enthält die Policy Engine, den
614 Authorization Server und eine Datenbank. Jeder TI 2.0 Dienst hat einen ZETA Guard zum
615 Schutz des Dienstes vor unberechtigtem Zugriff. Der ZETA Guard wird in der
616 Verantwortung des TI 2.0-Dienst-Anbieters betrieben.

4.2 Policy Enforcement Point (PEP)

618 Ein Policy Enforcement Point (PEP) ist eine Schlüsselkomponente im Zero Trust-
619 Paradigma, der darauf abzielt, dass Sicherheitsmodell von einem vertrauensbasierten auf
620 ein verifizierungsbasiertes umzustellen. Der PEP dient dazu, den Zugriff auf Ressourcen,
621 basierend auf vordefinierten Richtlinien, zu kontrollieren und durchzusetzen. Im Kontext
622 der TI 2.0 übernimmt der PEP folgende Funktionen:

- 623 • Der PEP agiert als HTTP Proxy, der den Datenverkehr zwischen Clientanwendungen
624 und den zu schützenden Ressourcen kontrolliert. Dadurch kann der PEP den gesamten
625 Datenverkehr überwachen und filtern, um sicherzustellen, dass er den festgelegten
626 Sicherheitsrichtlinien entspricht.
- 627 • Der PEP stellt die Außenschnittstelle des Dienstes dar, in den er integriert ist.

628 Insgesamt agiert der PEP als Kontrollpunkt in der Zero Trust-Architektur, der sicherstellt,
629 dass nur autorisierte Nutzer und Clients Zugriff auf die Ressourcen eines Dienstes
630 erhalten und dass dabei die definierten Sicherheitspolicies eingehalten werden. Die
631 Entscheidung zwischen verschiedenen Policies auf Basis der vom Client übergebenen
632 Signale, Sicherheitsnachweise und Token trifft der Policy Decision Point. Am PEP werden
633 Betriebsdaten erhoben, verarbeitet und dem Telemetriedaten Service im ZETA Guard zur
634 Verfügung gestellt.

635 4.3 Policy Decision Point (PDP)

636 Ein Policy Decision Point (PDP) ist die wesentliche Komponente im Zero Trust-Paradigma,
637 die Zugriffsentscheidungen trifft, indem sie Richtlinien (Policies) interpretiert und anhand
638 dieser Richtlinien Zugriffsanfragen bewertet. Folgende Funktionen eines PDP sind von
639 besonderer Bedeutung:

- 640 • Der PDP fungiert als OAuth2 Authorization Server und verwaltet die Autorisierung von
641 Nutzeranfragen auf geschützte Ressourcen. Zudem überwacht der Authorization
642 Server die Nutzersessions, um sicherzustellen, dass sie gültig sind und den
643 Sicherheitsrichtlinien entsprechen. Der Authorization Server ist als vertrauenswürdige
644 Relying Party im föderierten Identitätsmanagement registriert. Dadurch kann der
645 Authorization Server Identitätsinformationen von Nutzern sicher und
646 vertrauenswürdig beziehen und bei Bedarf eine (erneute) Nutzerauthentifizierung an
647 die IDPs delegieren. Der Authorization Server stellt sicher, dass nur authentifizierte
648 Nutzer mit registrierten Clients Zugriff auf die geschützten Ressourcen erhalten.
- 649 • Der PDP ermöglicht am Authorization Server die dynamische Registrierung von
650 Clients, die auf geschützte Ressourcen zugreifen möchten. Dies umfasst auch die
651 Offband-Bestätigung, bei der zusätzliche Sicherheitsmechanismen (Verifikation via E-
652 Mail) verwendet werden, um die Identität und Integrität (plattformabhängig) der
653 registrierten Clients zu überprüfen.
- 654 • Der PDP analysiert und interpretiert in der Policy Engine die Sicherheitsrichtlinien, die
655 im Rahmen des Zero Trust-Modells definiert sind. Diese Policies können Kriterien wie
656 Nutzeridentität, Gerätetyp, Standort, Zeitpunkt der Anfrage und andere
657 Kontextinformationen ("Signale") enthalten, die relevant für die Zugriffsentscheidung
658 sind. Basierend auf der Interpretation der Policies trifft die Policy Engine
659 Entscheidungen darüber, ob eine Zugriffsanfrage auf eine bestimmte Ressource
660 genehmigt oder abgelehnt wird. Diese Entscheidungen erfolgen auf Plattformebene,
661 was bedeutet, dass die Policy Engine die Zugriffsanfragen im Kontext der gesamten
662 Plattform oder des Netzwerks bewertet, und nicht isoliert betrachtet. Die
663 Zugriffsentscheidung resultiert dann in der Ausstellung eines Access Token, das für
664 den konkret angefragten Zugriff verwendet wird (siehe Policy Enforcement).
- 665 • Die Policy Engine verwendet dabei die Informationen, die sie von der ZETA Artifact
666 Registry bezieht und die Daten der Zugriffsanfrage vom Authorization Server, um die
667 Zugriffsentscheidung zu treffen. Dazu gehören nicht nur die Policies selbst, sondern
668 auch Echtzeitinformationen über den Zustand von Nutzeridentitäten, Clients und
669 andere Kontextinformationen, die für die Bewertung der Zugriffsanfrage relevant sind.

670 Am PDP werden Betriebsdaten erhoben, verarbeitet und dem Telemetriedaten Service im
671 ZETA Guard zur Verfügung gestellt.

672 4.4 ZETA Client

673 Im Kontext von Zero Trust der TI stellt der "ZETA Client" eine logische Komponente
674 innerhalb einer Clientanwendung (Primärsystem (PS), Frontend des Versicherten (FdV)
675 etc.) dar.

676 Ein ZETA Client im Zero Trust-Modell wird nicht als vertrauenswürdig angesehen, sondern
677 muss - genauso wie alle Komponenten im Netzwerk - kontinuierlich authentifiziert und
678 autorisiert werden. Die Zugriffsentscheidungen werden basierend auf aktuellen
679 Richtlinien, Kontextinformationen, Bedrohungsinformationen und insbesondere in
680 Kenntnis des diesen Client benutzenden Nutzers getroffen.

681 Die Aufgaben des ZETA Clients sind:

- 682 • Erzeugung, sichere Speicherung und Prüfung der kryptographischen App/Geräte
683 Identität
- 684 • Erzeugung der App/Gerät-Attestierung und Ermittlung und Übertragung der
685 Eigenschaften der Laufzeitumgebung (Betriebssystem, Betriebssystem Version, etc.)
- 686 • Implementierung des OAuth Flows
- 687 • Management der Sessions inkl. Verwaltung der Access und Refresh Token.
- 688 • Management der Client-Registrierungen

689 **4.5 Policy-Information und -Administration**

690 Im Zero Trust-Paradigma spielen der Policy Information Point (PIP) und der Policy
691 Administration Point (PAP) wichtige Rollen bei der Verwaltung und Durchsetzung von
692 Sicherheitsrichtlinien bzw. Policies. Zusammen ermöglichen der PIP und der PAP eine
693 zentrale Verwaltung und Bereitstellung von Policies im Zero Trust-Netzwerk.

694 Der PAP stellt Policies bereit und der PIP stellt die Daten für die Policies bereit, sodass sich
695 aus beiden ein Regelwerk ergibt, das die Policy Engine des PDP anwendet, um zu
696 entscheiden, ob eine Kommunikationsanfrage zulässig ist.

697 **4.5.1 Policy Information Point (PIP)**

698 Der PIP ist für die Bereitstellung von Informationen über Sicherheitsrichtlinien
699 zuständig. Er dient als zentraler Informationsdienst, der Policy Engines im Zero Trust-
700 Netzwerk Zugriff auf aktuelle Sicherheitsrichtlinien ermöglicht. Der PIP kann Attribute wie
701 Nutzerrollen, Zugriffsrechte, Clientzustände und andere Kontextinformationen
702 bereitstellen, die von den Policy Engines für die Zugriffsentscheidung benötigt
703 werden. Der PIP kann Daten aus verschiedenen Quellen beziehen, einschließlich einer
704 zentralen Richtliniendatenbank, externen Identitätsanbietern, Sicherheitsinformationen
705 von Clients und anderen Quellen.

706

707 **4.5.2 Policy Administration Point (PAP)**

708 Der PAP ist für die Verwaltung und Konfiguration von Sicherheitsrichtlinien
709 verantwortlich. Er bietet eine Schnittstelle oder eine Konsole, über die Richtlinien in
710 hoheitlicher Verantwortung definiert, geändert und gelöscht werden können. Policy-
711 Administratoren können im PAP Zugriffsregeln, Autorisierungsniveaus,
712 Bedrohungsabwehrmaßnahmen und andere Sicherheitsrichtlinien festlegen. Der PAP
713 ermöglicht es Policy-Administratoren, Richtlinien - basierend auf verschiedenen Kriterien
714 wie Nutzerrollen, Gruppenzugehörigkeit, Standorten und Clientattributen - zu
715 differenzieren. Änderungen an den Sicherheitsrichtlinien, die im PAP vorgenommen
716 werden, werden von den Policy Engines im Zero Trust Netzwerk übernommen. Das
717 Vertrauen in bereitgestellte und angepasste Policies wird über Signaturen für die
718 Sicherstellung von Integrität und Authentizität jeder Policy sichergestellt.

719 **4.5.3 PIP und PAP Ausprägung in ZETA**

720 In ZETA Guard wird als Policy Engine der Open Policy Agent (OPA) verwendet. OPA
721 bezieht Policies (PAP) und Daten (Wertebereiche für die Policies; PIP) zusammengefasst

722 (OPA Bundle) über ein OCI Container Image aus der ZETA Artifact Registry der gematik.
723 Die einzelnen Dateien im OPA Bundle sind mit einer Identität der gematik signiert.

724 Dadurch reduziert sich die Bereitstellung der PIP und PAP Daten zu einem Download in
725 einer OCI Registry. Die Anforderungen an den PIP und PAP Service (siehe [5.9- Policies und](#)
726 [Daten](#)) beziehen sich daher auf den Policies und Daten CI/CD-Prozess zur Entwicklung und
727 Bereitstellung der Policies und Daten.

728 **4.6 Client-Registrierung**

729 Die Client-Registrierung dient dazu, eine konkrete Installation eines Clientsystems
730 eindeutig zu identifizieren und, wenn möglich, mit einem Nutzer zu verknüpfen. Dabei
731 werden sowohl statische Eigenschaften des Clientsystems als auch dynamische
732 Eigenschaften der Client-Instanz übermittelt.

733 Die Clientattribute werden vom Client und von den Plattformen der Endgeräte geliefert.
734 Ihre Erhebung erfolgt im ZETA Client des Endgeräts mittels plattformspezifischer
735 Attestierungs- und Erhebungsmechanismen. Die Attribute sind daher für die jeweilige
736 Plattform und ihr Sicherheitsmodell spezifisch.

737 **4.7 Monitoring**

738 Das Monitoring im Kontext von Zero Trust ist ein entscheidender Aspekt, um die
739 Sicherheit des Netzwerks und der Ressourcen kontinuierlich zu überwachen und
740 potenzielle Bedrohungen oder Anomalien zu identifizieren.

741 **4.7.1 Security Information and Event Management (SIEM)**

742 SIEM-Systeme spielen eine zentrale Rolle im Monitoring im Zero Trust-Paradigma. Sie
743 sammeln Daten aus verschiedenen Quellen wie Protokollen, Ereignissen und Alarmen von
744 Sicherheitskomponenten im Netzwerk. Durch die Analyse dieser Daten in Echtzeit können
745 SIEM-Systeme potenzielle Sicherheitsvorfälle erkennen und Anomalien identifizieren.

746 **4.7.2 Shared Signals**

747 Shared Signals [Shared Signals] sind Hinweise oder Indikatoren für Sicherheitsvorfälle, die
748 von verschiedenen Systemen und Quellen im Netzwerk gemeinsam genutzt
749 werden. Diese Signale können von verschiedenen Sicherheitskomponenten wie Firewalls,
750 Endpunktschutzsystemen, Intrusion Detection Systems (IDS) und anderen generiert
751 werden.

752 SIEM-Systeme aggregieren und korrelieren diese Signale, um umfassende Einblicke in die
753 Sicherheitslage des Netzwerks zu erhalten und potenzielle Bedrohungen zu
754 identifizieren. Durch die Integration von Shared Signals in das Monitoring kann eine
755 umfassende und ganzheitliche Sicherheitsüberwachung gewährleistet werden, die
756 potenzielle Angriffe frühzeitig erkennt und darauf reagiert.

757 Aktuell ist die Shared Signals Unterstützung durch SIEM Systeme und
758 Infrastrukturkomponenten noch nicht hinreichend gegeben. Für die Session-Beendigung
759 wird daher eine anwendungsspezifische, serverseitige Auslösung der Session-Termination
760 über die Plug-In-Schnittstelle des Authorization Servers umgesetzt.

761 **A_29847 -ZETA Guard, Session-Beendigung in Mindestvariante**

762 Die Komponenten des ZETA Guard MÜSSEN eine Session-Beendigung durch den TI 2.0-
763 Dienstanbieter unterstützen.

764 Die Session-Beendigung MUSS den Entzug der zur Session gehörenden Refresh Token
765 bewirken. Bereits ausgestellte Access Token KÖNNEN bis zum Ablauf ihrer Gültigkeit
766 weiterverwendet werden.

767
768 Hinweis: In dieser Umsetzung wirkt die Session-Beendigung unmittelbar auf die Nutzung
769 von Refresh Token. Die maximale Restwirkdauer entspricht der Gültigkeitsdauer bereits
770 ausgestellter Access Token.

771 [**<=**]

772 **4.7.3 Telemetrie, Monitoring und Logging**

773 Betriebliche Daten zum Zwecke des Monitorings (Telemetrie) werden von den ZETA
774 Guard Komponenten erhoben und für die eingesetzten ZETA Guard Komponenten
775 übergreifend mittels dem Telemetriedaten-Service erfasst. Bei der Nachbereitung der
776 Telemetriedaten werden personenbezogene oder -beziehbare Daten anonymisiert, um
777 diese bereinigten Daten dem Anbieter regelhaft zugänglich zu machen. Das bereinigte
778 Monitoring-Log kann von dem Anbieter für sein eigenes betriebliches Monitoring und als
779 Quelle für sein SIEM-System verwendet werden. Das bereinigte Monitoring-Log wird unter
780 Anderem zur Generierung von Traces und Metriken für die gematik benutzt.

781 **4.8 Zusammenspiel mit Identity Provider**

782 Das Stichwort "Step-up-Authentifizierung" bezieht sich auf eine Sicherheitsmaßnahme,
783 bei der der Nutzer zusätzliche Authentifizierungsschritte durchlaufen muss, um auf
784 sensible Ressourcen zuzugreifen. Diese Maßnahme wird wie folgt realisiert:

- 785 1. **Nutzer stellt über den ZETA Client eine Anfrage:**Der Nutzer versucht auf eine
786 Ressource zuzugreifen, für die das aktuelle Access Token nicht ausreicht.
- 787 2. **PEP fängt Anfrage ab:**Der PEP empfängt die Anfrage.
- 788 3. **PEP validiert Access Token:**Der PEP prüft:
 - 789 • Ist das Access Token gültig (Signatur, Ablaufdatum etc.)?
 - 790 • Hat das Access Token den **erforderlichen Scope und die passende Resource-**
791 **Identifizierer (aud)** für die angeforderte Ressource?
- 792 4. **Zugriff verweigert:**Wenn der erforderliche Scope oder der passende Resource-
793 Identifizierer **nicht** im Access Token enthalten ist, verweigert der PEP den Zugriff.
- 794 5. **PEP antwortet mit Step-up-Anforderung:**Wenn die angefragte Ressource auf dem
795 Resource Server existiert, informiert der PEP den Nutzer, dass für den Zugriff ein
796 Access Token mit einem bestimmten Scope und einem bestimmten Resource-
797 Identifizierer benötigt wird.
- 798 6. **Nutzer initiiert die Step-up-Authentifizierung:**Der ZETA Client kommuniziert mit
799 dem Authorization Server, um die Authentifizierung für den benötigten Scope und den
800 benötigten Resource-Identifizierer zu beginnen.
- 801 7. **Authorization Server authentifiziert und gewährt neues Access Token:**Der
802 Authorization Server:
 - 803 • Steuert die Durchführung der Step-up-Authentifizierung.
 - 804 • Erstellt ein **neues Access Token** mit passendem **Scope** und passendem
805 **Resource-Identifizierer** (aud).

- 806
- Gibt das neue Access Token an den ZETA Client zurück.
- 807
- 808
- 809
- 810
- 811
- 812
- 813
- 814
- 815
- 816
- 817
8. **Nutzer stellt eine erneute Anfrage:** Der Nutzer sendet die Anfrage mit dem **neuen** Access Token an den PEP.
 9. **PEP prüft das neue Token:** Der PEP prüft wieder:
 - Ist das **neue** Access Token gültig?
 - Hat das **neue** Access Token den **Scope** und die **den Resource-Identifer (aud)** für die angeforderte Ressource?
 10. **Zugriff gewährt:** Wenn der erforderliche Scope und der passende Resource-Identifer im **neuen** Access Token enthalten ist, gewährt der PEP den Zugriff auf die Ressource.
- Die Step-up-Authentifizierung stellt sicher, dass zusätzliche Sicherheitsmaßnahmen - wenn erforderlich - ergriffen werden, um die Integrität und Vertraulichkeit der geschützten Daten zu gewährleisten.

818 **4.9 TI 2.0-Dienst-Backend**

819 Das TI 2.0-Dienst-Backend (im folgenden Resource Server genannt) stellt das Ziel jedes
820 Zugriffswunschs eines Nutzers über sein Clientsystem dar. Es stellt fachliche
821 Schnittstellen zur Nutzung durch Clientsysteme dar, die über die Mechanismen des Zero
822 Trust abgesichert werden.

823

5 Spezifikation

824 Dieses Kapitel beschreibt die technische Umsetzung der beschriebenen Konzepte an die
825 oben eingeführten Komponenten des Zero Trust (ZETA Guard Komponenten) als
826 generische Produkt- und Anbietertypen. Diese Anforderungen finden Anwendung in den
827 Steckbriefen von konkreten Produkt- und Anbietertypen der jeweiligen Fachanwendung
828 und erhalten erst in der dortigen Zuordnung ein konkretes Prüfverfahren.

829 Die Festlegungen zu Schemas, OpenAPI Schnittstellen und Abbildungen sind in
830 [gemAPI_ZETA] enthalten.

831 5.1 Anforderungen an die Sicherheit und den Datenschutz

832 5.1.1 Übergreifende Anforderungen für Datenschutz und 833 Sicherheit

834 **A_25400 -ZETA Guard - Umsetzung Sicherer Softwareentwicklungsprozess**

835 Der Hersteller des ZETA Guards MUSS einen sicheren Softwareentwicklungsprozess
836 umsetzen (siehe [gemSpec_DS_Hersteller#Kapitel 2.2 Sicherer
837 Softwareentwicklungsprozess]).[<=]

838 **A_25401 -ZETA Guard - Darstellung der Voraussetzungen für sicheren Betrieb 839 des Produkts im Produkthandbuch**

840 Der Hersteller des ZETA Guards MUSS für sein Produkt im dazugehörigen
841 Produkthandbuch leicht ersichtlich darstellen, welche Voraussetzungen vom Anbieter und
842 der Betriebsumgebung erfüllt werden müssen, damit ein sicherer Betrieb des Produktes
843 gewährleistet werden kann.[<=]

844 **A_28459 -ZETA Guard - Informationsobjekte im Produkthandbuch**

845 Der Hersteller des ZETA Guards MUSS alle vom ZETA Guard verarbeiteten
846 Informationsobjekte in seinem Produkthandbuch vollständig auflisten.[<=]

847 **A_28463 -ZETA Guard - Informationsobjekte des ZETA Clients im 848 Produkthandbuch**

849 Der Hersteller des ZETA Guards MUSS alle vom ZETA Client verarbeiteten
850 Informationsobjekte im Produkthandbuch des ZETA Guard vollständig auflisten.[<=]

851 **A_28460 -ZETA Guard - Datenschutzrechtliche Bewertung durch den 852 Dienstanbieter**

853 Der Anbieter eines TI 2.0 Dienstes MUSS sich über die Informationsobjekte, die im ZETA
854 Guard verarbeitet werden, aus dem Produkthandbuch informieren und als
855 Datenschutzverantwortlicher für sein Dienst bewerten.[<=]

856 **A_28461-01 -Informationspflicht des Client-Herstellers gegenüber Nutzern**

857 Der Hersteller eines TI 2.0-Clients MUSS

- 858 • sich über die Informationsobjekte aus dem Produkthandbuch des ZETA Guard
859 informieren
- 860 • und seine Nutzer über Verarbeitung der Informationsobjekte datenschutzkonform
861 informieren.

862 [**<=**]

863 Hinweis: Es gibt ein Kapitel in dem ZETA Guard Handbuch, das die Integration von
864 Clientsystemen mit Zeta Guard beschreibt. Das Kapitel beschreibt die Client-
865 Informationsobjekte, die der Client verarbeiten muss.

866 **A_25402 -ZETA Guard - Schutz der transportierten Daten**

867 ZETA Guard MUSS sicherstellen, dass die Vertraulichkeit und Integrität der transportierten
868 Daten gewährleistet ist.

869 Alle Endpunkte des ZETA Guards MÜSSEN TLS gesichert sein. [≤]

870 **A_28781 -ZETA Guard - Schutz der internen Kommunikation**

871 ZETA Guard MUSS sicherstellen, dass die interne Kommunikation zwischen ZETA Guard
872 Komponenten durch TLS abgesichert ist. [≤]

873 *Hinweis: Es wird empfohlen ein Service Mesh (z. B. Cilium, Istio oder linkerd) einzusetzen.*

874 **A_26517-01 -ZETA Guard - Unterstützung von mTLS**

875 ZETA Guard MUSS die Konfiguration und Nutzung von mTLS für die interne
876 Kommunikation und für die Kommunikation zum Resource Server unterstützen. [≤]

877 **A_25403 -ZETA Guard - Schutzmaßnahmen gegen die OWASP Top 10 Risiken**

878 ZETA Guard MUSS geeignete technische Maßnahmen zum Schutz vor den Risiken in der
879 aktuellen Version der [OWASP-Top-10-Risiken] umsetzen. [≤]

880 *Hinweis: Die Anforderungen gelten für die gesamte ZETA-Guard-Lösung. Entscheidet sich
881 ein Anbieter dafür, eine eigene Komponente anstelle einer von ZETA bereitgestellten
882 Komponente (z. B. Ingress) zu verwenden, legt der Sicherheitsgutachter (Anbieter) fest,
883 welche Risiken aus der Top-10-Liste für diese Komponente relevant sind, da deren
884 konkrete Umsetzung anbieterabhängig ist.*

885 **A_28961 -Maßnahmen gegen die OWASP Top 10 Kubernetes Risiken**

886 Der Anbieter eines TI 2.0-Dienstes MUSS geeignete Maßnahmen zum Schutz vor den
887 Risiken in der aktuellen Version der [OWASP-Top-Ten-Kubernetes] umsetzen.

888 [≤]

889 **A_25404 -ZETA Guard - Angriffe erkennen**

890 ZETA Guard MUSS Maßnahmen zur Erkennung, Kategorisierung und Protokollierung bzw.
891 Meldung von Angriffen umsetzen. Die Kategorisierung von Angriffen MUSS nach "CAPEC:
892 OWASP Related Patterns"[CAPEC OWASP] erfolgen. [≤]

893 **A_25405 -ZETA Guard - Angriffen entgegenwirken**

894 ZETA Guard MUSS Maßnahmen zur Schadensreduzierung und -verhinderung von
895 Angriffen umsetzen. [≤]

896 **A_25406 -ZETA Guard - Eingabe Validierung von Operationen**

897 ZETA Guard MUSS sicherstellen, dass alle Daten und Parameter, die über eine API
898 kommuniziert werden, sicherheitstechnisch validiert werden. [≤]

899 *Hinweis: Eine Eingabe-Validierung von Resource Server APIs erfolgt im Resource
900 Server und nicht in den Zero Trust-Komponenten.*

901 **A_25407 -ZETA Guard - Sicherheitstechnische Validierung von Policy und 902 Konfigurationen**

903 ZETA Guard MUSS sicherstellen, dass alle Daten und Parameter, die von einer
904 Konfigurationsdatei oder Policy gelesen werden, sicherheitstechnisch validiert werden.

905 [≤]

906 **A_25408-01 -ZETA Guard - Verbot Profilbildung**

907 Der Anbieter eines TI 2.0-Dienstes DARF Profile - außer zum Zweck des Security
908 Monitorings - NICHT bilden.

909 [≤]

910 **A_25409 -ZETA Guard - Privacy by Design**

911 ZETA Guard MUSS sicherstellen, dass bei Konfigurationsmöglichkeiten die
912 datenschutzfreundlichere Option vorausgewählt ist. [<=]

913 **A_25410 -ZETA Guard - Verbot von Werbe- und Usability-Tracking**

914 ZETA Guard DARF im Produktivbetrieb ein Werbe- und Usability-Tracking NICHT
915 verwenden. [<=]

916 **A_25411 -ZETA Guard - Verbot vom dynamischen Inhalt**

917 ZETA Guard DARF dynamischen Inhalt von Drittanbietern NICHT herunterladen und
918 verwenden. [<=]

919 **A_25412 -ZETA Guard - Zusätzliche Verschlüsselung bei der Persistierung**

920 Unabhängig davon, ob die Daten schon verschlüsselt vorliegen, MUSS ZETA Guard seine
921 Daten bei der Persistierung verschlüsseln. [<=]

922 **A_25413-01 -ZETA Guard - Ordnungsgemäße IT-Administration**

923 Der Anbieter eines TI 2.0-Dienstes MUSS die Maßnahmen für erhöhten Schutzbedarf aus
924 dem BSI-Bausteins „OPS.1.1.2 Ordnungsgemäße IT-Administration“ [BSI-Grundschutz]
925 während des gesamten Betriebs des ZETA Guards umsetzen. [<=]

926 **A_26479-02 -ZETA Guard - Ordnungsgemäße Änderung von Konfigurationen**

927 Der Anbieter eines TI 2.0-Dienstes MUSS durch technische und organisatorische Mittel
928 sicherstellen, dass eine Änderung der Konfiguration des ZETA Guards nur unter 4-Augen
929 erfolgen kann. [<=]

930

931 **A_25718 -ZETA Guard - Bereitstellung Security-KPIs**

932 ZETA Guard MUSS sicherstellen, dass die Security-KPIs in A_25484-* automatisch
933 bereitgestellt werden.

934 [<=]

935 *Hinweis: Die Anforderung ist besonders wichtig, falls die Zero Trust-Komponente in einer*
936 *VAU betrieben wird.*

937 **A_28406-01 -ZETA Guard - Verification der ZETA Guard Images**

938 Der Hersteller des TI 2.0-Dienstes MUSS vor der Aktualisierung von ZETA Guard die
939 Authentizität und Aktualität der zu aktualisierenden Komponenten auf der Grundlage
940 einer von der gematik vorgegebenen Signaturprüfung verifizieren und bei Fehlschlagen
941 der Verifikation die Aktualisierung abbrechen und gematik umgehend informieren. [<=]

942 *Hinweis: Die Images für ZETA Guard werden mit einem Zertifikat der Komponenten PKI*
943 *signiert. Das Zertifikat und die zugehörige Kette müssen für die Signaturprüfung*
944 *verwendet werden.*

945 **A_28407 -ZETA Guard - Nachweisbarkeit verwendete Version des ZETA-Images**

946 Der Hersteller eines TI 2.0-Dienstes MUSS ein SBOM für sein Produkt erstellen, aus dem
947 eindeutig hervorgeht, welches ursprüngliche ZETA Guard-Image verwendet wurde. [<=]

948 *Hinweis: Das SBOM für das gesamte Produkt (inkl. ZETA Guard) kann durch den Hersteller*
949 *über mehrere Dateien abgebildet werden. Für ZETA Guard wird immer ein eigenständiges*
950 *SBOM geliefert.*

951 ZETA Guard implementiert für Stufe 1 eine vereinfachte Prüfung des Clients auf gesperrte
952 IP-Adressen und Impossible Travel. Diese Vereinfachung ist möglich, da in Stufe 1 nur
953 eine Role (LEI) mit einem stationären Endgerät vorgesehen ist.

954 **A_28827 -ZETA-Guard - IP-Adresse Binding des Access-Token**

955 ZETA Guard MUSS für Stufe 1 die anfragende IP-Adresse des Clients im Access-Token bei
956 der Ausstellung des Access-Token hinterlegen. [<=]

957 **A_28803 -ZETA Guard - Prüfung von Impossible Travel**

958 ZETA Guard MUSS für Stufe 1 die im Access-Token hinterlegte IP-Adresse bei jedem
959 Client-Request gegen die tatsächliche IP-Adresse des anfragenden Clients validieren. Bei

960 einem negativen Ergebnis MUSS ZETA Guard das Access-Token und Refresh-Token des
961 Clients sperren und die aktuelle fachliche Operation abbrechen. [≤]

962 **A_29872 -ZETA Guard - IP-Binding bei IPv6**

963 Bei der Validierung gemäß A_28827 und A_28803 MUSS ZETA Guard IPv6-Adressierung
964 berücksichtigen. Der Anbieter MUSS ein Verfahren festlegen und dokumentieren, das
965 Fehlentscheidungen durch wechselnde IPv6-Adresspräfixe minimiert. [≤]

966 **A_28828 -ZETA Guard - Weiterleitung Client-IP**

967 Der Anbieter des TI 2.0-Dienstes MUSS alle in seiner Hoheit betriebenen
968 Netzwerkkomponenten (wie z. B. Load Balancer, Reverse Proxy, DDoS-Schutz, WAF, CDN
969 oder API Gateway) zwischen Client und ZETA Guard so konfigurieren, dass die originale
970 IP-Adresse des Clients über die gesamte Verarbeitungskette hinweg mittels
971 standardisierter HTTP-Header (vorzugsweise Forwarded gemäß RFC 7239, alternativ X-
972 Forwarded-For oder X-Real-IP) an ZETA Guard weitergeleitet wird. [≤]

973 **5.1.2 Sicherheits- und Datenschutzerfordernungen an Logging und** 974 **Monitoring**

975 *Hinweis: Die Anforderungen dieses Abschnitts könnten sich noch ändern, falls sich bei der*
976 *Umsetzung des Zero Trust herausstellt, dass weitere Protokollierungen auf Seiten des*
977 *Anbieters notwendig werden.*

978

979 **A_25744 -ZETA Guard - Datenschutzkonformes Logging und Monitoring**

980 ZETA Guard MUSS die für den Betrieb des Zero Trust erforderlichen Logging- und
981 Monitoring-Informationen in solcher Art und Weise erheben und verarbeiten, dass mit
982 technischen Mitteln ausgeschlossen ist, dass dem Anbieter eines TI 2.0-Dienstes
983 vertrauliche oder zur Profilbildung geeignete Daten zur Kenntnis gelangen. [≤]

984 *Hinweis: Der Telemetriedaten Service im ZETA Guard muss die vom PEP und PDP*
985 *gesammelten Telemetriedaten so verändert an das Monitoring System des Anbieters*
986 *weitergeben, dass eine Profilbildung nicht mehr möglich ist.*

987 **A_25745 -ZETA Guard - Keine medizinischen Informationen in Logging und** 988 **Monitoring**

989 ZETA Guard MUSS sicherstellen, dass in für den Betrieb erstellten Protokollen keine
990 personenbezogenen medizinischen Informationen enthalten sind (u. a. medizinische
991 Daten von Versicherten oder Informationen, aus denen sich ableiten lässt, bei welchen
992 Leistungserbringerinstitutionen ein Versicherter in Behandlung ist). [≤]

993 **A_25746 -ZETA Guard - Keine sicherheitsrelevanten Daten in Logging und** 994 **Monitoring**

995 ZETA Guard MUSS sicherstellen, dass in für den Betrieb erstellten Protokollen keine
996 sicherheitsrelevanten Daten enthalten sind. [≤]

997 *Hinweis: Sicherheitsrelevante Daten sind zum Beispiel, Kryptoschlüssel, Access/Refresh*
998 *Token usw.*

999 **A_25747-01 -ZETA Guard - Löschfristen Protokolle**

1000 Der Anbieter eines TI2.0 Dienstes MUSS sicherstellen, dass die zum Zwecke der
1001 Fehleranalyse erhobenen Protokolle des ZETA Guards nach Behebung des Fehlers
1002 unverzüglich gelöscht werden.

1003 [≤]

1004 5.1.3 Sicherheits- und Datenschutz-Anforderungen an das 1005 Security Monitoring

1006 Um eine lückenlose Nachvollziehbarkeit komplexer Systeminteraktionen zu
1007 gewährleisten, ist die nahtlose Verknüpfung aller anfallenden Telemetriedaten
1008 entscheidend. Jede einzelne Anfrage löst eine Vielzahl von Prozessen, Logs und Metriken
1009 aus, die erst durch eine konsistente Korrelation ihren vollen diagnostischen Wert
1010 entfalten. Ziel ist es, die gesamte Verarbeitungskette eines Requests über alle ZETA-
1011 Komponenten hinweg als zusammenhängendes Ereignis sichtbar zu machen, sodass die
1012 kausalen Zusammenhänge zwischen den verschiedenen Datenpunkten jederzeit
1013 transparent und ohne Informationsverlust nachverfolgbar bleiben.

1014 **A_25484-03 -Security Monitoring - Security KPIs**

1015 ZETA Guard MUSSeinmal täglich mittels dem Telemetriedaten-Service die folgende
1016 Sicherheits-KPIs automatisiert über die von der gematik angebotene Schnittstelle an das
1017 TI SIEM-System als OTel-Metric übermitteln:

- 1018 • Anzahl versuchter Zugriffe von nicht registrierten Clients (hier muss die KPIs zwischen
1019 Resource Server APIs und Client-Registrierung APIs unterscheiden)
- 1020 • Anzahl von Zugriffen von Botnetzen
- 1021 • Anzahl von Zugriffen aus jedem Land gezählt plus weitere Zugriffe, die separat in
1022 Versicherte und LE ausgewiesen werden
- 1023 • Anzahl fehlerhafter Clientfreischaltungen plus weitere breakdown in Versicherte und
1024 LE
- 1025 • Anzahl von Impossible travel Zugriffen (inkl. Land- und Ortsdaten) plus weitere
1026 breakdown in Versicherte und LE
- 1027 • Anzahl von Zugriffen über TOR Netzwerke plus weitere breakdown in Versicherte und
1028 LE
- 1029 • Anzahl von Zugriffen über VPNs plus weitere breakdown in Versicherte und LE
- 1030 • Anzahl erkannte Angriffe in Kategorie (siehe A_25404-*) plus weitere breakdown in
1031 Versicherte und LE
- 1032 • Anzahl fehlerhafte Authorization Codes vom IDP.

1033 **[<=]**

1034 *Hinweis: Security KPIs beinhalten anonyme Daten und sind nicht auf individuelle Nutzer*
1035 *zurückzuführen.*

1036 *Hinweis: Impossible Travel ist eine Methode zur Anomalieerkennung in der*
1037 *Cybersicherheit, die potenzielle Kompromittierungen identifiziert, indem sie*
1038 *Nutzeranmeldeaktivitäten analysiert und mit geografischen Standorten korreliert. Dabei*
1039 *werden Fälle markiert, in denen auf das Nutzerkonto innerhalb eines verdächtig kurzen*
1040 *Zeitraums aus zwei verschiedenen Ländern zugegriffen wird.*

1041 *Hinweis: Falls der Anbieter aufgrund einer Netzwerk-Sicherheitsrichtlinie Anfragen mit*
1042 *bestimmten Kommunikationsmerkmalen (z. B. Zugriffe aus einem TOR-Netzwerk oder*
1043 *Botnetz) am Netzwerk-Perimeter blockiert, hat diese Richtlinie Vorrang. Es wird nicht*
1044 *erwartet, dass entsprechende Requests dennoch an den ZETA Guard weitergeleitet, da*
1045 *diese Anfragen bereits durch den Schutzmechanismus am Netzwerk-Perimeter*
1046 *abgefangen werden.*

1047 *Hinweis: Forbidden Countries beinhaltet eine Liste von IP-Ranges der Länder-ASN.*

1048

1049 Hier ein Beispiel Metric für die Anzahl von Zugriffen von Botnetzen:

```

1050 {
1051   "resourceMetrics": [
1052     {
1053       "resource": {
1054         "attributes": [
1055           {
1056             "key": "service.name",
1057             "value": {
1058               "stringValue": "zeta-guard"
1059             }
1060           },
1061           {
1062             "key": "service.version",
1063             "value": {
1064               "stringValue": "1.0.0"
1065             }
1066           }
1067         ]
1068       },
1069       "scopeMetrics": [
1070         {
1071           "scope": {
1072             "name": "zeta-guard-kpis",
1073             "version": "1.0.0"
1074           },
1075           "metrics": [
1076             {
1077               "name": "zeta_guard_kpi_botnet_accesses_24h",
1078               "description": "Number of botnet accesses detected by ZETA
1079 Guard in the last 24 hours",
1080               "unit": "1",
1081               "sum": {
1082                 "isMonotonic": true,
1083                 "aggregationTemporality": "AGGREGATION_TEMPORALITY_DELTA",
1084                 "dataPoints": [
1085                   {
1086                     "attributes": [
1087                       {
1088                         "key": "date",
1089                         "value": {
1090                           "stringValue": "2026-03-09"
1091                         }
1092                       }
1093                     ],
1094                     "startTimeUnixNano": "1762128000000000000",
1095                     "timeUnixNano": "1762214400000000000",
1096                     "asInt": "123"
1097                   }
1098                 ]
1099             }
1100           ]
1101         }
1102       ]
1103     }
1104   ]
1105 }
1106 }

```

1107

1108 **A_28783 -ZETA Guard - Telemetriedaten für das Security Monitoring**

1109 Der PEP und PDP MÜSSEN über den Telemetriedaten-Service die folgende Daten zu jeder
 1110 Anfrage automatisiert über die von der gematik bereitgestellte Schnittstelle an das TI-
 1111 SIEM-System als Teil eines OTel-Traces übermitteln.

1113 **Tabelle 1: Security Telemetriedaten PEP und PDP**

Daten	Open Telemetry Attribute	Begründung und Zweck
IP-Adresse	https://opentelemetry.io/docs/specs/semconv/registry/attributes/client/#client-address	Identifikationsmerkmal der anfragenden Instanz
http_user_agent	https://opentelemetry.io/docs/specs/semconv/registry/attributes/user-agent/#user-agent-original	Erkennung von Angriffen und Anomalien.
http_method	https://opentelemetry.io/docs/specs/semconv/registry/attributes/http/#http-request-method-original	Erkennung von unautorisierten Aktionen oder abnormalen Anfragen. Ein unerwarteter HTTP-Methoden-Typ auf einer bestimmten Route kann auf einen Angriffsversuch hindeuten.
http_route	https://opentelemetry.io/docs/specs/semconv/registry/attributes/http/#http-route	Überwachung von Zugriffen auf spezifische Ressourcen und Erkennung von unautorisierten Zugriffen oder Versuchen, nicht existierende Routen zu erreichen (z.B. für Brute-Force-Angriffe oder Enumeration).
http_fqdn	https://opentelemetry.io/docs/specs/semconv/registry/attributes/server/#server-address	Identifikation des Zielservers bei Multi-Host-

		Umgebungen und Erkennung von Anfragen an nicht autorisierte oder verdächtige Domänen.
http_status	https://opentelemetry.io/docs/specs/semconv/registry/attributes/http/#http-response-status-code	Erkennung von Fehlern, Ausfällen oder potenziellen Angriffsversuchen (z.B. viele 401/403-Fehler bei Brute-Force-Angriffen, viele 5xx-Fehler bei Denial-of-Service-Angriffen).
client_id	https://opentelemetry.io/docs/specs/semconv/registry/attributes/app/#app-installation-id	Identifikationsmerkmal des anfragenden Clients

1114 [\leq]

A_28793 -Telemetriedaten für Policy Entscheidungen

Der PDP MUSS über den Telemetriedaten-Service die folgende Daten zu jeder Policy-Entscheidung automatisiert über die von der gematik bereitgestellte Schnittstelle an das TI-SIEM-System als Teil eines OTel-Traces übermitteln.

Tabelle 2: Telemetriedaten Policy Entscheidungen

Daten	Open Telemetry Attribute	Begründung und Zweck
produkt_id	nicht Verfügbar	Möglichkeit abweichende und verdächtige Produktversionen zu erkennen
produkt_version	https://opentelemetry.io/docs/specs/semconv/registry/attributes/app/#app-build-id	Möglichkeit abweichende und verdächtige Produktversionen zu erkennen
os	https://opentelemetry.io/docs/specs/semconv/registry/attributes/os/#os-name	Möglichkeit abweichende und verdächtige Betriebssystemv

		ersionen zu erkennen
os_version	https://opentelemetry.io/docs/specs/semconv/registry/attributes/os/#os-version	Möglichkeit abweichende und verdächtige Betriebssystemversionen zu erkennen
device_integrity	nicht Verfügbar	Indikator ob das Gerät gerootet/kompromittiert ist
Optional: Gerätmodel für Android und iOS Clients		
device_model		Möglichkeit abweichende und verdächtige Geräte zu erkennen.
Optional: Falls die Policyentscheidung fehlgeschlagen hat		
Ergebnis der Auswertung der Policy als getrennte OTEL Log	nicht Verfügbar	Verständnis über die Ablehnungsgrund von Policies plus Angriffe oder Manipulationen zu erkennen.
Optional: Falls ein Simulationspolicy		

existiert		
Ergebnis der Auswertung der Simluationspolicy als getrennte OTEL Log	nicht Verfügbar	Möglichkeit häufige Violators von Simulation Policy zu erkennen und verstehen.

1120 [**<=**]

1121 **A_28867 -Ergebnis der Policy Entscheidung als OTel-Log**

1122 Der PDP MUSS über den Telemetriedaten-Service das Ergebnis einer Policy-Entscheidung
 1123 automatisiert über die von der gematik bereitgestellte Schnittstelle an das TI-SIEM-
 1124 System als ein OTel-Log übermitteln. [**<=**]

1125 Hier ein Beispiel eines OTel-Logs für die Policyentscheidung:

```

1126 {
1127   "resourceLogs": [
1128     {
1129       "resource": {
1130         "attributes": [
1131           {
1132             "key": "service.name",
1133             "value": {
1134               "stringValue": "openpolicyagent"
1135             }
1136           },
1137           {
1138             "key": "service.version",
1139             "value": {
1140               "stringValue": "1.14.0"
1141             }
1142           },
1143           {
1144             "key": "opa.instance.id",
1145             "value": {
1146               "stringValue": "4794056b-0001-48e2-8acf-3a6fb0287384"
1147             }
1148           },
1149           {
1150             "key": "opa.bundle.authz",
1151             "value": {
1152               "stringValue": ""
1153             }
1154           }
1155         ]
1156       },
1157       "scopeLogs": [
1158         {
1159           "scope": {
1160             "name": "openpolicyagent.org/decision_logs",
1161             "version": "1.14.0"
1162           },
1163           "logRecords": [
1164             {
    
```

```

1165     "timeUnixNano": "1741170585787524800",
1166     "observedTimeUnixNano": "1741170585787524800",
1167     "severityNumber": 9,
1168     "severityText": "INFO",
1169     "body": {
1170       "stringValue": "Decision Log"
1171     },
1172     "attributes": [
1173       {
1174         "key": "opa.decision_id",
1175         "value": {
1176           "stringValue": "d88b7796-8e90-441e-a572-a2f6ea179c18"
1177         }
1178       },
1179       {
1180         "key": "opa.path",
1181         "value": {
1182           "stringValue": "policies/zeta/authz/decision"
1183         }
1184       },
1185       {
1186         "key": "opa.result.allow",
1187         "value": {
1188           "boolValue": false
1189         }
1190       },
1191       {
1192         "key": "opa.result.reasons",
1193         "value": {
1194           "arrayValue": {
1195             "values": [
1196               {
1197                 "stringValue": "User profession is not allowed"
1198               },
1199               {
1200                 "stringValue": "TelematikID is blocked"
1201               }
1202             ]
1203           }
1204         }
1205       }
1206     ],
1207     "traceId": "",
1208     "spanId": "",
1209     "flags": 0
1210   }
1211 ]
1212 }
1213 ]
1214 }
1215 ]
1216 }

```

1217

1218 *Hinweis: Im "opa.path" Attribut soll der Path der Simulation-Policy oder der Policy*
 1219 *eingetragen werden.*

1220 **A_28795-01 -Telemetriedaten Angriffserkennung**

1221 ZETA Guard MUSS über den Telemetriedaten-Service die folgende Daten zu jedem
 1222 erkannten Angriff (A_25404-*) automatisiert über die von der gematik bereitgestellte
 1223 Schnittstelle an das TI-SIEM-System als Teil eines OTel-Traces übermitteln

1224 **Tabelle 3: Telemetriedaten Angriffserkennung**

Daten	Open Telemetry Attribute	Datenschutzbeurteilung
Angriffstyp (OWASP CAPEC Klassifizierung) [CAPEC OWASP]	nicht Verfügbar	Verständnis über die Angriffsklassifizierung
Angriffsdaten	nicht Verfügbar	Verständnis über den aktuellen Angriff

1225 [**<=**]

1226 **A_28796 -ZETA Guard - Korrelation der Telemetrie des Security Monitorings**
 1227 ZETA Guard MUSS sicherstellen, dass alle während der Anfrageverarbeitung anfallenden
 1228 Telemetriedaten des Security Monitorings (A_28783-, A_28793- und A_28795-*) mittels
 1229 einer durchgängigen Korrelations-ID logisch miteinander verknüpft werden. [**<=**]

1230 Hinweis: Dies gewährleistet eine lückenlose und verlustfreie Nachverfolgung der
 1231 gesamten Verarbeitungskette eines Requests über alle Systemkomponenten hinweg,
 1232 analog zur Funktionsweise eines Spans in OpenTelemetry.

1233 **A_25485-02 -Security Monitoring - Sicherheitsmeldung bei Aktualisierung von**
 1234 **PIP-Daten oder PAP-Policies**

1235 ZETA Guard MUSS bei der erfolgreichen Aktualisierung der PIP-Daten und PAP-Policies
 1236 eine Sicherheitsmeldung automatisiert über die von der gematik angebotene Schnittstelle
 1237 an das TI SIEM-System übermitteln. [**<=**]

1238 **A_25606-02 -Security Monitoring - Fehlermeldung bei Aktualisierung von PIP-**
 1239 **Daten oder PAP-Policies**

1240 ZETA Guard MUSS bei folgenden Fehlern während der Aktualisierung der PIP-Daten und
 1241 PAP-Policies eine Fehlermeldung automatisiert über die von der gematik angebotene
 1242 Schnittstelle an das TI SIEM-System übermitteln:

- 1243 • Policy Download Fehler
- 1244 • Fehler bei der Integritätsprüfung der Policy-Signatur

1245 [**<=**]

1246 **A_28960-01 -Security Monitoring - Keine Weitergabe sicherheitsrelevanter**
 1247 **Telemetriedaten**

1248 ZETA Guard DARF sicherheitsrelevante Telemetriedaten (A_28783-*, A_28793-*, A_28867-
 1249 *, A_28795-*) NICHT an den Betreiber weitergeben. [**<=**]

1250 Hinweis: Betriebliche Telemetriedaten dürfen gemäß A_27260-* an den Betreiber
 1251 übermittelt werden.

1252 **A_28964 -Security Monitoring - Löschung sicherheitsrelevanter Telemetriedaten**

1253 Nach erfolgreicher Übermittlung der sicherheitsrelevanten Telemetriedaten an die
 1254 gematik MUSS ZETA Guard alle relevanten Logs unmittelbar löschen.

1255 [**<=**]

1256 **5.1.4 Sicherheits- und Datenschutz-Anforderungen an die** 1257 **Protokollierung von Administrationsaktivitäten**

1258 Administratoren haben weitreichende Zugriffsrechte in Kubernetes, die es ihnen
1259 ermöglichen, die Sicherheitskonfigurationen von TI 2.0-Diensten direkt zu ändern. Ohne
1260 Kontrolle und Protokollierung dieser Aktivitäten besteht das Risiko, dass unautorisierte
1261 oder fehlerhafte Änderungen vorgenommen werden, die die Sicherheit der TI 2.0-
1262 Diensten gefährden.

1263 **A_28749 -ZETA Guard - Tamper-Proof Protokollierung von** 1264 **Administrationsaktivitäten**

1265 Der Anbieter des TI 2.0-Dienstes MUSS eine revisionssichere Protokollierung (Tamper-
1266 Proof) aller TI 2.0-Dienst relevanten administrativen Vorgänge auf dem ZETA Kubernetes
1267 Cluster führen. [<=]

1268 Hinweis: Diese Anforderung umfasst alle Administrationsaktivitäten z.B. Anwendung,
1269 Plattform und Cluster Administration.

1270 **A_28750-01 -ZETA Guard - Löschfristen Auditeinträge des Admin-Audit-Logs**

1271 Der Anbieter des TI 2.0-Dienstes MUSS sicherstellen, dass die Löschung eines Admin-
1272 Auditeintrags den gesetzlichen Vorgaben entspricht und frühestens nach 6 Monaten
1273 erfolgt. [<=]

1274 **A_28751 -ZETA Guard - Kontrolle des Admin-Audit-Logs**

1275 Der Anbieter des TI 2.0-Dienstes MUSS das Admin-Audit-Log mindestens alle 3 Monate im
1276 Vieraugenprinzip kontrollieren. Diese Rollen DÜRFEN NICHT an der Administration des
1277 ZETA Clusters teilnehmen. Bei der Kontrolle ist insbesondere auf ungewöhnliche, nicht
1278 nachvollziehbare oder maliziöse Administratoraktivitäten zu achten. [<=]

1279 Hinweis: Die in A_28751-* und A_28749-* definierten Anforderungen dürfen durch
1280 Automatisierung bzw. unterstützendes Tooling anstelle manueller Kontrollen umgesetzt
1281 werden, sofern das Vier-Augen-Prinzip weiterhin gewährleistet ist.

1282 **A_28752 -ZETA Guard - Sicherheitsmeldung bei Aktualisierung der** 1283 **Konfiguration des ZETA Guard**

1284 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Anbieter des TI 2.0-Dienstes
1285 sicherstellen, dass

- 1286 • jede Konfigurationsänderung am ZETA Guard eine automatisierte Sicherheitsmeldung
1287 an das SIEM-System des Anbieters auslöst und
- 1288 • auf jede dieser Meldungen eine Reaktion gemäß dem vordefinierten Incident-
1289 Management-Prozess erfolgt

1290 [<=]

1291 **A_28753 -ZETA Guard - Incident-Management-Prozess bei** 1292 **Konfigurationsänderungen von ZETA Guard**

1293 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Anbieter des TI 2.0-Dienstes
1294 einen Incident-Management-Prozess etablieren, der bei jeder automatisierten SIEM-
1295 Meldung über eine Konfigurationsänderung am ZETA Cluster ausgelöst wird. Dieser
1296 Prozess MUSS eine unverzügliche Analyse und Bewertung der Änderung sicherstellen, um
1297 festzustellen, ob sie autorisiert war und ein Sicherheitsrisiko darstellt. Bei unautorisierten
1298 oder schädlichen Änderungen müssen definierte Korrekturmaßnahmen eingeleitet
1299 werden. [<=]

1300 **5.1.5 Sicherheits- und Datenschutz-Anforderungen an die**
1301 **Verarbeitung von Daten mit dem Schutzbedarf "sehr hoch"**

1302 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1303 Anbieter keine Kenntnis über die in dem ZETA Guard verarbeiteten Daten erlangen darf,
1304 gibt es Sonderanforderungen, um die Daten während der Verarbeitung zu schützen.

1305 ZETA Guard muss in einer VAU laufen können. Daher muss der ZETA Guard die
1306 Speicherung und Verwendung der Privatschlüssel von Komponenten-Identitäten in einem
1307 HSM unterstützen. Für die Kubernetes etcd Verschlüsselung unterstützt ZETA Guard KMS
1308 v2.

1309 **A_25608-01 -ZETA Guard - Verarbeitung von Daten mit Schutzbedarf "sehr**
1310 **hoch"**

1311 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1312 Anbieter des TI 2.0-Dienstes keine Kenntnis über die in dem ZETA Guard verarbeiteten
1313 Daten erlangen darf, MUSS der Anbieter den ZETA Guard in einer VAU umsetzen.

1314 [\leq]

1315 **A_25763-01 -Zero Trust-Komponenten - Private Schlüssel der Komponenten-**
1316 **Identitäten in einem HSM**

1317 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1318 Anbieter des TI 2.0-Dienstes keine Kenntnis über die in dem ZETA Guard verarbeiteten
1319 Daten erlangen darf, MUSS der Anbieter die privaten Schlüssel der Identitäten des ZETA
1320 Guards in einem HSM speichern.

1321 [\leq]

1322 **A_25764 -Zero Trust-Komponenten - Sicherer Betrieb und Nutzung eines HSMs**

1323 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1324 Anbieter des TI 2.0-Dienstes keine Kenntnis über die in dem ZETA Guard verarbeiteten
1325 Daten erlangen darf, MUSS der Anbieter beim Einsatz eines HSMs sicherstellen, dass die
1326 auf dem HSM verarbeiteten privaten Schlüssel, Konfigurationen und eingesetzte Software
1327 nicht unautorisiert ausgelesen, unautorisiert verändert, unautorisiert ersetzt oder in
1328 anderer Weise unautorisiert benutzt werden können. [\leq]

1329 **A_25765 -Zero Trust-Komponenten - Einsatz zertifizierter HSM**

1330 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1331 Anbieter des TI 2.0-Dienstes keine Kenntnis über die in dem ZETA Guard verarbeiteten
1332 Daten erlangen darf, MUSS der Anbieter beim Einsatz eines HSMs sicherstellen, dass
1333 dessen Eignung durch eine erfolgreiche Evaluierung nachgewiesen wurde. Als
1334 Evaluierungsschemata kommen dabei Common Criteria oder Federal Information
1335 Processing Standard (FIPS) in Frage.

1336 Die Prüftiefe MUSS mindestens:

- 1337 1. FIPS 140-2 Level 3 oder
1338 2. FIPS 140-3 Level 3 oder
1339 3. Common Criteria EAL 4+ (mit AVA_VAN.5)

1340 entsprechen. [\leq]

1341 **A_26065-01 -Nur zugelassene Images in Produktion**

1342 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1343 Anbieter des TI 2.0-Dienstes keine Kenntnis über die in dem ZETA Guard verarbeiteten
1344 Daten erlangen darf, MUSS der Anbieter eine VAU verwenden, die mit technischen Mitteln
1345 sicherstellt, dass nur von gematik signierte und für den Einsatz in der PU vorgesehene
1346 Produktimages in der PU laufen können. [\leq]

1347 Das ZETA Guard Image ist unabhängig von einer spezifischen VAU-Architektur. Eine VAU
1348 ist allerdings in der Regel auf eine bestimmte Architektur (wie z. B. SGX, TDX, SEV usw.)

1349 ausgelegt. Da ZETA Guard aus mehreren Komponenten besteht, kann der Hersteller des
1350 TI 2.0-Dienstes flexibel entscheiden, wie ZETA Guard auf seiner VAU-Plattform
1351 implementiert wird. Dies kann beispielsweise die Nutzung von Confidential Containern,
1352 virtuellen Maschinen (VMs) oder die vollständige Ausführung von ZETA Guard innerhalb
1353 einer VAU umfassen. Entsprechend dieser Entscheidung muss der Hersteller das ZETA
1354 Guard Image für die jeweils eingesetzte VAU-Architektur umwandeln.

1355 **A_28756 -Umsetzung ZETA Guard in Sicherheits- und Datenschutzkonzept**

1356 Falls der ZETA Guard in einer VAU umgesetzt wird, muss der Hersteller des TI 2.0-
1357 Dienstes die Umsetzung von ZETA auf seiner VAU-Plattform in seinem Sicherheits- und
1358 Datenschutzkonzept beschreiben. [<=]

1359 1360 **A_28405 -ZETA Guard - Umwandlung für Ziel-VAU-Architektur**

1361 Falls der ZETA Guard in einer VAU umgesetzt wird, muss der Hersteller des TI2.0-Dienstes
1362 sicherstellen:

- 1363 • dass das ZETA Guard-Image in einer manipulationssicheren Build-Pipeline für die Ziel-
1364 VAU-Architektur erstellt und umgewandelt wird, und
- 1365 • dass der Build-Log sämtliche VAU-spezifischen Ergänzungen am ZETA Guard-Image
1366 protokolliert und für die gematik auditierbar ist.

1367 [<=]

1368 **A_28962 -ZETA GUARD - Nachweis der Umsetzung von Anforderungen in dem** 1369 **Build-Pipeline**

1370 Falls der ZETA Guard innerhalb einer VAU umgesetzt wird, können die Sicherheits- und
1371 Datenschutzerfordernungen an den ZETA Guard über die Build-Pipeline erfüllt werden. In
1372 diesem Fall MUSS der Hersteller des TI-2.0-Dienstes die Umsetzung dieser Anforderungen
1373 im Sicherheits- und Datenschutzkonzept nachvollziehbar beschreiben, sodass ein
1374 Gutachter die konforme Umsetzung eindeutig prüfen kann.[<=]

1375 **A_28744 -ZETA Guard - Ressourcenserver-Zugriff nur von freigegebenen** 1376 **Komponenten**

1377 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Hersteller des TI 2.0-
1378 Dienstes sicherstellen, dass der Zugriff auf den Ressourcenserver ausschließlich durch
1379 attestierte ZETA-Komponenten erfolgt. Die Attestierungswerte dieser Komponenten
1380 müssen vorab eindeutig geprüft und explizit für den Zugriff freigegeben worden sein.

1381 [<=]

1382 **A_28745 -ZETA Guard - Zugriff nur von freigegebenen Ressourcenservern**

1383 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Hersteller des TI 2.0-
1384 Dienstes sicherstellen, dass der Zugriff auf die ZETA-Komponenten ausschließlich durch
1385 einen attestierten Ressourcenserver erfolgt. Die Attestierungswerte dieses
1386 Ressourcenservers müssen zuvor eindeutig geprüft und explizit für diesen Zugriff
1387 freigegeben worden sein.

1388 [<=]

1389 **A_28757 -ZETA Guard - Zugriff nur zwischen attestierten Komponenten**

1390 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Hersteller des TI 2.0-
1391 Dienstes sicherstellen, dass der Zugriff zwischen ZETA-Komponenten ausschließlich durch
1392 attestierte ZETA-Komponenten erfolgt. Die Attestierungswerte dieser Komponenten
1393 müssen vorab eindeutig geprüft und explizit für den Zugriff freigegeben worden sein.

1394 [<=]

1395 **A_28809 -ZETA Guard - Abgesicherte Kommunikation zwischen ZETA-** 1396 **Komponenten in einer VAU**

1397 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und der
1398 Anbieter des TI 2.0 Dienstes keine Kenntnis über die im ZETA Guard verarbeiteten Daten
1399 erlangen darf, MUSS der Anbieter sicherstellen, dass:

1400 • die Kommunikation zwischen den ZETA Guard Komponenten durch mTLS abgesichert
1401 ist oder

1402 • sich die Komponenten innerhalb einer gemeinsamen VAU befinden, sodass die
1403 unverschlüsselte Kommunikation den geschützten Bereich der VAU zu keinem
1404 Zeitpunkt verlässt.

1405 [\leq]

1406 5.1.6 Sicherheits- und Datenschutz Anforderungen an dem ZETA 1407 Client in FdVs

1408 **A_25802 -ZETA Client - Einhaltung der BSI [TR-03161-1]**

1409 Der ZETA Client eines FdVs MUSS die BSI [TR-03161-1] erfüllen, sofern sie für den ZETA
1410 Client anwendbar ist. [\leq]

1411 *Hinweis: Nicht anwendbar können zum Beispiel sein: O.Paid .. Die Anwendbarkeit ist*
1412 *zwischen Hersteller des ZETA Clients und dem Gutachter zu klären. Der Gutachter gibt*
1413 *sein Votum über die Erfüllung der BSI [TR-03161-1] in Form der Bewertung der Erfüllung*
1414 *der A_25802 ab, wobei die A_25802 als „umgesetzt“ bewertet werden kann, wenn die*
1415 *anwendbaren Abschnitte der BSI [TR-03161-1] aus Sicht des Gutachters erfüllt sind.*

1416 5.2 Schlüssel Management und Verwaltung in ZETA

1417 Teil der Sicherheitsarchitektur von ZETA ist ein robustes Schlüsselmanagement. Um
1418 Entwicklern, Betreibern und Auditoren ein klares Verständnis der kryptografischen
1419 Architektur zu vermitteln, verwendet diese Spezifikation standardisierte **Schlüssel-IDs**
1420 (z.B. PrK.AuthS.Sig für den privaten Signaturschlüssel des Authorization Servers).

1421 5.2.1 Nomenklatur für Schlüssel-IDs

1422 Jeder Schlüssel erhält eine eindeutige ID nach folgendem Schema: **[Typ].[Komponente].**
1423 **[Zweck]**

1424 **Typ:**

- 1425 • PrK: Privater Schlüssel (Private Key)
- 1426 • PuK: Öffentlicher Schlüssel (Public Key)
- 1427 • C: Zertifikat (Public Key inkl. Identitätsbindung, Certificate)
- 1428 • SymK: Symmetrischer Schlüssel (Symmetric Key)

1429 **Komponente oder Schlüssel:**

- 1430 • AuthS: Authorization Server (PDP)
- 1431 • PEP: Policy Enforcement Point (HTTP Proxy)
- 1432 • DB: Datenbank (PDP DB)
- 1433 • Client: ZETA Client (App / Primärsystem)
- 1434 • K8s: Kubernetes / Infrastruktur
- 1435 • Sys: Gematik / Zentrales System
- 1436 • AK: Attestation Keys
- 1437 • EK: TPM Endorsement Keys

- 1438 • DPoP: DPoP Keys
- 1439 • SM(C)-B: SM(C)-B Keys
- 1440 • TI-RootCA: TI Root CA Signer Keys
- 1441 • TI-TSL: TI TSL Signer Keys
- 1442 • TI-KompCA: TI Komponenten PKI CA Keys
- 1443 • TI-FedMaster: Federation Master Signer Keys
- 1444 • ZETA-IK: ZETA OCI Container Image Signer Keys
- 1445 • ZETA-DK: ZETA OCI Data Image Signer Keys
- 1446 • ZETA-PAK: ZETA OPA Bundle Autor Signer Keys
- 1447 • ZETA-PFK: ZETA OPA Bundle Freigeber Signer Keys

Zweck:

- 1449 • TLS: Verschlüsselung des externen Traffics
- 1450 • mTLS: Interne Komponenten Authentifizierung
- 1451 • ASL: Application Security Layer
- 1452 • Sig: Signatur (Token, Policies, Entity Statements)
- 1453 • Enc: Payload-/Daten-Verschlüsselung (JWE, DB At-Rest)

5.2.2 Übersicht der ZETA Guard Schlüssel (Anbieter-Seite)

Diese Tabelle fasst alle Schlüssel zusammen, die vom Betreiber des TI 2.0-Dienstes (ZETA Guard) verwaltet werden müssen. Bei Verarbeitung von Daten mit Schutzbedarf "sehr hoch" greifen strenge VAU- und HSM-Pflichten.

Hinweis: Schlüssel mit dem Vermerk "Die Schlüsselverwaltung muss dokumentiert werden", werden vom Anbieter des TI 2.0 Dienstes verwaltet. Die Schlüssel ohne Vermerk verwaltet ZETA Guard automatisch.

Betriebsmodi und Schlüsselschutz:

Der ZETA Guard unterscheidet zwei primäre Betriebsmodi:

1. **Regulärer Betrieb (ohne VAU):**Schlüssel werden in sicheren Software-Keystores (z.B. KMS v2) der jeweiligen Container-Umgebung abgelegt.
2. **Betrieb mit Schutzbedarf "sehr hoch" (mit VAU):**Sobald sensible Daten verarbeitet werden, muss der ZETA Guard in einer Vertrauenswürdigen Ausführungsumgebung (VAU) betrieben werden (siehe A_25608-01). In diesem Fall **MÜSSEN** asymmetrische private Schlüssel der TI- und Internet-Identitäten zwingend in einem Hardware Security Module (HSM) verbleiben. Schlüssel, die in den Arbeitsspeicher der VAU geladen werden müssen (z.B. Token-Signatur- oder Datenbank-Schlüssel), müssen durch HSM-generierte Key-Encryption-Keys (KEK) "gewrappt" (Sealing) übergeben werden.

Tabelle 4: Tab-ZETA-Guard-Keys

Schlüssel-ID	Bezeichnung & Zweck	Speicherung / Betrieb OHNE VAU	Speicherung / Betrieb MIT VAU (Schutzbedarf "sehr hoch")
PrK.AuthS.Sig	AuthS Token-	PrK: Sicherer	PrK: Muss durch HSM

PuK.AuthS.Sig	Signaturschlüssel Zur Signatur von Access- und Refresh-Token sowie des Entity Statements. Der PuK wird im JWKS bereitgestellt. Die Schlüsselverwaltung muss dokumentiert werden.	Software-Keystore des AuthS.	geschützt sein. (Darf in den AuthS geladen werden, wenn z.B. mittels KEK aus dem HSM gesichert).
PrK.AuthS.TLS C.AuthS.TLS	AuthS Internet-Identität (TLS) Terminierung der externen TLS-Verbindung am Authorization Server. Die Schlüsselverwaltung muss dokumentiert werden.	Sicherer Software-Keystore.	PrK: Verbleibt zwingend im HSM (via HSM-Proxy), falls kein ASL Tunnel zum AuthS existiert.
SymK.DB.Enc	Datenbank-Verschlüsselung Verschlüsselung der Session-, Nutzer- und Client-Daten At-Rest im Authorization Server.	Sicherer Software-Keystore (z.B. K8s Secrets).	Muss durch HSM geschützt sein. Übergabe an AuthS darf nur an attestierte Instanzen erfolgen.
PrK.PEP.TLS C.PEP.TLS	PEP Internet-Identität (TLS) Terminierung der externen TLS-Verbindung am HTTP Proxy. Die Schlüsselverwaltung muss dokumentiert werden.	Sicherer Software-Keystore.	PrK: Verbleibt zwingend im HSM, falls kein ASL Tunnel zum HTTP Proxy existiert.
PrK.PEP.Sig C.PEP.Sig	PEP Signatur-Identität Zertifikat aus der Komponenten-PKI. Dient als Identität des PEP (für Signatur des ASL Master-Schlüssels). Die Schlüsselverwaltung muss dokumentiert werden.	Sicherer Software-Keystore.	Privater Schlüssel verbleibt zwingend im HSM (Zugriff via HSM-Proxy).
PrK.PEP.ASL PuK.PEP.ASL	PEP ASL-Identität Semi-statische Schlüsselpaare für den ZETA/ASL-Kanal (maximal 1 Monat gültig).	Im RAM des PEP.	Wird im PEP generiert, aber vom PrK.PEP.Sig im HSM beglaubigt.
SymK.PEP.ASL	PEP ASL Session-Key Abgeleiteter kurzlebiger Schlüssel für den eigentlichen ASL-Traffic.	Im RAM des PEP.	Wird im PEP generiert.
PrK.ZG.IDP PuK.ZG.IDP	IDP für die ZETA Guard Workload Identity Zur Signatur von Subject Token für die	Innerhalb der Kubernetes-Infrastruktur des	Wenn langlebiger Schlüssel, dann wird der private Schlüssel im HSM

	Kommunikation mit gematik-Diensten (SIEM, Telemetrie) und für die Dienst-zu-Dienst Kommunikation. Initial: Kubernetes IDP oder langlebiger Schlüssel Zukünftig: AuthS als IDP des ZETA Guard mit PrK.AuthS.Sig und PuK.AuthS.Sig Die Schlüsselverwaltung muss dokumentiert werden.	Anbieters.	gespeichert.
PrK.Ingress.TLS C.Ingress.TLS	Ingress TLS TLS-Terminierung am vorgeschalteten Ingress (falls genutzt). Die Schlüsselverwaltung muss dokumentiert werden.	Sicherer Software-Keystore.	Verbleibt zwingend im HSM, , falls kein ASL Tunnel zum AuthS und zum HTTP Proxy existiert.
PrK.K8s.mTLS C.K8s.mTLS	Interne Mesh-Identitäten Zur Absicherung der microservice-internen Kommunikation (z. B. AuthS <-> PE).	K8s Service Mesh (z.B. Istio).	Verwaltung innerhalb der VAU (Zugriff strikt limitiert).
PuK.Client.Sig	Öffentlicher Client Instance Key Wird bei der initialen Client-Registrierung an den AuthS gesendet. Dient der Validierung der "Client Assertion" bei zukünftigen Logins.	Gespeichert in der PDP Datenbank.	Gespeichert in der PDP Datenbank. Muss vor unautorisierter Manipulation (Austausch durch Angreifer) geschützt sein (Integrität).

1474

A_28826 -HSM Proxy, Übergabe PDP-Datenbank-Schlüssel an ZETA Guard Authorization Server

1475
1476
1477
1478
1479

Falls ZETA Guard in einer VAU umgesetzt wird, MUSS der HSM-Proxy sicherstellen, dass die Übergabe der PDP-Datenbank-Schlüssel (SymK.DB.Enc) ausschließlich an einen attestierten und freigegebenen ZETA Guard Authorization Server erfolgt.[<=]

A_28863 -HSM Proxy, Verwendung AuthS-Schlüssel nur durch ZETA Guard Authorization Server

1480
1481
1482
1483
1484
1485

Falls ZETA Guard in einer VAU umgesetzt wird, MUSS der HSM-Proxy sicherstellen, dass die Verwendung der AuthS-Schlüssel (PrK.AuthS.Sig, PrK.AuthS.TLS) im HSM ausschließlich durch einen attestierten und freigegebenen ZETA Guard Authorization Server möglich ist.[<=]

A_28864 -HSM Proxy, Verwendung PEP-Schlüssel nur durch ZETA Guard HTTP Proxy

1486
1487
1488
1489
1490

Falls ZETA Guard in einer VAU umgesetzt wird, MUSS der HSM-Proxy sicherstellen, dass die Verwendung der PEP-Schlüssel (PrK.PEP.TLS, PrK.PEP.Sig) im HSM ausschließlich durch einen attestierten und freigegebenen ZETA Guard HTTP Proxy möglich ist.[<=]

1491 **A_28865 -HSM Proxy, Verwendung Ingress-Schlüssel nur durch ZETA Guard**
 1492 **Ingress**
 1493 Falls ZETA Guard in einer VAU umgesetzt wird, MUSS der HSM-Proxy sicherstellen, dass
 1494 die Verwendung der Ingress-Schlüssel (PrK.Ingress.TLS) im HSM ausschließlich durch
 1495 einen attestierten und freigegebenen ZETA Guard Ingress möglich ist.【<=】

1496 **5.2.3 ZETA Client Schlüssel (Nutzer-Seite)**

1497 Diese Schlüssel verbleiben in der Verfügungsgewalt des Endnutzers (Smartphone /
 1498 Primärsystem) bzw. der Institution.

1499 **Tabelle 5: Tab-ZETA-Client-Keys**

Schlüssel-ID	Bezeichnung & Zweck	Speicherung / Verwendung
PrK.Client.Sig PuK.Client.Sig	Client Instance Key Pair Langlebiges ECC-Schlüsselpaar zur eindeutigen Identifikation der Client-Installation. Dient der Signatur der Client Assertion.	PrK: Zwingend in Hardware (TPM, Secure Enclave, TEE) generiert und gespeichert. Erhält strikt das Attribut sign (kein Export möglich). PuK: Wird bei der Registrierung (DCR) an den ZETA Guard übertragen.
PrK.DPoP.Sig PuK.DPoP.Sig	DPoP Schlüsselpaar Kurzlebiger (Session-basierter) Schlüssel zum Signieren von Anfragen als Beweis für den Besitz des Access Token (Proof of Possession).	PrK: Wird für die Dauer der Session sicher lokal gehalten (flüchtig im RAM oder durch native OS-Sicherheitsmechanismen / TPM-Wrapping geschützt). Eine Verschlüsselung des DPoP-Keys durch den PrK.Client.Sig ist aus Gründen der Schlüsseltrennung nicht zulässig. PuK: Wird im HTTP-Header gesendet.
PrK.AK.Sig PuK.AK.Sig	Plattform Attestation Key Zur Signatur des Client-Zustands.	PrK: Wird zur Signatur des Client-Zustands verwendet. Hardware-gebunden (TPM / Secure Enclave). PuK: Wird bei der Registrierung (DCR) an den ZETA Guard übertragen und zur Prüfung der Systemintegrität gesendet (inkl. Zertifikatskette zur Validierung gegen Root-CA des Herstellers).
PrK.EK.Enc PuK.EK.Enc C.EK.Enc	TPM 2.0 Endorsement Keys Wird bei der TPM Attestation verwendet.	PrK: Hardware-gebunden (TPM / Secure Enclave). PuK: Wird bei der Registrierung (DCR) an den ZETA Guard übertragen und zur Bindung des AK an den EK verwendet. C: Wird bei der Registrierung (DCR) an den ZETA Guard übertragen und zur Bindung des AK an den EK verwendet.

PrK.SM(C)-B.Sig C.SM(C)-B.Sig	SMC-B Institutionsidentität Ausgestellt von der SMC-B CA. Zur Signatur des subject_token beim Token Exchange, um die Identität der Institution (z.B. Praxis) nachzuweisen. Das Subject Token enthält ein Binding des PuK.Client.Sig und des PuK.DPoP.Sig.	PrK: Verbleibt hardwaregebunden auf der Smartcard (SMC-B) oder im HSM-B. C: Wird übermittelt und durch AuthS gegen TSL validiert.
----------------------------------	--	--

1500 **5.2.4 gematik verwaltete Schlüssel (TI)**

1501 Diese Zertifikate und Schlüssel spannen den Vertrauensraum der TI 2.0 auf. Die privaten
1502 Schlüssel liegen hochsicher bei der gematik (bzw. den zugelassenen Trust Centern). Die
1503 ZETA Guards und Clients nutzen die öffentlichen Teile/Zertifikate zur Validierung. Diese
1504 Schlüssel dienen der Sicherstellung der Supply-Chain-Security und der Integrität von
1505 Regelwerken.

1506 **Tabelle 6: Tab-Gematik-Keys**

Schlüssel-ID	Bezeichnung & Zweck	Speicherung / Verteilung
PrK.TI-RootCA.Sig C.TI-RootCA.Sig	TI Root CA Oberster Vertrauensanker der TI. Alle Zertifikate im System leiten sich hiervon ab.	C: Lokal in Truststores hinterlegt. Wird mit dem ZETA Guard Provisioning OCI Image in den ZETA Guard geladen. PrK: Offline/Hochsicher bei der gematik.
PrK.TI-TLSig C.TI-TSL.Sig	TSL Signer Zertifikat zur Validierung der Signatur der Trust-Service Status List (TSL), welche die aktuell gültigen CAs definiert.	C: Im ZETA Guard (über lokales Artifact Registry) zur TSL-Verifikation. PrK: Bei der gematik.
PrK.TI-KompCA.Sig C.TI-KompCA.Sig	Komponenten PKI CA Sub-CA der Root CA. Stellt die Zertifikate für die TI-Dienste (PEP, AuthS) aus.	C: Über die TSL als vertrauenswürdig verteilt.
PrK.TI-SMCB-CA.Sig C.TI-SMCB-CA.Sig	SMC-B CA Sub-CA der Root CA. Stellt die Institutionszertifikate für Leistungserbringer aus.	C: Über die TSL als vertrauenswürdig verteilt (zur Prüfung im ZETA Guard).
PrK.TI-FedMaster.Sig PuK.TI-FedMaster.Sig	Federation Master Signer Zur Signatur der Entity Statements (Trustmarks) in der OIDC-Föderation.	PuK: Im ZETA Guard hinterlegt, um die Föderations-Zugehörigkeit anderer AuthS zu prüfen. PrK: Bei der gematik.
PrK.ZETA-IK.Sig C.TI-ZETA-IK.Sig	Signaturzertifikat Ausführbare ZETA Images Signatur der ausführbaren ZETA	C: Im Admission Controller validiert (Signaturkette bis zur Root CA). PrK: In gematik CI/CD-Pipeline.

	OCI Container (PEP, AuthS, OPA, etc.).	
PrK.ZETA-DK.Sig C.TI-ZETA-DK.Sig	Signaturzertifikat Daten-Images Signatur von Datencontainern (z.B. Konfigurationsdaten, Provisioning-Daten).	C: Im Admission Controller / durch ZETA Guard validiert. PrK: In gematik CI/CD-Pipeline.
PrK.ZETA-PAK.Sig C.TI-ZETA-PAK.Sig	OPA Policy-Autor Signatur Signaturzertifikat des Policy- Erstellers für OPA Bundles.	C: Im ZETA Guard (OPA Engine) zur Signaturprüfung konfiguriert. PrK: Beim Datenbearbeiter.
PrK.ZETA-PFK.Sig C.TI-ZETA-PFK.Sig	OPA Policy-Freigeber Signatur Zweitsignatur (4-Augen-Prinzip) für OPA Bundles.	C: Im ZETA Guard (OPA Engine) konfiguriert. PrK: Beim Freigeber.
PrK.K8s.IDP C.K8s.IDP	Kubernetes IDP (Workload Identity) Zur Signatur von Subject Token für die Kommunikation mit gematik-Diensten (SIEM, Telemetrie).	PrK: Innerhalb der Kubernetes- Infrastruktur des Anbieters. C: Innerhalb des gematik Workload Identity Pools

1507

1508 **5.3 ZETA Abläufe**

1509 Dieses Kapitel spezifiziert die dynamischen Kommunikations- und Prozessabläufe
1510 innerhalb der ZETA-Architektur. Um die Interaktionen zwischen dem ZETA Client, dem
1511 ZETA Attestation Service (ZAS), dem ZETA Guard (Authorization Server) und dem
1512 Resource Server nachvollziehbar darzustellen, sind die Unterkapitel chronologisch anhand
1513 des Lebenszyklus einer Client-Instanz strukturiert.

1514 Die Beschreibung folgt dem Weg von stationären und mobilen Clients von der initialen
1515 Bereitstellung (Installation und Start) über die netzwerktechnische Ermittlung der
1516 benötigten Endpunkte (Service Discovery) bis hin zur initialen Schlüsselbindung und
1517 Registrierung (Dynamic Client Registration). Darauf aufbauend wird detailliert dargestellt,
1518 wie der Client unter fortlaufendem Nachweis seiner Systemintegrität (Attestierung)
1519 Access Tokens bezieht und diese für den sicheren Zugriff auf geschützte Fach-dienste
1520 (Resource Server) nutzt. Den Abschluss bilden die Prozesse zur Session-Erneuerung sowie
1521 die Besonderheiten bei der Dienst-zu-Dienst-Kommunikation.

1522 **5.3.1 Abläufe für stationäre Clients**

1523 Dieses Unterkapitel beschreibt die spezifischen ZETA-Abläufe für stationäre Endgeräte
1524 (Primärsysteme). Im Zentrum steht dabei der Nachweis der Systemintegrität, welcher
1525 primär hardwaregestützt über ein Trusted Platform Module (TPM) oder eine Secure
1526 Enclave erfolgt. Für Systeme ohne entsprechende Hardware-Voraussetzungen wird eine
1527 Software-Attestierung (SW Attestation) als Fallback unterstützt. Auf Basis dieser
1528 Attestierung erfolgt die Autorisierung beim ZETA Guard nach OAuth Token Exchange

1529 Grant, um die sichere Ausstellung und den Austausch von Access Tokens für den Zugriff
1530 auf geschützte Ressourcen zu regeln.

1531 Abbildung 3: Attestierungsablauf nach Betriebssystem gibt einen vereinfachten Überblick
1532 über die ganzheitlichen Prozesspfad über die verschiedenen Betriebssystemvarianten und
1533 dient als Leitfaden für die folgenden Detailbeschreibungen.

1534

Überblick Attestierungsabläufe nach Betriebssystem

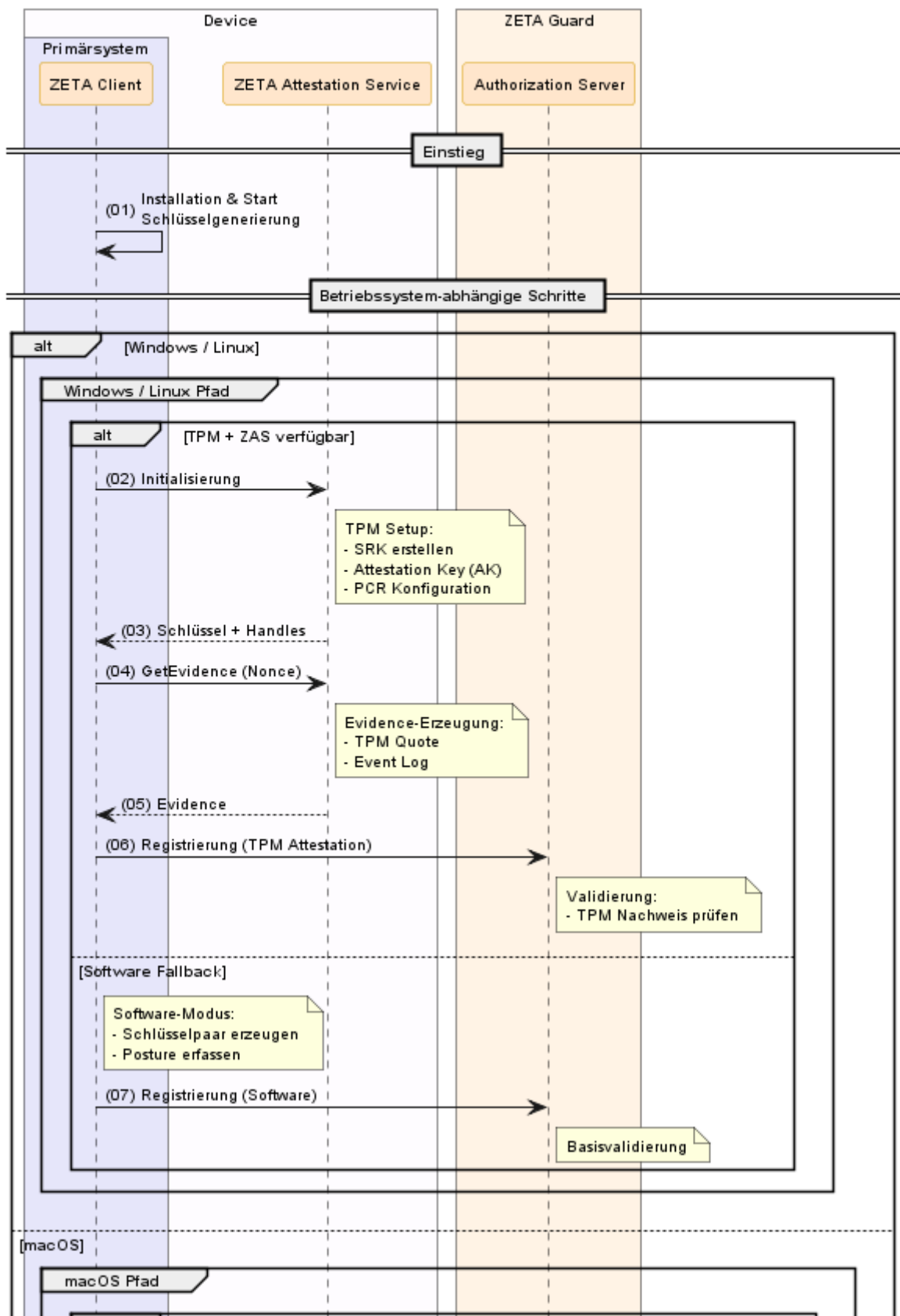


Abbildung 3 : Abb-Attestierungsablauf-nach-Betriebssystem

1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565

Der Ablauf kann wie folgt zusammengefasst werden.

1. Initialisierung

- Installation des Clients und Generierung eines Client-Schlüsselpaares
- Aufbau einer initialen System-Baseline (Messwerte, Schlüsselmaterial)

2. Hardware-abhängiger Attestierungspfad

- Windows / Linux
 - Primär: TPM-basierte Attestierung durch den ZETA Attestation Service (ZAS)
 - Fallback: Software-basierte Attestierung ohne TPM
- macOS
 - Primär: Secure Enclave + Apple App Attest / DeviceCheck
 - Fallback: Software-basierte Attestierung

3. Vorbereitung der Registrierung

- Bereitstellung der attestierungsrelevanten Schlüssel und Nachweise
- Plattform-spezifische Attestierungsobjekte:
 - TPM: EK-, AK- und Client-Schlüssel + Zertifikate
 - macOS: Apple Attestation Object
 - Software: ausschließlich Client-Schlüssel

4. Client-Registrierung beim ZETA Guard

- Durchführung eines Dynamic Client Registration (DCR)
- Ablauf abhängig vom Attestierungstyp:
 - TPM (Windows/Linux)
 - Secure Enclave (macOS)
 - Software (Fallback)

5. Laufende Attestierung und Authentifizierung

- Regelmäßige Erhebung von Zustandsdaten (Evidence)
- Token-Abruf über OAuth Token Exchange
- optionales Attestation Token bei hardwaregestützter Vertrauensbildung

5.3.1.1 Client Installation und Schlüsselgenerierung

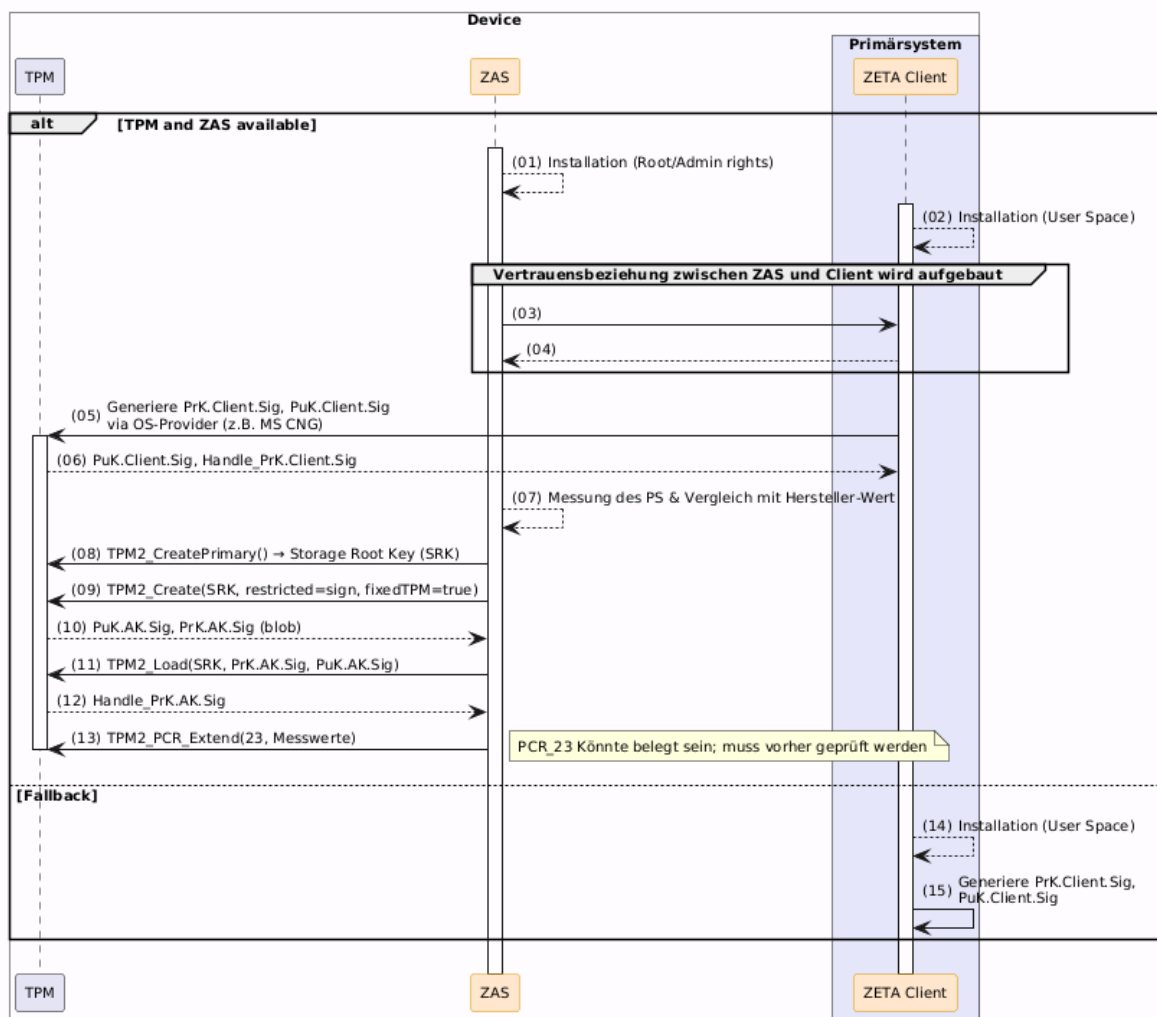
Nach der Installation werden beim ersten Start des Primärsystems die Schlüssel generiert, die vom ZETA Client als Identitäts-Nachweis gegenüber ZETA Guard Instanzen verwendet werden.

Das Ergebnis des Prozesses ist eine initiale Baseline des Clientsystems, die – abhängig vom gewählten Attestierungsmodus – aus hardware- oder software-gebundenen Schlüsseln sowie initial erhobenen Zustands- und Messinformationen besteht. Diese Baseline bildet die Grundlage für alle weiteren Schritte des Attestierungslebenszyklus.

1574 5.3.1.1.1 Schlüsselgenerierung auf Windows und Linux Systemen

1575 Die Installation des Primärsystems (inkl. ZETA Client) erfolgt in der Regel ohne Admin
 1576 Rechte. Um Zugriff auf die TPM Funktionen zu erhalten, ist jedoch eine Komponente
 1577 erforderlich, die unter Admin-Rechten ausgeführt wird. Dafür wird der ZETA Attestation
 1578 Service (ZAS) verwendet. Zwischen ZAS und ZETA Client muss eine Vertrauensbeziehung
 1579 bestehen, damit nicht beliebige Clients vom ZAS Attestation-Daten anfragen können und
 1580 damit der ZETA Client nicht mit einem kompromittierten ZAS kommuniziert.

1581 Das folgende Sequenzdiagramm beschreibt den Prozess der Schlüsselgenerierung und
 1582 Attestierung auf Windows und Linux Betriebssystemen. Der Ablauf gliedert sich in einen
 1583 primären, hardwaregestützten Pfad über das TPM und einen Software-Attestation
 1584 Fallback-Pfad.



1585
 1586 **Abbildung 4 : Abb-ZETA-Schlüsselgenerierung-Windows-und-Linux**

1587 Die Schritte (01)-(13) werden nur durchgeführt, wenn auf dem Clientsystem ein TPM
 1588 vorhanden ist und das Primärsystem ein ZETA Attestation Service beinhaltet.

- 1589 • (01) Privilegierte Installation: Der ZETA Attestation Service wird mit administrativen
 1590 Rechten installiert, damit Messungen des Primärsystems in ein TPM Register
 1591 geschrieben werden können.
- 1592 • (02) Die Installation des Primärsystems (mit Ausnahme des ZAS) erfolgt ohne Admin
 1593 Rechte.

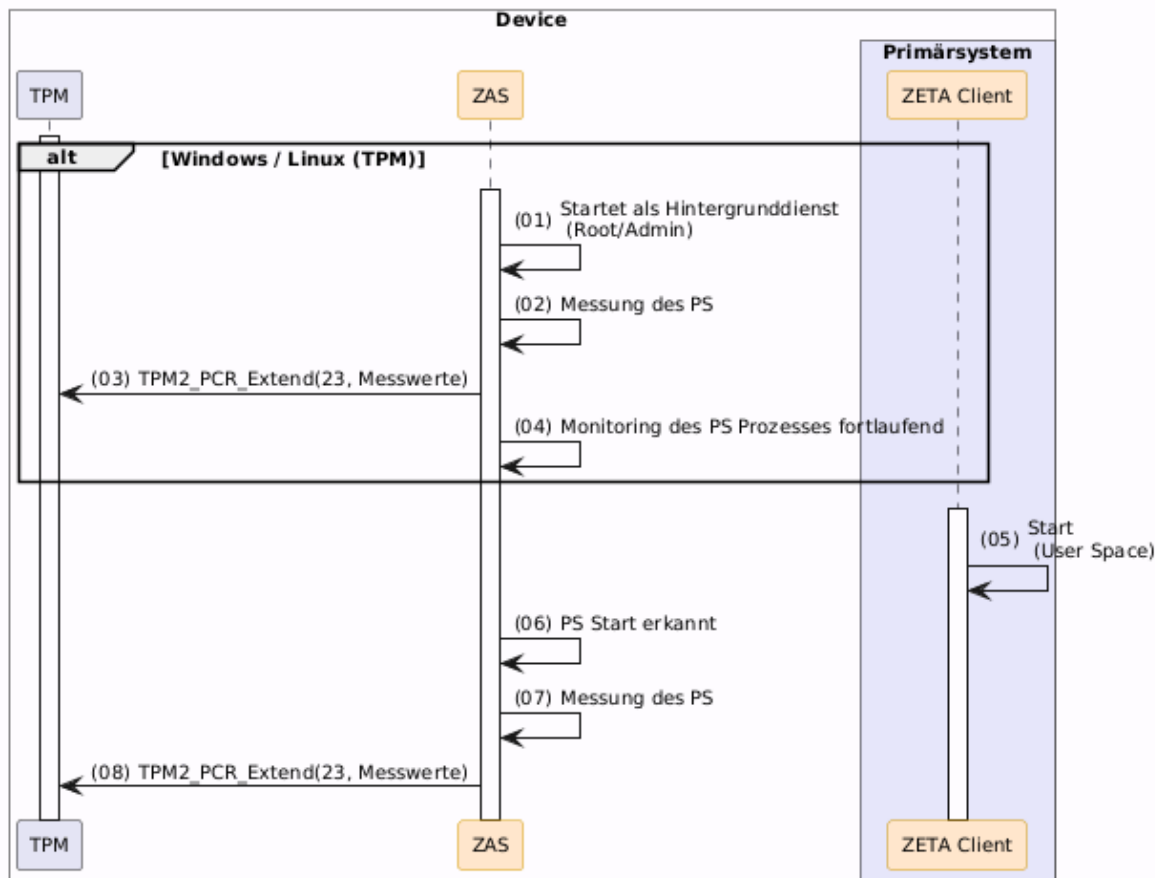
- 1594 • (03)-(04) Um eine exklusive und sichere Vertrauensbeziehung zwischen dem ZAS
1595 (läuft privilegiert als Root/Admin) und dem Primärsystem (läuft im User Space)
1596 herzustellen, müssen Betriebssystem-Mechanismen (IPC-Sicherheit) mit
1597 kryptographischer Bindung kombiniert werden. Da beide Komponenten vom selben
1598 Hersteller stammen, können zudem Code-Signatur-Prüfungen genutzt werden.
- 1599 • (05) Erzeugung eines Client-Signaturschlüsselpaares: Auf Windows- oder
1600 Linux-Systemen wird ein Client-Signaturschlüsselpaar über einen Betriebssystem-
1601 oder TPM-Provider erzeugt.
- 1602 • (06) Übergabe von Schlüsselmaterial: Die öffentlichen Schlüsselanteile sowie
1603 zugehörige Schlüssel-Handles werden dem ZETA Client bereitgestellt.
- 1604 • (07) Messung des Primärsystems: Der Zustand des Primärsystems wird gemessen. Die
1605 Messwerte KÖNNEN optional mit herstellerseitig bereitgestellten Referenzwerten
1606 verglichen werden.
- 1607 • (08) Erzeugung einer Storage Root Key (SRK): Eine Storage Root Key wird innerhalb
1608 des TPM als Vertrauensanker erzeugt.
- 1609 • (09) Erzeugung des Attestierungsschlüssels (AK): Ein Attestierungsschlüssel wird
1610 erzeugt und fest an das TPM gebunden.
- 1611 • (10) Bereitstellung der öffentlichen AK-Anteile: Die öffentlichen Anteile des
1612 Attestierungsschlüssels werden exportiert.
- 1613 • (11) Laden des Attestierungsschlüssels: Der Attestierungsschlüssel wird in das TPM
1614 geladen und aktiviert.
- 1615 • (12) Übergabe des AK-Handles: Der ZETA Client erhält einen Handle zur
1616 Referenzierung des Attestierungsschlüssels.
- 1617 • (13) Erweiterung von PCR-Registern: Die ermittelten Messwerte werden in geeignete
1618 Platform Configuration Registers (PCRs) erweitert. Vor der Erweiterung prüft der ZAS,
1619 ob die jeweiligen PCR-Register bereits belegt sind. Nur wenn ein PCR frei ist, kann der
1620 ZAS das TPM verwenden. Wenn kein PCR frei ist, muss die Fallback Alternative ohne
1621 TPM Nutzung verwendet werden. In den weiteren Schritten ist dann nur die Software-
1622 Attestation nutzbar.
- 1623 Die Schritte (14)-(15) stellen eine Fallback Alternative zu den TPM-spezifischen Schritten
1624 dar.
- 1625 • (14) Die Installation des Primärsystems erfolgt ohne Admin Rechte.
- 1626 • (15) Der ZETA Client erzeugt das Client-Signaturschlüsselpaar lokal im
1627 Software-Kontext.

1628 5.3.1.1.2 Schlüsselgenerierung auf macOS Systemen

1629 Siehe [5.3.2.1.2- Apple Secure Enclave](#)

1630 5.3.1.2 Client Start mit TPM und ZAS

1631 Wenn unter Windows oder Linux ein TPM 2.0 und der ZAS verfügbar sind, dann wird nach
1632 dem Booten und bei jedem Start des Primärsystems eine Messung durch den ZAS
1633 durchgeführt. Die Messung wird im weiteren Verlauf bei der Remote Attestierung im ZETA
1634 Guard mit einem vom Hersteller vorgegebenen Wert verglichen.



1635
1636

Abbildung 5 Abb-ZETA-Client-Start-mit-TPM-und-ZAS

1637 (01) Der ZAS wird während des Starts des Betriebssystems als Service gestartet.
 1638 (02)-(03) Der ZAS führt eine Messung der unveränderlichen Teile des Primärsystems
 1639 durch und trägt sie in ein TPM PCR ein.
 1640 (04) Durch ein kontinuierliches Monitoring prüft der ZAS, ob das Primärsystem gestartet
 1641 wurde.
 1642 (05)-(08) Wenn ein Start des Primärsystems erkannt wurde, führt der ZAS eine Messung
 1643 der unveränderlichen Teile des Primärsystems durch und trägt sie in ein TPM PCR ein.

1644 **5.3.1.3 Service Discovery**

1645 Das folgende Sequenzdiagramm beschreibt den sogenannten Discovery-Prozess
 1646 (Metadaten-Abruf) eines ZETA Clients in der TI 2.0 Architektur. Bevor der Client Zugriffs-
 1647 Tokens anfordern kann, muss er dynamisch herausfinden, welcher Autorisierungsserver
 1648 für eine bestimmte Ressource zuständig ist, welche Konfiguration für den Zugriff
 1649 verwendet werden muss (z. B. ZETA/ASL Nutzung) und welche Endpunkte und
 1650 Konfigurationen der TI 2.0 Dienst nutzt.

1651 Dieser Prozess basiert auf etablierten IETF/OAuth-Standards (RFCs) und teilt sich in zwei
 1652 aufeinanderfolgende Abfragen auf.

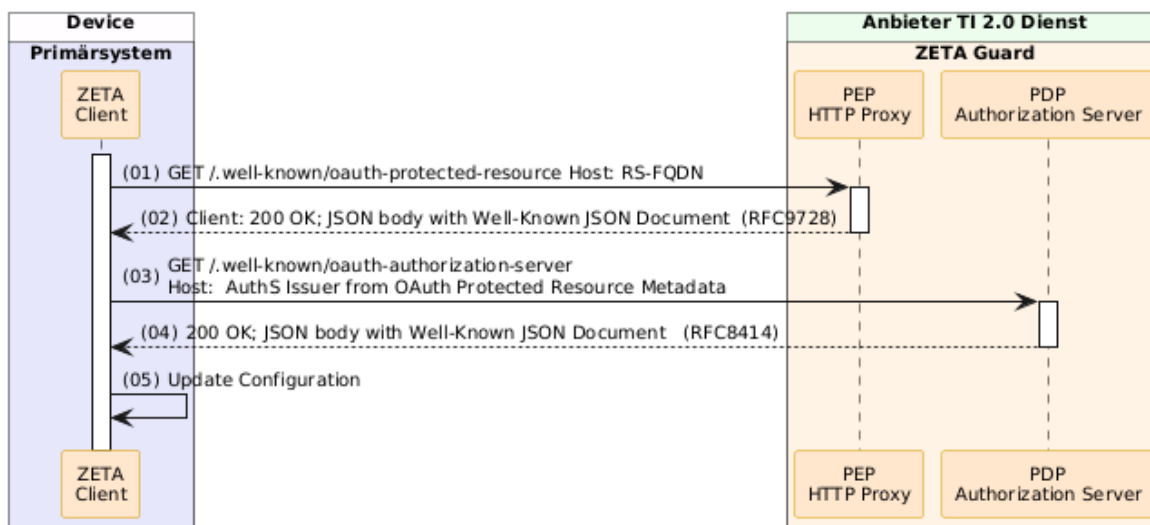


Abbildung 6 Abb-ZETA-Service-Discovery

1654
1655

1656 (01) Der ZETA Client sendet einen HTTP GET Request an den PEP HTTP Proxy. Der Proxy
1657 agiert hier als Stellvertreter für die eigentliche Fachanwendung (Resource Server / RS-
1658 FQDN). Der Pfad /.well-known/oauth-protected-resource ist standardisiert und dient exakt
1659 diesem Zweck.
1660 (02) Antwort des Proxys: Der PEP Proxy antwortet mit 200 OK und liefert ein JSON-
1661 Dokument zurück. Gemäß RFC 9728 (OAuth 2.0 Protected Resource Metadata) enthält
1662 dieses Dokument Metadaten über die Ressource. Die für den Client wichtigste Information
1663 darin ist der Issuer (die URL) des Autorisierungsservers (PDP), der diese spezielle
1664 Ressource schützt. Das Schema für das OAuth Protected Resource Well-known JSON
1665 Dokument ist in [opr-well-known.yaml] festgelegt.

1666 Nachdem der Client nun weiß, mit wem er für die Authentifizierung sprechen muss, fragt
1667 er diesen Server nach seinen Fähigkeiten und Endpunkten.
1668 (03) Der ZETA Client sendet nun einen GET Request an den in Schritt (02) ermittelten PDP
1669 Authorization Server. Er fragt den standardisierten Pfad /.well-known/oauth-authorization-
1670 server ab.
1671 (04) Der Authorization Server antwortet mit 200 OK und einem weiteren JSON-Dokument.
1672 Nach RFC 8414 (OAuth 2.0 Authorization Server Metadata) enthält dieses Dokument die
1673 vollständige Konfiguration des Servers. Dazu gehören z.B. die URLs für den Token-
1674 Endpoint (um Tokens anzufordern), den Registration-Endpoint (DCR), unterstützte
1675 Verschlüsselungsalgorithmen, Scopes und die URL zu den öffentlichen Schlüsseln (JWKS)
1676 des Servers. Über das Feld api_versions_supported gibt der Authorization Server zudem
1677 bekannt, welche Vertragsversionen der Token-Ausstellung er unterstützt; hieran erkennt
1678 der ZETA Client, ob er den Parameter resource (Contract v2) verwenden kann oder den
1679 Parameter audience (Contract v1, Legacy) verwenden muss (siehe 5.4.2.5). Das Schema
1680 für das OAuth Authorization Server Well-known JSON Dokument ist in [as-well-
1681 known.yaml] festgelegt.

1682 (05) Update Configuration: Der ZETA Client verarbeitet die empfangenen JSON-
1683 Dokumente aus Schritt (02) und (04) lokal. Er aktualisiert seine interne Konfiguration mit
1684 den abgerufenen Endpunkten und Parametern.

A_29374 -ZETA, Ablauf Service Discovery

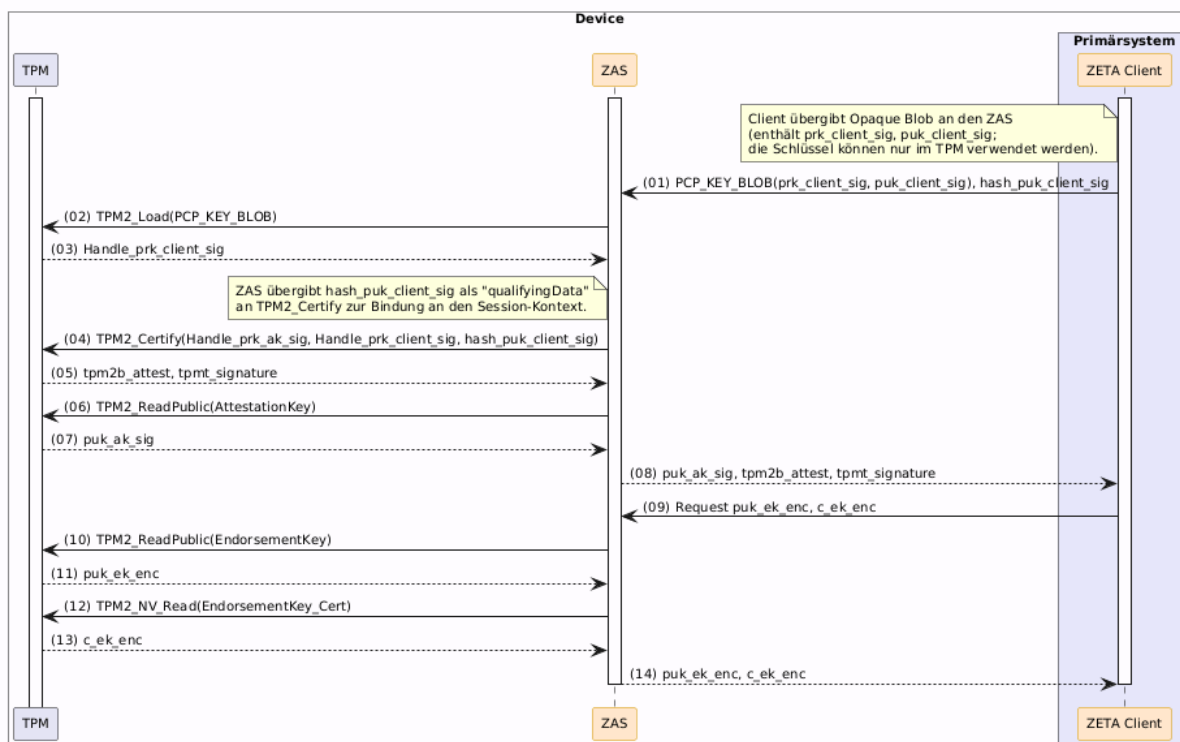
1685 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
1686 *Service-Discovery* unterstützen. [<=]
1687

1688 **5.3.1.4 Vorbereitung der Client-Registrierung beim ZETA Guard**

1689 Wenn ein ZAS und TPM oder eine Secure Enclave verwendet werden, dann müssen
 1690 Schlüssel für die Attestierung beim Authorization Server bekannt gemacht werden, um
 1691 die Attestierung dort verifizieren zu können.

1692 **5.3.1.4.1 ZAS und TPM**

1693 Der folgende Ablauf zeigt die erforderlichen Schritte, um alle benötigten Schlüssel für die
 1694 Verifikation per TPM Attestation einzusammeln.

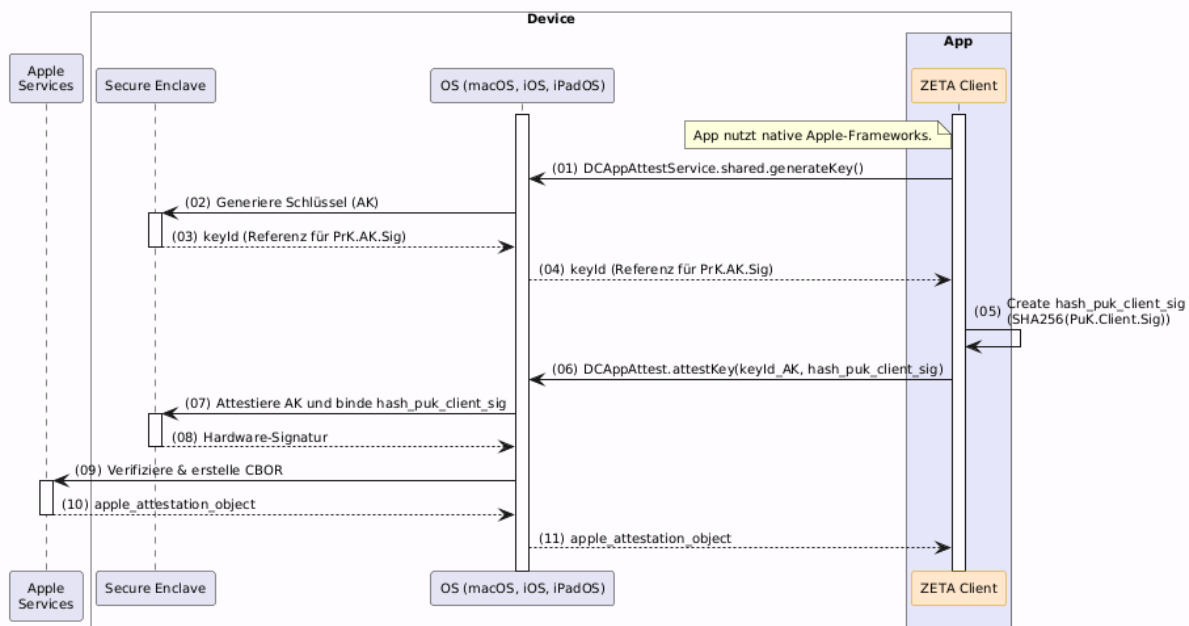


1695
 1696

Abbildung 7 Abb-ZETA-TPM-Attestation-Key

1697 (01)-(03) Das TPM hat nur wenig Speicher. Daher werden anwendungsspezifische
 1698 Schlüssel außerhalb des TPM als PCP_KEY_BLOB gespeichert. Die Schlüssel können
 1699 dennoch nur im TPM verwendet werden. Vor der Verwendung müssen die Schlüssel in das
 1700 TPM geladen werden.
 1701 (04)-(05) Es wird ein TPM2 Attest erzeugt, das das Schlüsselpaar PrK.Client.Sig und
 1702 PuK.Client.Sig als TPM-Schlüssel bestätigt.
 1703 (06)-(14) Aus dem TPM werden der PuK.AK.Sig und der Puk.EK.Enc Schlüssel sowie das
 1704 C.EK.Enc Zertifikat ausgelesen und an den ZETA Client übergeben.

1705 5.3.1.4.2 Secure Enclave



1706
1707

Abbildung 8 Abb-ZETA-SE-Attestation-Key

1708 (01)-(11) Der Ablauf zeigt, wie ein Apple Attestation Object erzeugt wird, das den Hash
1709 des PuK.Client.Sig enthält.

1710 **5.3.1.5 Client-Registrierung**

1711 Das folgende Sequenzdiagramm beschreibt den Registrierungsprozess eines ZETA Clients
1712 bei einem ZETA Guard Authorization Server. Der Ablauf unterscheidet sich grundlegend
1713 danach, ob bereits ein ZETA Guard Attestation Token vorhanden ist oder nicht. Dies wird
1714 durch den äußeren alt-Block dargestellt. Für eine bessere Übersichtlichkeit sind nicht alle
1715 erforderlichen Parameter des DCR Requests dargestellt (siehe [dcr-request.yaml] für
1716 detaillierte Informationen).

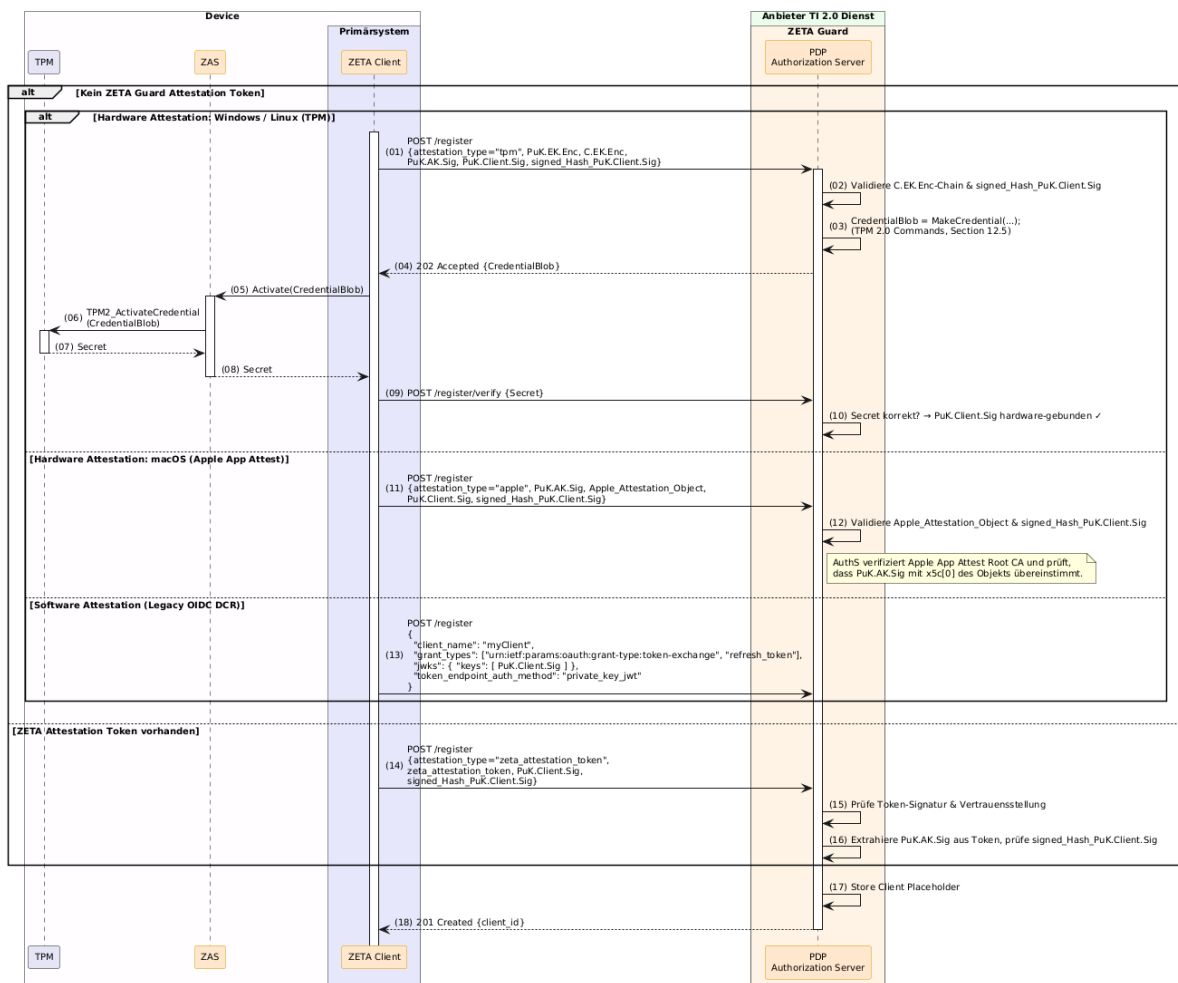


Abbildung 9 Abb-ZETA-DCR-für-stationäre-Clients

Wenn der Client noch kein Token besitzt, muss er seine Vertrauenswürdigkeit durch eine Attestierung nachweisen. Hierfür gibt es drei Unterfälle (innerer alt-Block), abhängig von der genutzten Hardware/Software.

(01)-(10) Hardware-Attestation: Windows / Linux (TPM)

Der ZETA Client sendet einen Request an den PDP Authorization Server mit folgenden Parametern:

Der Authorization Server validiert die Zertifikatskette von sowie . Anschließend erzeugt er ein gemäß dem TPM 2.0 Befehl (vgl. TCG TPM Library Part 3, Section 12.5) und antwortet mit gemäß [dcr-response-202.yaml].

Der ZETA Client leitet das an den ZAS weiter. Der ZAS führt aus und erhält das entschlüsselte . Das wird an den Authorization Server via übermittelt. Nach erfolgreicher Prüfung gelten und PuK.AK.Sigals hardware-gebunden.

(11)-(12) Hardware-Attestation: macOS (Secure Enclave)

Der ZETA Client sendet insbesondere mit:

Der Authorization Server validiert das und . Dabei wird sichergestellt, dass identisch mit dem Schlüssel aus des Attestation-Objekts ist und die Apple Root CA-Kette gültig ist.

1717

1718

1719

1720

1721

1722

1723

1724

1725

1726

1727

1728

1729

1730

1731

1732

1733

1734

1735

1736

1737 DerWert vonsigned_Hash_PuK.Client.Sig enthält die Signatur des Hashes von
 1738 PuK.Client.Sig mit dem *eigenen privaten Client-Schlüssel* (PrK.Client.Sig).
 1739 DerHash_PuK.Client.Sig ist im Apple Attestation Object enthalten.

1740 **(13)-(14) Software-Attestation (Legacy OIDC DCR)**

1741 Im Software-Attestation-Pfad sendet der ZETA Client einen Standard-OIDC-DCR-Request:

```
1742 {"client_name": "<Name des Clients>", "grant_types": ["urn:ietf:params:oauth:grant-  

  1743 type:token-exchange", "refresh_token"], "jwks": {"keys": [ PuK.Client.Sig  

  1744 ]}, "token_endpoint_auth_method": "private_key_jwt" }
```

1745 Der Authorization Server legt den Client in der PDP DB an. Erst wenn beim Token Request
 1746 die Policy Engine dem Authorization Server eine allow: true Decision übergibt, wird der
 1747 Client als unterstützter Client akzeptiert.

1748 **Vorbedingung: ZETA Guard Attestation Token vorhanden**

1749 Verfügt der ZETA Client über ein gültiges ZETA Guard Attestation Token aus einer
 1750 vorherigen Authentifizierung, kann die erneute Registrierung beschleunigt erfolgen.

1751 **Beispiel**

```
1752 {  

  1753 "attestation_type": "zeta_attestation_token",  

  1754 "client_name": "ZETA Client App",  

  1755 "token_endpoint_auth_method": "private_key_jwt",  

  1756 "grant_types": [  

  1757   "urn:ietf:params:oauth:grant-type:token-exchange",  

  1758   "refresh_token"  

  1759 ],  

  1760 "redirect_uris": [  

  1761   "https://app.example-hersteller.de/cb/zeta-client/oidc",  

  1762   "https://app.example-hersteller.de/cb/zeta-client/app"  

  1763 ],  

  1764 "jwks": {  

  1765   "keys": [  

  1766     {  

  1767       "kty": "EC",  

  1768       "crv": "P-256",  

  1769       "x": "11qYAYKxCrfVS_7TyWQHOG7hcvPapiMlrwlaaPcHURo",  

  1770       "y": "eZXwxvO1hvCY0KucrPfKo7yAyMT6Ajc3N7OkAB6VYy8"  

  1771     }  

  1772   ]  

  1773 },  

  1774 "zeta_attestation_token": "eyJ...",  

  1775 "signed_hash_puk_client_sig": "MEQ..."  

  1776 }
```

1778 Der Authorization Server prüft die Token-Signatur, die Vertrauensstellung sowie anhand
 1779 des im Token enthaltenen .

1780 **Abschluss der Registrierung**

1781 Nach erfolgreichem Durchlauf eines der obigen Pfade antwortet der Authorization Server
 1782 mit .

1783 **A_29375 -ZETA, Ablauf Dynamic Client Registration**

1784 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
 1785 *DCR-für-stationäre-Clients* unterstützen. [<=]

1786 *Hinweis: Ein ZETA Client muss nur die für das eigene Betriebssystem passende Variante*
 1787 *des Dynamic Client Registration Flows implementieren.*

5.3.1.6 Authentifizierung

1788 Die Authentifizierung für Primärsysteme erfolgt per OAuth Token Exchange mit einem
 1789 SM(C)-B signiertem Subject Token. Eine erfolgreiche Authentifizierung führt auch zu
 1790 einem Abschluss der Client-Registrierung, da die Clients grundsätzlich einem Nutzer
 1791 zugeordnet werden.
 1792

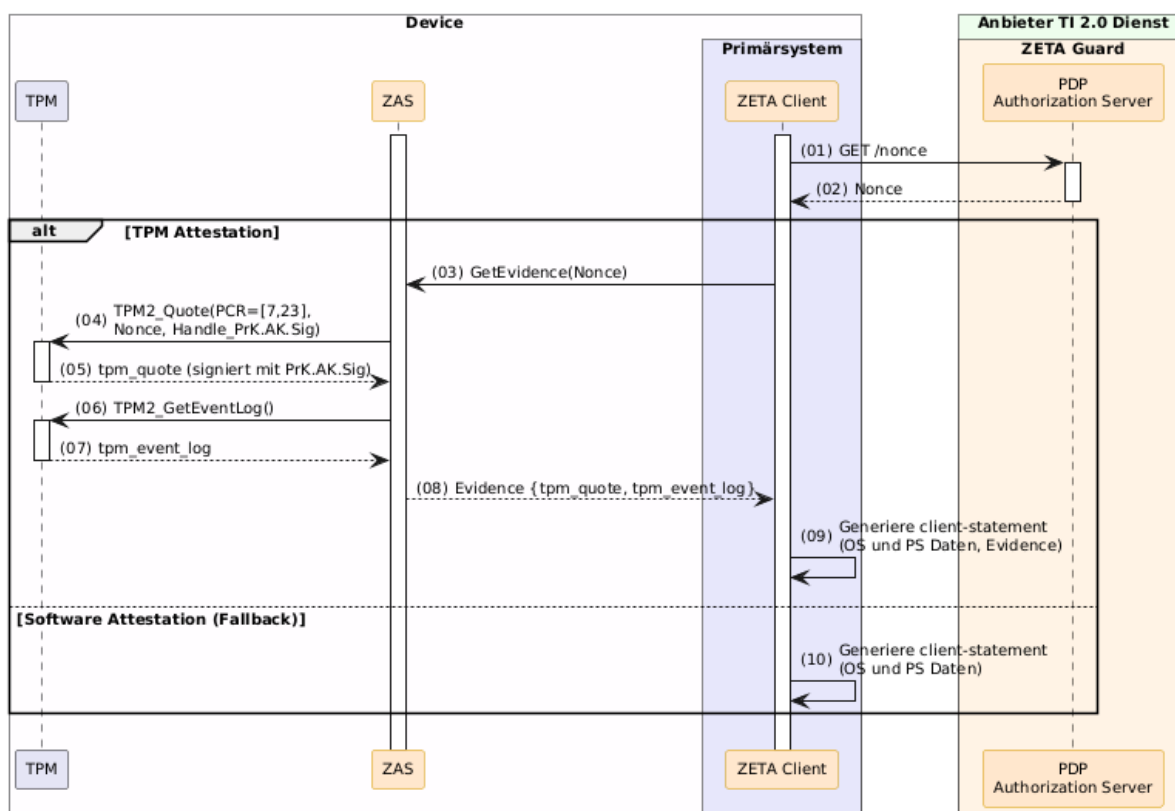
1793 ZETA Guard prüft nicht nur, ob der Nutzer für den Zugriff berechtigt ist, sondern auch ob
 1794 der Client Zugriff erhalten darf. Daher muss der ZETA Client neben dem Subject Token
 1795 auch ein Client Assertion JWT erstellen und mit dem Schlüssel PrK.Client.Sig signieren.
 1796 Das Client Assertion JWT dient zur Authentifizierung des Clients gegenüber dem ZETA
 1797 Guard Authorization Server und enthält im cLient_statement die Attestation Daten des
 1798 Clients (siehe [client-assertion-jwt.yaml]).

5.3.1.6.1 Client Statement mit ZAS und TPM Attestation

1799 Das folgende Sequenzdiagramm beschreibt den Prozess zur Erhebung von Integritäts-
 1800 und Zustandsdaten (Evidence/Posture) eines Gerätes durch einen ZETA Client mit ZAS
 1801 und TPM. Dieser Prozess dient dazu, dem ZETA Guard kryptografisch abzusichern, in
 1802 welchem Sicherheitszustand sich der Client aktuell befindet.
 1803

1804 Im Schema [client-statement.yaml] sind die Daten, die bei der Attestierung verwendet
 1805 werden, festgelegt.

1806 Der Ablauf beginnt mit der Einholung einer kryptografischen Herausforderung (Nonce)
 1807 und teilt sich danach auf, je nachdem, welche Hardware-Sicherheitsfeatures das Gerät
 1808 bietet.



1809

1810 **Abbildung 10 Abb-ZETA-Client-Statement-mit-TPM-Attestation**

- 1811 (01) Der ZETA Client sendet einen Request (GET /nonce) an den PDP Authorization Server
1812 (ZETA Guard), um eine Nonce (eine einmalige Zufallszahl) anzufordern. Dies schützt den
1813 späteren Prozess vor Replay-Attacks.
- 1814 (02) Der Server generiert die Nonce und sendet sie an den Client zurück.
- 1815 (03) Der ZETA Client beauftragt den ZAS mit der Erhebung der Nachweise (GetEvidence)
1816 und übergibt dabei die erhaltene Nonce.
- 1817 (04)-(05) Der ZAS fordert vom TPM ein sogenanntes Quote an (TPM2_Quote). Dabei
1818 fordert er die Signierung bestimmter Platform Configuration Registers (hier PCR=[7,23],
1819 welche typischerweise Boot-State und Application-State repräsentieren) an. In diesen
1820 Aufruf fließen die Nonce (zur Sicherstellung der Aktualität) und das Handle des privaten
1821 Attestation Keys (Handle_PrK.AK.Sig) ein.
- 1822 (06)-(07) Zusätzlich ruft der ZAS das TCG Event Log (TPM2_GetEventLog()) vom System
1823 ab. Das Log enthält die detaillierte Historie, wie die Werte in den PCRs zustande
1824 gekommen sind.
- 1825 (08) Der ZAS bündelt das Quote und das Event Log zu einer Evidence und sendet
1826 diese an den ZETA Client zurück.
- 1827 (09) Der ZETA Client generiert das `client_statement`, in dem Daten über das
1828 Primärsystem, das Betriebssystem und die Evidence Daten enthalten sind.
- 1829 (10) Software-Attestation: Der ZETA Client sammelt die OS- und Primärsystem-Daten
1830 selbst und packt sie in ein `client_statement`. In diesem Fall findet keine kryptografische
1831 Signatur durch ein TPM statt.

1832 *5.3.1.6.2 Client Statement mit Apple AppAttest*

1833 Das folgende Sequenzdiagramm beschreibt detailliert den Prozess zur Erhebung von
1834 Integritäts- und Zustandsdaten (Evidence) auf einem macOS-Gerät durch einen ZETA
1835 Client. Es zeigt den korrigierten, architektonisch sauberen Ablauf unter Verwendung der
1836 Apple App Attest (bzw. DeviceCheck) API in Verbindung mit der Secure Enclave.

1837 Im Schema [`client-statement.yaml`] sind die Daten, die bei der Attestierung verwendet
1838 werden, festgelegt.

1839 Der Ablauf ist in eine initiale Phase und eine Fallunterscheidung (Attestierungs-Methode)
1840 gegliedert.

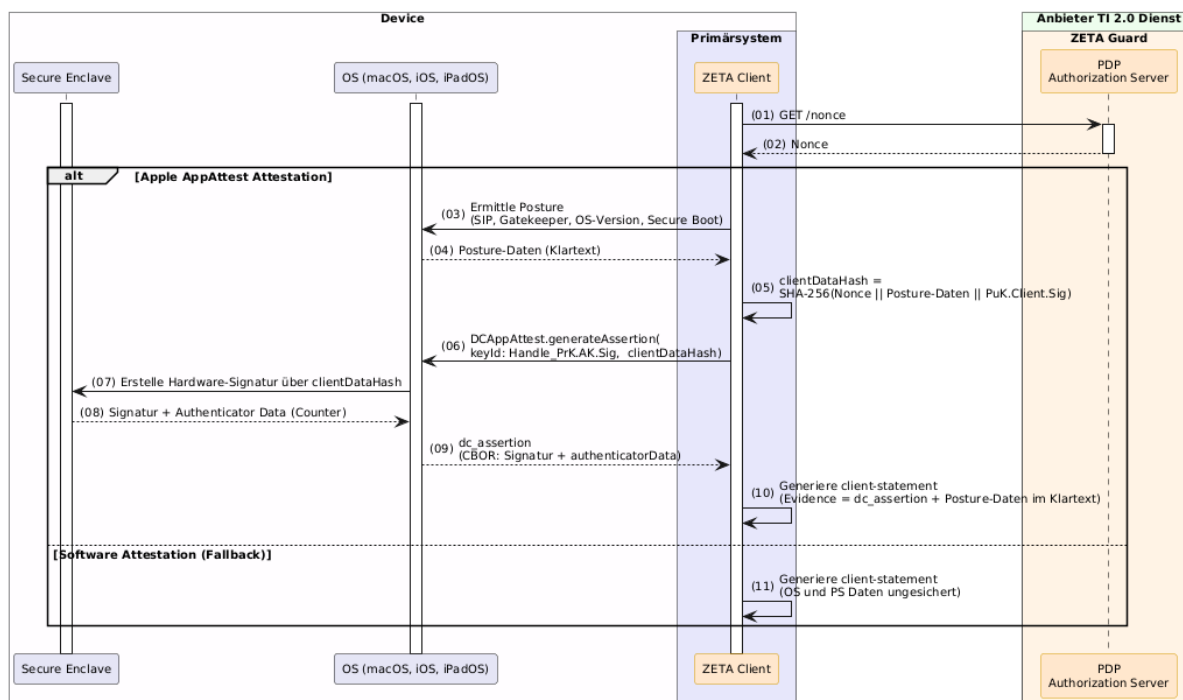


Abbildung 11 Abb-ZETA-Client-Statement-mit-Apple-AppAttest

1841

1842

1843

Vorbereitungsphase: Nonce-Bezug

Um Replay-Angriffe zu verhindern und die Frische des Integritätsnachweises zu garantieren, benötigt der Client eine kryptografische Challenge vom Server.

(01) GET /nonce: Der ZETA Client fragt eine neue Nonce beim ZETA Guard (Authorization Server) an.

(02) Nonce: Der ZETA Guard generiert eine zufällige Nonce und liefert diese zurück.

1850

Primärer Pfad: Apple AppAttest Attestation (Hardwaregestützt)

Dieser Pfad wird durchlaufen, wenn das Gerät über eine Secure Enclave verfügt und das App Attest Framework bei der initialen Registrierung (DCR) erfolgreich eingerichtet wurde.

(03) Ermittle Posture: Der ZETA Client fragt über native Apple-APIs sicherheitsrelevante Systemparameter ab. Dazu gehören beispielsweise der Status der System Integrity Protection (SIP), Gatekeeper-Einstellungen, die exakte OS-Version und der Status des Secure Boots.

(04) Posture-Daten (Klartext): Das Betriebssystem übergibt diese Systemdaten im Klartext an den ZETA Client.

(05) Hash-Berechnung (clientDataHash): Dies ist der zentrale kryptografische Sicherheitsanker. Der ZETA Client berechnet einen SHA-256 Hash aus der Verkettung der serverseitigen Nonce, den ermittelten Posture-Daten und seinem öffentlichen Client-Schlüssel (PuK.Client.Sig).

(Architektur-Hinweis: Dadurch werden die Frische des Requests, der Systemzustand und die Identität des Clients untrennbar aneinander gebunden).

(06) Assertion anfordern: Der Client ruft die API DCAppAttest.generateAssertion auf. Er übergibt das Handle seines hardwaregebundenen Attestation Keys (Handle_PrK.AK.Sig) sowie den soeben berechneten clientDataHash.

(07) Hardware-Signatur: Das Betriebssystem leitet den Hash an die Secure Enclave weiter. Die Secure Enclave signiert diesen Hash mit dem privaten Attestation Key.

(08) Rückgabe der Signatur & Metadaten: Die Secure Enclave gibt die Signatur sowie zusätzliche Authenticator-Daten (insbesondere einen monoton steigenden Counter zum

1872

1873 Schutz vor Klon-Angriffen) an das OS zurück.
1874 (09) dc_assertion (CBOR): Das OS verpackt die Signatur und die Authenticator-Daten in
1875 ein standardisiertes CBOR-Objekt (dc_assertion) und reicht es an den ZETA Client weiter.
1876 (10) Generiere client_statement: Der ZETA Client baut das finale client_statement
1877 (Evidence) für den Request an den ZETA Guard zusammen. Es besteht aus der
1878 kryptografischen dc_assertion und den Posture-Daten im Klartext.
1879 (Der ZETA Guard kann später den Hash aus der ihm bekannten Nonce, den Klartext-
1880 Posture-Daten und dem registrierten Client-Key selbst berechnen und die Hardware-
1881 Signatur in der dc_assertion validieren).

Alternativer Pfad: Software Attestation (Fallback)

1882 Dieser Pfad dient als Rückfallebene, falls keine Hardware-Unterstützung vorliegt.
1883 (11) Generiere client_statement (Software): Der ZETA Client sammelt die OS- und
1884 Primärsystem-Daten (PS-Daten) und fügt sie ungesichert in das Statement ein. Da hierbei
1885 keine Signatur durch eine Secure Enclave stattfindet, wird die Policy Engine des ZETA
1886 Guards diesen Request mit einem entsprechend niedrigen Trust-Score bewerten.
1887

1888

1889 *5.3.1.6.3 Token Exchange mit Attestation*

1890 Das folgende Sequenzdiagramm beschreibt den zentralen OAuth Token Exchange
1891 Authentifizierungs- und Autorisierungsprozess eines ZETA Clients am ZETA Guard
1892 Authorization Server. Der Prozess verknüpft die Nutzer-/Institutions-Authentifizierung
1893 (mittels SM(C)-B) mit der Client-Authentifizierung (Client Assertion & Evidence) und
1894 bindet die resultierenden Tokens kryptografisch an die aktuelle Sitzung (DPoP).

1895 Das Subject Token zur Authentifizierung der Institution ist nach Schema [subject-token-
1896 smb.yaml] festgelegt. Die Client Assertion zur Authentifizierung des Clients ist in [client-
1897 assertion-jwt.yaml] festgelegt.

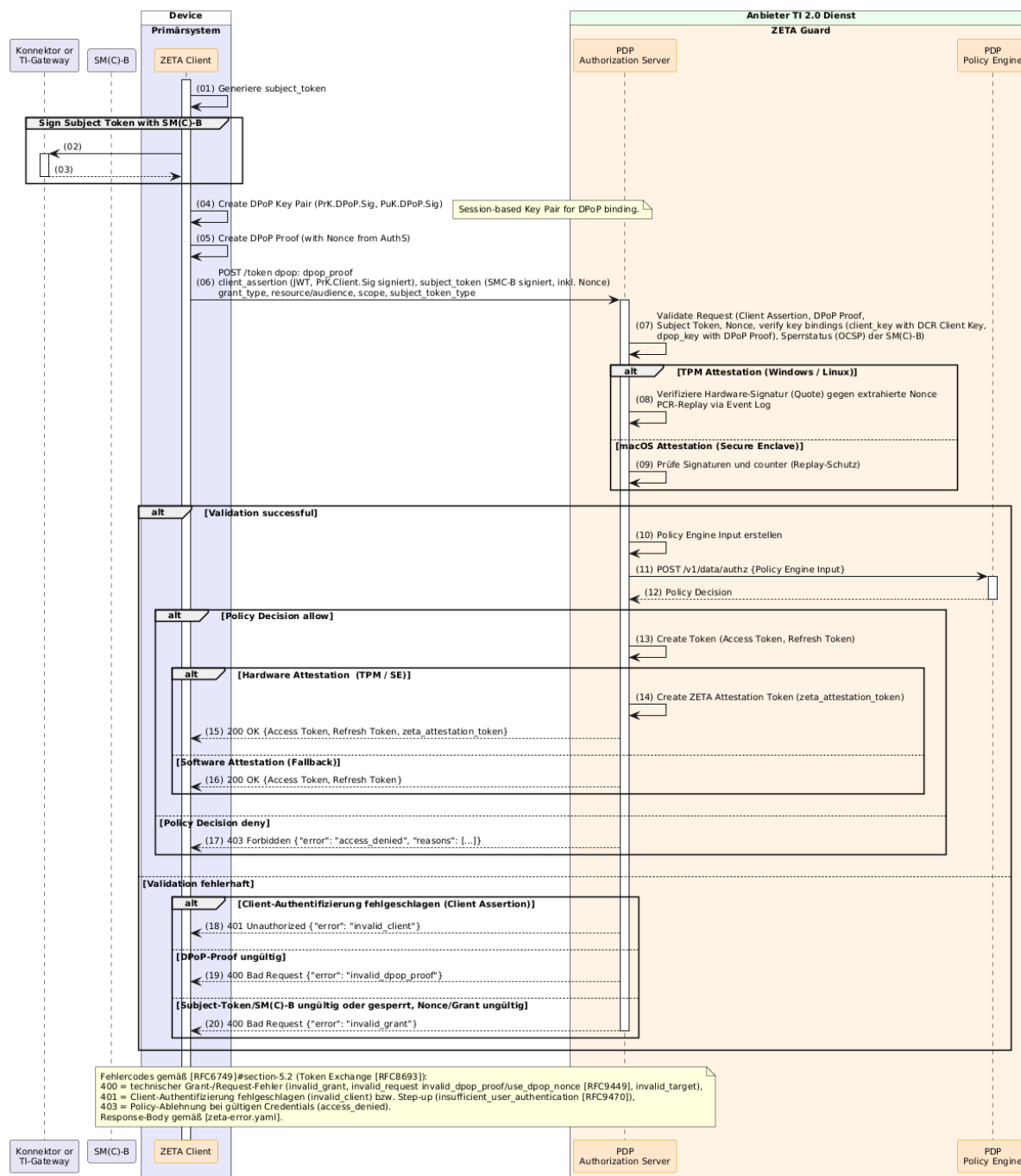


Abbildung 12 Abb-ZETA-Token-Exchange-mit-Attestation

Das Diagramm ist in mehrere logische Phasen und (verschachtelte) Fallunterscheidungen (alt-Blöcke) unterteilt.

Phase 1: Vorbereitung der Tokens und Schlüssel (Schritte 01 - 05)

(01 - 03) Subject Token Erstellung: Der ZETA Client generiert zunächst ein subject_token. Um die Identität des Nutzers bzw. der Institution (z.B. einer Arztpraxis in der Telematikinfrastruktur) nachzuweisen, wird dieses Token an die lokale Smartcard/Sicherheitsmodul (SM(C)-B) gesendet, dort signiert und an den Client zurückgegeben.

(04) DPoP Key Pair: Der Client generiert ein temporäres, sitzungsbasiertes Schlüsselpaar (PrK.DPoP.Sig, PuK.DPoP.Sig). Dieses dient dem Demonstrating Proof-of-Possession

1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909

1910 (DPoP) – einem Mechanismus, der verhindert, dass gestohlene Access Tokens von
1911 Angreifern genutzt werden können.
1912 (05) DPoP Proof: Der Client erstellt mit dem privaten DPoP-Schlüssel und einer vom
1913 Server stammenden Nonce einen DPoP Proof.

1914 **Phase 2: Der Token-Request (Schritt 06)**

1915 (06) POST /token: Der ZETA Client sendet die eigentliche Anfrage an den PDP
1916 Authorization Server. Dieser Request enthält ein umfassendes Sicherheitspaket:

- 1917 • Im Body: den Parameter resource [RFC8707] (Endpunkt-URL der Zielressource aus
1918 [opr-well-known.yaml]) sowie scope. Aus Gründen der Abwärtskompatibilität kann
1919 stattdessen der Parameter audience verwendet werden (siehe 5.4.2.5).
- 1920 • Im Header (dpop): Den in (05) erstellten dpop_proof.
- 1921 • client_assertion: Ein JWT, signiert mit dem privaten Client-Schlüssel (PrK.Client.Sig),
1922 um den Client selbst zu authentifizieren.
- 1923 • subject_token: Das von der SMC-B signierte Token inkl. Nonce aus Schritt (03).
- 1924 • [Evidence]: Optional/Bedingt mitgeschickte Integritäts- und Zustandsdaten des Geräts
1925 (wie in den vorherigen Diagrammen ermittelt).

1926 **Phase 3: Validierung durch den ZETA Guard (Schritte 07 - 09)**

1927 (07) Basis-Validierung: Der Authorization Server prüft die Client Assertion, den DPoP
1928 Proof, das Subject Token, die Nonces und gleicht ab, ob die Schlüssel zu der vorherigen
1929 Registrierung (DCR) passen.

1930 Hardware-Attestierungsprüfung (Innerer alt-Block): Falls Evidence mitgeliefert wurde,
1931 wird diese hardware-spezifisch geprüft:

1932 (08) [TPM Attestation (Windows / Linux)]: Die TPM-Hardware-Signatur (Quote) wird
1933 verifiziert und die Systemzustände (PCRs) werden anhand des Event Logs nachvollzogen.

1934 (09) [macOS Attestation (Secure Enclave)]: Die Apple App Attest Signaturen werden
1935 geprüft und der Counter ausgelesen, um Replay-Attacken (Wiederholungsangriffe) zu
1936 verhindern.

1937 **Phase 4: Policy-Entscheidung und Response (Schritte 10 - 18)**

1938 Ab hier spaltet sich der Ablauf je nach Erfolg der Validierung auf (äußerer alt-Block):

1939 **Fall A: [Validation successful] (Schritte 10 - 16)**

1940 Wenn die technische und kryptografische Validierung (07-09) erfolgreich war, muss
1941 inhaltlich über den Zugriff entschieden werden.

1942 (10 - 11): Der Authorization Server bereitet die validierten Daten auf (Policy Engine Input)
1943 und sendet sie via POST /v1/data/authz an die PDP Policy Engine.

1944 (12): Die Policy Engine evaluiert die Zugriffsregeln und liefert eine Entscheidung (Policy
1945 Decision) zurück.

1946 **Unterfall A1: [Policy Decision allow] (Schritte 13 - 16)**

1947 Die Policy Engine erlaubt den Zugriff.

1948 (13): Der Server erstellt die Standard-OIDC-Tokens (Access Token, Refresh Token). Diese
1949 sind durch den Request an den DPoP-Schlüssel gebunden. Der aud-Claim des Access
1950 Tokens wird dabei aus der Policy Engine Decision übernommen (Contract v2, ver: 2); bei
1951 einem Legacy-Request über audience wird der Wert verbatim übernommen (ver: 1).

1952 **Unterscheidung nach Attestierungs-Level (Innerster alt-Block):**

1953 (14 - 15) [Hardware Attestation (TPM / SE)]: Wenn das Gerät seinen Zustand erfolgreich
1954 durch Hardware (TPM/Secure Enclave) bewiesen hat, generiert der Server zusätzlich ein
1955 ZETA Attestation Token (zeta_attestation_token). Er antwortet mit 200 OK und liefert
1956 Access Token, Refresh Token und das ZETA attestation Token an den Client.

1957 (16) [Software Attestation (Fallback)]: Wenn das Gerät nur auf Software-Basis attestiert
1958 wurde, erhält der Client zwar Zugriff (200 OK), bekommt aber nur das Access Token und
1959 das Refresh Token (kein Attestation Token).

1960 **Unterfall A2: [Policy Decision deny] (Schritt 17)**
1961 (17): Wenn die Policy Engine den Zugriff verweigert (z.B. weil der Gerätezustand nicht
1962 den Sicherheitsrichtlinien entspricht, obwohl die Signaturen gültig sind), antwortet der
1963 Server dem Client mit 403 Forbidden (error: access_denied).

1964 **Fall B: [Validation fehlerhaft] (Schritt 18)**
1965 (18)-(20): Wenn bereits die technische Prüfung in Schritt (07) bis (09) fehlschlägt (z.B.
1966 ungültige Signatur der SMC-B, DPoP-Proof falsch, Attestation manipuliert), bricht der
1967 Server sofort ab und antwortet dem ZETA Client mit

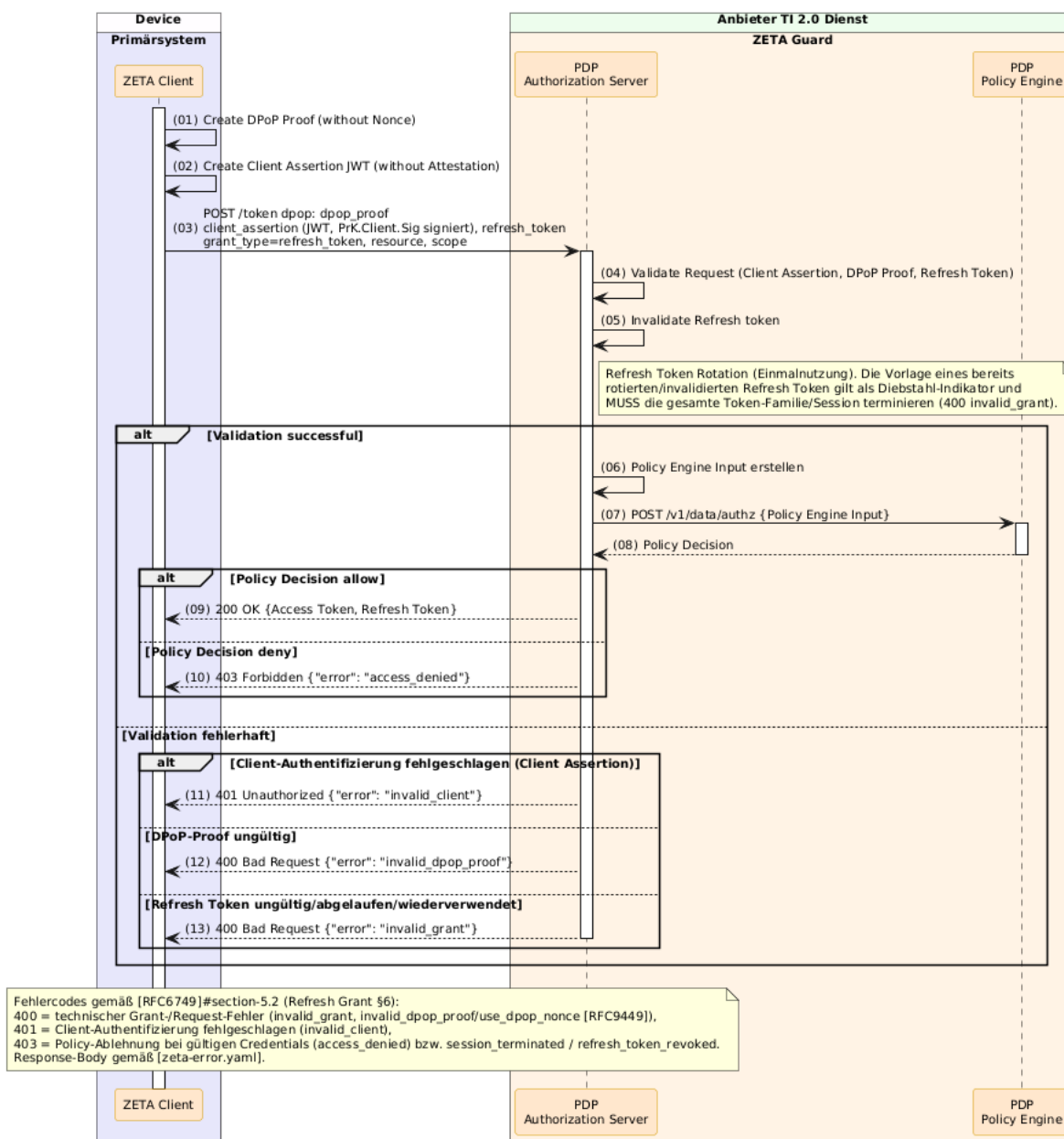
- 1968 • 400 Bad Request (invalid_grant bei ungültiger/gesperrter SMC-B bzw. abgelaufenem
1969 Grant, invalid_dpop_proof bei fehlerhaftem DPoP-Proof); Client-
1970 Authentifizierungsfehler
- 1971 • 401 (invalid_client).

1972 Die Policy Engine wird in diesem Fall nicht befragt.

1973 **A_29376 -ZETA, Ablauf Token Exchange mit Attestation**
1974 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
1975 *Token-Exchange-mit-Attestation* unterstützen. [<=]

1976 *5.3.1.6.4 Token Request mit grant_type=refresh_token*

1977 Das folgende Sequenzdiagramm beschreibt den Refresh-Token-Prozess (Token Renewal)
1978 eines ZETA Clients am ZETA Guard Authorization Server. Es zeigt, wie ein Client, der
1979 bereits eine Sitzung etabliert hat, mit einem gültigen Refresh Token neue Zugriffs-Token
1980 (Access Tokens) anfordert, ohne den vollständigen (und zeitaufwendigen) Hardware-
1981 Attestierungs- und Smartcard-Signaturprozess erneut durchlaufen zu müssen.



1982

1983

1984

Abbildung 13 Abb-ZETA-Token-Request-mit-Refresh-Token

1985 Das Diagramm gliedert sich in die Vorbereitung der Anfrage, die Validierung durch den
 1986 Server und die finale Autorisierungsentscheidung.

1987 **Phase 1: Vorbereitung und Token-Request**

1988 (01) DPoP Proof: Der ZETA Client erstellt einen DPoP Proof (Demonstrating Proof-of-
 1989 Possession). Bemerkenswert ist hier der Hinweis (without Nonce): Um einen zusätzlichen
 1990 Netzwerk-Roundtrip zu sparen, wird beim Refresh oft zunächst auf eine frische Nonce des
 1991 Servers verzichtet.

1992 (02) Client Assertion: Der Client erstellt ein JWT (JSON Web Token) zur eigenen
 1993 Authentifizierung (Client Assertion). Hier geschieht dies explizit (without Attestation), da
 1994 der Gerätezustand bei einem bloßen Refresh nicht zwingend neu aufwendig hardware-
 1995 attestiert wird.

1996 (03) POST /token: Der Client sendet die Anfrage an den PDP Authorization Server. Dieser

1997 Request enthält im Header den dpop_proof und in der Payload die signierte
1998 client_assertion sowie das bisherige refresh_token.

1999 **Phase 2: Validierung durch den ZETA Guard**

2000 (04) Basis-Validierung: Der Server prüft die kryptografische Gültigkeit der Client
2001 Assertion, des DPOP Proofs und verifiziert, ob das eingereichte Refresh Token gültig und
2002 dem Client zugeordnet ist.

2003 (05) Token Invalidation: Der Server invalidiert das eingereichte Refresh Token. Dies
2004 implementiert das Konzept der Refresh Token Rotation – ein Refresh Token kann nur
2005 exakt einmal benutzt werden. Bei einem erneuten Versuch würde dies als Diebstahl-
2006 Indikator gewertet.

2007 **Phase 3: Policy-Entscheidung und Response**

2008 Der weitere Ablauf spaltet sich auf (äußerer alt-Block), je nachdem, ob die technische
2009 Validierung in Phase 2 erfolgreich war.

2010 **Fall A: [Validation successful]**

2011 Wenn die Token gültig sind und die SMC-B nicht gesperrt ist, wird die inhaltliche
2012 Autorisierung geprüft.

2013 (06 - 07): Der Authorization Server bereitet die Daten auf (Policy Engine Input) und
2014 befragt via POST /v1/data/authz die PDP Policy Engine.

2015 (08): Die Policy Engine liefert ihre Entscheidung (Policy Decision) zurück.

2016 **Unterfall A1: [Policy Decision allow]**

2017 (09): Die Policy Engine erlaubt die Verlängerung der Sitzung. Der Server antwortet dem
2018 Client mit 200 OK und stellt ein brandneues Paar aus Access Token und einem neuen
2019 Refresh Token aus.

2020 **Unterfall A2: [Policy Decision deny]**

2021 (10): Verweigert die Policy Engine den Zugriff (z.B. weil sich übergeordnete Zugriffsregeln
2022 geändert haben), antwortet der Server mit 403 (access_denied). Der Client muss sich
2023 dann ggf. komplett neu authentifizieren.

2024 **Fall B: [Validation fehlerhaft]**

2025 (11)-(13): Schlägt bereits die technische Prüfung (Schritte 04-06) fehl – beispielsweise
2026 weil das Refresh Token abgelaufen ist, der DPOP-Proof falsch ist oder die SMC-B
2027 zwischenzeitlich gesperrt wurde – bricht der Server den Vorgang ab. Die Policy Engine
2028 wird nicht befragt und der Client erhält sofort ein

2029 • 400 (invalid_grant / invalid_dpop_proof); reuse eines rotierten Refresh Token

2030 • 400 invalid_grant und Terminierung der Token-Familie. Client-Auth-Fehler

2031 • 401 invalid_client.

2032 Die Sitzung ist damit beendet.

2033 **A_29377 -ZETA, Ablauf Token Exchange mit Refresh Token**

2034 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-
2035 Token-Request-mit-Refresh-Token* unterstützen. [<=]

2036 **5.3.1.7 Zugriff auf den Resource Server**

2037 Voraussetzung für den Zugriff auf eine geschützte Ressource (Resource Server) durch
2038 einen Client in der ZETA-Architektur ist, dass der vorgeschaltete Authentifizierungs- und
2039 Autorisierungsprozess bereits erfolgreich war und der ZETA Client über ein gültiges
2040 Access Token verfügt.

2041

2042 Durch die Auswertung des OAuth Protected Resource Well-known JSON Dokuments kann
2043 der ZETA Client erkennen, ob eine ZETA/ASL Verbindung aufgebaut werden muss und
2044 welche Token benötigt werden.

2045 5.3.1.7.1 Zugriff auf den Resource Server mit ZETA/ASL

2046 Das folgende Diagramm zeigt, wie eine Anfrage vom Fach-Client über den ZETA Client
2047 zum Resource Server geleitet wird.

2048 Das Hauptmerkmal dieses Diagramms ist die Nutzung des ZETA/ASL (Application Security
2049 Layer), welcher eine zusätzliche Verschlüsselungsschicht auf Applikationsebene darstellt.
2050 Das Diagramm zeigt in einem großen alt-Block zwei unterschiedliche Architektur-Ansätze,
2051 je nachdem, wo dieser verschlüsselte Tunnel endet (terminiert). Je nach Einstellung im
2052 OAuth Protected Resource Well-known JSON Dokument wird der Tunnel entweder am
2053 ZETA Guard HTTP Proxy oder am Resource Server terminiert.

2054 Vorbedingung: Der ZETA Client hat bereits erfolgreich den Authentifizierungsprozess
2055 durchlaufen und besitzt ein gültiges Access Token für den Resource Server und ggf. für
2056 den ASL Tunnel.

2057

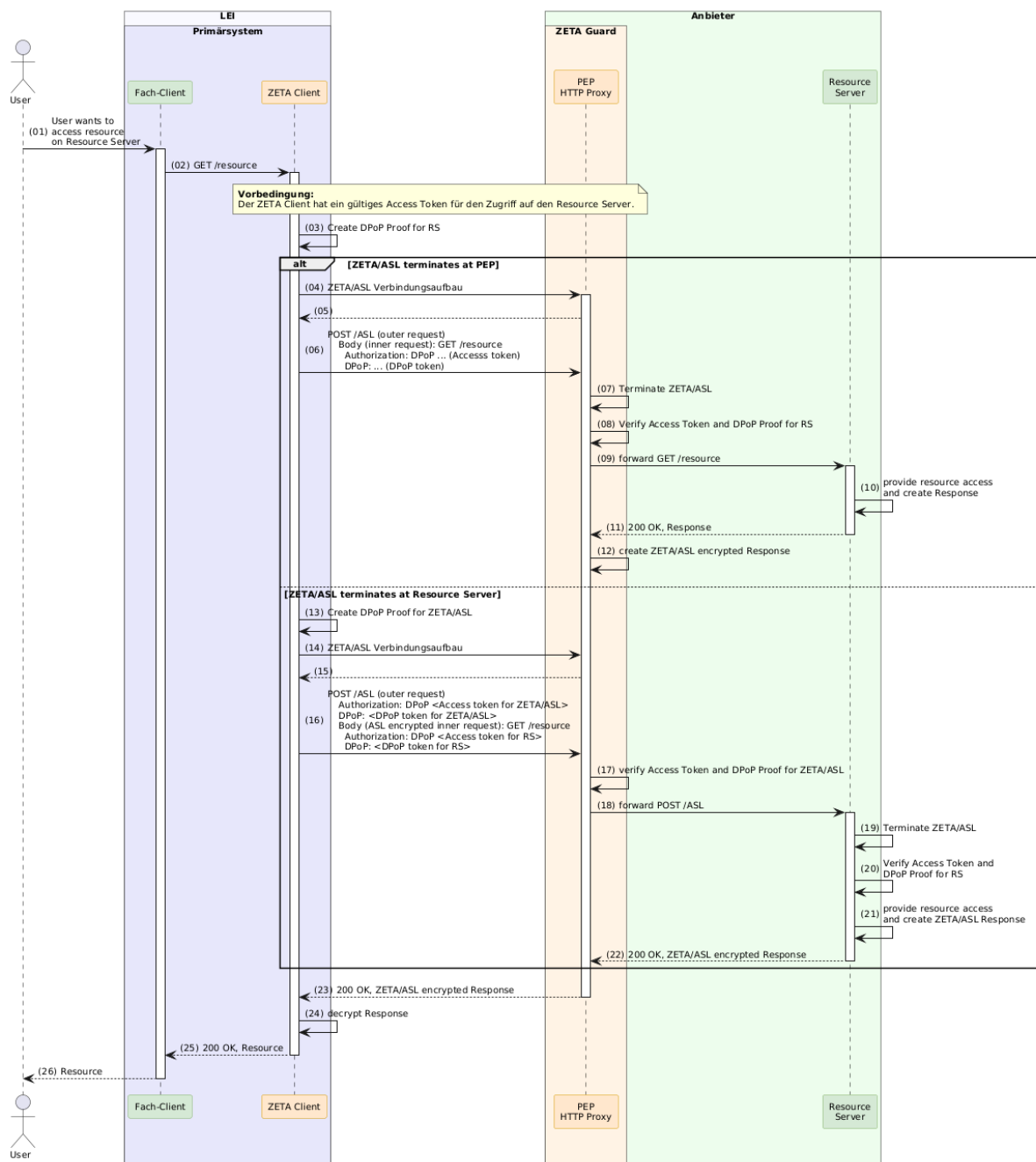


Abbildung 14 Abb-ZETA-Zugriff-auf-RS-mit-ASL

2058

2059

2060

2061

2062

2063

2064

2065

2067

2068

2069

2070

(01)-(02): Der Nutzer (User) möchte auf eine Ressource zugreifen. Das Primärsystem (Fach-Client) sendet eine normale, unverschlüsselte HTTP-Anfrage (GET /resource) an den lokalen ZETA Client.
 (03): Der ZETA Client erstellt einen kryptografischen DPoP-Proof (Demonstrating Proof-of-Possession), der an die URL des Resource Servers gebunden ist, um das Access Token vor Diebstahl/Replay zu schützen.

Fall A: [ZETA/ASL terminates at PEP]

In diesem Szenario dient ASL als sicherer Tunnel zwischen dem lokalen Client und dem Netzübergang (PEP) des Anbieters. Im internen Netz des Anbieters (vom PEP zum Resource Server) fließen die Daten nur mit TLS verschlüsselt.

2071 (04)-(05): Es wird ein ZETA/ASL-Verbindungsaufbau zwischen dem ZETA Client und dem
2072 PEP HTTP Proxy initiiert.
2073 (06): Der ZETA Client verpackt die eigentliche Anfrage. Er sendet einen "äußeren"
2074 Request (POST /ASL) an den PEP. Im verschlüsselten Body (inner request) liegt der
2075 eigentliche GET /resource-Aufruf, zusammen mit dem Access Token und dem in (03)
2076 generierten DPoP-Proof.
2077 (07): Der PEP terminiert den ASL-Tunnel. Das bedeutet, er entschlüsselt den Body.
2078 (08): Der PEP verifiziert nun das entpackte Access Token und den DPoP-Proof.
2079 (09): Ist die Prüfung erfolgreich, leitet der PEP den ursprünglichen Klartext-Request
2080 (GET /resource) an den eigentlichen Resource Server weiter.
2081 (10 - 11): Der Resource Server verarbeitet die Anfrage und sendet die Response an den
2082 PEP zurück.
2083 (12): Der PEP verschlüsselt diese Antwort wieder mit ASL, um sie für den sicheren
2084 Rückweg über das Internet vorzubereiten.

2085 **Fall B: [ZETA/ASL terminates at Resource Server]**

2086 In diesem hochsicheren Szenario wird der Payload Ende-zu-Ende verschlüsselt. Der PEP
2087 agiert nur als Router und kann den Inhalt der Fachnachricht nicht mitlesen.
2088 (13): Der ZETA Client muss hier einen zweiten DPoP-Proof generieren. Einer ist für den
2089 Resource Server (für die Fachdaten), der neue ist für den PEP (um überhaupt den ASL-
2090 Endpunkt nutzen zu dürfen).
2091 (14)-(15): Ein ZETA/ASL-Verbindungsaufbau findet logisch zwischen dem ZETA Client und
2092 dem Resource Server statt.
2093 (16): Der Request (POST /ASL) ist komplexer verschachtelt:
2094 Auf der äußeren Ebene (für den PEP sichtbar) liegen ein Access Token und ein DPoP-Proof,
2095 die nur für den Zugang zum ASL-Endpunkt berechtigen.
2096 Der verschlüsselte Body enthält den eigentlichen Request (GET /resource) samt eigenen
2097 Autorisierungsdaten (Access Token und DPoP-Proof für den RS).
2098 (17)-(18): Der PEP prüft nur die äußeren Header. Da er den Body nicht entschlüsseln
2099 kann, leitet er den POST /ASL Request ungesehen an den Resource Server weiter.
2100 (19): Erst der Resource Server terminiert den ASL-Tunnel und entschlüsselt die Nachricht.
2101 (20): Der Resource Server prüft nun die inneren Header (Access Token und DPoP-Proof für
2102 die Ressource).
2103 (21): Er verarbeitet die Anfrage, erzeugt die Antwort und verschlüsselt diese direkt wieder
2104 in eine ASL-Response.

2105 (22)-(23): Unabhängig davon, ob der PEP oder der Resource Server die ASL-
2106 Verschlüsselung der Antwort übernommen hat, empfängt der ZETA Client nun die
2107 verschlüsselte ASL-Response.
2108 (24): Der ZETA Client entschlüsselt die Antwort lokal.
2109 (25): Er reicht die Response im Klartext an den Fach-Client weiter.
2110 (26): Der Fach-Client stellt dem User die angeforderten Daten dar.

2111 **A_29380 -ZETA, Ablauf Zugriff auf den RS mit ZETA/ASL**

2112 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
2113 *Zugriff-auf-RS-mit-ASL* unterstützen. [<=]

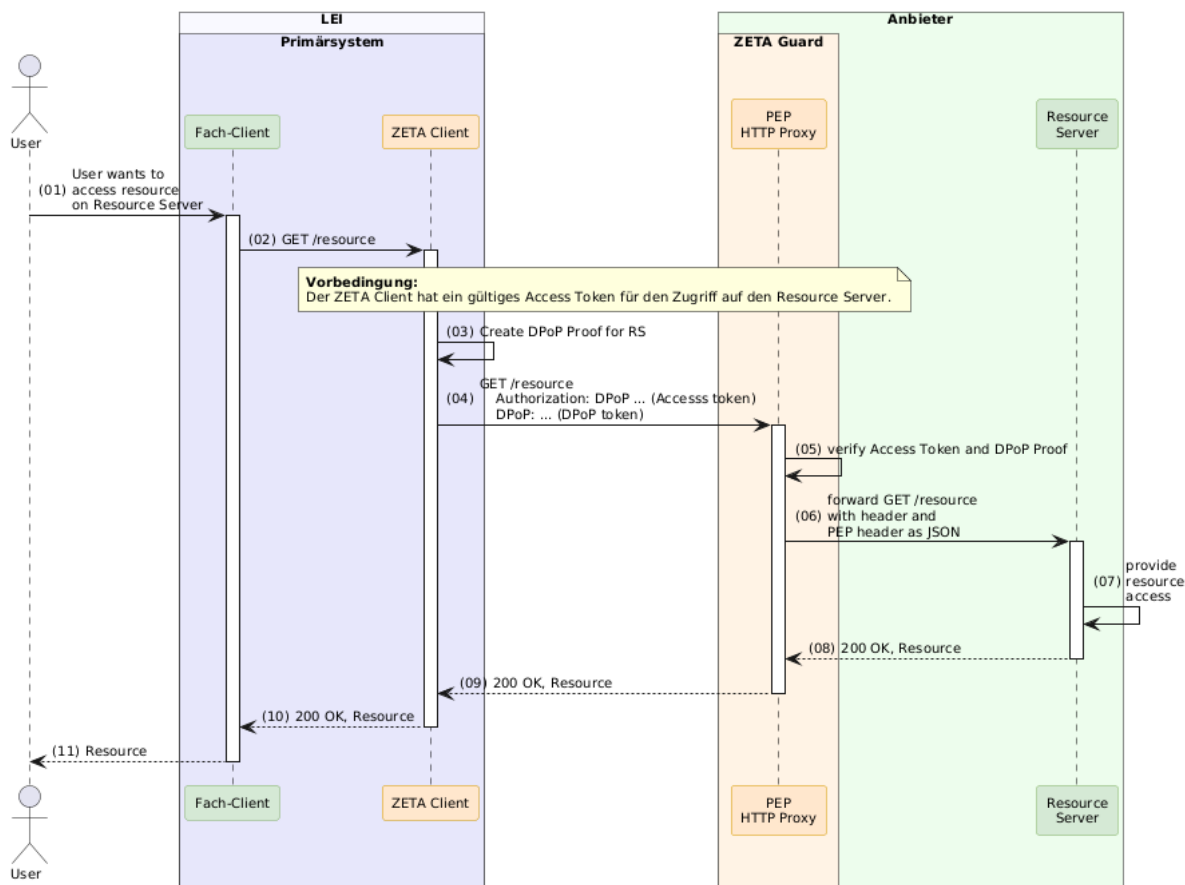
2114 *5.3.1.7.2 Zugriff auf den Resource Server ohne ZETA/ASL*

2115 Das folgende Sequenzdiagramm beschreibt den direkten Zugriff auf eine geschützte
2116 Ressource (Resource Server) durch einen Client in der ZETA-Architektur.

2117 Er zeigt das Routing über den PEP, bei dem die Absicherung primär über tokenbasierte
2118 HTTP-Header (Authorization, DPoP) erfolgt, ohne dass ein zusätzlicher ASL-
2119 Verschlüsselungstunnel aufgebaut wird.

2120 Vorbedingung: Der ZETA Client hat im Vorfeld bereits erfolgreich einen
2121 Authentifizierungsprozess durchlaufen und verfügt über ein gültiges Access Token für den
2122 entsprechenden Resource Server.

2123



2124

2125

Abbildung 15 Abb-ZETA-Zugriff-auf-RS-ohne-ASL

2126 (01): Der Endnutzer (User) möchte auf eine bestimmte Ressource zugreifen und
 2127 interagiert dafür mit seiner Fachanwendung (Fach-Client, z. B. ein
 2128 Praxisverwaltungssystem).
 2129 (02): Der Fach-Client sendet eine reguläre, ungesicherte HTTP-Anfrage (GET /resource)
 2130 an den lokal installierten ZETA Client. Der Fach-Client selbst muss sich nicht um die
 2131 komplexe ZETA-Kryptografie kümmern.

2132 (03): Der ZETA Client bereitet die Anfrage vor. Er erstellt einen DPoP Proof
 2133 (Demonstrating Proof-of-Possession). Dies ist eine kryptografische Signatur, die beweist,
 2134 dass der Client im Besitz des privaten Schlüssels ist, an den das Access Token gebunden
 2135 wurde. Dieser Proof wird speziell für die Ziel-URL des Resource Servers (RS) generiert
 2136 (Schutz vor Replay-Attacks).

2137 (04): Der ZETA Client leitet den GET /resource Request an den PEP HTTP Proxy (den ZETA
 2138 Guard des Anbieters) weiter. Dabei fügt er zwei Sicherheits-Header hinzu:

- 2139 • Authorization: DPoP ...: Enthält das eigentliche Access Token.
- 2140 • DPoP:: Enthält den in Schritt 03 generierten DPoP-Token (Proof).

2141 (05): Der PEP HTTP Proxy empfängt die Anfrage und verifiziert die Gültigkeit des Access
 2142 Tokens und des DPoP Proofs.

2143 (06): Verläuft die Prüfung erfolgreich, leitet der PEP den Request in das interne Netz an
 2144 den eigentlichen Resource Server weiter. Um dem Resource Server mitzuteilen, wer
 2145 anfragt, fügt der PEP verifizierte Kontextdaten (Identität, Berechtigungen) in Form von
 2146 ZETA headers as JSON an die Anfrage an.

2147 (07): Der Resource Server verarbeitet die nun vollständig autorisierte Anfrage und stellt

2148 die angeforderten Daten bereit (provide resource access).
2149 (08): Der Resource Server antwortet dem PEP mit einer Response.
2150 (09): Der PEP HTTP Proxy leitet diese Antwort transparent an den ZETA Client zurück.
2151 (10): Der ZETA Client reicht die Response an den Fach-Client durch.
2152 (11): Der Fach-Client nimmt die Daten entgegen und präsentiert die Ressource dem User.

2153 **A_29381 -ZETA, Ablauf Zugriff auf den RS ohne ZETA/ASL**

2154 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
2155 *Zugriff-auf-RS-ohne-ASL* unterstützen. [\leq]

2156 **5.3.2 Abläufe für mobile Clients**

2157 In diesem Unterkapitel werden die Kommunikations- und Attestierungsprozesse für
2158 mobile ZETA Clients (z. B. auf Basis von Android oder iOS) detailliert. Aufgrund der
2159 Architektur mobiler Betriebssysteme werden für den Integritätsnachweis
2160 plattformspezifische Mechanismen wie die Android Key and ID Attestation oder die Secure
2161 Enclave von Apple genutzt. Als Fallback-Lösung ist eine Software-Attestierung
2162 vorgesehen, falls hardwarebasierte Nachweise nicht erbracht werden können. Im
2163 Unterschied zu stationären Systemen erfordert das mobile Umfeld in der Regel eine
2164 interaktive Nutzerauthentisierung. Daher erfolgt die Autorisierung hier standardisiert über
2165 OpenID Connect (OIDC) in Verbindung mit dem OAuth Authorization Code Flow.

2166 Im Unterschied zu stationären Clients basieren mobile Clients auf plattformintegrierten
2167 Sicherheitsmechanismen und erfordern in der Regel eine interaktive
2168 Nutzerauthentisierung. Während bei stationären Clients TPM/ZAS im Fokus steht, erfolgt
2169 der Integritätsnachweis bei mobilen Clients über plattformabhängige
2170 Attestierungsverfahren. Die Autorisierung erfolgt über den OIDC Authorization Code Flow
2171 in Kombination mit OAuth 2.0.

2173 Der Lebenszyklus eines mobilen Clients gliedert sich analog zu stationären Clients in
2174 folgende Phasen:

- 2175 1. Installation und Initialisierung
- 2176 2. Schlüsselgenerierung
- 2177 3. Service Discovery
- 2178 4. Key Preparation und Attestation
- 2179 5. Dynamic Client Registration (DCR)
- 2180 6. Authentifizierung (OIDC Flow)
- 2181 7. Token Exchange und Zugriff
- 2182 8. Session-Erneuerung

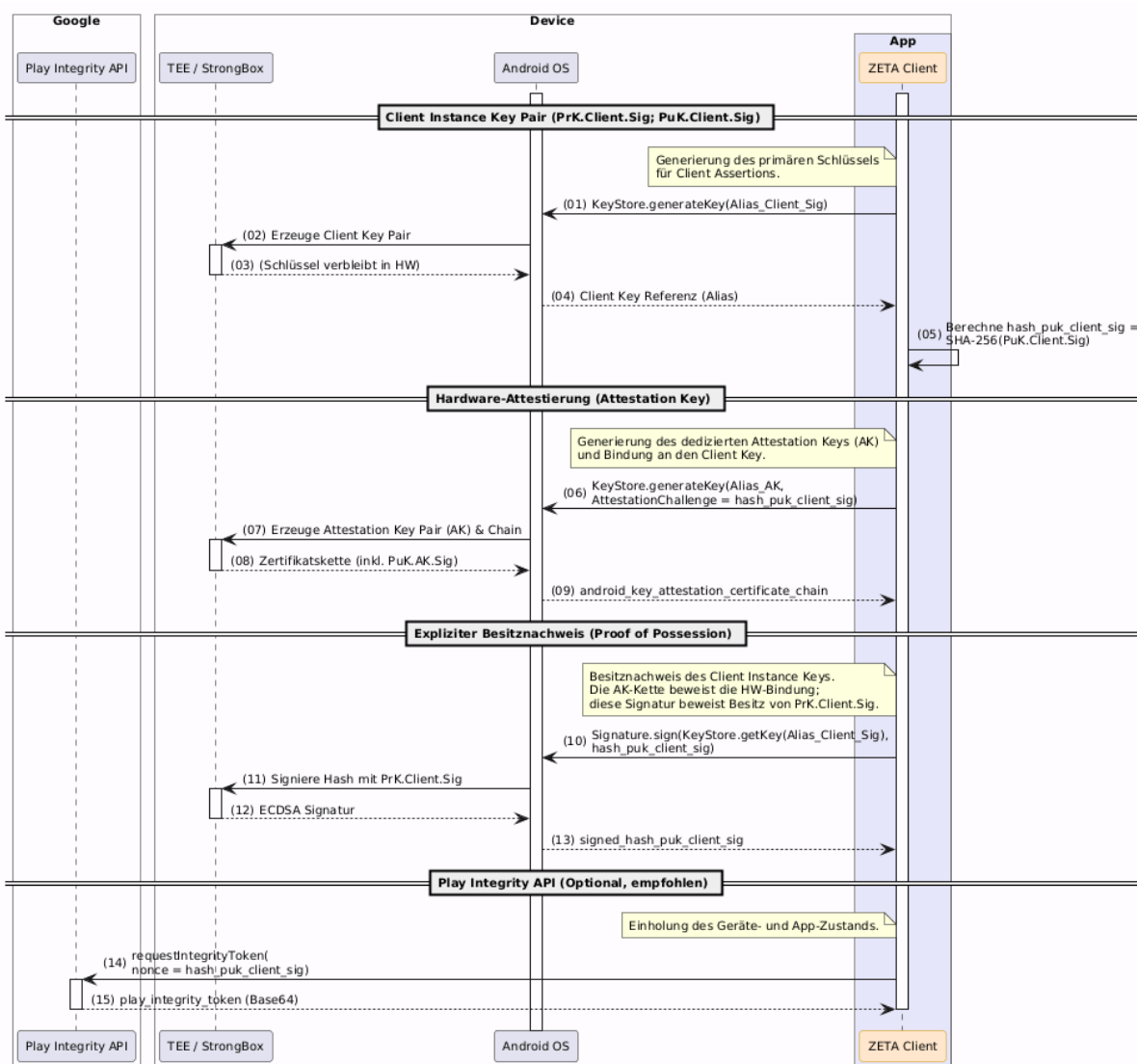
2183 Hardware-Attestation SOLL bevorzugt eingesetzt werden und Software-Attestation dient
2184 als Fallback-Mechanismus.

2186 **5.3.2.1 Initialisierung und Schlüsselgenerierung**

2187 Nach der Installation der mobilen Anwendung wird beim ersten Start ein initialer Satz
2188 kryptografischer Schlüssel erzeugt, der als langlebige Identität der Client-Instanz dient.
2189 Im Gegensatz zu stationären Clients erfolgt die Schlüsselgenerierung vollständig
2190 innerhalb der durch das mobile Betriebssystem bereitgestellten Sicherheitsmechanismen.

2191 5.3.2.1.1 Android TEE oder Strongbox

2192 Das Sequenzdiagramm zeigt den grundlegenden Prozess der kryptografischen
 2193 Schlüsselgenerierung und Hardware-Attestierung auf einem Android-Gerät. Es
 2194 veranschaulicht das Zusammenspiel zwischen der App (mit ZETA Client), dem
 2195 Betriebssystem (Android OS) und dem isolierten Hardware-Sicherheitsmodul (TEE /
 2196 StrongBox). Es nutzt ein Zwei-Schlüssel-Modell (beide in der Hardware verankert) und
 2197 integriert die externe Google Play Integrity API.



2198
 2199 **Abbildung 16 Abb-ZETA-Schlüsselgenerierung-Android**

2200 **1. Generierung des Client Instance Key Pairs**

2201 In dieser Phase erzeugt die App den primären kryptografischen Schlüssel, der später für
 2202 die fortlaufende Authentisierung (Client Assertions) genutzt wird.
 2203 (01) Schlüsselgenerierung anfordern: Der ZETA Client ruft die native Android-API
 2204 KeyStore.generateKey auf, um ein asymmetrisches Schlüsselpaar (Alias: Alias_Client_Sig)
 2205 zu erzeugen.
 2206 (02) Erzeugung in der Hardware: Das Android OS leitet den Befehl an das Hardware-
 2207 Sicherheitsmodul (TEE oder StrongBox) weiter.
 2208 (03) Schlüssel verbleibt in HW: Die StrongBox erzeugt das Schlüsselpaar (PrK.Client.Sig,
 2209 PuK.Client.Sig). Der private Schlüssel (PrK.Client.Sig) ist hardwaregebunden und kann

2210 nicht in den Arbeitsspeicher des Betriebssystems oder der App ausgelesen werden.
2211 (04) Rückgabe der Referenz: Das Android OS gibt dem ZETA Client lediglich eine Referenz
2212 (den Alias) auf diesen Schlüssel zurück.
2213 (05) Hash-Berechnung: Der ZETA Client berechnet den SHA-256 Hash des öffentlichen
2214 Schlüssels (`hash_puk_client_sig = SHA-256(PuK.Client.Sig)`). Dieser Hash dient im
2215 weiteren Verlauf als zentrales kryptografisches Bindeglied (Key-Binding).

2216 **2. Hardware-Attestierung durch den Attestation Key**

2217 Um dem ZETA Guard (AuthS) später beweisen zu können, dass der Client-Schlüssel
2218 tatsächlich in einem sicheren Hardware-Modul liegt, wird ein dedizierter
2219 Attestierungsschlüssel (AK) erzeugt.
2220 (06) AK anfordern mit Challenge: Der ZETA Client fordert die Generierung eines weiteren
2221 Schlüssels an (Alias_AK). Das entscheidende Sicherheitsmerkmal: Der zuvor berechnete
2222 Hash des Client-Schlüssels (`hash_puk_client_sig`) wird als AttestationChallenge an das OS
2223 übergeben.
2224 (07) AK-Generierung & Zertifikatserstellung: Die StrongBox erzeugt das Attestation Key
2225 Pair. Gleichzeitig erstellt die Hardware ein X.509-Attestierungszertifikat und schreibt die
2226 übergebene AttestationChallenge unveränderlich in die ASN.1-Erweiterung
2227 (KeyDescription) des Zertifikats.
2228 (08) & (09) Rückgabe der Zertifikatskette: Die Hardware liefert die Zertifikatskette
2229 (`android_key_attestation_certificate_chain`), die bis zur Google Root CA reicht, über das
2230 OS an den ZETA Client zurück. (Diese Kette beweist später dem ZETA Guard die
2231 Hardware-Bindung).

2232 **3. Expliziter Besitznachweis / Proof of Possession**

2233 Der Client muss beweisen, dass er den soeben erzeugten privaten Client-Schlüssel auch
2234 tatsächlich physisch kontrolliert.
2235 (10) Signatur anfordern: Der ZETA Client ruft `Signature.sign` auf und nutzt dabei den Alias
2236 des Client Instance Keys (`Alias_Client_Sig`), um den Hash (`hash_puk_client_sig`) zu
2237 signieren.
2238 (11) Signatur in der HW: Das Android OS instruiert die StrongBox, den Hashwert mit dem
2239 privaten Schlüssel (`PrK.Client.Sig`) zu signieren.
2240 (12) & (13) Rückgabe der Signatur: Die erzeugte ECDSA-Signatur
2241 (`signed_hash_puk_client_sig`) wird an den Client zurückgeliefert.

2242 **4. Play Integrity API - Optional, empfohlen**

2243 Zur zusätzlichen Absicherung (Software- und Geräteintegrität) wird ein serverseitiges
2244 Urteil von Google eingeholt.
2245 (14) Integrity Token anfordern: Der ZETA Client ruft die Google Play Integrity API auf
2246 (`requestIntegrityToken`). Auch hier übergibt er den Hash des Client-Schlüssels
2247 (`hash_puk_client_sig`) als nonce. Dadurch wird die Bewertung kryptografisch untrennbar
2248 an diesen spezifischen ZETA Client gebunden (Schutz vor Token-Injection).
2249 (15) Rückgabe des Tokens: Die Play Integrity API liefert das signierte Base64-Token
2250 (`play_integrity_token`) an den Client zurück.

2251 Der ZETA Client verfügt nun über alle notwendigen kryptografischen Artefakte, um die
2252 Dynamic Client Registration (DCR) am ZETA Guard durchzuführen).

2253 **A_29781 -ZETA Client, Android Ablauf zur Schlüsselgenerierung**

2254 Der ZETA Client MUSS den Ablauf gemäß Abbildung *Abb-ZETA-Schlüsselgenerierung-*
2255 *Android* unterstützen. [≤]

2256 *5.3.2.1.2 Apple Secure Enclave*

2257 Das Sequenzdiagramm beschreibt die initiale Generierung des Client Key Pair durch eine
2258 ZETA Client App auf einem Apple-Gerät. Der Vorgang ist für mobile und stationäre Clients
2259 identisch (iOS, iPadOS, macOS).

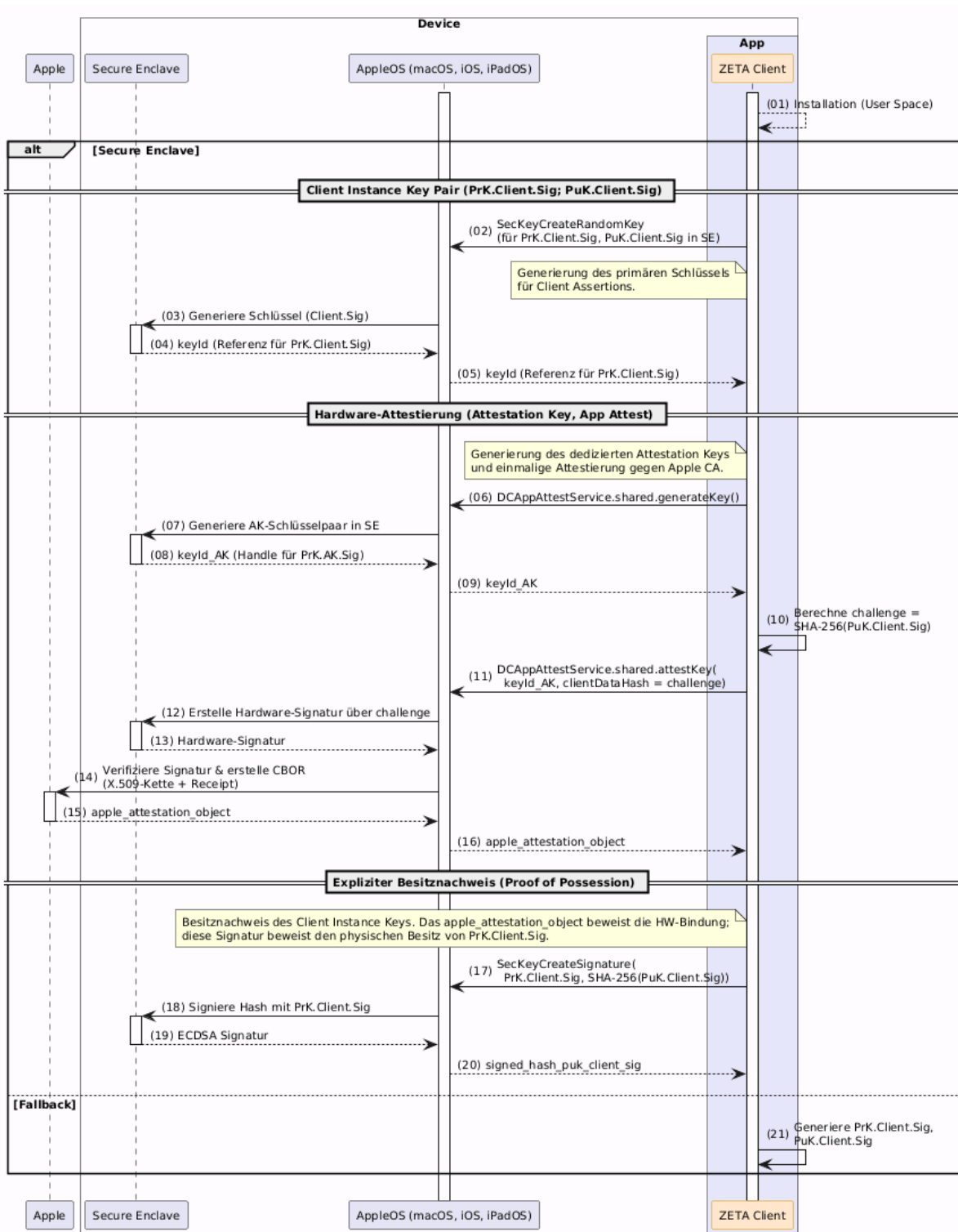


Abbildung 17 Abb-ZETA-Schlüsselgenerierung-Apple

1. Generierung des Client Instance Key Pairs

In dieser Phase erzeugt die App den primären kryptografischen Schlüssel, der zukünftig für die Authentisierung gegenüber dem ZETA Guard (Client Assertions) genutzt wird. (01) Installation: Der ZETA Client wird im User Space des Betriebssystems installiert und gestartet.

2260
2261
2262
2263
2264
2265
2266

2267 (02) Schlüsselgenerierung anfordern: Der Client ruft die native Apple-API
2268 SecKeyCreateRandomKey auf, um ein asymmetrisches Schlüsselpaar zu erzeugen. Die
2269 Parameter erzwingen die Speicherung zwingend in der hardwareisolierten Secure Enclave
2270 (SE).
2271 (03) Erzeugung in der SE: Das Betriebssystem leitet den Befehl an die Secure Enclave
2272 weiter, welche das Schlüsselpaar (PrK.Client.Sig, PuK.Client.Sig) erzeugt. Der private
2273 Schlüssel verlässt die Secure Enclave niemals.
2274 (04) & (05) Rückgabe der Referenz: Die Secure Enclave und das Betriebssystem geben
2275 lediglich einen Handle bzw. eine Referenz (keyId) für diesen Schlüssel an den ZETA Client
2276 zurück.

2277 **2. Hardware-Attestierung durch den Attestation Key**

2278 Um dem ZETA Guard (AuthS) beim späteren DCR-Prozess beweisen zu können, dass der
2279 Client-Schlüssel tatsächlich in der Secure Enclave dieses spezifischen Apple-Geräts liegt,
2280 wird das App Attest Framework genutzt.

2281 (06) AK-Schlüsselpaar anfordern: Der ZETA Client fordert über
2282 DCAppAttest.shared.generateKey() die Generierung eines dedizierten
2283 Attestierungsschlüssels (AK) an.

2284 (07) bis (09) AK in der SE: Die Secure Enclave erzeugt den Attestation Key und gibt
2285 dessen Referenz (keyId_AK) an den Client zurück.

2286 (10) Hash-Berechnung (Challenge): Der ZETA Client berechnet den SHA-256 Hash seines
2287 öffentlichen Client-Schlüssels (challenge = SHA-256(PuK.Client.Sig)). Dies ist das
2288 entscheidende Key-Binding.

2289 (11) Attestierung initiieren: Der Client ruft DCAppAttest.shared.attestKey auf und übergibt
2290 die Referenz des Attestation Keys (keyId_AK) sowie den Hash als clientDataHash
2291 (Challenge).

2292 (12) bis (16) Erstellung des Attestation Objects: Die Secure Enclave signiert die Challenge
2293 mit dem Attestation Key. Das Betriebssystem kommuniziert im Hintergrund mit den Apple
2294 Services, um ein standardisiertes, von Apple signiertes CBOR-Objekt zu erstellen. Dieses
2295 finale apple_attestation_object (welches eine X.509-Zertifikatskette, das Apple-Receipt
2296 und die eingebettete Challenge enthält) wird dem Client übergeben. (Dieses Objekt
2297 beweist später dem ZETA Guard die Hardware-Bindung).

2298 **3. Expliziter Besitznachweis / Proof of Possession**

2299 Obwohl das Attestation Object beweist, dass der Schlüssel sicher in der Hardware liegt,
2300 muss der ZETA Client zusätzlich beweisen, dass genau diese App-Instanz autorisiert ist,
2301 den Schlüssel zu verwenden (Besitznachweis).

2302 (17) Signatur anfordern: Der ZETA Client ruft die Standard-Krypto-API
2303 SecKeyCreateSignature auf. Er nutzt die Referenz des primären Client Instance Keys
2304 (PrK.Client.Sig), um den Hash seines eigenen Public Keys zu signieren.

2305 (18) Signatur in der HW: Die Secure Enclave wird angewiesen, den Hash mit dem privaten
2306 Client-Schlüssel zu signieren.

2307 (19) & (20) Rückgabe der Signatur: Die erzeugte ECDSA-Signatur
2308 (signed_hash_puk_client_sig) wird an den ZETA Client zurückgeliefert.

2309 **4. Software Fallback**

2310 (21) Fallback: Für den Fall, dass das Apple-Gerät keine Secure Enclave besitzt oder App
2311 Attest nicht verfügbar ist (z. B. in bestimmten Simulator-Umgebungen oder bei veralteter
2312 Hardware), generiert der ZETA Client das Schlüsselpaar rein in Software. Es existiert dann
2313 keine kryptografische Hardware-Bindung, was bei der Registrierung (DCR) am ZETA
2314 Guard zu einer Abwertung des Trust-Scores durch die Policy Engine führt.

2315 **A_29782 -ZETA Client, Ablauf Apple OS Schlüsselgenerierung**

2316 Der ZETA Client für Apple Betriebssysteme MUSS den Ablauf gemäß Abbildung *Abb-ZETA-*
2317 *Schlüsselgenerierung-Apple* unterstützen. [<=]

2318
2319
2320
2321
2322
2323

5.3.2.2 Client Registrierung und Authentifizierung

Das Sequenzdiagramm zeigt den Prozess der Dynamic Client Registration (DCR) für mobile ZETA Clients. Es vereint die kryptografische Hardware-Attestierung mit der nutzerzentrierten E-Mail-Verifikation (TOFU - Trust On First Use) und die Generierung eines Wiederherstellungs-codes (Recovery Code / Faktor F3) für den Fall eines Geräteverlusts.



2324
2325
2326
2327

Abbildung 18 Abb-ZETA-DCR-für-mobile-Clients

Der Gesamtprozess teilt sich in zwei Hauptpfade auf:

2328 • Pfad 1: Kein ZETA Attestation Token vorhanden (Erstregistrierung mit Hardware- oder
2329 Software-Attestierung sowie einer E-Mail-Verifizierung mittels OTP).

2330 • Pfad 2: ZETA Attestation Token vorhanden (Fast-Path) (Schnellere Registrierung unter
2331 Wiederverwendung eines bereits ausgestellten Attestierungstokens).

2332 **Hauptpfad 1: [Kein ZETA Attestation Token]**

2333 Falls noch kein Token vorhanden ist, muss sich der Client erst registrieren. Hierbei gibt es
2334 drei verschiedene Attestierungsmethoden (Sub-Pfade):

2335 **[Hardware Attestation: iOS, iPadOS, macOS (Apple App Attest)]**

2336 (01) POST /register: Der ZETA Client sendet eine Registrierungsanfrage an den
2337 Authorization Server. Übermittelt werden unter anderem: attestation_type="apple",
2338 client_name, jwks (mit dem öffentlichen Schlüssel PuK.Client.Sig), das apple attestation
2339 object, die Signatur über den Hash des Client-Signaturschlüssels

2340 (signed_hash_puk_client_sig), die Redirect-URLs und die E-Mail-Adresse des Nutzers.

2341 (02) Validiere apple_attestation_object: Der Authorization Server prüft das empfangene
2342 Apple-Attestierungsobjekt (Verifikation der Apple App Attest Root CA, Abgleich des
2343 extrahierten clientDataHash mit dem SHA-256-Hash des öffentlichen Client-Schlüssels
2344 PuK.Client.Sig).

2345 (03) Prüfe signed_hash_puk_client_sig mit PuK.Client.Sig: Der Server verifiziert die
2346 Signatur (Selbstsignatur bzw. Proof-of-Possession des Client Instance Keys).

2347 **[Hardware Attestation: Android (Keystore, TEE, StrongBox)]**

2348 (04) POST /register: Der ZETA Client sendet eine Registrierungsanfrage mit
2349 attestation_type="android", client_name, jwks (mit PuK.Client.Sig), der android key
2350 attestation certificate chain, der Signatur über den Hash des Client-Signaturschlüssels,
2351 dem play integrity token, den Redirect-URLs und der Nutzer-E-Mail.

2352 (05) Validiere Zertifikatskette gegen Google Root CA: Der Server überprüft die
2353 Vertrauenskette der Android-Schlüsselattestierung.

2354 (06) Extrahiere attestationChallenge aus KeyDescription ASN.1-Extension des Blatt-
2355 Zertifikats: Der Server liest die Challenge aus den Zertifikatsdetails aus.

2356 (07) Prüfe attestationChallenge == SHA-256(PuK.Client.Sig): Der Server stellt sicher, dass
2357 die Challenge dem Hash des übermittelten öffentlichen Schlüssels entspricht.

2358 (08) Prüfe signed_hash_puk_client_sig mit PuK.Client.Sig: Verifikation des Proof-of-
2359 Possession (PoP) des Client Instance Keys.

2360 **Optionaler Block: [Play Integrity Prüfung]**

2361 (09) Entschlüssele play_integrity_token (lokal/remote): Der Server entschlüsselt
2362 den Play-Integrity-Token von Google.

2363 (10) Verifiziere Nonce: Der Server prüft, ob die Nonce im Token dem SHA-256-
2364 Hash von PuK.Client.Sig entspricht.

2365 (11) Werte Verdicts aus: Auswertung der Integritätsurteile (applIntegrity,
2366 deviceIntegrity, accountDetails).

2367 **[Software Attestation (Legacy OIDC DCR)]**

2368 (12) POST /register: Der Client sendet eine klassische Registrierungsanfrage ohne
2369 Hardware-Attestierung mit Angaben zu client_name, unterstützten grant_types (Token-
2370 Exchange, Refresh Token, JWT-Bearer), jwks, der Authentifizierungsmethode
2371 (private_key_jwt), den Redirect-URLs und der Nutzer-E-Mail.

2372 (13) Store Client Placeholder (state: pending_verification): Der Server legt einen
2373 vorläufigen Client-Eintrag an.

2374 (14) Generiere OTP & transaction_id, Speichere Request im temporären Cache: Erstellung
2375 eines Einmalpassworts (OTP) und Speicherung des Vorgangs.

2376 (15) E-Mail mit OTP-Bestätigungscode (Out-of-Band): Der Server sendet das OTP direkt an
2377 die E-Mail-Adresse des Nutzers.

2378 (16) 202 Accepted (transaction_id, message_id: otp_verification): Der Server meldet dem
2379 Client, dass die Registrierung läuft und auf die Verifizierung wartet.

2380 (17) Gibt OTP-Bestätigungscode ein: Der Nutzer liest die E-Mail und gibt das OTP in der

2381 App (ZETA Client) ein.
2382 (18) POST /register/verify (transaction_id, code): Der Client sendet das eingegebene OTP
2383 zusammen mit der Transaktions-ID an den Server.
2384 (19) Verifiziere OTP-Code gegen Cache: Der Server prüft die Gültigkeit des Codes.

2385 **Hauptpfad 2: [ZETA Attestation Token vorhanden (Fast-Path)]**

2386 Wenn der Client bereits über ein gültiges ZETA-Attestierungstoken verfügt, kann der
2387 Prozess stark verkürzt werden:
2388 (20) POST /register: Der Client sendet eine Anfrage mit
2389 attestation_type="zeta_attestation_token", client_name, dem ZETA Attestation Token
2390 selbst (zeta_attestation_token), dem jwks (mit PuK.Client.Sig) und zur Absicherung
2391 die Selbstsignatur des SHA-256-Hashes von puk_client_sig (signed_hash_puk_client_sig)
2392 und dem Besitznachweis des Attestation Keys (attestation_pop).
2393 (21) Prüfe Token-Signatur & Vertrauensstellung: Der Server verifiziert das eingereichte
2394 ZETA Attestation Token.
2395 (22) Prüfe signed_hash_puk_client_sig mit PuK.Client.Sig: Prüfung der Selbstsignatur (PoP)
2396 des neuen Client Instance Keys.
2397 (23) Extrahiere attestation_type und PuK.AK.Sig (cnf.jwk) aus dem Token: Der Server liest
2398 den Attestierungstyp und den öffentlichen Attestierungsschlüssel (PuK.AK.Sig) aus dem
2399 Token aus.
2400 Bedingte Verifizierung (abhängig vom extrahierten Typ):
2401 Bei attestation_type == "apple":
2402 (24) Validiere attestation_pop als CBOR-Assertion (generateAssertion): Validierung der
2403 Signatur über die Authentifikatordaten und den Client-Daten-Hash unter Verwendung von
2404 PuK.AK.Sig. Zudem Prüfung des rpldHash und der Monotonie des Counters.
2405 Bei attestation_type == "android" oder "tpm":
2406 (25) Validiere attestation_pop als Signatur über SHA-256(PuK.Client.Sig) mit PuK.AK.Sig:
2407 Direkte Signaturprüfung zur Bindung des (neuen) Client-Schlüssels an den bereits
2408 bekannten Attestierungsschlüssel.

2409 **Abschluss der Registrierung**

2410 Egal welcher Pfad erfolgreich durchlaufen wurde, der Prozess endet mit der Bestätigung:
2411 (26) 201 Created (client_id): Der Authorization Server sendet die Antwort 201 Created
2412 inklusive der zugewiesenen client_id an den ZETA Client zurück. Damit ist die Client-
2413 Registrierung abgeschlossen.

2414 **A_29658 -ZETA, Ablauf Client Registrierung für mobile Clients**

2415 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
2416 *DCR-für-mobile-Clients* unterstützen. [**<=**]

2417 **5.3.2.3 Plattformabhängige Attestierung**

2418 Die Attestierungsdaten werden in der Client Assertion (siehe [client-assertion-jwt]) im
2419 Attribut client_statement (siehe [client-statement.yaml]) eingetragen.

2420 *5.3.2.3.1 Android*

2421 Auf Android-Geräten erfolgt die Attestierung über plattformspezifische
2422 Sicherheitsmechanismen, die eine Bindung zwischen Anwendung, Schlüsselmaterial und
2423 Gerät herstellen.

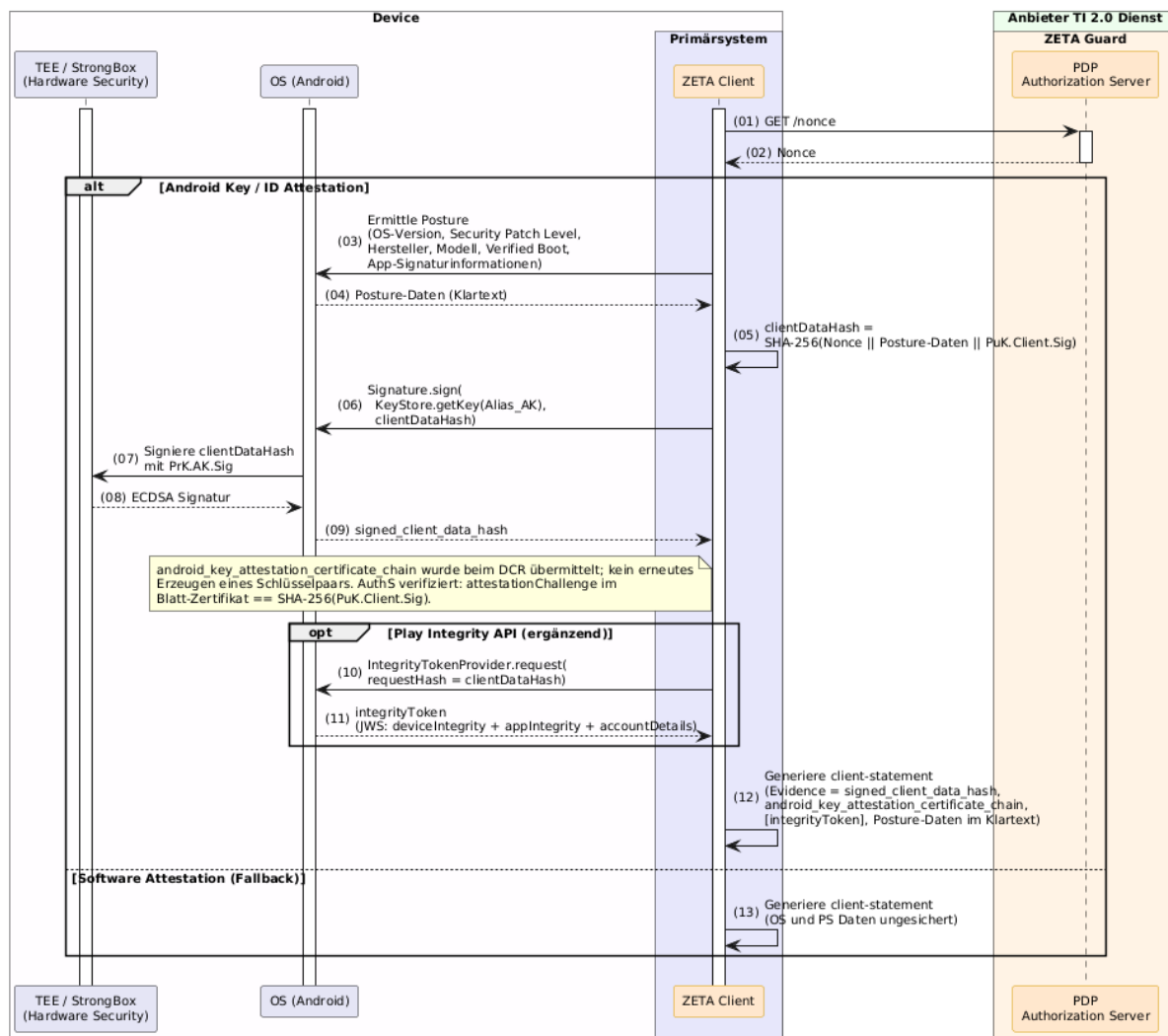


Abbildung 19 Abb-ZETA-Client-Statement-mit-Android-Attestation

Vorbereitungsphase: Nonce-Bezug

Um Replay-Angriffe zu verhindern und die Frische des Nachweises sicherzustellen, benötigt der Client eine kryptografische Challenge vom Server.

(01) GET /nonce: Der ZETA Client fragt eine neue Nonce beim ZETA Guard (Authorization Server) an.

(02) Nonce: Der ZETA Guard generiert eine zufällige Nonce und liefert diese zurück.

Primärer Pfad: Android Key / ID Attestation (Hardwaregestützt)

Dieser Pfad wird durchlaufen, wenn das Gerät über eine Trusted Execution Environment (TEE) oder StrongBox verfügt und die Hardware-Attestierung bei der Registrierung erfolgreich war.

(03) Ermittle Posture: Der ZETA Client fragt über native Android-APIs die aktuellen, sicherheitsrelevanten Systemparameter ab (z. B. OS-Version, Security Patch Level, Hersteller, Modell, Verified Boot, App-Signaturinformationen).

(04) Posture-Daten (Klartext): Das Android OS übergibt diese Systemdaten im Klartext an den Client.

(05) Hash-Berechnung (clientDataHash): Dies ist der zentrale kryptografische Sicherheitsanker. Der ZETA Client berechnet einen SHA-256 Hash aus der Verkettung der serverseitigen Nonce, den ermittelten Posture-Daten und seinem öffentlichen Client-Schlüssel (PuK.Client.Sig).

2424

2425

2426

2427

2428

2429

2430

2431

2432

2433

2434

2435

2436

2437

2438

2439

2440

2441

2442

2443

2444

2445 (Dadurch werden die Frische des Requests, der Systemzustand und die Identität des
2446 Clients untrennbar aneinander gebunden).
2447 (06) Signatur anfordern: Der Client ruft Signature.sign auf. Hierbei referenziert er den
2448 dedizierten Attestation Key (Alias_AK), der beim DCR-Prozess in der Hardware erzeugt
2449 wurde, und fordert die Signatur des clientDataHash an.
2450 (07) Signatur in der HW: Das Android OS instruiert die TEE / StrongBox, den Hash mit dem
2451 privaten Attestation Key (PrK.AK.Sig) zu signieren.
2452 (08) & (09) Rückgabe der Signatur: Die TEE liefert die erzeugte ECDSA-Signatur
2453 (signed_client_data_hash) an den ZETA Client zurück.
2454 (Erklärung gemäß Notiz: Da der ZETA Guard bereits beim DCR verifiziert hat, dass der
2455 Attestation Key kryptografisch an den Client-Schlüssel gebunden ist [attestationChallenge
2456 == SHA-256(PuK.Client.Sig)], beweist diese Laufzeit-Signatur dem Server, dass die
2457 Posture-Daten vom selben physischen Gerät stammen).

2458 **Optionaler Block: Play Integrity API (ergänzend)**

2459 Da die Posture-Daten im Laufzeit-Flow durch Software-APIs ermittelt wurden, kann (und
2460 sollte) dieser Status durch eine serverseitige Google-Prüfung verifiziert werden.
2461 (10) Request an Play Integrity: Der ZETA Client ruft den IntegrityTokenProvider auf.
2462 Entscheidend: Er übergibt den exakt selben clientDataHash als requestHash.
2463 (11) integrityToken (JWS): Das System liefert das von Google signierte Token zurück,
2464 welches unabhängige Bewertungen (Verdicts) zur Geräte- und App-Integrität enthält und
2465 kryptografisch an den aktuellen Request gebunden ist.

2466 **Zusammenbau des Statements**

2467 (12) Generiere client-statement: Der ZETA Client baut das finale client-statement
2468 (Evidence) für den ZETA Guard zusammen. Es besteht aus der Hardware-Signatur
2469 (signed_client_data_hash), der beim DCR ausgetauschten
2470 android_key_attestation_certificate_chain, dem optionalen integrityToken und den
2471 Posture-Daten im Klartext.

2472 **Alternativer Pfad: Software Attestation (Fallback)**

2473 (13) Generiere client-statement (Software): Falls keine TEE/StrongBox verfügbar ist,
2474 sammelt der ZETA Client die OS- und Primärsystem-Daten (PS-Daten) und fügt sie
2475 ungesichert in das Statement ein. Die Policy Engine des ZETA Guards wird diesen
2476 Request mit einem deutlich geringeren Vertrauensniveau (Trust-Score) bewerten.

2477 **A_29809 -ZETA Client, Ablauf Android Attestierung**

2478 Der ZETA Client MUSS den Ablauf gemäß Abbildung *Abb-ZETA-Client-Statement-mit-*
2479 *Android-Attestation* unterstützen.
2480 **[<=]**

2481 **5.3.2.3.2 Apple**

2482 Für Apple-basierte Geräte erfolgt die Attestation durch die Secure Enclave in Kombination
2483 mit Apple App Attest.

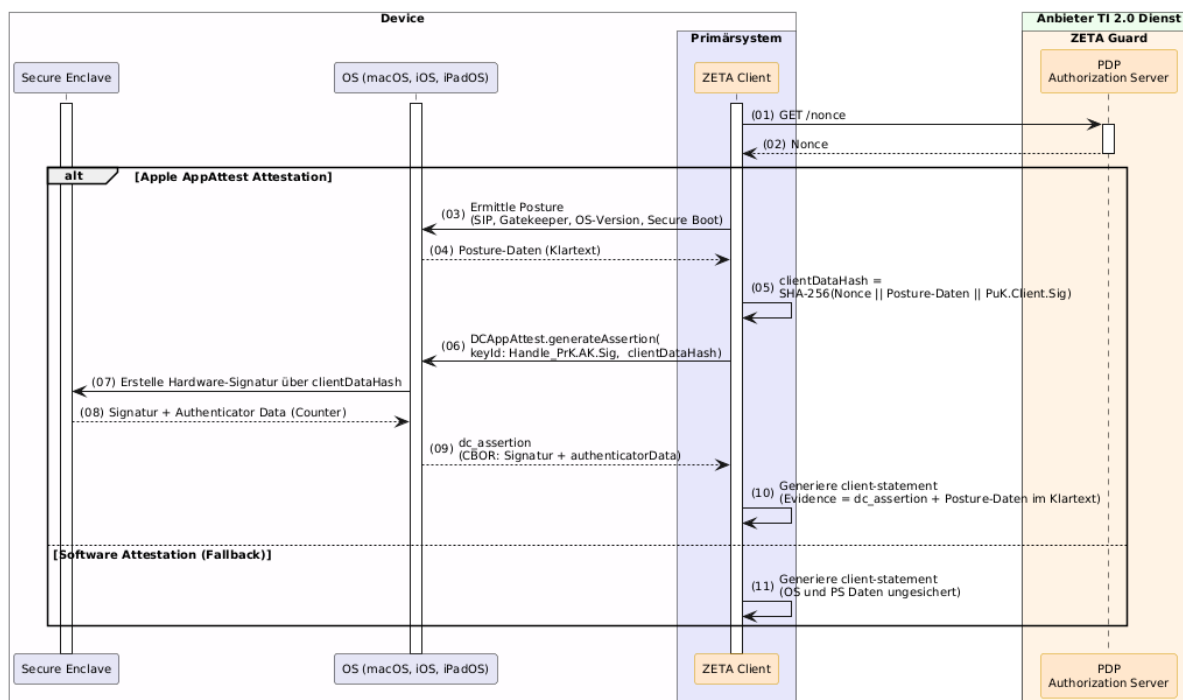


Abbildung 20 Abb-ZETA-Client-Statement-mit-Apple-AppAttest

Vorbereitungsphase: Nonce-Bezug

Um Replay-Angriffe zu verhindern und die Frische des Integritätsnachweises zu garantieren, benötigt der Client eine kryptografische Challenge vom Server.
 (01) GET /nonce: Der ZETA Client fragt eine neue Nonce beim ZETA Guard (Authorization Server) an.
 (02) Nonce: Der ZETA Guard generiert eine zufällige Nonce und liefert diese zurück.

Primärer Pfad: Apple AppAttest Attestation (Hardwaregestützt)

Dieser Pfad wird durchlaufen, wenn das Gerät über eine Secure Enclave verfügt und das App Attest Framework bei der initialen Registrierung (DCR) erfolgreich eingerichtet wurde.
 (03) Ermittle Posture: Der ZETA Client fragt über native Apple-APIs sicherheitsrelevante Systemparameter ab. Dazu gehören beispielsweise der Status der System Integrity Protection (SIP), Gatekeeper-Einstellungen, die exakte OS-Version und der Status des Secure Boots.
 (04) Posture-Daten (Klartext): Das Betriebssystem übergibt diese Systemdaten im Klartext an den ZETA Client.
 (05) Hash-Berechnung (clientDataHash): Dies ist der zentrale kryptografische Sicherheitsanker. Der ZETA Client berechnet einen SHA-256 Hash aus der Verkettung der serverseitigen Nonce, den ermittelten Posture-Daten und seinem öffentlichen Client-Schlüssel (PuK.Client.Sig).
 (Architektur-Hinweis: Dadurch werden die Frische des Requests, der Systemzustand und die Identität des Clients untrennbar aneinander gebunden).
 (06) Assertion anfordern: Der Client ruft die API DCAppAttest.generateAssertion auf. Er übergibt das Handle seines hardwaregebundenen Attestation Keys (Handle_PrK.AK.Sig) sowie den soeben berechneten clientDataHash.
 (07) Hardware-Signatur: Das Betriebssystem leitet den Hash an die Secure Enclave weiter. Die Secure Enclave signiert diesen Hash mit dem privaten Attestation Key.
 (08) Rückgabe der Signatur & Metadaten: Die Secure Enclave gibt die Signatur sowie zusätzliche Authenticator-Daten (insbesondere einen monoton steigenden Counter zum Schutz vor Klon-Angriffen) an das OS zurück.
 (09) dc_assertion (CBOR): Das OS verpackt die Signatur und die Authenticator-Daten in

2517 ein standardisiertes CBOR-Objekt (dc_assertion) und reicht es an den ZETA Client weiter.
2518 (10) Generiere client_statement: Der ZETA Client baut das finale client_statement
2519 (Evidence) für den Request an den ZETA Guard zusammen. Es besteht aus der
2520 kryptografischen dc_assertion und den Posture-Daten im Klartext.
2521 (Der ZETA Guard kann später den Hash aus der ihm bekannten Nonce, den Klartext-
2522 Posture-Daten und dem registrierten Client-Key selbst berechnen und die Hardware-
2523 Signatur in der dc_assertion validieren).

2524 **Alternativer Pfad: Software Attestation (Fallback)**

2525 Dieser Pfad dient als Rückfallebene, falls keine Hardware-Unterstützung vorliegt.
2526 (11) Generiere client_statement (Software): Der ZETA Client sammelt die OS- und
2527 Primärsystem-Daten (PS-Daten) und fügt sie ungesichert in das Statement ein. Da hierbei
2528 keine Signatur durch eine Secure Enclave stattfindet, wird die Policy Engine des ZETA
2529 Guards diesen Request mit einem entsprechend niedrigen Trust-Score bewerten.

2530 **A_29810 -ZETA Client, Ablauf Apple OS Attestierung**

2531 Der ZETA Client MUSS den Ablauf gemäß Abbildung *Abb-ZETA-Client-Statement-mit-*
2532 *Apple-AppAttest* unterstützen.
2533 [**<=**]

2534 *5.3.2.3.3 Software Attestierung (Fallback)*

2535 Falls keine hardwaregestützte Attestierung möglich ist, wird eine softwarebasierte
2536 Attestierung durchgeführt.
2537 Dies betrifft insbesondere:

- 2538 • ältere Geräte
- 2539 • Emulatoren oder spezielle Umgebungen
- 2540 • Geräte ohne verfügbare Attestation-APIs oder Hardware

2541 Der Ablauf ist wie folgt:

2542 (01) Generierung des Client-Schlüssels
2543 Das Schlüsselpaar wird im Software-Kontext erzeugt.
2544 (02) Erhebung von Basisinformationen
2545 Der ZETA Client sammelt einfache Kontextdaten:

- 2546 • Betriebssystem und Version
- 2547 • Architektur
- 2548 • Produkt- und Versionsinformationen

2549 Eine hardwarebasierte Vertrauensbindung besteht nicht. Der resultierende
2550 Sicherheitsgrad ist daher geringer. Die finale Zugriffserlaubnis erfolgt unter zusätzlicher
2551 Policy-Prüfung.

2552 **5.3.2.4 Service Discovery**

2553 Die Service Discovery ist identisch für stationäre und mobile Clients. Siehe [5.3.1.3- Service](#)
2554 [Discovery](#).

2555 **5.3.2.5 Authentifizierung**

2556 Mobile ZETA Clients authentisieren den Nutzer interaktiv über einen sektoralen Identity
2557 Provider (IDP) der TI-Föderation. Hierzu wird ein OpenID Connect (OIDC) Authorization
2558 Code Flow mit Pushed Authorization Request (PAR, RFC 9126) und Proof Key for Code
2559 Exchange (PKCE, RFC 7636) ausgeführt.

2560 Auf dem mobilen Endgerät wirken zwei Komponenten zusammen:

- 2561 • Der **Fach-Client** enthält die fachliche Logik für die Arbeit mit dem Resource Server.
2562 Er wählt bzw. lässt den Nutzer den zuständigen sektoralen IDP wählen (idp_iss),
2563 erstellt den vollständigen fachlichen HTTPS-Request inklusive aller Header und
2564 übergibt ihn an den ZETA Client.
- 2565 • Der **ZETA Client** kümmert sich um Client-Registrierung, Session-, Schlüssel- und
2566 Token-Verwaltung sowie die Authentifizierung. Er verarbeitet den Request, beschafft
2567 bei Bedarf ein gültiges Access Token und gibt die Response an den Fach-Client
2568 zurück.

2569 Diese Aufgabenteilung entspricht dem Zusammenspiel von Fach-Client und ZETA Client
2570 beim Zugriff auf den Resource Server (siehe Kapitel 5.3.1.7).

2571 Der PDP Authorization Server des ZETA Guard übernimmt zwei Rollen:

- 2572 • Gegenüber dem mobilen ZETA Client agiert er als der für den Fachdienst zuständige
2573 Authorization Server (äußerer Flow, code_challenge_app).
- 2574 • Gegenüber dem sektoralen IDP agiert er als OpenID-Connect-Relying-Party
2575 (Fachdienst, innerer Flow, code_challenge_as). Die Client-Authentifizierung am IDP
2576 erfolgt per mTLS (self_signed_tls_client_auth), die Vertrauensbeziehung über OpenID
2577 Federation 1.0.

2578 Das vom PDP Authorization Server ausgestellte Access Token ist über DPoP an die Sitzung
2579 gebunden.

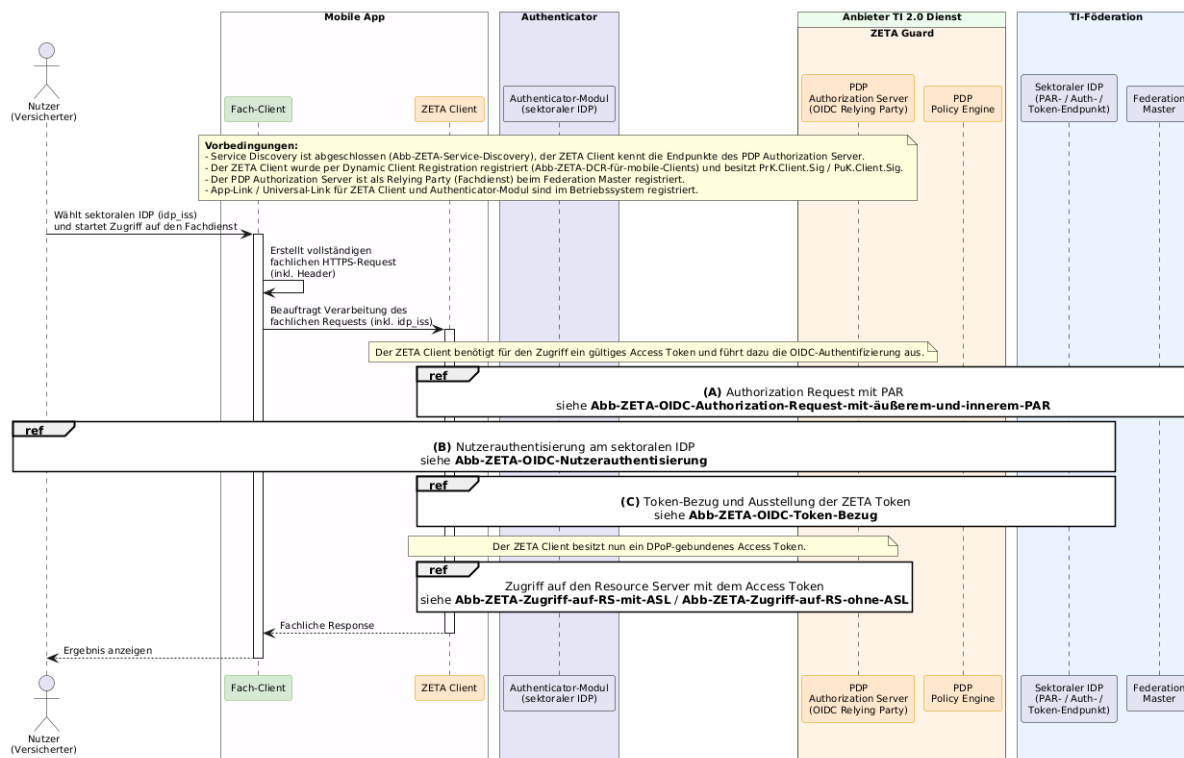
2580 5.3.2.5.1 Voraussetzungen

- 2581 • **Service Discovery abgeschlossen:** Der Client kennt die Endpunkte des PDP
2582 Authorization Server (authorization_endpoint, token_endpoint); siehe Kapitel [5.3.1.3-](#)
2583 [Service Discovery](#).
- 2584 • **Dynamic Client Registration** abgeschlossen: Der Client besitzt eine client_id sowie
2585 das Schlüsselpaar PrK.Client.Sig / PuK.Client.Sig.
- 2586 • **App-Link / Universal-Link** für ZETA Client und Authenticator-Modul sind im
2587 Betriebssystem registriert.
2588 Sektoraler IDP wird vom Fach-Client ausgewählt und als idp_iss an den ZETA Client
2589 übergeben.

2590 5.3.2.5.2 Übersicht

2591 Der Ablauf besteht aus einem äußeren Authorization-Code-Flow (ZETA Client ↔ PDP
2592 Authorization Server) und einem inneren Authorization-Code-Flow (PDP Authorization
2593 Server ↔ sektoraler IDP) und ist in drei Teilabläufe (A), (B) und (C) zerlegt:

- 2594 • **(A) Authorization Request mit PAR:** Der ZETA Client startet den Authorization
2595 Request; der PDP Authorization Server stellt einen Pushed Authorization Request
2596 (PAR) am IDP und erhält eine request_uri.
- 2597 • **(B) Nutzerauthentisierung:** Der ZETA Client öffnet das Authenticator-Modul; der
2598 Nutzer authentisiert sich (eGK+PIN / eID) und gibt den Consent frei; der IDP stellt
2599 einen AUTHORIZATION_CODE (IDP) aus.
- 2600 • **(C) Token-Bezug und Ausstellung der ZETA Token:** Der PDP Authorization Server
2601 löst den Code am IDP ein, entscheidet über die Policy Engine und stellt dem ZETA
2602 Client ein DPoP-gebundenes Access Token und ein Refresh Token aus.



2603

2604

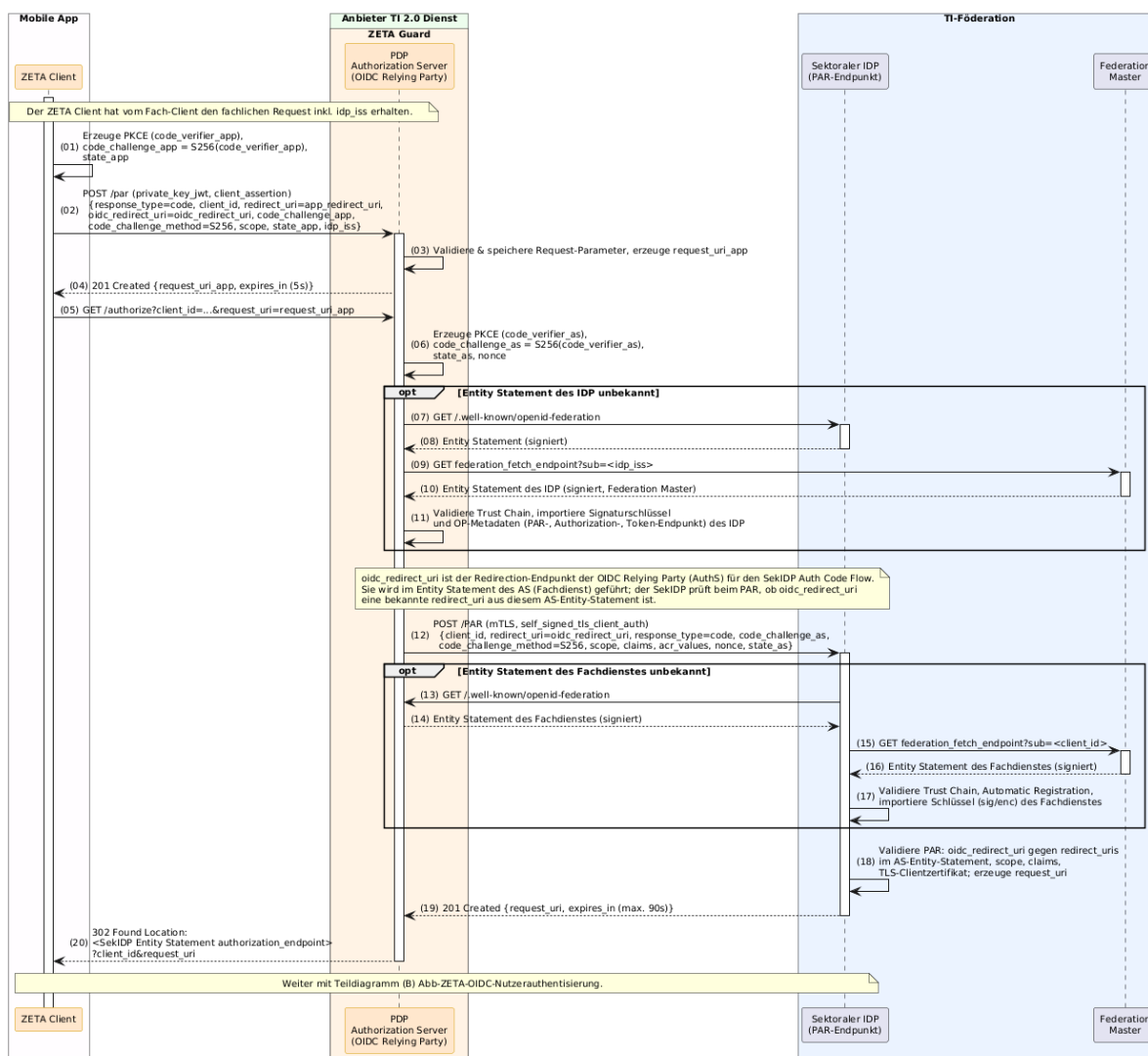
Abbildung 21 Abb-ZETA-OIDC-Authentifizierung-mobiler-Clients

A_29659 -ZETA, Ablauf OIDC Authentifizierung für mobile Clients

Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-OIDC-Authentifizierung-mobiler-Clients* unterstützen. [**=**]

5.3.2.5.3 Teilablauf (A): Authorization Request mit äußerem und innerem PAR

Das folgende Sequenzdiagramm beschreibt die initiale Phase der Nutzerauthentisierung für mobile Clients (Apps) innerhalb der ZETA-Architektur. Im Kern zeigt es den Aufbau eines OIDC Authorization Requests, erweitert um Sicherheitsmechanismen wie PKCE (Proof Key for Code Exchange), PAR (Pushed Authorization Requests) und die dynamische Vertrauensbildung über die TI-Föderation (OpenID Federation).



2614

2615

Abbildung 22 Abb-ZETA-OIDC-Authorization-Request-mit-äußerem-und-innerem-PAR

2616 **Ausgangssituation:** Der ZETA Client hat vom aufrufenden Fach-Client (der eigentlichen
 2617 Fach-App) den Auftrag erhalten, einen Request abzusichern. Dabei wurde auch der
 2618 Identifier des gewünschten Identity Providers (idp_iss) übergeben.

2619 **1. PAR und Initiierung durch den ZETA Client**

- 2620 (01) PKCE & State Generierung (Client): Der ZETA Client generiert einen zufälligen
- 2621 code_verifier_app, leitet daraus über SHA-256 die code_challenge_app ab und erstellt
- 2622 einen state_app zur Verhinderung von CSRF-Angriffen.
- 2623 (02) POST /par am ZETA Guard: Der Client initiiert den Autorisierungsablauf über Pushed
- 2624 Authorization Requests (RFC 9126) und sendet einen POST /par an den ZETA Guard. Er
- 2625 authentisiert sich dabei über eine Client Assertion (private_key_jwt). Im Body überträgt er
- 2626 die Autorisierungsparameter (u. a. response_type=code, app_redirect_uri, die eigene
- 2627 PKCE-Challenge, scope, state_app und idp_iss).
- 2628 (03) Validierung & request_uri Erzeugung: Der ZETA Guard validiert die Client Assertion
- 2629 sowie die Request-Parameter, speichert diese zwischen und generiert eine eindeutige
- 2630 request_uri_app.
- 2631 (04) 201 Created (request_uri_app): Der Guard antwortet mit HTTP 201 und übergibt die
- 2632 request_uri_app mit einer sehr kurzen Gültigkeitsdauer von 5 Sekunden.
- 2633 (05) GET /authorize: Der ZETA Client ruft nun den Authorization-Endpoint des ZETA
- 2634 Guards auf (GET /authorize), wobei lediglich die eigene client_id und die zuvor erhaltene

2635 sichere request_uri_app übergeben werden.

2637 **2. Vorbereitung durch den ZETA Guard (als Relying Party)**

2638 (06) PKCE & Nonce Generierung (Guard): Da der Guard als Vermittler (Relying Party)
2639 gegenüber dem sektoralen IDP auftritt, generiert er einen eigenen, zweiten PKCE-Satz
2640 (code_verifier_as, code_challenge_as), einen eigenen state_as sowie einen nonce für das
2641 spätere ID-Token.

2642 **3. Dynamischer Vertrauensaufbau: IDP (OpenID Federation)**

2643 (Dieser optionale Block wird nur ausgeführt, wenn dem ZETA Guard das Entity Statement
2644 des angeforderten IDP noch nicht bekannt ist oder dieses abgelaufen ist).

2645 (07) & (08) Fetch IDP Entity Statement: Der Guard ruft das signierte Entity Statement des
2646 sektoralen IDPs über dessen .well-known/openid-federation Endpunkt ab.

2647 (09) & (10) Fetch from Federation Master: Um die Vertrauenswürdigkeit des IDP zu
2648 überprüfen, fragt der Guard beim zentralen Federation Master das Entity Statement für
2649 diesen IDP (Subjekt = idp_iss) an.

2650 (11) Validierung & Metadaten-Import: Der Guard validiert die erhaltene Trust Chain,
2651 importiert den Signaturschlüssel des IDP sowie dessen OP-Metadaten (PAR-,
2652 Authorization- und Token-Endpunkt).

2653 **4. Pushed Authorization Request (PAR) am sektoralen IDP**

2654 (12) POST /PAR am SekIDP: Der Guard sendet die Autorisierungsdaten per mTLS (unter
2655 Nutzung seiner self_signed_tls_client_auth) an den PAR-Endpunkt des sektoralen IDP. Im
2656 Payload übergibt er seine eigene client_id, die oidc_redirect_uri (der eigene Redirection-
2657 Endpunkt für den SekIDP Auth Code Flow), seine PKCE-Daten (code_challenge_as), scope,
2658 claims, acr_values, nonce und state_as.

2659 **5. Dynamischer Vertrauensaufbau: ZETA Guard (OpenID Federation)**

2660 (Dieser optionale Block auf Seiten des IDP wird nur ausgeführt, wenn dem IDP das Entity
2661 Statement des anfragenden Fachdienstes/Guards noch nicht bekannt ist).

2662 (13) & (14) Fetch Guard Entity Statement: Der IDP ruft das Entity Statement des
2663 anfragenden ZETA Guards ab.

2664 (15) & (16) Fetch from Federation Master: Der IDP verifiziert den Guard beim Federation
2665 Master (Subjekt = client_id).

2666 (17) Automatic Registration: Der IDP validiert die Trust Chain, führt die Automatic
2667 Registration durch und importiert die Schlüssel (Signatur und Verschlüsselung) des
2668 Fachdienstes.

2669 **6. PAR-Antwort am IDP und Umleitung an den Authenticator**

2670 (18) PAR-Validierung (SekIDP): Der sektorale IDP validiert den PAR-Request (u. a. Prüfung
2671 der oidc_redirect_uri gegen die im AS-Entity-Statement hinterlegten redirect_uris, Scopes,
2672 Claims und das TLS-Clientzertifikat) und erzeugt eine eindeutige request_uri.

2673 (19) 201 Created (SekIDP): Der IDP antwortet dem Guard mit HTTP 201 und übergibt die
2674 generierte request_uri (Gültigkeit max. 90 Sekunden).

2675 (20) 302 Found (HTTP-Redirect an App): Der ZETA Guard beantwortet nun den
2676 Authorization Request des ZETA Clients (aus Schritt 05) mit einem HTTP 302 Redirect. Die
2677 Location-URL verweist auf den Authorization-Endpunkt des sektoralen IDP (aus dessen
2678 Entity Statement), parametrisiert mit der client_id des Guards und der neuen request_uri.
2679 (Dieser Redirect löst auf dem mobilen Endgerät den App-Wechsel in das Authenticator-
2680 Modul aus).

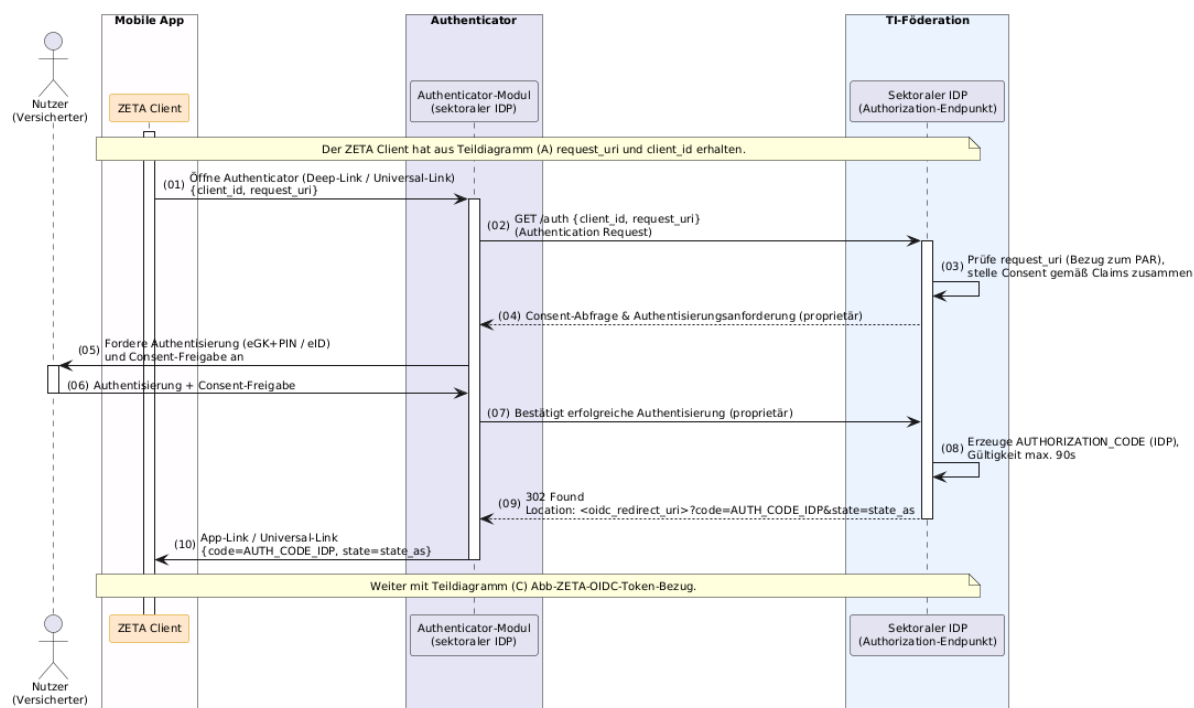
2681 **A_29660 -ZETA, Ablauf Authorization Request mit PAR**

2682 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
2683 *OIDC-Authorization-Request-mit-äußerem-und-innerem-PAR* unterstützen.

2684 [**<=**]

2685 5.3.2.5.4 Teilablauf (B): Nutzerauthentisierung am sektoralen IDP

2686 Der ZETA Client delegiert die interaktive Nutzerauthentisierung an das Authenticator-
 2687 Modul des sektoralen IDP. Im Zentrum dieses Ablaufs steht der Wechsel aus der
 2688 aufrufenden App (ZETA Client) in eine dedizierte Authenticator-App (z. B. die App der
 2689 Krankenkasse), in welcher der Versicherte sich sicher authentisiert (z. B. via eGK und PIN
 2690 oder eID) und dem Datenzugriff zustimmt. Eine In-App-Authentifizierung ist ebenfalls
 2691 möglich, wenn der Herausgeber der Fach-App (z. B. der ePA-App) identisch mit dem
 2692 Anbieter des Authenticator-Moduls (z. B. der Krankenkasse als sektoraler IDP) ist.



2693
 2694 **Abbildung 23 Abb-ZETA-OIDC-Nutzerauthentisierung**

- 2695 (01) Der ZETA Client öffnet das Authenticator-Modul des sektoralen IDP per Deep-Link
- 2696 bzw. Universal-Link und übergibt client_id und request_uri.
- 2697 (02) Das Authenticator-Modul übermittelt den Authentication Request (GET /auth mit
- 2698 client_id und request_uri) an den Authorization-Endpoint des sektoralen IDP.
- 2699 (03) Der IDP prüft die request_uri (Bezug zum zuvor gestellten PAR) und stellt die
- 2700 Consent-Abfrage gemäß den angeforderten claims zusammen.
- 2701 (04) Der IDP fordert über das Authenticator-Modul die Nutzerauthentisierung und die
- 2702 Consent-Freigabe an (proprietäres Protokoll des sektoralen IDP).
- 2703 (05)–(06) Der Nutzer authentisiert sich (z. B. eGK+PIN oder eID) und gibt den Consent
- 2704 frei.
- 2705 (07) Das Authenticator-Modul bestätigt dem IDP die erfolgreiche Authentisierung.
- 2706 (08) Der IDP erzeugt einen AUTHORIZATION_CODE (IDP) mit einer Gültigkeit von max. 90
- 2707 Sekunden.
- 2708 (09) Der IDP antwortet mit einem 302 Found. Die Location verweist auf die
- 2709 oidc_redirect_uri des Fachdienstes und enthält code=AUTH_CODE_IDP und
- 2710 state=state_as.
- 2711 (10) Das Authenticator-Modul ruft den ZETA Client per App-Link bzw. Universal-Link auf
- 2712 und übergibt code=AUTH_CODE_IDP und state=state_as.

2713 **A_29671 -ZETA, Ablauf Nutzerauthentisierung am sektoralen IDP**

2714 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
 2715 *OIDC-Nutzerauthentisierung* unterstützen. [<=]

2716 **5.3.2.5.5 Teilablauf (C): Token-Bezug und Ausstellung der ZETA Token**

2717 Das folgende Sequenzdiagramm (Teildiagramm C) spezifiziert den finalen Abschnitt der
 2718 Nutzerauthentisierung: die Auflösung des sektoralen IDP-Codes im Backend (Innerer
 2719 Flow), die dynamische Policy-Evaluation und die anschließende Ausstellung des DPoP-
 2720 gebundenen ZETA Access Tokens (Äußerer Flow).

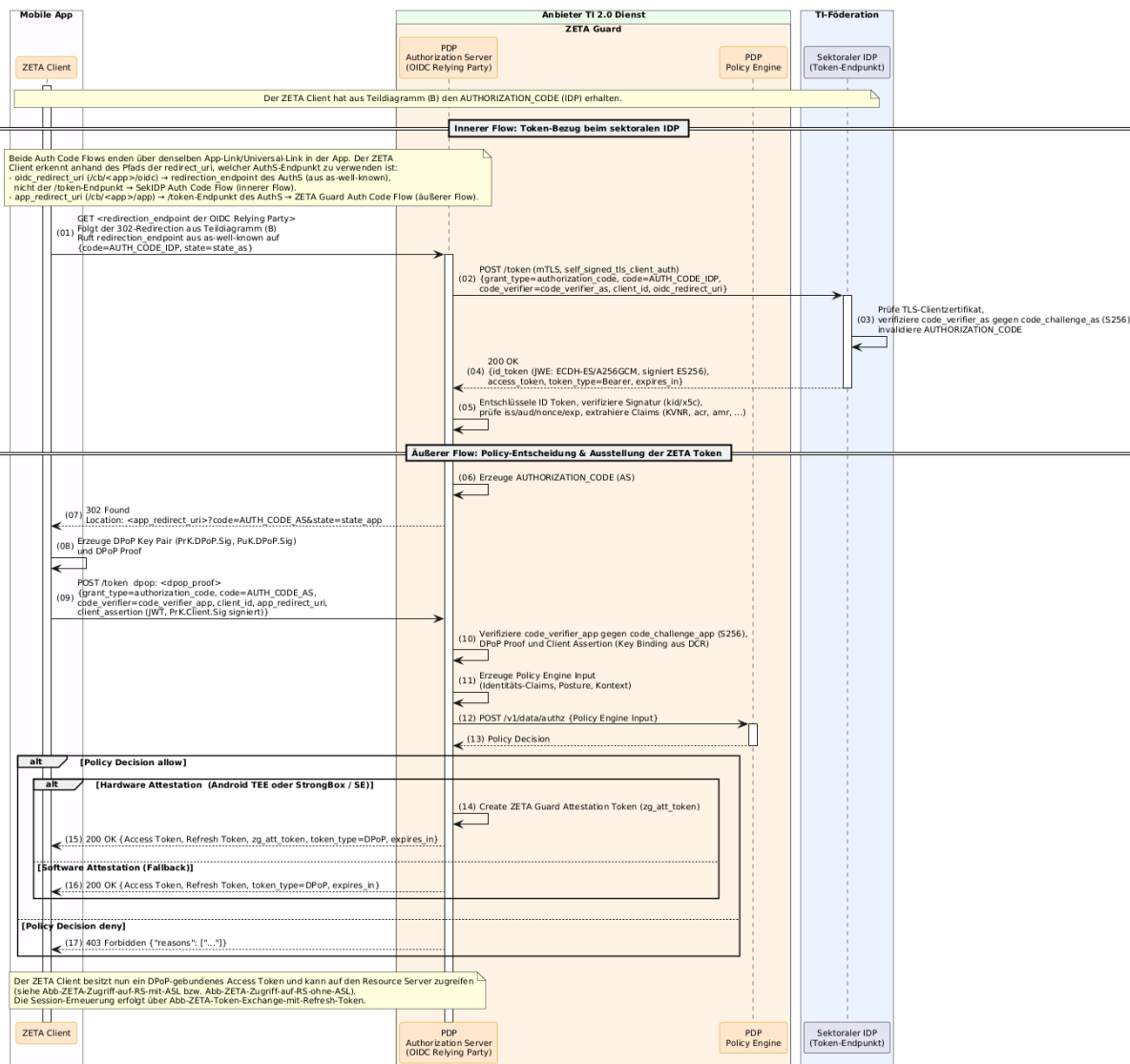


Abbildung 24 Abb-ZETA-OIDC-Token-Bezug

2721 **Vorab-Definition: Routing-Logik der Redirection URIs**

2722 Da sowohl der innere Auth Code Flow (SekIDP) als auch der äußere Auth Code Flow (ZETA
 2723 Guard) über App-Links/Universal-Links auf dem mobilen Endgerät terminieren, nutzt der
 2724 ZETA Client eine strikte Pfad-Unterscheidung, um den korrekten Folge-Endpunkt
 2725 anzusteuern:
 2726 **oidc_redirect_uri (Schema: /cb/<app>/oidc):** Signalisiert dem ZETA Client den
 2727 Abschluss des inneren Flows. Der Client erkennt am Pfad-Suffix /oidc, dass der
 2728 empfangene Code an den Redirection-Endpunkt des ZETA Guards (aus dessen as-well-
 2729
 2730

2731 known) weitergeleitet werden muss - und ausdrücklich nicht an dessen /token-Endpunkt.
2732 **redirect_uri (Schema: /cb/<app>/app):** Signalisiert dem ZETA Client den Abschluss
2733 des äußeren Flows. Der Client erkennt am Pfad-Suffix /app, dass der empfangene Code
2734 nun am /token-Endpunkt des ZETA Guards eingetauscht werden muss.

2735 **Ablaufbeschreibung: OIDC Token-Bezug (Teildiagramm C)**

2736 Ausgangssituation: Der ZETA Client hat am Ende von Teildiagramm B den
2737 AUTHORIZATION_CODE (AUTH_CODE_IDP) des sektoralen IDP erhalten.

2738 **1. Innerer Flow: Token-Bezug beim sektoralen IDP**

2739 (01) Code-Übergabe an den Guard: Folgend der 302-Redirection aus Teildiagramm B ruft
2740 der ZETA Client den Redirection-Endpunkt des ZETA Guards auf (GET
2741 <oidc_redirect_uri>) und überträgt den AUTH_CODE_IDP sowie den state_as.
2742 (02) POST /token am SekIDP: Der ZETA Guard (als Relying Party) ruft per mTLS (unter
2743 Nutzung seiner self_signed_tls_client_auth) den Token-Endpunkt des sektoralen IDP auf.
2744 Er überträgt den AUTH_CODE_IDP, seinen eigenen code_verifier_as, seine client_id und
2745 die oidc_redirect_uri.
2746 (03) Validierung durch den SekIDP: Der sektorale IDP prüft das TLS-Clientzertifikat des
2747 Guards, verifiziert den code_verifier_as gegen die code_challenge_as (S256) und
2748 invalidiert den einmalig gültigen Authorization Code.
2749 (04) 200 OK (Token-Ausstellung SekIDP): Der SekIDP antwortet mit HTTP 200 und
2750 übermittelt ein verschlüsseltes ID-Token (id_token als JWE: ECDH-ES/A256GCM, signiert
2751 mit ES256), ein kurzlebigen Access Token (token_type=Bearer) sowie die Gültigkeitsdauer
2752 (expires_in).
2753 (05) Token-Entschlüsselung & Extraktion: Der ZETA Guard entschlüsselt das ID-Token,
2754 verifiziert die Signatur über die Zertifikatskette (kid/x5c), prüft iss, aud, nonce sowie exp
2755 und extrahiert die im Token enthaltenen fachlichen Identitäts-Claims (u. a. die
2756 Krankenversichertennummer (KVNR), acr und amr).

2757 **2. Äußerer Flow: Policy-Entscheidung (Policy Engine)**

2758 (06) Erzeuge AUTHORIZATION_CODE (AS): Bei positiver Policy-Entscheidung generiert der
2759 ZETA Guard einen neuen Authorization Code (AUTH_CODE_AS) für den äußeren Flow.
2760 (07) 302 Found (Redirect an App): Der Guard beantwortet den Request des Clients aus
2761 Schritt (01) mit einem HTTP 302 Redirect auf die <app_redirect_uri>, parametrisiert mit
2762 dem AUTH_CODE_AS und dem ursprünglichen state_app.
2763 (08) DPoP-Schlüsselpaar generieren: Der ZETA Client fängt den Aufruf der
2764 app_redirect_uri ab. Er erzeugt im Arbeitsspeicher ein neues, asymmetrisches DPoP-
2765 Schlüsselpaar (PrK.DPoP.Sig, PuK.DPoP.Sig) sowie einen frischen DPoP Proof.
2766 (09) POST /token am ZETA Guard: Der Client ruft den Token-Endpunkt des ZETA Guards
2767 auf (POST /token), setzt den HTTP-Header DPoP und authentisiert sich via Client Assertion
2768 (signiert mit PrK.Client.Sig). Im Body überträgt er den AUTH_CODE_AS, seinen
2769 code_verifier_app, seine client_id und die app_redirect_uri.
2770 (10) Validierung durch den ZETA Guard: Der Guard verifiziert den code_verifier_app
2771 gegen die code_challenge_app (S256), validiert den DPoP Proof und verifiziert die Client
2772 Assertion (Prüfung der Key-Bindung aus der Dynamic Client Registration).
2773 (11) Erzeuge Policy Engine Input: Der ZETA Guard aggregiert die extrahierten Identitäts-
2774 Claims des Versicherten, die im Vorfeld erfassten Geräte-Integritätsdaten (Posture) sowie
2775 Kontextmetadaten zu einem Bewertungs-Payload.
2776 (12) POST /v1/data/authz: Der Guard ruft die PDP Policy Engine (OPA) auf und übergibt
2777 diesen Input.
2778 (13) Policy Decision: Die Policy Engine evaluiert die unternehmensweiten
2779 Zugriffsrichtlinien und gibt ihre Entscheidung (allow oder deny) an den Authorization
2780 Server zurück.

2781 **3. Ausstellung des ZETA Tokens (Fall: [Policy Decision allow])**

2782 (14)-(16) Ausstellung ZETA Token: Der Guard antwortet mit HTTP 200 und stellt das
2783 finale, DPoP-gebundene Access Token (token_type=DPoP), ein Refresh Token und wenn

2784 HW-Attestation verwendet wurde zusätzlich ein ZETA Guard Attestation Token
2785 (zeta_attestation_token) aus.

2786 **4. Fehlerpfad (Fall: [Policy Decision deny])**

2787 (17) 403 Forbidden: Entscheidet die Policy Engine auf Ablehnung, bricht der ZETA Guard
2788 den Prozess ab und beantwortet den Request des Clients aus Schritt (01) direkt mit HTTP
2789 403 Forbidden unter Angabe der spezifischen Ablehnungsgründe (reasons).

2791 **A_29672 -ZETA, Ablauf Token-Bezug und Ausstellung der ZETA Token**

2792 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-*
2793 *OIDC-Token-Bezug* unterstützen.

2794 [\leq]

2795 **A_29842 -Authorization Server, ZETA Attestation Token**

2796 Der Authorization Server MUSS, zusätzlich zur Ausstellung von Access und Refresh Token,
2797 ein ZETA Attestation Token gemäß [zeta-attestation-token.yaml] ausstellen, wenn bei der
2798 Client Registrierung eine Hardware basierte Attestation verwendet und eine erfolgreiche
2799 Nutzer-Authentifizierung durchgeführt wurde. [\leq]

2800 *5.3.2.5.6 Authentifizierung für eine andere Resource am gleichen ZETA Guard*

2801 Die Gültigkeitsdauer der Nutzer-Authentifizierung wird bei ZETA durch die Session des
2802 Authorization Servers abgebildet. Innerhalb einer gültigen Session kann der ZETA Client
2803 das Refresh Token einsetzen, um Zugriff auf andere durch den ZETA Guard geschützte
2804 Ressourcen (z. B. Notification Service Endpunkt oder Client Management Endpunkt) zu
2805 beantragen.

2806 **A_29843 -PDP Authorization Server, Zugriff auf andere Resource**

2807 Der Authorization Server MUSS Token Requests mit grant_type=refresh_token und
2808 Angabe der Parameter resource und scope verarbeiten, auch wenn das Refresh Token
2809 ursprünglich für eine andere resource und einen anderen scope beantragt wurde. [\leq]

2810 **A_29952 -PDP Authorization Server, Parameter audience anstatt resource im** 2811 **Token Request**

2812 Der Authorization Server MUSS Token Requests mit dem Parameter audience anstatt
2813 resource unterstützen und die audience als resource Parameter interpretieren.

2814 [\leq]

2815 **A_29844 -PDP Authorization Server, Abfrage der Policy Engine bei Einsatz des** 2816 **Refresh Tokens**

2817 Der Authorization Server MUSS Token Requests mit grant_type=refresh_token und
2818 nach erfolgreicher Verifikation des Refresh Token, über die Policy Engine abfragen, ob
2819 neue Token ausgestellt werden dürfen. [\leq]

2820 **A_29848 -PDP Authorization Server, Prüfung Session-Status bei Verwendung** 2821 **des Refresh Token**

2822 Der Authorization Server MUSS bei Token Requests mit grant_type=refresh_token vor
2823 der Ausstellung eines neuen Token-Sets prüfen, ob die referenzierte Session terminiert
2824 wurde. Ist die Session terminiert, MUSS der Authorization Server den Request ablehnen.

2825 [\leq]

2826 **5.3.3 Authentifizierung ohne Nutzer-Identität**

2827 Das folgende Sequenzdiagramm spezifiziert den Prozess zur Autorisierung von Client-
2828 Instanzen bei fachlichen Aufrufen, die keinen unmittelbaren Nutzerkontext erfordern (z.
2829 B. Abfrage des Verzeichnisdienstes). Der Nachweis der Berechtigung stützt sich
2830 ausschließlich auf die kryptografische Identität und die Geräteintegrität (Posture) des
2831 anfragenden ZETA Clients.

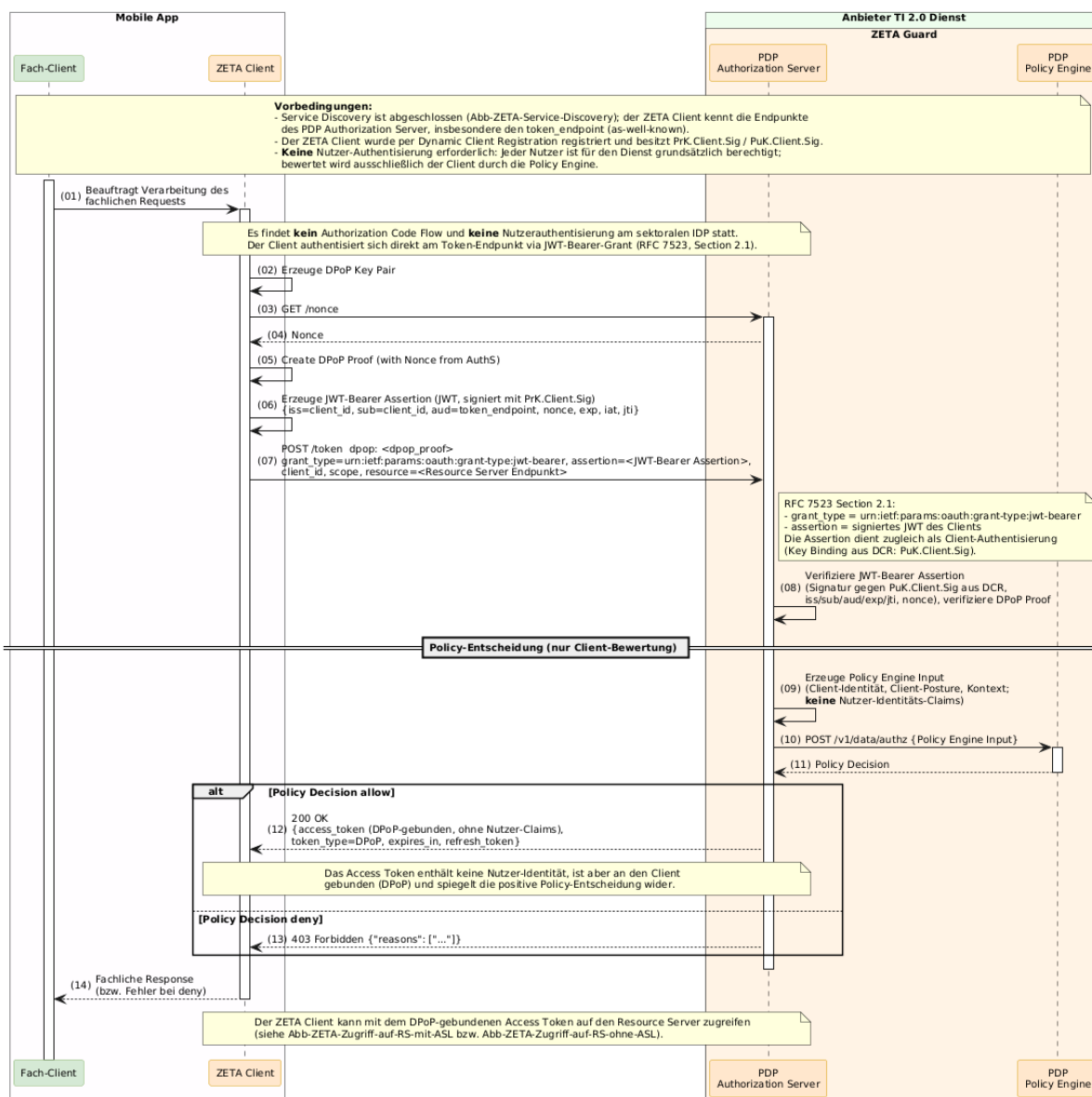


Abbildung 25 Abb-ZETA-OAuth-Client-Authentifizierung-ohne-Nutzer

Vorbedingungen:

Die Service Discovery ist abgeschlossen; der ZETA Client kennt die Endpunkte des ZETA Guards (insbesondere den token_endpoint).
 Der ZETA Client wurde über die Dynamic Client Registration (DCR) registriert und besitzt ein gerätegebundenes Signaturschlüsselpaar (PrK.Client.Sig / PuK.Client.Sig).
 Für den aufgerufenen Fachdienst ist keine Nutzerauthentisierung erforderlich; bewertet wird ausschließlich der Client durch die Policy Engine.

1. Initiierung und Vorbereitung der Client-Assertion (Schritte 01 bis 06)

- (01) Beauftragung: Der Fach-Client beauftragt den ZETA Client mit der Absicherung eines maschinellen Requests.
- (02) DPoP-Schlüsselpaar erzeugen: Der ZETA Client generiert im Arbeitsspeicher ein flüchtiges DPoP-Schlüsselpaar (PrK.DPoP.Sig, PuK.DPoP.Sig).
- (03) & (04) Nonce-Bezug: Der ZETA Client ruft proaktiv eine serverseitige Nonce beim ZETA Guard ab (GET /nonce), um Replay-Angriffe auf den nachfolgenden DPoP-Proof zu verhindern.

2832
 2833
 2834
 2835
 2836
 2837
 2838
 2839
 2840
 2841
 2842
 2843
 2844
 2845
 2846
 2847
 2848

2849 (05) DPoP Proof erstellen: Der Client erstellt den DPoP Proof unter Einbindung der
2850 erhaltenen Nonce.

2851 (06) JWT-Bearer Assertion erzeugen: Der Client erstellt ein selbst-signiertes JWT gemäß
2852 RFC 7523 (Section 2.1) inklusive der Nonce und signiert dieses mit seinem privaten
2853 Instanz-Schlüssel (PrK.Client.Sig). Die Claims iss und sub entsprechen der eigenen
2854 client_id, der Claim aud der URL des Token-Endpunkts des ZETA Guards.

2855 **2. Token Request am ZETA Guard (Schritte 07 und 08)**

2856 (07) POST /token (JWT-Bearer-Grant): Der ZETA Client ruft den Token-Endpunkt des
2857 Guards auf (POST /token). Er setzt den HTTP-Header DPoP und beantragt über den Grant-
2858 Typ urn:ietf:params:oauth:grant-type:jwt-bearer ein Access Token. Im Body überträgt er
2859 die erzeugte assertion, seine client_id, den angeforderten scope sowie die Ziel-audience
2860 (im Bild noch resource).

2861 (08) Validierung am AuthS: Der Authorization Server verifiziert die Signatur der JWT-
2862 Bearer Assertion gegen den bei der DCR hinterlegten öffentlichen Schlüssel
2863 (PuK.Client.Sig). Er prüft die zeitlichen Claims (iat, exp), schützt über den Claim jti vor
2864 Replays und validiert den DPoP Proof.

2865 **3. Reine Client-Policy-Entscheidung (Schritte 09 bis 11)**

2866 (09) Erzeuge Policy Engine Input: Da kein Nutzer involviert ist, aggregiert der Guard
2867 ausschließlich die Client-Identität, die übermittelten Posture-Daten des Geräts sowie
2868 Kontextmetadaten zu einem Bewertungs-Input. Nutzer-Identitäts-Claims werden nicht
2869 erstellt.

2870 (10) & (11) Policy-Abfrage: Der Guard ruft die PDP Policy Engine auf (POST
2871 /v1/data/authz). Diese bewertet den Gerätezustand und gibt ihre Entscheidung (allow
2872 oder deny) zurück.

2873 **4. Token-Ausstellung & Rückgabe (Schritte 12 bis 14)**

2874 (12) 200 OK (Erfolgspfad): Bei positiver Policy-Entscheidung stellt der ZETA Guard ein
2875 DPoP-gebundenes Access Token (token_type=DPoP) und ein Refresh Token aus. Das
2876 Access Token enthält ausschließlich Client-Claims und keine Nutzer-Identität.

2877 (13) 403 Forbidden (Fehlerpfad): Entscheidet die Policy Engine auf Ablehnung, abweist
2878 der Guard den Request unter Angabe der Ablehnungsgründe (reasons).

2879 (14) Fachliche Rückgabe: Der ZETA Client quittiert dem aufrufenden Fach-Client den
2880 Vorgang (entweder durch Bereitstellung des abgesicherten Kanals/Tokens oder durch
2881 Weitergabe des Fehlers).

2883 **A_29733 -ZETA, Ablauf OAuth Client Authentifizierung ohne Nutzer**

2884 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb-ZETA-
2885 OAuth-Client-Authentifizierung-ohne-Nutzer* unterstützen. [**<=**]

2886 **5.3.4 Dienst-zu-Dienst Kommunikation**

2887 Für die sichere Maschine-zu-Maschine Interaktion zwischen Backends wird die Workload
2888 Identity Federation zur Authentifizierung verwendet. Ein Backend-Dienst authentifiziert
2889 sich mit einem Subject Token am token_endpoint des Ziel-Dienstes und erhält nach
2890 erfolgreicher Prüfung ein Access Token für den Zugriff auf den Ziel-Dienst.

2891 Wenn eine ZETA Guard geschützte Workload (z. B. der Resource Server) auf einen
2892 externen Dienst zugreift, der eine Authentifizierung gemäß Workload Identity Federation
2893 verlangt, dann stellt der ZETA Guard Authorization Server als OpenID Provider die
2894 erforderlichen Subject Token aus. Der ZETA Guard erstellt regelmäßig einen cronjob Pod,
2895 der ein Subject Token vom Authorization Server bezieht und dieses per Token Exchange
2896 beim Zieldienst in ein Access Token tauscht. Das Access Token wird der Workload als
2897 Secret bereitgestellt und jeweils vor Ablauf der Gültigkeit durch erneuten Start des
2898 cronjob Pods und Token Exchange erneuert. Dadurch hat die Workload permanent Zugriff
2899 auf ein gültiges Access Token und kann mit dem externen Ziel-Dienst kommunizieren.

2900 Wenn ein ZETA Guard geschützter Resource Server als Ziel-Dienst genutzt wird, dann
 2901 übernimmt der ZETA Guard Authorization Server die Rolle eines Secure Token Service in
 2902 der Workload Identity Federation und tauscht beim Token Exchange das Subject Token
 2903 eines anfragenden Dienstes gegen ein Access Token für den Zugriff auf den Resource
 2904 Server.

2905

2906 **A_28431 -ZETA Guard, Ablauf Dienst-zu-Dienst Kommunikation**

2907 Der ZETA Guard MUSS den Ablauf gemäß Abbildung *Abb-ZETA-Dienst-zu-Dienst-*
 2908 *Kommunikation* unterstützen.[<=]

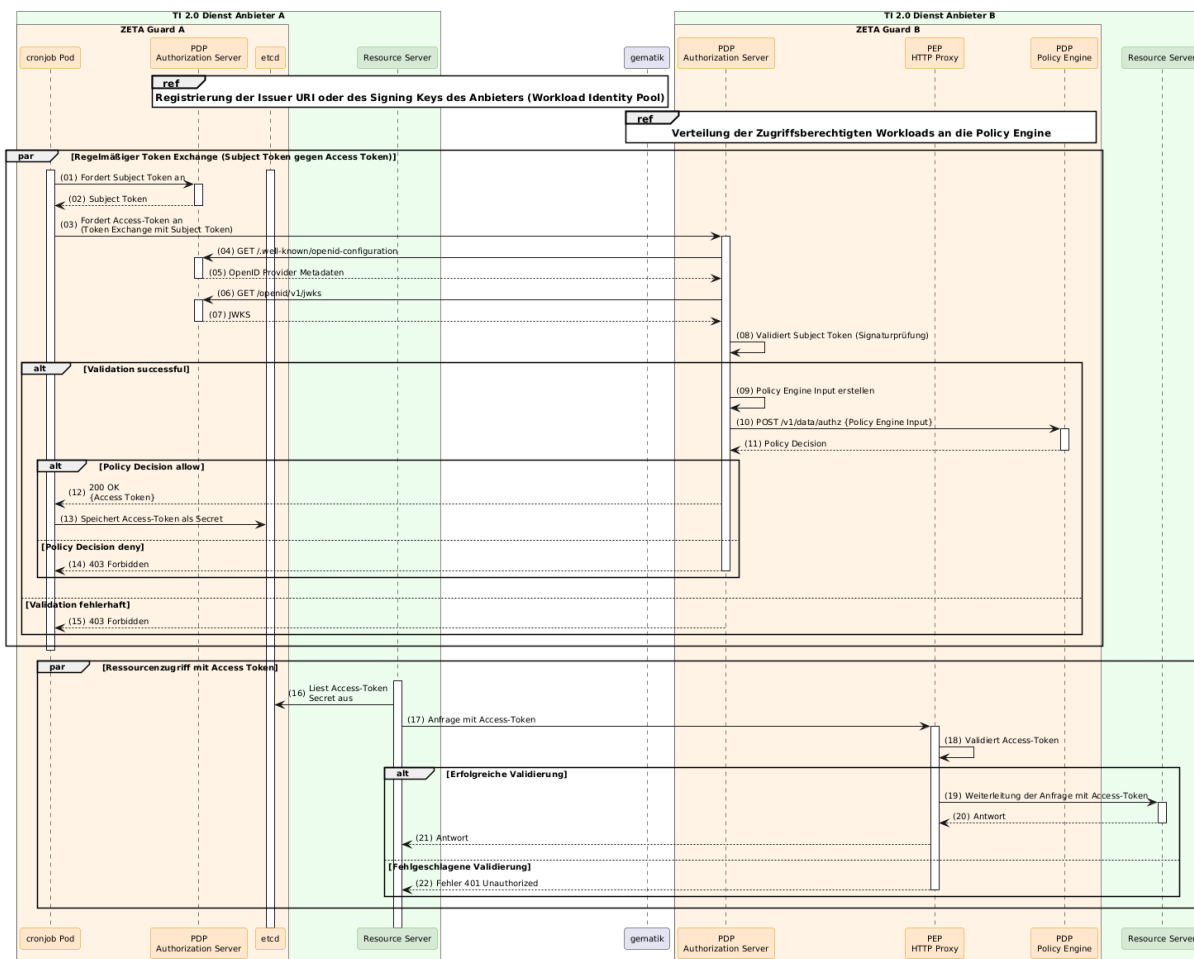


Abbildung 26 : Abb-ZETA-Dienst-zu-Dienst-Kommunikation

2909
 2910 Voraussetzung für die Dienst-zu-Dienst-Kommunikation ist:
 2911

- Die Registrierung der Issuer URI oder des Signing Keys des Anbieters im Workload Identity Pool (unter Einbindung der gematik).
- Die Verteilung der zugriffsberechtigten Workloads an die Policy Engine von Anbieter B.

2916 **Regelmäßiger Token Exchange (Subject Token gegen Access Token)**

2917 Dieser parallele Block (par) beschreibt, wie Anbieter A ein gültiges Access-Token von
 2918 Anbieter B erhält:

2919 Lokaler Token-Abruf (01-02): Der cronjob Pod fordert ein lokales Subject Token vom
 2920 eigenen PDP Authorization Server (Anbieter A) an und erhält dieses.

2921 Token-Austausch-Anfrage (03): Der cronjob Pod sendet dieses Subject Token an den PDP

2922 Authorization Server von Anbieter B, um es gegen ein Access-Token einzutauschen.
2923 Metadaten- & Schlüsselabruf (04-07): Zur Verifizierung des Tokens ruft der
2924 Autorisierungsserver von Anbieter B die OpenID-Konfiguration und die Signaturschlüssel
2925 (JWKS) direkt vom Autorisierungsserver von Anbieter A ab.
2926 Validierung (08): Der Server von Anbieter B validiert die Signatur des Subject Tokens.
2927 Falls die Validierung erfolgreich ist wird ein Input für die Policy Engine erstellt (09).
2928 Dieser wird per POST-Request an die PDP Policy Engine übermittelt (10), welche eine
2929 Entscheidung zurückgibt (11).
2930 Entscheidung „allow“ (12-13): Anbieter B antwortet mit 200 OK und liefert das Access
2931 Token aus. Der cronjob Pod von Anbieter A speichert dieses Token als Secret im lokalen
2932 etcd (oder in einem anderen sicheren Speicher).
2933 Entscheidung „deny“ (14): Es wird ein 403 Forbidden zurückgegeben.
2934 Falls die Validierung fehlschlägt (15): Es wird direkt ein 403 Forbidden an den cronjob Pod
2935 zurückgesendet.

2936 **Ressourcenzugriff mit Access Token**

2937 Der zweite parallele Block (par) beschreibt den eigentlichen Zugriff auf die geschützte
2938 Ressource:
2939 Token auslesen (16): Der Resource Server von Anbieter A liest das zuvor im etcd
2940 gespeicherte Access-Token aus.
2941 Anfrage senden (17): Der Resource Server von Anbieter A stellt eine Anfrage inklusive
2942 des Access-Tokens an den PEP HTTP Proxy von Anbieter B.
2943 Token-Prüfung (18): Der Proxy validiert das Access-Token.
2944 Abzweigung (alt-Block):
2945 Erfolgreiche Validierung (19-21): Die Anfrage wird an den eigentlichen Resource Server
2946 von Anbieter B weitergeleitet. Dieser antwortet, und die Antwort wird über den Proxy an
2947 den Resource Server von Anbieter A zurückgegeben.
2948 Fehlgeschlagene Validierung (22): Der Proxy blockiert die Anfrage und liefert einen Fehler
2949 401 Unauthorized zurück.

2950 **5.3.4.1 Ausgehende Verbindungen der Dienst-zu-Dienst Kommunikation** 2951 **mit ZG Client**

2952 *5.3.4.1.1 Motivation und Abgrenzung*

2953 Die Dienst-zu-Dienst-Kommunikation funktioniert auf Basis eines regelmäßig laufenden
2954 Cronjob-Pods, der ein Subject Token bezieht, per Token Exchange gegen ein Access
2955 Token tauscht und dieses als Secret für die Workload bereitstellt. Dieses Muster eignet
2956 sich für Zielsysteme, die selbst Workload Identity Federation (WIF) gemäß RFC 8693
2957 anbieten, erfordert jedoch, dass jede aufrufende Workload den Lebenszyklus des Secrets
2958 (Ablage, Rotation, Aktualisierung) kennt und im Fehlerfall behandelt.

2959 Um Resource Server und andere ZETA Guard Komponenten (z. B. den Telemetriedaten-
2960 Service) vollständig von dieser Aufgabe zu entlasten, wird eine neue Komponente ZG
2961 Client (ZETA Guard Client) eingeführt. Der ZG Client übernimmt für ausgehende Aufrufe
2962 an externe Dienste - diese müssen kein TI-Dienst sein - alle notwendigen Schritte der
2963 Authentifizierung mittels Workload Identity Federation. Die aufrufende Fachkomponente
2964 sendet ausschließlich den fachlichen Request an den ZG Client; dieser reichert den
2965 Request um ein gültiges Access Token an und leitet ihn an den Zieldienst weiter. Die
2966 Fachkomponente muss weder Token-Beschaffung noch -Erneuerung noch
2967 Schlüsselmaterial verwalten.

2968 Der ZG Client ist damit das Pendant zum ZETA Client (Kapitel 5.4) auf der Seite
2969 ausgehender Server-zu-Server-Kommunikation. Während der ZETA Client die Absicherung
2970 von Anfragen eines Clientsystems gegenüber einem ZETA Guard übernimmt, übernimmt
2971 der ZG Client die Absicherung ausgehender Anfragen einer serverseitigen Workload
2972 gegenüber einem beliebigen externen Dienst.

2973 5.3.4.1.2 Architektur

2974 Der ZG Client wird als eigenständiger Prozess (z. B. Sidecar-Container im selben Pod oder
2975 lokal erreichbarer Proxy)

2976 neben der Fachkomponente betrieben. Er stellt einen lokalen Endpunkt (Egress-Proxy)
2977 bereit, an den die

2978 Fachkomponente ihre ausgehenden Requests adressiert. Der ZG Client:

- 2979 • ermittelt anhand der Ziel-Adresse des Requests die zuständige Konfiguration
2980 (Zieldienst, erforderlicher Scope/Audience, Token-Endpunkt des Zieldienstes, Subject-
2981 Token-Quelle),
- 2982 • bezieht bei Bedarf ein Subject Token vom lokalen ZETA Guard Authorization Server
2983 und tauscht das Subject Token per Token Exchange (RFC 8693) beim Zieldienst gegen
2984 ein Access Token,
- 2985 • cached das Access Token bis kurz vor Ablauf der Gültigkeit und erneuert es danach
2986 automatisch im Hintergrund,
- 2987 • fügt das Access Token dem fachlichen Request hinzu und leitet diesen an den
2988 Zieldienst weiter,
- 2989 • gibt die Antwort des Zieldienstes unverändert an die Fachkomponente zurück.

2990 Im Unterschied zum Cronjob-Pod-Muster erfolgt die Token-Beschaffung beim ZG Client
2991 requestgetrieben bzw.
2992 proaktiv im Hintergrund innerhalb desselben Prozesses, ohne Ablage des Access Tokens
2993 als Kubernetes Secret.

2994 5.3.4.1.3 Konfiguration über die Policy Engine

2995 Die Konfiguration des ZG Client (zulässige Zieldienste, Scopes/Audiences, Token-
2996 Endpunkte, Caching-Parameter)

2997 erfolgt nicht statisch, sondern wird zur Laufzeit über die Policy Engine (PAP/PDP)
2998 bereitgestellt. Dadurch lassen sich

2999 Berechtigungen für ausgehende Verbindungen zentral pflegen, ohne Deployments der
3000 Fachkomponenten anzupassen,
3001 und Änderungen wirken sich ohne Neustart des ZG Client aus.

3002 5.3.4.1.4 Ablauf

3003 Das Sequenzdiagramm beschreibt die maschinelle Dienst-zu-Dienst-Kommunikation über
3004 System- und Anbietergrenzen hinweg (hier: von Anbieter A zu Anbieter B) mittels ZG
3005 Client.

3006 Es visualisiert das Konzept der Workload Identity Federation. Dabei agiert der ZETA Guard
3007 des aufrufenden Anbieters (A) als lokaler Identity Provider (IdP) und stellt ein Subject
3008 Token aus. Dieses Token wird über einen OAuth 2.0 Token Exchange beim Secure Token
3009 Service (STS) des fremden Anbieters (B) gegen ein dort gültiges Access Token
3010 eingetauscht. Den Prozess steuert der ZETA Guard Client (ZG Client), der typischerweise
3011 als Egress-Proxy (Sidecar) oder SDK für die Fachkomponente fungiert.

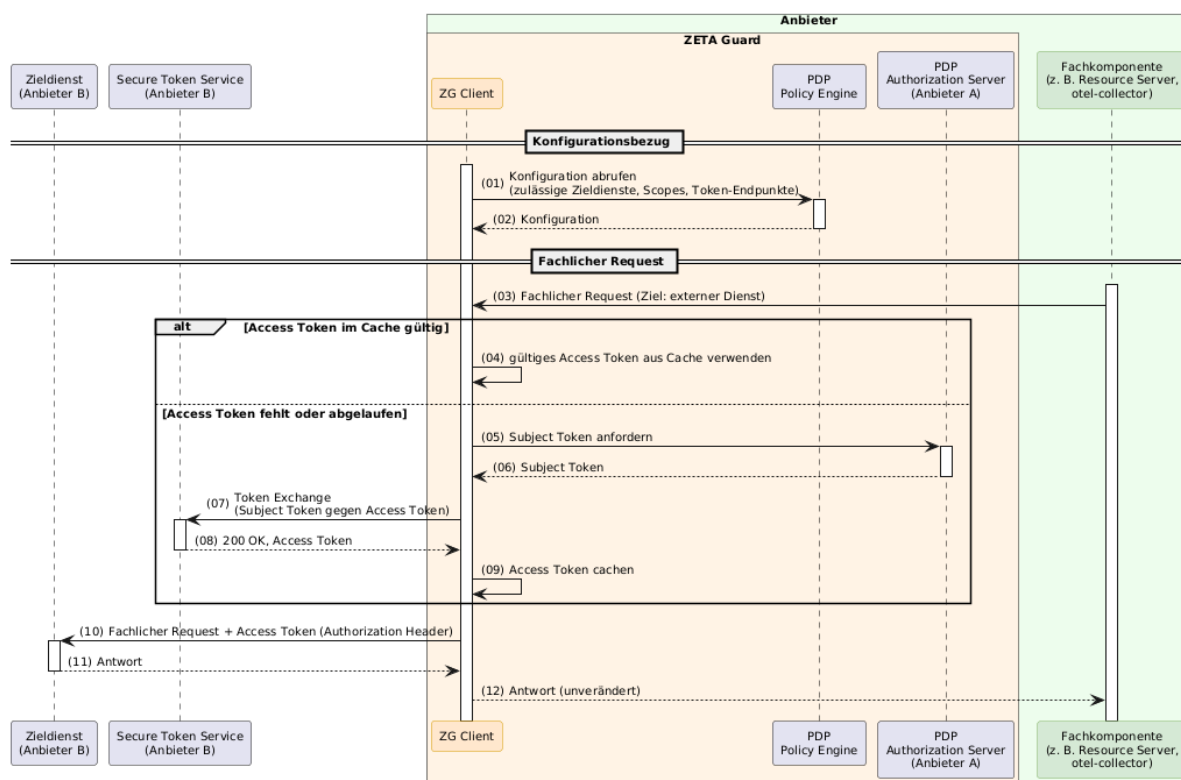


Abbildung 27 Abb-ZETA-Dienst-zu-Dienst-Kommunikation-mit-ZG-Client

1. Konfigurationsbezug

Bevor der ZG Client externe Aufrufe absichern kann, muss er die geltenden Föderationsrichtlinien kennen. Die Konfiguration wird regelmäßig aktualisiert.
 (01) Konfiguration abrufen: Der ZG Client fragt die lokale PDP Policy Engine (von Anbieter A) nach den aktuellen Konfigurationsdaten ab.
 (02) Konfiguration: Die Policy Engine liefert eine Liste der zulässigen Zieldienste, die erlaubten Scopes sowie die Endpunkt-URLs der fremden Secure Token Services (STS), denen vertraut wird.

2. Initiierung des fachlichen Requests

(03) Fachlicher Request: Eine interne Fachkomponente (z. B. ein Resource Server oder ein Telemetrie-Dienst wie der otel-collector) initiiert einen Aufruf, der an einen externen Dienst (Zieldienst bei Anbieter B) gerichtet ist. Der ZG Client fängt diesen Request ab bzw. nimmt ihn entgegen, um die Autorisierungsschicht transparent hinzuzufügen.

3. Token-Management und Token Exchange

Der ZG Client prüft nun, ob er bereits über ein gültiges Access Token für das angeforderte externe Zielsystem verfügt.

Pfad A: Access Token im Cache gültig

(04) Token aus Cache verwenden: Liegt ein gültiges Access Token für den Zieldienst im lokalen Cache des ZG Clients vor, wird dieses direkt für den Aufruf verwendet (Weiter mit Schritt 10).

Pfad B: Access Token fehlt oder ist abgelaufen (Workload Identity Federation)

(05) Subject Token anfordern: Der ZG Client ruft den lokalen PDP Authorization Server (ZETA Guard von Anbieter A) auf und fordert ein Identitätstoken für den eigenen Dienst an.

(06) Subject Token: Der lokale AuthS stellt ein signiertes Subject Token aus. Dieses Token bestätigt die Identität der lokalen Fachkomponente gegenüber der Außenwelt.

(07) Token Exchange: Der ZG Client ruft den externen Secure Token Service (STS) von Anbieter B auf. Er nutzt den OAuth 2.0 Token Exchange (RFC 8693) und reicht das

3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041

3042 Subject Token ein, um im Gegenzug ein lokales Access Token von Anbieter B zu erhalten.
3043 (08) 200 OK (Access Token): Der fremde STS validiert das Subject Token (basierend auf
3044 der Föderations-Vertrauensstellung), akzeptiert es und stellt ein neues Access Token aus,
3045 das für den Zugriff auf den Zieldienst bei Anbieter B berechtigt.

3046 (09) Access Token cachen: Der ZG Client speichert das neu erhaltene Access Token in
3047 seinem lokalen Cache, um Folgeaufrufe zu beschleunigen.

3048 **4. Ausführung des fachlichen Requests**

3049 (10) Fachlicher Request + Access Token: Der ZG Client fügt das (entweder aus dem
3050 Cache stammende oder frisch eingetauschte) Access Token als Authorization: DPoP
3051 <Token> Header in den ursprünglichen HTTP-Request ein und leitet diesen an den
3052 externen Zieldienst (Anbieter B) weiter.

3053 (11) Antwort: Der externe Zieldienst verarbeitet die Anfrage und sendet die fachliche
3054 HTTP-Antwort zurück an den ZG Client.

3055 (12) Antwort (unverändert): Der ZG Client reicht die empfangene Antwort transparent
3056 und unverändert an die aufrufende interne Fachkomponente zurück.

3057 *5.3.4.1.5 Anforderungen*

3058 **A_29878 -ZETA Guard, Unterstützung des Ablaufs der Dienst-zu Dienst** 3059 **Kommunikation mit ZG Client**

3060 Der ZETA Guard MUSS für ausgehende Dienst-zu-Dienst-Kommunikation die
3061 Bereitstellung einer ZG Client-Komponente
3062 gemäß Abbildung *Abb-ZETA-Dienst-zu-Dienst-Kommunikation-mit-ZG-Client* unterstützen.
3063 [**<=**]

3064 **A_29879 -ZG Client, Transparenz für Fachkomponente**

3065 Der ZG Client MUSS fachliche Requests einer aufrufenden Fachkomponente
3066 entgegennehmen, ohne dass diese
3067 Token-Beschaffung, Token-Erneuerung oder Schlüsselmaterial selbst verwalten muss.
3068 [**<=**]

3069 **A_29880 -ZG Client, Authentifizierung mittels Workload Identity Federation**

3070 Der ZG Client MUSS sich gegenüber externen Diensten mittels Workload Identity
3071 Federation gemäß RFC
3072 8693 (Token Exchange) authentifizieren. [**<=**]

3073 *Hinweis: Der externe Zieldienst muss nicht zwingend ein ZETA Guard geschützter TI-*
3074 *Dienst sein. Es werden jedoch nur externe Dienste unterstützt, zu denen alle TI 2.0*
3075 *Fachdienste des gleichen Typs eine Dienst-zu-Dienst Kommunikation benötigen.*

3076 **A_29881 -ZG Client, Konfiguration über die Policy Engine**

3077 Der ZG Client MUSS seine Konfiguration (zulässige Zieldienste, Scopes/Audiences, Token-
3078 Endpunkte sowie
3079 Caching-Parameter) zur Laufzeit von der Policy Engine beziehen und für die in der
3080 Response der Policy Engine angegebene Gültigkeitsdauer cachen. Nach Ablauf der
3081 Gültigkeitsdauer MUSS der ZG Client die aktuelle Konfiguration von der Policy Engine
3082 beziehen. Eine ausschließlich statische, lokale Konfiguration ist nicht zulässig. [**<=**]

3083 **A_29882 -ZG Client, Aktualisierung der Konfiguration**

3084 Der ZG Client MUSS Änderungen der von der Policy Engine bereitgestellten Konfiguration
3085 zeitnah übernehmen, ohne dass hierfür ein Neustart des ZG Client erforderlich ist. [**<=**]

3086 **A_29883 -ZG Client, Caching und Erneuerung von Access Tokens**

3087 Der ZG Client MUSS bezogene Access Tokens bis zu deren Ablauf zwischenspeichern und
3088 VOR Ablauf der Gültigkeit automatisiert erneuern, sodass für die Fachkomponente
3089 durchgehend ein gültiges Access Token zur Verfügung steht. [**<=**]

3090 **A_29884 -ZG Client, Fehlerbehandlung**

3091 Lehnt der Secure Token Service des Zieldienstes den Token Exchange ab oder schlägt die
3092 Validierung des Subject Tokens fehl, so MUSS der ZG Client den Fehler unverändert (inkl.
3093 HTTP-Statuscode) an die aufrufende Fachkomponente weiterleiten. [<=]

3094 **A_29885 -ZG Client, Geltungsbereich der Berechtigung**

3095 Der ZG Client MUSS Access Tokens ausschließlich für die in der Policy-Engine-
3096 Konfiguration als zulässig hinterlegten Zieldienste und Scopes beziehen und verwenden.
3097 [<=]

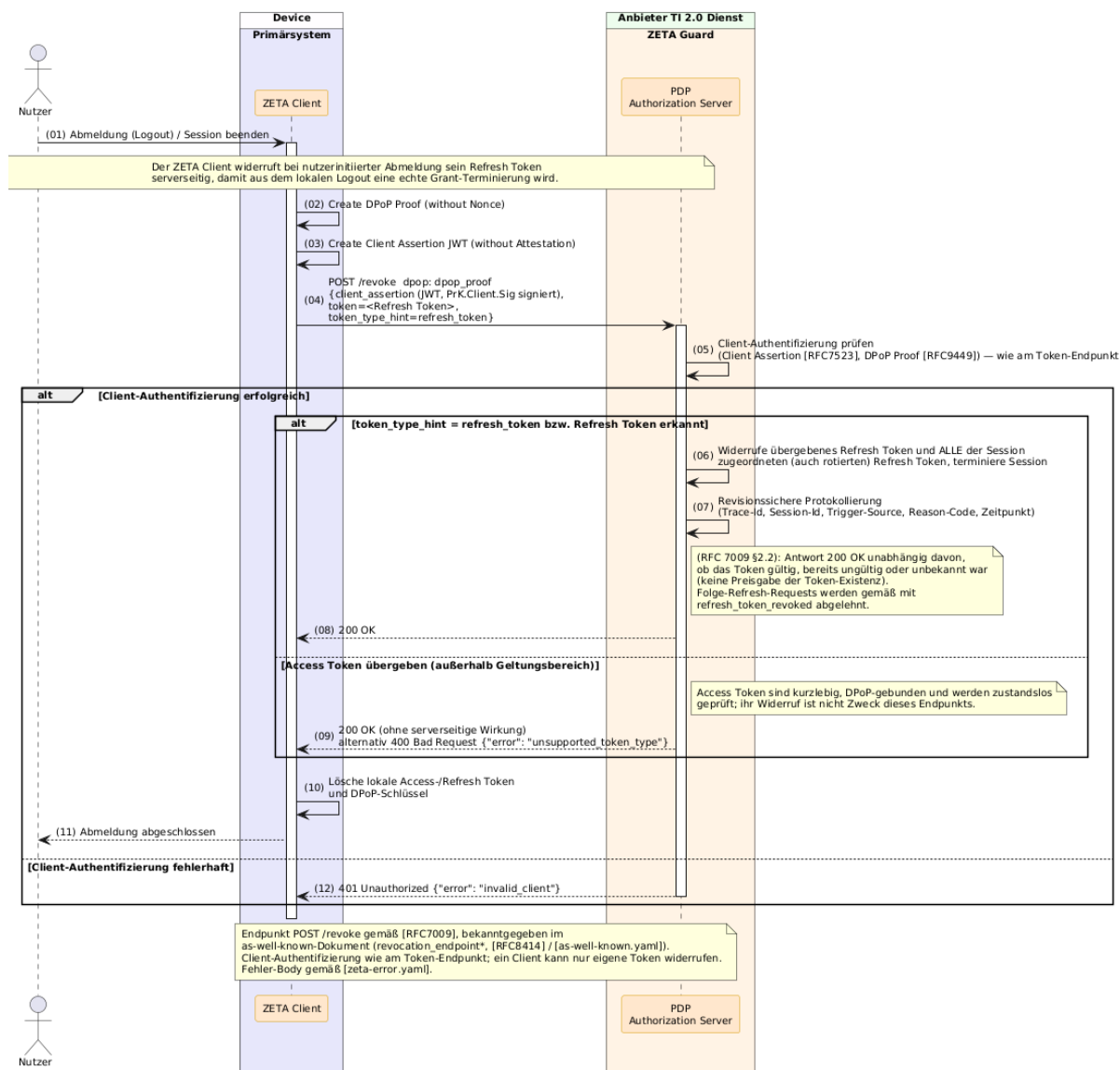
3098 **A_29886 -ZG Client, Erreichbarkeit**

3099 Der ZG Client MUSS für die ihm zugeordnete(n) Fachkomponente(n) als lokal erreichbarer
3100 Egress-Endpunkt bereitgestellt werden. [<=]

3101 **5.3.5 Token Revocation**

3102 Das Sequenzdiagramm spezifiziert den Prozess der Token Revocation (Token-Widerruf),
3103 welcher bei einer nutzerinitiierten Abmeldung (Logout) durchlaufen wird.

3104 Das Diagramm setzt die Vorgaben aus RFC 7009 (OAuth 2.0 Token Revocation) im
3105 Kontext der ZETA-Architektur um. Ziel ist es, aus einem reinen lokalen Logout der App
3106 eine echte, serverseitige Terminierung der Sitzung (Grant-Terminierung) beim ZETA
3107 Guard zu machen.



3108
3109

Abbildung 28 Abb-ZETA-Token-Revocation

3110 Ausgangssituation: Der Nutzer ist am ZETA Client angemeldet, die App besitzt ein
3111 gültiges Refresh Token und ein zugehöriges DPoP-Schlüsselpaar.

3112 **1. Initiierung und Request-Vorbereitung**

3113 (01) Abmeldung: Der Nutzer initiiert in der Benutzeroberfläche des Primärsystems (ZETA
3114 Client) die Abmeldung. Gemäß Anforderung A_30002 ist der Client nun verpflichtet, das
3115 Refresh Token serverseitig zu widerrufen.
3116 (02) DPoP Proof: Der Client erstellt einen DPoP Proof für den /revoke-Endpoint. (Da dieser
3117 Endpoint in der Regel keine Nonce erfordert, wird der Proof ohne Nonce generiert).
3118 (03) Client Assertion: Der Client generiert ein JWT (gemäß RFC 7523) zur eigenen
3119 Authentisierung und signiert dieses mit seinem privaten Instanzschlüssel (PrK.Client.Sig).
3120 Hardware-Attestierungsdaten (Evidence) sind für diesen Endpoint nicht erforderlich.
3121 (04) POST /revoke: Der Client sendet den Widerrufs-Request an den Token Revocation
3122 Endpoint des ZETA Guards. Er übergibt den DPoP Proof im Header sowie die
3123 client_assertion, das zu widerrufende token (das Refresh Token) und den Hinweis
3124 token_type_hint=refresh_token im Body.

3125 **2. Authentisierung am ZETA Guard**
3126 (05) Client-Authentifizierung prüfen: Der ZETA Guard prüft die Identität und Berechtigung
3127 des aufrufenden Clients. Analog zum Token-Endpoint werden die Client Assertion und der
3128 DPoP Proof kryptografisch validiert. (Ein Client darf ausschließlich seine eigenen Tokens
3129 widerrufen).
3130 Der weitere Ablauf verzweigt sich anhand des Ergebnisses der Client-Authentifizierung
3131 (A_29997).

3132 **3. Erfolgreiche Client-Authentifizierung (Hauptpfad)**
3133 Wenn die Authentifizierung des Clients erfolgreich war, prüft der Server, welcher Token-
3134 Typ widerrufen werden soll.

3135 **Pfad A: Refresh Token erkannt (Der Standardfall)**
3136 (06) Token und Session terminieren: Der ZETA Guard identifiziert das übergebene Refresh
3137 Token. Gemäß A_29998 widerruft er nicht nur dieses spezifische Token, sondern alle der
3138 aktuellen Session zugeordneten (auch bereits rotierten) Refresh Tokens. Die Sitzung wird
3139 serverseitig endgültig beendet.
3140 (07) Protokollierung: Der ZETA Guard protokolliert den Vorgang revisionsicher (inkl.
3141 Trace-Id, Session-Id, Trigger-Source und Reason-Code) (A_29857).
3142 (08) 200 OK: Der ZETA Guard antwortet mit HTTP 200 OK.
3143 (Wichtiger Hinweis gem. RFC 7009 §2.2 / A_29999: Diese Erfolgsmeldung wird immer
3144 gesendet, unabhängig davon, ob das Token gültig, bereits abgelaufen oder dem Server
3145 völlig unbekannt war. Dies verhindert das Ausspähen von Token-Zuständen durch
3146 Angreifer).

3147 **Pfad B: Access Token übergeben (Außerhalb des Geltungsbereichs)**
3148 (09) Umgang mit Access Tokens: ZETA Access Tokens sind extrem kurzlebig, zustandslos
3149 und DPoP-gebunden. Ein serverseitiger Widerruf (Revocation List) ist für sie nicht
3150 vorgesehen (A_30000). Sendet der Client versehentlich ein Access Token, antwortet der
3151 Server entweder mit 200 OK (ohne serverseitige Wirkung) oder weist den Request mit
3152 400 Bad Request ("unsupported_token_type") ab.

3153 **Abschluss auf Client-Seite**
3154 (10) Lokale Bereinigung: Unabhängig von der Antwort des Servers (solange es kein
3155 401/403 ist), löscht der ZETA Client nach dem Aufruf das Refresh Token, etwaige Access
3156 Tokens sowie die zugehörigen flüchtigen DPoP-Schlüssel aus seinem lokalen Speicher
3157 (A_30002).
3158 (11) Abmeldung abgeschlossen: Der ZETA Client meldet dem Nutzer den erfolgreichen
3159 Logout.
3160 **4. Fehlerhafte Client-Authentifizierung (Fehlerpfad)**
3161 (12) 401 Unauthorized: Schlägt die Signaturprüfung der Client Assertion oder des DPoP
3162 Proofs fehl (Schritt 05), bricht der ZETA Guard ab und antwortet mit 401 Unauthorized
3163 (Fehlercode invalid_client).

3165 **A_30001 -ZETA, Ablauf Token Revocation**
3166 Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung Abb-ZETA-
3167 Token-Revocation unterstützen.[<=]

3168 5.4 Clientsystem und ZETA Client

3169 Ein Clientsystem ist eine Softwarekomponente in der Verwendung eines Nutzers zum
3170 Ausführen fachlicher Anwendungsfälle z.B. als Primärsystem (PVS, AVS, LIS, KIS etc.) oder
3171 als Frontend des Versicherten (ePA-App, E-Rezept-App, TI-Messenger etc.). Dieses
3172 Clientsystem wird dem Nutzer durch einen Hersteller zur Verfügung gestellt.

3173 Ein ZETA Client ist ein Teil des Clientsystems, der die Kommunikation mit dem PDP und
3174 dem PEP eines Dienstes übernimmt.

3175 Mobile iOS und Android Apps werden unterstützt und setzen ebenfalls einen ZETA Client
3176 um. Anforderungen, die nur für diese Clientsysteme und ZETA Clients gelten werden,
3177 verwenden die Bezeichnung "mobiler Client" oder "mobiles Clientsystem" und "mobiler
3178 ZETA Client".

3179 **5.4.1 Hersteller**

3180 **A_25335 -Hersteller Clientsystem - Hinweise und Maßnahmen sicherer Betrieb**

3181 Der Hersteller eines Clientsystems MUSS den Nutzer über Maßnahmen zum sicheren
3182 Betrieb seines Clientsystems vor der Inbetriebnahme informieren und während des
3183 Betriebs stets zum Abruf durch den Nutzer bereithalten. [≤]

3184 **A_25336 -Hersteller Clientsystem - Regelmäßige Updates**

3185 Der Hersteller eines Clientsystems MUSS, solange das Produkt nicht abgekündigt ist, dem
3186 Nutzer regelmäßig (z. B. quartalsweise) Updates für das Clientsystem bereitstellen, um
3187 das Clientsystem dauerhaft auf dem Stand der Technik zu halten und Sicherheitslücken
3188 zu schließen. [≤]

3189 **A_25337 -Hersteller Clientsystem - Registrierung für product_id**

3190 Der Hersteller eines Clientsystems MUSS sich über einen organisatorischen Prozess bei
3191 der gematik für die Nutzung von Diensten, für welche Token abgerufen werden sollen,
3192 registrieren. Der Hersteller eines Clientsystems bekommt dabei eine "product_id"
3193 zugewiesen, die in jeder Instanz des Clientsystems verwendet werden MUSS. [≤]

3194 **A_29723 -Hersteller Clientsystem, Meldung von Client-Metadaten**

3195 Der Hersteller eines Clientsystems MUSS über einen Prozess der gematik die folgenden
3196 Client-Metadaten an die gematik melden:

- 3197 • fortlaufend im Betrieb alle Änderungen von aktiv unterstützten Versionen seines
3198 Clients (Claim product_version)
- 3199 • bei Nutzung von TPM Attestation, die kryptografischen Hashes der unveränderlichen
3200 Bestandteile (Immutable Code / Binärdateien) jeder gemeldeten Produktversion des
3201 Clients
- 3202 • bei mobilen Applikationen:
 - 3203 • die oidc_redirect_uri nach dem Schema /cb/<app>/oidc (für die Weiterleitung an
3204 den Redirection-Endpunkt des ZETA Guards),
 - 3205 • die app_redirect_uri nach dem Schema /cb/<app>/app (für die Weiterleitung an
3206 den Token-Endpunkt des ZETA Guards).

3207 [≤]

3208 **5.4.2 Verbindungsaufbau**

3209 Bevor ein ZETA Client auf geschützte Fachdienste zugreifen kann, muss der
3210 netzwerktechnische Verbindungsaufbau sowie die korrekte Adressierung und
3211 Autorisierung über Token-Strukturen sichergestellt werden. Die nachfolgenden
3212 Definitionen bilden die Grundlage für die Ausgestaltung der Token-Requests und den
3213 Einsatz des ZETA/ASL-Kanals.

3214 **5.4.2.1 Definition der Endpunkte**

3215 Für den Verbindungsaufbau und die Anforderung von Token werden die folgenden
3216 Endpunkte unterschieden:

3217 **Resource Server Endpunkt:** Der finale Endpunkt des aufgerufenen Fachdienstes. Die
3218 URL dieses Endpunkts MUSS mit dem Parameter resource der OAuth Protected Resource
3219 Metadata ([opr-well-known.yaml]) des jeweiligen Dienstes übereinstimmen.

3220 **ZETA/ASL Endpunkt:** Der fest definierte Endpunkt für die Terminierung des ZETA
3221 Application Security Layer (ASL) Protokolls. Die URL dieses Endpunkts MUSS nach dem
3222 Schema https://<hostname>/ASL gebildet werden, wobei <hostname> exakt dem
3223 Hostnamen des ermittelten Resource Server Endpunkts entspricht.

3224 **ZETA Guard Token Endpunkt:** Der Autorisierungsendpunkt des ZETA Guards
3225 (Authorization Server), an den der Token Exchange Request gesendet wird.

3226 **5.4.2.2 Adressierung und Gültigkeitsbereich (Audience und Scope)**

3227 Um sicherzustellen, dass Tokens nur für den vorgesehenen Empfänger gültig sind, muss
3228 bei der Token-Anforderung strikt zwischen Parametern des HTTP-Requests und den
3229 Claims innerhalb der JWT-Strukturen unterschieden werden.

3230 **Claim aud im Subject Token:** Das Subject Token wird an den ZETA Guard gesendet,
3231 damit dieser es auswertet. Folglich MUSS der Claim aud (Audience) innerhalb des Subject
3232 Tokens exakt die URL des ZETA Guard Token Endpunkts enthalten.

3233 **Parameter resource im Token Request (HTTP Body):** Dieser Parameter gemäß
3234 [RFC8707] enthält die absolute URL des aufgerufenen Endpunkts am Resource Server
3235 und wird der OAuth Protected Resource Metadata ([opr-well-known.yaml], Feld
3236 resource) entnommen. Der ZETA Guard ermittelt aus dem angeforderten resource über
3237 die Policy Engine den logischen Bezeichner der Zielressource und stellt das Access Token
3238 mit diesem Wert im Claim aud aus (siehe 5.4.2.4). Die Ressourcen-URL wird NICHT in den
3239 aud-Claim übernommen; sie wird beim späteren Aufruf der Ressource über den Claim htu
3240 des DPoP Proofs [RFC9449] an den konkreten Request gebunden.

3241 **Parameter audience im Token Request (HTTP Body):** Aus Gründen der
3242 Abwärtskompatibilität DARF ein ZETA Client anstelle von resource den Parameter
3243 audience (logischer Bezeichner) senden. In diesem Fall wird der angegebene Wert
3244 unverändert als aud-Claim in das Access Token übernommen (Legacy-Verhalten, siehe
3245 5.4.2.5). Der Parameter audience ist als veraltet gekennzeichnet; neue ZETA Clients
3246 SOLLEN ausschließlich resource verwenden.

3247 **Parameter scope im Token Request (HTTP Body):** Dieser Parameter MUSS die
3248 spezifischen Berechtigungen (Scopes) anfordern, die für den Zugriff auf den unter
3249 resource definierten Endpunkt erforderlich sind. Die unterstützten Scopes werden der
3250 OAuth Protected Resource Metadata ([opr-well-known.yaml], Feld scopes_supported)
3251 entnommen.

3252 **5.4.2.3 Zusammenhang der Token Exchange Komponenten**

3253 Stationäre ZETA Clients nutzen für den Bezug eines Access Tokens den OAuth 2.0 Token
3254 Exchange Grant [RFC 8693]. Ein vollständiger, valider Token Request an den ZETA Guard
3255 (POST /token) setzt sich aus vier ineinandergreifenden kryptografischen Komponenten
3256 zusammen:

3257 **Der Token Request (HTTP Body):** Die Klammer um alle Parameter. Er definiert über
3258 grant_type=urn:ietf:params:oauth:grant-type:token-exchange die Art der Anfrage,
3259 benennt über resource [RFC8707] die Endpunkt-URL der Zielressource und über scope die
3260 angeforderten Berechtigungen und bindet das Subject Token sowie die Client Assertion
3261 ein. Aus der angegebenen resource leitet der ZETA Guard über die Policy Engine den
3262 logischen aud-Claim ab; die resource-URL selbst wird nicht in den aud-Claim
3263 übernommen, sondern bestimmt den späteren Ziel-Endpunkt (Bezug zum DPoP-Claim
3264 htu, siehe 5.4.2.4).

3265 **Das Subject Token:** Ein JWT, welches den Kontext der Anfrage (z.B. den Nutzer oder die
3266 Session) repräsentiert. Es ist (wie oben beschrieben) über den aud-Claim an den ZETA
3267 Guard Token Endpunkt gebunden.

3268 **Die Client Assertion:** Ein JWT (`client_assertion_type=...jwt-bearer`), welches die
3269 Authentisierung und Integrität (Attestierung) des ZETA Clients gegenüber dem ZETA
3270 Guard nachweist. Es enthält das `client_statement` mit den plattformsspezifischen
3271 Integritätsnachweisen (z. B. Apple App Attest oder das Software-Fallback).

3272 **Der DPoP Proof für den Request:** Ein über den HTTP-Header (DPoP) übergebenes JWT.
3273 Es beweist, dass der Client im Besitz des privaten DPoP-Schlüssels ist. In diesem DPoP-
3274 Proof MUSS der Claim `htm` den Wert `POST` und der Claim `htu` die URL des ZETA Guard
3275 Token Endpunkts enthalten (da dieser Request an den Guard gerichtet ist).

3276 Hinweis: Über den Parameter `resource` können in getrennten Token Requests
3277 verschiedene Access Tokens für unterschiedliche Zielressourcen bezogen werden. Alle so
3278 bezogenen Access Tokens werden über denselben DPoP-Schlüssel an die Client-Instanz
3279 gebunden (`Claimjkt`); sie unterscheiden sich im logischen `aud-Claim`, den die Policy
3280 Engine je Zielressource vergibt.

3281 **5.4.2.4 Resultierende Token für den Aufruf von Fachdiensten**

3282 Nach erfolgreicher Validierung des Token Requests stellt der ZETA Guard ein Access
3283 Token aus. Der `aud-Claim` dieses Access Tokens enthält den von der Policy Engine
3284 festgelegten logischen Bezeichner der Zielressource, der aus dem im Request
3285 angegebenen Parameter `resource` abgeleitet wird. Tokens, die auf diesem Weg
3286 ausgestellt werden, tragen den Claim `ver` mit dem Wert `2`. Wurde der Token Request im
3287 Legacy-Verfahren über den Parameter `audience` gestellt, so entspricht der `aud-Claim`
3288 dem verbatim übernommenen `audience`-Wert und das Token trägt `ver` mit dem Wert `1`
3289 (siehe 5.4.2.5).

3290 Wenn der ZETA Client anschließend den eigentlichen Request an den Resource Server
3291 (oder den ZETA/ASL Endpunkt) durchführt, MUSS er einen neuen DPoP Proof generieren.
3292 Für diesen neuen DPoP Proof gilt zwingend: Der Claim `htu` MUSS exakt der URL des
3293 aufgerufenen Endpunkts entsprechen gemäß [RFC9449]; wurde im Token Request der
3294 Parameter `resource` angegeben, MUSS `htu` mit dieser URL übereinstimmen. Während
3295 der `aud-Claim` die Zielressource über ihren logischen Bezeichner benennt, adressiert
3296 der `htu-Claim` den konkret aufgerufenen Endpunkt und bindet das DPoP-gebundene
3297 Access Token an den tatsächlichen Request. Der `claim htm` MUSS der HTTP Methode des
3298 fachlichen Requests entsprechen.

3299 Wenn im OAuth Protected Resource Well-known JSON Dokument angegeben ist
3300 `zeta_asl_use: required_passthrough`, dann hat der ZETA Client beim Token Request
3301 neben dem Access und Refresh Token für den Zugriff auf den Resource Server auch ein
3302 zusätzliches Access Token für den Zugriff auf den /ASL Endpunkt erhalten (Parameter
3303 `asl_access_token`).

3304 **A_29862 -ZETA Client, zusätzliches DPoP Token bei ZETA/ASL Terminierung am 3305 Resource Server**

3306 Der ZETA Client MUSS beim durchführen eines fachlichen Requests zum Resource Server
3307 ein zusätzliches DPoP Token für den /ASL Endpunkt gemäß Tabelle Tab-ZETA-Token-
3308 Ausstellung-ASL-am-Resource-Server erzeugen und im äußeren HTTP Request im DPoP
3309 Header verwenden. [`<=`]

3310 **5.4.2.5 Versionierung des Token-Ausstellungsverfahrens**

3311 Da bereits produktive ZETA Clients und ZETA Guards im Einsatz sind, MUSS das Verfahren
3312 zur Ableitung des `aud-Claims` versioniert und abwärtskompatibel betrieben werden. Es
3313 werden zwei Vertragsversionen (Contract Versions) unterschieden:

- 3314 • Contract v1 (Legacy): Der ZETA Client sendet den Parameter audience (logischer
3315 Bezeichner). Der ZETA Guard übernimmt diesen Wert verbatim in denaud-Claim und
3316 stellt das Access Token mit dem Claimver: 1 aus. Fehlt der Claimver, ist das Token
3317 wiever: 1 zu behandeln.
- 3318 • Contract v2: Der ZETA Client sendet den Parameter resource [RFC8707] sowie scope.
3319 Der ZETA Guard leitet den logischenaud-Claim aus der Policy Engine Decision ([pdp-
3320 decision.yaml], Feldaud) ab und stellt das Access Token mit dem Claimver: 2 aus.
- 3321 Der ZETA Guard Authorization Server MUSS die von ihm unterstützten Vertragsversionen
3322 in seinem OAuth Authorization Server Well-known JSON Dokument ([as-well-
3323 known.yaml], Feld api_versions_supported) bekannt geben. Der ZETA Client MUSS
3324 dieses Feld vor dem Token Request auswerten und das Verfahren entsprechend wählen.
- 3325 Unterstützt der ZETA Guard Contract v2 und wird der Parameterresource für eine nicht
3326 unterstützte Zielressource angegeben, so MUSS der Authorization Server den Token
3327 Request mit dem Fehlerinvalid_target gemäß [RFC8707] ablehnen.
- 3328 Der PEP HTTP Proxy MUSS während der Übergangsphase sowohl Access Tokens mitver:
3329 1 als auch mit ver: 2 akzeptieren und die jeweils zutreffende Regel zum Audience-
3330 Matching anwenden.

3331 5.4.2.6 Anforderungen

3332

3333 **A_25338-01 -ZETA Client - Authentifizierung beim Token Exchange**

3334 Der ZETA Client MUSS sich gegenüber dem ZETA Guard Authorization Server beim Token
3335 Exchange mit einem JWT Bearer Token gemäß [RFC7523] und gemäß[client-
3336 assertion-jwt.yaml] authentifizieren. Dabei MUSS die von der gematik für das
3337 Clientsystem ausgestellte product_id nach [A_25337] und die Versionskennzeichnung
3338 des Clients nach folgendem Schema verwendet werden:

- 3339 • <product_id> gemäß Registrierung bei der gematik mit Länge maximal 20 Zeichen,
3340 Zeichenvorrat [0-9a-zA-Z\-.],
- 3341 • <product_version> gemäß Produktidentifikation mit Länge 1-20
3342 Zeichen, Zeichenvorrat [0-9a-zA-Z\-.].

3343 [**<=**]

3344 **A_25339 -ZETA Client - Exponential Backoff**

3345 Der ZETA Client SOLL bei Server-seitigen Fehlermeldungen, die auf eine Überlastung des
3346 Zielsystems schließen lassen (z. B. HTTP-status 5xx, 429 - too many requests etc.),
3347 erneute Verbindungsversuche nach dem Prinzip des Exponential Backoffs [ExpBack]
3348 durchführen.[**<=**]

3349 *Hinweis: Die Parameter für das Verfahren des Exponential Backoffs werden vom*
3350 *Hersteller des ZETA Clients festgelegt.*

3351 **A_27378-01 -ZETA Client - TLS**

3352 Der ZETA Client MUSS mindestens TLS 1.2 oder höher unterstützen.

3353 [**<=**]

3354 **A_25340-02 -ZETA Client- Zertifikatsprüfung**

3355 Der ZETA Client MUSS alle Zertifikate, die es aktiv verwendet (bspw. TLS-
3356 Verbindungsaufbau), auf Integrität und Authentizität prüfen. Falls ein Zertifikat ungültig
3357 ist, so MUSS der ZETA Client die von dem Zertifikat und den darin enthaltenen Attributen
3358 (bspw. öffentliche Schlüssel) abhängenden Arbeitsabläufe ablehnen.

3359 Der ZETA Client MUSS alle öffentlichen Schlüssel, die es verwenden will, auf eine positiv
3360 verlaufene Zertifikatsprüfung zurückführen können.

- 3361
3362 Die Zertifikatsprüfung MUSS folgende Prüfungen enthalten:
- 3363 • Der ZETA Client MUSS den Subject Alternative Name (SAN, Extension dNSName) des
3364 Zertifikats (oder falls nicht vorhanden den Common Name (CN)) mit dem erwarteten
3365 Hostnamen vergleichen.
 - 3366 • Der Client MUSS die Gültigkeitsdauer des Zertifikats (Nicht vor und Nicht nach)
3367 überprüfen.
 - 3368 • Vertrauenswürdigkeit der Zertifikatskette: Sicherstellen, dass die Kette zu einer
3369 vertrauenswürdigen Root-CA führt.
 - 3370 • Revocation-Prüfung für das End-Entity-Zertifikat (Server-Zertifikat): Der Client MUSS
3371 den Widerrufsstatus prüfen. Hierbei SOLL primär eine vom Server bereitgestellte
3372 OCSP-Response (OCSP Stapling) ausgewertet werden. Fehlt diese, MUSS die Prüfung
3373 über gecachte CRLs oder OCSP-Responses erfolgen.
 - 3374 • Revocation-Prüfung für Sub-CA-Zertifikate: Der Client MUSS den Widerrufsstatus der
3375 Sub-CAs in der Kette prüfen. Um Lastspitzen und Latenzen zu vermeiden, SOLLEN
3376 hierfür vom Betriebssystem bereitgestellte Mechanismen (z. B. CRLite, lokale Trust-
3377 Stores) oder serverseitiges Multi-Stapling (TLS 1.3) verwendet werden. Falls Live-
3378 Abfragen (OCSP/CRL) für Sub-CAs notwendig sind, MÜSSEN deren Antworten zwingend
3379 entsprechend ihrer Gültigkeitsdauer (Feld nextUpdate) lokal gecacht werden.
- 3380 Der ZETA Client MUSS die Root-Zertifikate, die von den Mitgliedern des [CAB-Forum]
3381 ausgestellt werden, als Vertrauensanker standardmäßig unterstützen. Dies impliziert:
- 3382 • Eine Standard-Liste von [CAB-Forum] Root-Zertifikaten ist im ZETA Client enthalten
3383 und wird regelmäßig aktualisiert oder der ZETA Client verwendet den Truststore des
3384 Betriebssystems.

3385 [**<=**]

3386 **A_27379-02 -ZETA Client - TLS OCSP Stapling Unterstützung**

3387 Der ZETA Client MUSS beim TLS Verbindungsaufbau OCSP Stapling erkennen und nutzen,
3388 wenn es vom Server bereitgestellt wird.

3389 Der ZETA Client MUSS die Gültigkeit der OCSP-Antwort (Signatur, Signierer) prüfen.

3390 Der ZETA Client MUSS die Gültigkeitsdauer der OCSP-Antwort prüfen.

3391 Der ZETA Client MUSS überprüfen, ob die OCSP-Antwort zum angefragten Zertifikat passt.
3392 OCSP-Antworten MÜSSEN im Cache für die Zeit bis NextUpdate der OCSP Response
3393 gespeichert werden, um unnötige Abfragen zu vermeiden. Die Dauer der Speicherung im
3394 Cache MUSS konfigurierbar sein und mindestens 1 Stunde betragen.

3395 Wenn OCSP Stapling nicht angeboten wird, MUSS der ZETA Client entweder die CRL laden
3396 oder den OCSP Responder des Zertifikats direkt abfragen.

3397 Wenn weder eine OCSP Response noch eine CRL verfügbar sind, MUSS der ZETA Client
3398 den Verbindungsaufbau abbrechen.

3399 [**<=**]

3400 **A_26681-02 -ZETA Client - Umsetzen eines ZETA/ASL-Kanals**

3401 Der ZETA Client MUSS einen ZETA/ASL-Kanal (Client-Seite) umsetzen können.

3402 Der ZETA/ASL Kanal muss aufgebaut werden, wenn in [opr-well-known.yaml] der
3403 claimzeta_asl_use den Wert required oder required_passthrough hat. Der innere
3404 Request MUSS ein Access Token und ein DPoP Token enthalten. Das DPoP Tokenhtu claim
3405 MUSS die URI des Resource Server Endpunktes enthalten [RFC9449]. Das Access
3406 Tokenaud claim MUSS den logischen Bezeichner (logische Audience) des Resource Server
3407 Endpunktes enthalten, den die Policy Engine gemäß 5.4.2.4 vergibt; das zugehörige
3408 Token trägtver: 2 (Contract v2). Bei Legacy-Clients (Contract v1) entspricht deraud claim
3409 dem verbatim aus dem Parameteraudience übernommenen Wert und das Token
3410 trägtver: 1 (siehe 5.4.2.5). Die URI des Resource Server Endpunktes wird nicht in denaud
3411 claim übernommen, sondern ausschließlich über denhtu claim des DPoP Proofs

3412 gebunden.
3413 Bei `zeta_asl_use: required_passthrough` gilt:
3414 Im äußeren HTTP Request MÜSSEN zusätzlich ein Access Token und ein DPoP Token
3415 vorhanden sein. Das `claim` im DPoP Token im äußeren HTTP Request MUSS die URI
3416 des ZETA/ASL Endpunktes enthalten. Das `aud claim` im Access Token im äußeren HTTP
3417 Request MUSS den logischen Bezeichner des ZETA/ASL Endpunktes enthalten.
3418 [`<=`]

3419 *Hinweis: Ob ein ZETA/ASL Kanal zu verwenden ist, wird im OAuth Protected Resource*
3420 *Well-known Dokument des TI 2.0 Dienstes festgelegt (`claim zeta_asl_use`). Die*
3421 *Anforderungen für den ZETA/ASL-Kanal sind in [`gemSpec_Krypt#8`] zu finden.*

3422 **A_28426 -ZETA Client, Service Discovery**
3423 Der ZETA Client MUSS die Well-known und JWKS JSON-Dokumente regelmäßig einmal alle
3424 24 Stunden neu laden, wenn im HTTP-Response-Header kein Cache-Control-Header vom
3425 ZETA Guard eingefügt wurde. Der ZETA Client MUSS die Service Discovery erneut
3426 durchführen, wenn beim Kommunikationsaufbau zu den geschützten Ressourcen der
3427 HTTP-Statuscodes 404 (Not Found) empfangen wird. [`<=`]

3428 **A_28425-01 -ZETA Client, Service Discovery - if-none-match und etag**
3429 Der ZETA Client MUSS bei der Abfrage der Well-known JSON-Dokumente die HTTP-Header
3430 `if-none-match` und `etag` verwenden. [`<=`]

3431 **A_29691 -ZETA Guard - Versionierung der Token Ausstellung**
3432 Der ZETA Guard Authorization Server MUSS im OAuth Authorization Server Well-known
3433 JSON Dokument (`[as-well-known.yaml]`) über das Feld `api_versions_supported` die
3434 unterstützten Vertragsversionen der Token-Ausstellung bekannt geben.
3435 [`<=`]

3436 **A_29692 -ZETA Client - Versionierung der Token Ausstellung**
3437 Der ZETA Client MUSS das Feld `api_versions_supported` auswerten und

- 3438 • bei Unterstützung von Contract v2 den
- 3439 Parameter `resource [RFC8707]` und `scope` verwenden,
- 3440 • andernfalls den Parameter `audience (Contract v1)` verwenden

3441 [`<=`]

3442 **A_29693 -ZETA Guard - Versionsabhängige Policy Engine Decision**
3443 Der ZETA Guard MUSS bei einem Token Request mit `resource denaud-Claim` aus der
3444 Policy Engine Decision ableiten und das Access Token mit `ver: 2` ausstellen. Bei einem
3445 Token Request mit `audience` MUSS der ZETA Guard `denaud-Claim` verbatim übernehmen
3446 und das Access Token mit `ver: 1` ausstellen.
3447 [`<=`]

3448 **A_29694 -ZETA Guard - Fehlermeldung bei fehlerhaftem Parameter im Token**
3449 **Request**
3450 Der ZETA Guard MUSS einen Token Request mit `resource` für eine nicht unterstützte
3451 Zielressource mit dem Fehler `invalid_target` gemäß [RFC8707] ablehnen.
3452 [`<=`]

3453 5.4.3 Client-Registrierung

3454 **A_28465 -ZETA Client, Registrierung mit mehreren ZETA Guard**
3455 Der ZETA Client MUSS sich einmalig am Authorization Server pro ZETA Guard registrieren.
3456 [`<=`]

3457 *Hinweis: Dabei ist es unerheblich, ob der TI 2.0-Dienst einen oder mehrere Ressource*
3458 *Server bereitstellt. Der PDP im ZETA Guard kann für mehrere Resource Server eines TI*

3459 2.0-Fachdienste zuständig sein. Diese Realisierung wird durch die Verwendung des API-
3460 Katalog in der Service Discovery ermöglicht.

3461 **A_28524 -ZETA Client, Konfigurationsdaten pro ZETA Guard Registrierung**

3462 Der ZETA Client MUSS die Registrierungs- und Konfigurationsdaten inklusive der client-
3463 id pro ZETA Guard Registrierung verwalten.[<=]

3464 *Hinweis: Grundsätzlich kann ZETA Guard den Zugriff auf mehrere Ressourcen*
3465 *kontrollieren.*

3466 **A_25432-01 -ZETA Client - Ablauf Client-Registrierung**

3467 Der mobile ZETA Client (oder ein stationärer Client) für Versicherte MUSS, sofern er an
3468 Schnittstellen der Telematikinfrastruktur wegen einer ungültigen/fehlenden Client-
3469 Registrierung abgewiesen wird, eine Registrierung am Authorization Server durchführen,
3470 indem er

- 3471 • kryptografische Client-Credentials lokal generiert,
- 3472 • den/die Nutzer:in mittels OpenID Connect authentifiziert,
- 3473 • die generierten Credentials sowie die Clientintegrität attestiert und
- 3474 • eine zusätzliche Nutzerbestätigung mittels One-Time-Passwort (gemäß [TR-03107-1])
3475 über einen zweiten Kommunikationsweg (E-Mailbestätigung) startet.

3476 [<=]

3477 *Hinweis: Für die Client-Registrierung muss das Vertrauensniveau hoch, nicht erreicht*
3478 *werden.*

3479 **A_25766 -ZETA Client - Client Credentials in TI Qualität**

3480 Der ZETA Client MUSS die Client-Credentials im Form von kryptografischen Schlüsseln
3481 gemäß der Festlegungen in [gemSpec_Krypt] (Verfahren, Algorithmen, Schlüssellängen
3482 etc.) unterstützen.[<=]

3483 **A_25769 -ZETA Client - Client Credentials sicher generieren und schützen**

3484 Der ZETA Client auf mobilem Gerät mit Apple- oder Android-basierter Betriebsumgebung
3485 MUSS die Generierung der Client-Credentials derart generieren und speichern, dass ein
3486 Kopieren und Exportieren der Schlüssel verhindert wird.[<=]

3487 *Hinweis: Eine Speicherung der Schlüssel in einem Hardware-Modul ist gegenüber einer*
3488 *Software-Lösung (z. B. Android TEE) zu bevorzugen.*

3489 *Hinweis: Das Client Instance Key Pair soll initial 2 Jahre gültig sein. Der private Client*
3490 *Instance Key PrK.Client.Sig soll im TPM2 Modul verwendet werden (oder Secure Enclave*
3491 *bei macOS und iOS sowie TEE bei Android).*

3492 **A_25767 -ZETA Client - Clientkey in JWT**

3493 Das ZETA Client MUSS Private Key JWT [RFC7521] und [RFC7523] sowie DPoP [RFC9449]
3494 zur Authentifizierung unterstützen.[<=]

3495 **A_25434 -ZETA Client - Client-Registrierung mit bestätigten** 3496 **Umgebungseigenschaften Android**

3497 Der ZETA Client für eine Google-Android basierte Betriebsumgebung MUSS seine Client-
3498 Credentials, App-Integrität und -Authentizität sowie OS-/FW und HW-Eigenschaften über
3499 Key and ID Attestation gegenüber PDP Client-Registrierung bestätigen..[<=]

3500 **A_25768 -ZETA Client - Client-Registrierung mit bestätigten** 3501 **Umgebungseigenschaften Apple**

3502 Der ZETA Client für eine Apple-basierte Betriebsumgebung (iOS, macOS) MUSS die Client-
3503 Credentials, App Integrität und Authentizität über DCAppAttest gegenüber dem PDP
3504 Authorization Server bestätigen. Eigenschaften der Laufzeitumgebung MÜSSEN durch das
3505 Clientsystem über einen geprüften Prozess bestätigt werden.[<=]

3506 **A_25758 -ZETA Client - Erfassung Kontaktinformation für Offband-Verifikation**

3507 Der mobile ZETA Client MUSS vom Nutzer eine strukturell valide Kontaktinformation (E-
3508 Mailadresse) abfragen und für eine Offband-Verifikation (Trust on First Use) an den
3509 Endpunkt des Authorization Servers übertragen.[<=]

3510 **A_25732 -ZETA Client - Unterstützung des Nutzers bei der Client-Registrierung**

3511 Der mobile ZETA Client MUSS den Nutzer bei der Client-Registrierung und -Verwaltung
3512 geeignet unterstützen (z. B. mittels Guide, Tutorial o. ä.).[<=]

3513 **A_25733 -ZETA Client - Clientverwaltung und manuelle De-Registrierung**

3514 Der ZETA Client MUSS dem Nutzer eine Übersicht aller beim Client-Registrierungsdienst
3515 registrierten Clients darstellen und die Möglichkeit zur De-Registrierung einzelner Clients
3516 anbieten.[<=]

3517 **A_25734 -ZETA Client - Zugriffsprotokoll Client-Registrierung**

3518 Der ZETA Client MUSS dem Nutzer einen Einblick in das Zugriffsprotokoll der
3519 Schnittstellen des Client-Registrierungsdienstes für genutzte Clients dieses Nutzers
3520 geben.[<=]

3521 **A_25735-01 -mobiler Client- Push-Benachrichtigung**

3522 Das Clientsystem MUSS dem Nutzer die Möglichkeit geben, Push-Benachrichtigungen für
3523 Aktivitäten über registrierte Clients und Neuregistrierungen für diesen Nutzer zu
3524 aktivieren, zu ändern und zu deaktivieren.[<=]

3525 **5.4.4 Nutzerauthentifizierung**

3526 **A_25761 -ZETA Client - Nutzerauthentifizierung mittels etablierter Standards**

3527 Der ZETA Client MUSS die Mechanismen OAuth Authorization Code Flow mit OpenID
3528 Connect und OpenID Federation (Auswahl des zuständigen sektoralen IDP) oder OAuth2
3529 Token Exchange mit private_key_jwt Client Authentifizierung unterstützen.[<=]

3530 *Hinweis: Perspektivisch sollen ZETA Clients auch OpenID for Verifiable Credentials*
3531 *(OIDC4VC) unterstützen. OAuth2 Token Exchange wird für stationäre Clients mit SM(C)-B*
3532 *Authentisierung verwendet. OAuth Authorization Code Flow, OpenID Connect und OpenID*
3533 *Federation wird grundsätzlich von mobilen Clients verwendet, kann aber auch von*
3534 *stationären Clients verwendet werden.*

3535 **A_25762-01 -ZETA Client - Nutzerauthentifizierung - Unterstützung etablierter**

3536 **Identitäten und Dienste**
3537 Der ZETA Client MUSS zur Authentifizierung des Nutzers mindestens eines der folgenden
3538 Verfahren unterstützen:

- 3539 • Authentifizierung des Nutzers gegenüber einem Sektoralen IDP der IDP Föderation
- 3540 • gemäß [gemSpec_IDP_Sek] (GesundheitsID)
- 3541 • Authentifizierung des Nutzers mittels SM(C)-B signiertem Subject Token.

3542 [**<=**]

3543 **5.4.5 Session Management**

3544 **A_25781-01 -ZETA Clients - OAuth2 Autorisierung**

3545 Der ZETA Client MUSS die Rolle eines OAuth2 Clients [RFC6749] übernehmen und eine
3546 Autorisierung vom Authorization Server einholen. Dabei MUSS beim Authorization-Code-
3547 Flow der PKCE Flow [RFC7636] verwendet werden.

3548 [**<=**]

3549 **A_25782 -ZETA Client - OAuth2 Session Management**

3550 Der ZETA Client MUSS

3551 • die vom Authorisation Server ausgestellten Access- und Refresh Token gemäß
3552 [RFC6749#1.5] sowie die DPoP Schlüssel gemäß [RFC9449] bis zur nächsten
3553 Aufforderung zur Autorisierung oder Authentifizierung als User-Session sicher
3554 aufbewahren,

3555 • nach Bedarf abgelaufene Access Token über Refresh Token erneuern und

3556 • eine Refresh Token-Rotation gemäß [RFC6749#10.4] unterstützen.

3557 [**<=**]

3558 *Hinweis: Die Session des ZETA Clients zum ZETA Guard Authorization Server wird über*
3559 *den claim sid im Access Token abgebildet.*

3560 **A_25783-01 -ZETA Client - Anweisungen aus HTTP Response Status Codes und** 3561 **Header folgen**

3562 Der ZETA Client MUSS die HTTP Response Status Codes und HTTP Header entsprechend
3563 der Vorgaben der Resource Server und Zero Trust-APIs auswerten und den Anweisungen
3564 daraus folgen und insbesondere

3565 • eine Step-Up- oder erneute Authentifizierung des Nutzers,

3566 • eine Re-Autorisierung und erneute Attestation der Client-Instanz und

3567 • eine Anzeige der Warnungen aufgrund der Policy-Entscheidungen

3568 umsetzen.[**<=**]

3569 **A_29849 -ZETA Client, Reaktion auf terminierte Session**

3570 Der ZETA Client MUSS bei einer Ablehnung aufgrund terminierter Session eine erneute
3571 Authentifizierung des Nutzers einleiten.

3572 Der ZETA Client DARF keinen weiteren Refresh-Request mit demselben Session-Kontext
3573 durchführen.

3574 [**<=**]

3575 **A_30002 -ZETA Client - Serverseitiger Token-Widerruf bei Logout**

3576 Der ZETA Client MUSS bei einer nutzerinitiierten Abmeldung bzw. Beendigung der Session
3577 sein Refresh Token über POST /revoke [RFC7009] am Authorization Server widerrufen,
3578 damit die clientseitige Abmeldung zu einer echten serverseitigen Grant-Terminierung
3579 wird, und MUSS lokal gespeicherte Access-/Refresh Token sowie DPoP-Schlüssel
3580 anschließend löschen.[**<=**]

3581 **5.4.6 Fehlerbehandlung**

3582 **5.4.6.1 Liste der HTTP-Statuscodes**

3583 Der folgende Abschnitt enthält die HTTP-Statuscodes, die ZETA Clients von Zero Trust-
3584 Komponenten erhalten können, basierend auf den spezifischen Schritten wie
3585 Authentifizierung, Client-Registrierung und HTTP Proxy.

3586 Die Fehlercodes sollen dem Client die Möglichkeit geben, angemessen auf die jeweilige
3587 Fehlersituation zu reagieren.

3588 Dabei wird unterschieden zwischen den folgenden Ergebnisklassen, die über
3589 verschiedene Statuscodes angezeigt werden:

3590 1. **Erfolgreiche Aktion:** dies sind die Statuscodes

3591 a. 2xx

3592 b. 3xx

3593 2. **Abschließend nicht erfolgreiche Aktion:** dies sind insb. die Statuscodes:

- 3594 a. 400 Bad Request: sollte ein Request mit 400 abgelehnt werden, kann der Client
3595 nicht entscheiden was anders aufzurufen wäre und würde den Request daher nur
3596 wiederholen können. Daher wird hier abgebrochen.
- 3597 b. Alle weiteren 4xx Statuscodes die nicht in den anderen Klassen gelistet sind (insb.
3598 nicht 401 und 403, siehe nächsten Punkt).
- 3599 3. **Step-Up Authentifizierung:** Eine Step-Up-Authentifizierung wird grundsätzlich nur
3600 einmal automatisch durchgeführt, bevor abgebrochen wird. Dies dient auf der einen
3601 Seite der erneuten Prüfung gegen die OPA-Regeln z.B. im Falle von
3602 Netzwerkwechseln, und auf der anderen Seite dem Vermeiden einer Überlastung
3603 durch nur einmalige Wiederholung. Hier wird unterschieden zwischen Requests gegen
3604 den PDP, bei denen die OPA Regeln bereits abgefragt werden – diese erlauben keinen
3605 Retry bei Forbidden, und anderen Requests, bei denen nach dem 403 Forbidden die
3606 OPA Regeln neu abgefragt werden sollen:
 - 3607 a. Bei einem 401 Unauthorized Statuscode bei einem Request gegen den PDP im
3608 Rahmen der Client-Registrierung, der Authentifizierung oder des Token Refreshs
3609 kann der Client einen Request nach einer erneuten Authentifizierung einmalig
3610 wiederholen. Die angenommenen Fehlersituationen hier sind z.B. während des
3611 Requests ablaufende Zertifikate/Token (time-of-check to time-of-use Situation),
3612 oder die Invalidierung eines Access Token durch Impossible-Travel Detektion oder
3613 ähnliches. Die Authentifizierung soll maximal einmal wiederholt werden, bevor ein
3614 401/403 Status nach einem wiederholten Request als abschließend nicht
3615 erfolgreich gewertet wird.
 - 3616 b. Beim Zugriff auf eine Ressource über den PEP HTTP Proxy bzw. andere PDP-APIs
3617 wird unterschieden:
 - 3618 (i) Step-up (401 mit error=insufficient_user_authentication, [RFC9470]): Der Client
3619 führt einmalig eine Authentisierung auf dem geforderten Niveau (acr_values aus
3620 dem WWW-Authenticate-Header) durch und wiederholt den Request. Schlägt auch
3621 dieser fehl, wird abgebrochen.
 - 3622 (ii) Ungültiges/abgelaufenes Token (401 mit error=invalid_token) oder erneute
3623 Policy-Prüfung (403): Der Client DARF einmalig einen Token-Refresh unter Nutzung
3624 eines vorhandenen Refresh-Token versuchen (dies löst eine erneute OPA-Prüfung
3625 aus, z. B. nach Netzwerkwechsel/Impossible-Travel). Wird erneut 401/403
3626 erhalten, wird einmalig eine volle Authentifizierung durchgeführt, bevor der
3627 ursprüngliche Request mit neuem Access Token wiederholt werden kann.
- 3628 4. **Temporäre Fehlersituation:**
 - 3629 a. Bei 5xx Statuscode kann der Client wie in ZT_HTTP_Statuscodes den Request ggf.
3630 nach Wartezeit wiederholen
 - 3631 b. 429 Too Many Requests – auch hier kann der Client – nach Wartezeit – den
3632 Request wiederholen.

A_27007 -ZETA Client - HTTP Statuscodes

Der ZETA Client MUSS die HTTP Statuscodes gemäß Tabelle "ZT_HTTP_Statuscodes" unterstützen.[<=]

Tabelle 7: ZT_HTTP_Statuscodes

Endpunkte	HTTP Statuscodes
Authentifizierung mit SM(C)-B signiertem Subject Token	200 OK: Authentifizierung erfolgreich. Der Client kann mit der gewünschten Operation fortfahren, z.B. Zugriff auf geschützte Ressourcen. 400 Bad Request: Fehlerhafte oder ungültige JWT-Assertion (z.B. falsches Format oder fehlende Claims).

	<p>401 Unauthorized: Authentifizierung fehlgeschlagen, z.B. aufgrund einer ungültigen Signatur oder eines abgelaufenen JWT (error Code <code>invalid_client</code> oder <code>insufficient_user_authentication</code>).</p> <p>403 Forbidden: Der Client hat keine Berechtigung, auf die angeforderte Ressource zuzugreifen. Der Client darf keine weiteren Versuche unternehmen und soll den Nutzer entsprechend informieren (error Code <code>access_denied</code>, <code>session_terminated</code> oder <code>refresh_token_revoked</code>).</p>
Authentifizierung mit OIDC und Authorization Code Flow	<p>200 OK: Erfolgreiche Authentifizierung und Autorisierung. Der Client kann mit der gewünschten Operation fortfahren, z.B. Zugriff auf geschützte Ressourcen.</p> <p>302 Found: Der Client wird zum Autorisierungs-Endpunkt des Identitätsproviders umgeleitet. Der Client sollte der Weiterleitungs-URL folgen, um den nächsten Schritt im Autorisierungscode-Austausch abzuschließen.</p> <p>400 Bad Request: Fehlerhafte Anfrage, z.B. fehlende oder ungültige Parameter (z.B. falscher <code>redirect_uri</code>, <code>client_id</code>).</p> <p>401 Unauthorized: Authentifizierung fehlgeschlagen, z.B. ungültiger Code oder Token (error Code <code>invalid_client</code> oder <code>insufficient_user_authentication</code>).</p> <p>403 Forbidden: Autorisierung fehlgeschlagen, z.B. fehlende Zugriffsrechte. Der Client hat möglicherweise keine Berechtigung, die angeforderte Ressource zu nutzen. Der Client sollte den Nutzer darüber informieren und keine weiteren Anfragen stellen (error Code <code>access_denied</code>, <code>session_terminated</code> oder <code>refresh_token_revoked</code>).</p>
Token-Refresh	<p>200 OK: Authentifizierung erfolgreich. Der Client kann mit der gewünschten Operation fortfahren, z.B. Zugriff auf geschützte Ressourcen.</p> <p>400 Bad Request: Fehlerhafte oder ungültige JWT-Assertion (z.B. falsches Format oder fehlende Claims).</p> <p>401 Unauthorized: Authentifizierung fehlgeschlagen, z.B. aufgrund einer ungültigen Signatur oder eines abgelaufenen JWT. Der Client initiiert eine einmalige Step-Up Authentifizierung, siehe oben zu Retry der Authentifizierung (error Code <code>invalid_client</code> oder <code>insufficient_user_authentication</code>).</p> <p>403 Forbidden: Die referenzierte Session wurde serverseitig terminiert oder das zugehörige Refresh Token wurde entzogen. Der Authorization Server liefert im Response-Body einen standardisierten Fehlercode (<code>session_terminated</code> oder <code>refresh_token_revoked</code>). Der Client MUSS eine erneute Authentifizierung durchführen und DARF keinen weiteren Refresh-Request mit demselben Session-Kontext senden (error Code <code>access_denied</code>, <code>session_terminated</code> oder <code>refresh_token_revoked</code>).</p>
Client-Registrierung	<p>201 Created: Client erfolgreich registriert. Der Client sollte die Registrierungsdaten sicher speichern und mit der weiteren Interaktion fortfahren.</p> <p>400 Bad Request: Fehlerhafte Anfrage, z.B. ungültige oder unvollständige Clientdaten.</p>

	<p>401 Unauthorized: Fehlende oder ungültige Authentifizierung bei der Registrierung. Der Client muss sicherstellen, dass er korrekt authentifiziert ist. Der Client initiiert eine einmalige Step-Up Authentifizierung, siehe oben zu Retry der Authentifizierung.</p> <p>403 Forbidden: Zugriff verweigert, z.B. wenn der Client nicht berechtigt ist, sich zu registrieren. Der Client sollte prüfen, ob er die Berechtigung zur Registrierung hat. Wenn nicht, sollte er keine weiteren Versuche unternehmen und den Nutzer informieren.</p> <p>409 Conflict: Konflikt bei der Registrierung, z.B. ein Client mit der gleichen ID existiert bereits. Der Client darf keine weiteren Registrierungsversuche senden und soll den Nutzer über die das Serverproblem informieren.</p>
<p>HTTP Proxy</p>	<p>200 OK: Anfrage erfolgreich durch den Proxy weitergeleitet. Der Client kann die angeforderte Ressource wie gewohnt verwenden.</p> <p>301 Moved Permanently: Permanente Weiterleitung der Anfrage durch den Proxy. Der Client sollte die neue URL speichern und zukünftige Anfragen an diese Adresse senden.</p> <p>302 Found: Temporäre Weiterleitung durch den Proxy. Der Client sollte der Weiterleitung folgen, um die angeforderte Ressource zu erhalten, aber die ursprüngliche URL für zukünftige Anfragen beibehalten.</p> <p>400 Bad Request: Ungültige Anfrage an den Proxy. Der Client sollte die Anfrage überprüfen und sicherstellen, dass sie korrekt formatiert und vollständig ist, bevor er sie erneut sendet.</p> <p>401 Unauthorized: Das Access/DPoP-Token ist fehlend, ungültig oder abgelaufen (error=invalid_token) oder das nachgewiesene Authentifizierungsniveau reicht nicht aus (error=insufficient_user_authentication, Step-up gemäß [RFC9470], signalisiert im WWW-Authenticate-Header). Im ersten Fall führt der Client einmalig eine erneute Authentifizierung/Token-Refresh durch; im zweiten Fall initiiert er das Step-up-Verfahren gemäß A_29859 und wiederholt den Request einmalig.</p> <p>403 Forbidden: Endgültige Ablehnung durch den Proxy — der Zugriff ist mit dem vorliegenden, korrekt authentisierten Token nicht gestattet (z. B. aud/acr/htu-Mismatch nach A_29676, Policy-Deny, session_terminated, refresh_token_revoked). Dies ist kein Step-up-Fall. Der Client DARF einmalig einen Token-Refresh versuchen (erneute OPA-Prüfung, z. B. nach Netz-/IP-Wechsel); bleibt die Antwort 403, bricht er ab und informiert den Nutzer. Bei session_terminated/refresh_token_revoked MUSS er eine vollständige Neu-Authentifizierung durchführen.</p> <p>404 Not Found: Die angeforderte Ressource wurde nicht gefunden.</p> <p>405 Method Not Allowed: Die verwendete HTTP Methode wird nicht unterstützt für den Endpunkt.</p> <p>502 Bad Gateway: Der Proxy hat eine ungültige Antwort vom Upstream-Server erhalten. Der Client sollte die Anfrage später erneut senden, da der Fehler auf einem Problem des Upstream-Servers beruhen könnte. Gegebenenfalls kann der Nutzer informiert werden.</p> <p>504 Gateway Timeout: Der Proxy hat auf eine Antwort vom Upstream-Server gewartet, diese aber nicht innerhalb des Timeouts erhalten. Der Client sollte die Anfrage nach einer angemessenen Wartezeit erneut versuchen und den Nutzer darüber informieren, dass der Server nicht rechtzeitig geantwortet hat.</p>

Alle Endpunkte	<p>429 Too Many Requests: Zu viele Anfragen innerhalb eines bestimmten Zeitraums (Rate Limiting). Der Client sollte die Anzahl der Anfragen reduzieren und eine geeignete Wartezeit (Retry-After Header beachten) einhalten, bevor er die Anfrage erneut sendet.</p> <p>500 Internal Server Error: Allgemeiner Serverfehler. Der Client sollte den Vorgang möglicherweise zu einem späteren Zeitpunkt wiederholen oder den Nutzer auf ein Problem auf dem Server hinweisen.</p>
-----------------------	---

3637 **5.4.6.2 Behandlung von PEP-Fehlern mittels HTTP-Header**

3638 Für die korrekte Fehlerbehandlung im ZETA-Client (SDK) ist es erforderlich, Fehler, die
 3639 direkt durch den Policy Enforcement Point (PEP) verursacht werden, von fachlichen
 3640 Fehlern des dahinterliegenden Resource Servers (Fachdienstes) zu unterscheiden.

3641 Step-up-Bedarf (unzureichendes Sicherheitsniveau) signalisiert der PEP mit 401
 3642 (Unauthorized) gemäß [RFC9470]; 403 (Forbidden) kennzeichnet eine endgültige
 3643 Ablehnung. Beide Codes können sowohl vom PEP (Header zeta-error-origin: pep) als auch
 3644 vom Fachdienst stammen und werden über diesen Header unterschieden.

3645 Während fachliche Fehler des Resource Servers unverändert an den Fachclient
 3646 durchgereicht werden müssen, erfordern Fehler des PEP eine dedizierte Behandlung
 3647 durch das ZETA-SDK (z. B. das Initiieren eines Step-Up-Verfahrens oder ein kontrollierter
 3648 Retry).

3649 Um diese Unterscheidung ohne komplexe Payload-Analysen (insb. im Kontext von JSON-
 3650 vs. CBOR-Sicherheitsprüfungen in ASL) zu ermöglichen, signalisiert das PEP eigene Fehler
 3651 über einen spezifischen HTTP-Response-Header.

3652 **A_29852 -ZETA Client, Auswertung des PEP-Fehler-Headers**

3653 Empfängt der ZETA Client eine HTTP-Fehlerantwort (Statuscodes 401 oder 403), MUSS es
 3654 prüfen, ob der HTTP-Header zeta-error-origin: pep in der Antwort enthalten ist.
 3655 Ist der HTTP-Header zeta-error-origin: pep in der Fehlerantwort vorhanden, MUSS der
 3656 ZETA Client die für den spezifischen Fehlerfall definierte interne Fehlerbehandlung
 3657 einleiten. [`<=`]

3658 **A_29853 -ZETA Client, Weiterleitung von Fachdienst-Fehlern**

3659 Ist der HTTP-Header zeta-error-origin in einer Fehlerantwort (auch bei den
 3660 Statuscodes 401 oder 403) nicht enthalten, MUSS der ZETA Client den Fehler unverändert
 3661 und transparent an den aufrufenden Fach-Client durchreichen. [`<=`]

3662 **A_29859 -ZETA Client, Behandlung von Step-Up-Fehlermeldungen**

3663 Empfängt der ZETA Client eine HTTP-Fehlerantwort mit HTTP-Statuscode 401 und zeta-
 3664 error-origin: pep, bei der das Step-up über den WWW-Authenticate-Header
 3665 error="insufficient_user_authentication" gemäß
 3666 [RFC9470] oder error="insufficient_scope" signalisiert wird, MUSS der ZETA Client
 3667 den Fall als Step-up erkennen und dabei das geforderte Niveau acr_values und oder
 3668 scopes sowie ggf. max_age aus dem WWW-Authenticate-Header übernehmen und
 3669 folgende Schritte durchführen:

- 3670 • Fehlererkennung: Den Fehler als dedizierten Step-Up-Fehler identifizieren (und nicht
 3671 als regulären Verbindungs- oder Token-Fehler).
- 3672 • Initiierung des Upgrades: Die Prozedur zum Erlangen eines neuen Access Tokens mit
 3673 erforderlichem Sicherheitsniveau (z. B. durch erneute Authentifizierung mit den
 3674 geforderten Parametern) einleiten.
- 3675 • Fachclient-Interaktion: Kann das SDK den Step-Up-Prozess nicht vollautomatisch im
 3676 Hintergrund durchführen (z. B. weil eine Benutzerinteraktion für die Multi-Faktor-
 3677 Authentifizierung nötig ist), MUSS der ZETA Client dem aufrufenden Fach-Client eine

3678 spezifische Exception bzw. ein definiertes Fehlerobjekt zurückgeben, das den Step-Up-
3679 Bedarf signalisiert.

- 3680 • Retry-Option: Nach erfolgreicher Durchführung des Step-Ups MUSS der ZETA Client
3681 die Möglichkeit bieten, den ursprünglichen Request mit dem neuen, höherwertigen
3682 Access Token zu wiederholen.

3683 [**<=**]

3684 *Hinweis: Nach RFC 6750 §3.1 ist insufficient_scope klassisch mit 403 assoziiert. Die*
3685 *bewusste Behandlung als 401-Step-up ist hier korrekt, weil der fehlende Scope in ZETA*
3686 *über eine (Step-up-)Authentisierung erlangbar ist.*

3687 **5.4.7 ZETA Attestation Service**

3688 Der ZETA Attestation Service stellt einen Dienst zur Verfügung, der es stationären Clients
3689 (Primärsysteme auf Basis von Windows oder Linux) ermöglicht, TPM-signierte
3690 Attestierungsinformationen für den Client abzurufen. Diese Informationen basieren auf
3691 Integritätsmessungen, die in ausgewählten Platform Configuration Registers (PCRs) des
3692 Trusted Platform Module (TPM) gespeichert sind. Der ZETA Guard Authorization Server
3693 verwendet diese Attestierungsdaten, um die Integrität und Authentizität der
3694 Softwareumgebung des Clients zu verifizieren, bevor Zugriff auf geschützte Ressourcen
3695 gewährt wird.

3696 Der ZETA Attestation Service wird vom Hersteller des stationären Clients bereitgestellt
3697 und es muss eine manipulationsgeschützte Vertrauensbeziehung zwischen stationären
3698 Client und ZETA Attestation Service bestehen, um zu gewährleisten, dass die Attestation
3699 über die vorgesehenen Software-Komponenten erfolgt.

3700 Der ZETA Attestation Service muss gemäß [API-ZETA-Attestation-Service] implementiert
3701 werden.

3702 *Hinweis: Während der Installation oder bei Updates des stationären Clients muss auch ein*
3703 *Update des ZETA Attestation Service erfolgen um eine neue Baseline für die Integrität des*
3704 *stationären Clients zu setzen. Die Baseline besteht aus einem Hash über alle*
3705 *unveränderlichen Komponenten des stationären Clients, inkl. ZETA Attestation Service.*

3706 *Hinweis: Der ZETA Attestation Service muss bei jedem Start des Clients die Messung über*
3707 *die Integrität des Clients durchführen und in das PCR schreiben.*

3708 *Hinweis: Der ZETA Attestation Service ist nicht für mobile Clients vorgesehen. Mobile*
3709 *Clients verwenden eine andere Attestierungsmethode, die auf den jeweiligen Plattformen*
3710 *basiert (z.B. Android und iOS).*

3711 *Hinweis: Wenn die Messung des Clients von der Baseline abweicht, dann soll der Zeta*
3712 *Attestation Service automatisch den Support des Herstellers informieren.*

3713 **5.5 Client Management**

3714 **5.5.1 Client Registrierungsdaten**

3715 Dieses Kapitel beschreibt die Datenstrukturen für die Kommunikation zwischen ZETA
3716 Client und ZETA Guard bei der Client-Registrierung.

3717 Die Client-Registrierung erfolgt über ein vom Client erzeugtes und signiertes Client
3718 Assertion JWT, das Informationen über den Client sowie über die aktuelle Gerätesituation
3719 (Posture) enthält.

3720 Die übermittelten Informationen werden vom Authorization Server geprüft und
 3721 anschließend in der Client-Registry gespeichert. Der Authorization Server stellt diese
 3722 Informationen dem Policy Decision Point für Autorisierungsentscheidungen zur Verfügung.

3723 Die Client-Registrierung besteht aus drei logischen Datenstrukturen:

- 3724 • Client Assertion JWT – kryptographisch signierte Identifikation der Client-Instanz
 3725 gemäß [client-assertion-jwt.yaml]
- 3726 • Client Statement – statische Eigenschaften des Clientsystems gemäß [client-
 3727 statement.yaml]
- 3728 • Posture Informationen – Eigenschaften der Laufzeitumgebung des Clients gemäß
 3729 [posture.yaml] sowie den referenzierten plattformabhängigen Informationen
 3730 gemäß [posture-apple.yaml], [posture-android.yaml], [posture-tpm.yaml]
 3731 und [posture-software.yaml].

3732

3733 5.5.2 Client Assertion JWT

3734 Die konkrete Installation eines Clients wird durch ein **Client Assertion JWT** identifiziert.
 3735 Das JWT wird vom Client signiert und beim Authorization Server übermittelt.

3736 **Tabelle 8: Tab-Client-Assertion-JWT**

Claim	Beschreibung
iss	Aussteller des JWT. Muss die vom AuthS vergebene die OAuth 2.0 client_id des Clients sein.
sub	Subject. Muss die OAuth 2.0 client_id des Clients sein.
aud	Zielgruppe. Muss die Token-Endpoint-URL des Authorization Servers enthalten.
exp	Ablaufzeitpunkt des JWT (Sekunden seit Epoch).
jti	JWT-ID – eindeutiger Bezeichner des Tokens.
client_statement	Optionales Client Statement. Nur im Registrierungs-Ablauf vorhanden. Schema [client-statement.yaml]

3737 5.5.3 Client Statement

3738 Das Client Statement (Schema: [client-statement.yaml]) enthält Informationen über die
 3739 Client-Instanz und wird als Claim client_statement in den Payload des Client Assertion
 3740 JWT eingebettet.

3741 **Tabelle 9: Tab-Client-Statement**

Claim	Beschreibung
sub	Name / Bezeichnung des Clients.
platform	Plattform der Client-Instanz. Erlaubte Werte: android, apple, windows,

	linux, other.
posture_type	Art der übermittelten Posture. Erlaubte Werte: android, apple, software, tpm.
posture	Posture der Client-Instanz. Struktur abhängig vom posture_type (s. Tabellen 10-13). Schema: posture.yaml.
attestation_timestamp	Zeitstempel der Attestierung.

3742

3743 **5.5.4 Posture Informationen**

3744 Das Posture-Objekt dient dem Nachweis der Integrität der Laufzeitumgebung des
 3745 Clients. Diese Informationen können sich im Laufe der Zeit ändern, beispielsweise durch
 3746 Updates des Betriebssystems oder der Clientsoftware.

3747 **5.5.4.1 TPM Posture**

3748 Die TPM-Posture gilt für stationäre Clients (Windows oder Linux) mit TPM-basierter
 3749 Attestierung.

3750 **Tabelle 10: Tab-Posture-TPM**

Claim	Beschreibung
product_id	gematik-Produktbezeichner.
product_version	Produktversion.
os	Name des Betriebssystems.
os_version	Version des Betriebssystems.
arch	Hardware-Architektur.
tpm_attestation_key	Öffentlicher Schlüssel des im TPM residenten Attestierungsschlüssels (PEM oder base64-DER).
tpm_quote	TPM Quote der Client-Instanz.
tpm_event_log	TPM Event Log der Client-Instanz.
tpm_ek_certificate_chain	Endorsement-Key-Zertifikatskette des TPM-Herstellers (PEM oder base64-DER).
platform_product_id	Plattform-spezifischer Produktbezeichner (Schema: product-id-windows.yaml oder product-id-linux.yaml).

3751 **5.5.4.2 Software Posture**

3752 Die Software-Posture gilt für generische Software-Clients ohne TPM-Attestierung
 3753 (Windows oder Linux).

3754

Tabelle 11: Tab-Posture-Software

Claim	Beschreibung
product_id	gematik-Produktbezeichner.
product_version	Produktversion.
os	Name des Betriebssystems.
os_version	Version des Betriebssystems.
arch	Hardware-Architektur.
public_key	Öffentlicher selbst-signierter Signierungsschlüssel (PEM oder base64-DER).
attestation_challenge	Attestierungs-Challenge der Client-Instanz; dient zur Überprüfung des öffentlichen Client-Instanz-Schlüssels und des Nonce vom AS.
platform_product_id	Plattform-spezifischer Produktbezeichner (Schema: product-id-windows.yaml oder product-id-linux.yaml).

3755

5.5.4.3 Apple Posture

3756

Die Apple-Posture enthält die dekodierten Inhalte des Attestation- bzw. Assertion-Objekts einer iOS-App (Apple App Attest).

3757

3758

Tabelle 12: Tab-Posture-Apple

Claim	Beschreibung
product_id	gematik-Produktbezeichner.
product_version	Produktversion.
system_version	Betriebssystemversion.
system_name	Name des Betriebssystems auf dem Gerät.
device_model	Gerätemodell.
key_id	Schlüsselbezeichner des kryptographischen Schlüssels auf dem Gerät (base64-encodiert).
platform_product_id	Plattform-spezifischer Produktbezeichner für Apple-Apps (Schema: product-id-apple.yaml).
fmt	Format des Attestation-Statements. MUSS den Wert apple-appattest haben. Optional, bei initialer Attestation verwendet.
attStmt	Attestation-Statement: x5c (array of base64-Strings, X.509-Zertifikatskette) und receipt (base64, Apple-Receipt für Device Risk Assessment). Beide Unterfelder sind Pflicht, wenn attStmt vorhanden.

	Optional, bei initialer Attestation verwendet
authData	Strukturierte Authenticator-Daten der Initial-Attestation Optional, bei initialer Attestation verwendet.
signature	Kryptographische Signatur des privaten Geräteschlüssels. Optional, bei nachfolgender Attestation.
assertionAuthenticatorData	Vereinfachte Authenticator-Daten für Folge-Assertions: rpidHash (byte, SHA-256 der App-ID) und counter (integer, pro Assertion inkrementiert). Beide Unterfelder sind Pflicht, wenn das Objekt vorhanden ist. Optional, bei nachfolgender Attestation.
client_data_json	JSON-String mit den zu signierenden Request-Daten einschließlich eines server-seitig bereitgestellten Challenge. Optional, bei nachfolgender Attestation.

3759
3760
3761
3762
3763

5.5.4.4 Android Posture

Die Android-Posture enthält plattformspezifische Geräte- und Sicherheitseigenschaften gemäß den Android-APIs (vgl. android.os.Build).

Tabelle 13: Tab-Posture-Android

Claim	Beschreibung
product_id	gematik-Produktbezeichner.
product_version	Produktversion.
build	Android-Build-Informationen; (Build.VERSION.SDK_INT, Build.VERSION.SECURITY_PATCH, Build.MANUFACTURER, Build.PRODUCT, Build.MODEL, Build.BOARD).
key_attestation_certificate_chain	Zertifikatskette aus der Android Key Attestation.
platform_product_id	Plattform-spezifischer Produktbezeichner für Android-Apps (Schema: product-id-android.yaml).
ro	Systemeigenschaften (ro.crypto.state, ro.product.first_api_level).
packageManager	Informationen des Package Managers (packageManager.feature_verified_boot, packageManager.mainline_patch_level).
keyguardManager	Keyguard-Zustand (isDeviceSecure: boolean).
biometricManager	Biometrische Fähigkeiten (deviceCredential: boolean, biometricStrong: boolean).
devicePolicyManager	Richtlinien des Device Policy Managers (passwordComplexity:

	integer; erlaubte Werte: 0, 1, 2, 3).
play_integrity_token	Von der Google Play Integrity API ausgestelltes und base64- enkodiertes Token. Wird vom AuthS validiert, um die meets_*_integrity-Claims zu generieren.

3764 5.5.5 Vertrauensmodell, Faktoren und Geltungsbereich

3765 Die eigentliche **Client-Registrierung** (dynamische Registrierung nach [RFC7591], Client
3766 Assertion JWT, Client Statement, Posture/Attestierung sowie der TOFU-Ablauf mit E-Mail-
3767 OTP) ist bereits in den vorangehenden Kapiteln normativ definiert (insb. 5.3.2, 5.5.1-
3768 5.5.3 sowie A_29658, A_25432-01). Dieses Kapitel definiert **kein** eigenes
3769 Registrierungsverfahren, sondern setzt darauf das **Client Management** auf: die faktor-
3770 gebundene Verwaltung *bestehender* Registrierungen über ihren Lebenszyklus hinweg
3771 (Folgerregistrierung, E-Mail-Bindung, Schlüssel-Rollover, Recovery, Client-Verwaltung und
3772 außerordentliche Löschung).

3773 Grundlage dieses Managements ist ein Trust-On-First-Use-Verfahren (TOFU)
3774 nach [RFC7591]/[RFC7592] mit dem Faktorsatz:

- 3775 • **F1 - verifizierte E-Mail:** auf Identitätsebene gebundener Nachweis, einmalig bei
3776 Erstnutzung gepinnt.
- 3777 • **F2 - Instanzschlüssel (PoP):** je Client-Instanz gebundenes asymmetrisches
3778 Schlüsselpaar.

3779 Schutzgut sind **bestehende** Client-Registrierungen gegenüber Übernahme durch
3780 einen kompromittierten sektoralen IDP. Die Grenze des Verfahrens (First-Use-Capture vor
3781 Erstnutzung) ist akzeptiert.

3782 **A_29892 -Authorization Server - Bereitstellung der TOFU- 3783 Erweiterungsschnittstellen**

3784 Der Authorization Server MUSS zusätzlich zur bereits definierten Client-Registration-
3785 API (**POST /register**, **POST /register/verify**; definiert durch A_29658) die darauf
3786 aufbauenden TOFU-Erweiterungen gemäß Annex [zeta-guard-client-management] (RFC
3787 7592-Registrierungsverwaltung/**register/{client_id}**, Schlüssel-Rollover, E-Mail-Änderung,
3788 Recovery, Client-Verwaltung,
3789 clientseitiges Veto) sowie die privilegierte, IDP-unabhängige Operator-Schnittstelle für
3790 die außerordentliche Löschung gemäß Annex [zeta-guard-admin-oob.yaml](#) bereitstellen.
3791 Die Bereitstellung der Basis-Registrierung selbst richtet sich unverändert nach A_29658.
3792 [**<=**]

3793 **A_29893 -Authorization Server - Geltungsbereich des TOFU-Faktormodells 3794 (mobile Clients)**

3795 Der Authorization Server MUSS das TOFU-Faktormodell dieses Kapitels ausschließlich auf
3796 mobile Client-Registrierungen anwenden, die im TOFU-Verfahren an eine TI-Identität
3797 (KVNR oder Telematik ID) gebunden werden. Für stationäre Clients mit Authentisierung
3798 per SMC-B Token Exchange (vgl. A_25762-01) sowie für Clients ohne Nutzer-Identität (vgl.
3799 A_29733) gelten die faktorgebundenen Anforderungen NICHT; deren Absicherung erfolgt
3800 gemäß den dort genannten Verfahren. Der Schlüssel-Rollover (A_29921 ff.) gilt
3801 clientübergreifend. [**<=**]

3802 **A_29894 -Authorization Server - Identitätslose Clients: reduziertes 3803 Vertrauensmodell**

3804 Der Authorization Server MUSS einen ohne TI-Identität registrierten Client (vgl.
3805 A_29733) ausschließlich über den Instanzschlüssel (F2, Proof-of-Possession via JWT-
3806 Bearer Assertion nach [RFC7523]) und die plattformabhängige Attestierung (Posture)
3807 absichern. Für solche Clients DARF NICHT eine identitätsgebundene E-Mail (F1) erzeugt

3808 oder verlangt werden; und es findet KEINE identitätsbezogene außerordentliche Löschung
3809 (A_29940 ff.) statt. [<=]

3810 **A_29895 -Authorization Server - Identitätslose Clients: Schlüssel-Rollover**
3811 **anwendbar**

3812 Der Authorization Server MUSS den Schlüssel-Rollover (A_29921 ff.) auch für
3813 identitätslose Clients zulassen, da er nur den Besitz des bisherigen Instanzschlüssels
3814 voraussetzt. Bei Verlust des Instanzschlüssels eines identitätslosen Clients existiert kein
3815 Recovery-Pfad; der Client MUSS sich neu registrieren. [<=]

3816 **A_29896 -Authorization Server - IDP-Authentisierung notwendig, aber nicht**
3817 **hinreichend**

3818 Der Authorization Server MUSS sicherstellen, dass eine erfolgreiche Authentisierung des
3819 Nutzers gegenüber dem sektoralen IDP allein NICHT ausreicht, um eine bestehende
3820 Client-Registrierung zu übernehmen, zu verändern oder wiederherzustellen. Jede solche
3821 Operation MUSS zusätzlich den Besitz bzw. die Kontrolle mindestens eines bei der
3822 Erstnutzung gebundenen Faktors aus {F1, F2} nachweisen (vgl. A_25649). [<=]

3823 **A_29897 -Authorization Server - Keine Faktoränderung allein aus IDP**

3824 Der Authorization Server DARF NICHT eine Änderung oder Neubindung eines Faktors
3825 allein auf Basis einer IDP-Authentisierung zulassen. [<=]

3826 **A_29898 -ZETA Guard - Eigenständige Vertrauensregistry je Fachdienst**

3827 Jeder ZETA Guard MUSS eine eigenständige, von anderen Guards unabhängige
3828 Vertrauensregistry führen. Der ZETA Guard DARF NICHT TOFU-Vertrauenszustände
3829 (gepinnte E-Mail, Instanzschlüssel, Tombstone-/Karenzzeit-Zustände) mit anderen
3830 Guards teilen oder von diesen beziehen. [<=]

3831 **A_29899 -ZETA Guard - Keine fachdienstübergreifende Korrelation**

3832 Der ZETA Guard DARF NICHT Identitätsdatensätze fachdienstübergreifend verknüpfen
3833 oder zur
3834 Korrelation eines Nutzers über mehrere Fachdienste hinweg verwenden. [<=]

3835 Hinweis (Per-Guard-Modell): Dieselbe Identität ist je ZETA Guard ein eigenständiger
3836 Datensatz. Die gebundene E-Mail KANN zwischen Fachdiensten abweichen. Es existiert
3837 kein fachdienstübergreifendes Token; jeder Guard stellt eigene, DPoP-/audience-
3838 gebundene Token aus. Die fachdienstübergreifende Authentisierung ergibt sich
3839 ausschließlich aus dem SSO der
3840 sektoralen IDPs, NICHT aus geteiltem ZETA-Zustand.

3841 **A_29900 -Authorization Server - Bindung an die TI-Identität**

3842 Ergänzend zu A_25650 (Bindung des registrierten Clients an eine TI-Identität) MUSS
3843 der Authorization Server diese Identität (KVNR oder Telematik-ID) als Ankerpunkt
3844 des TOFU-Identitätsdatensatzes führen, an den Faktor F1 auf Identitätsebene gebunden
3845 werden (vgl. A_29901). Die Ableitung der Identität aus der Nutzerauthentisierung
3846 regelt A_29903. [<=]

3847 **A_29901 -Authorization Server - Trennung von Identitäts- und Client-Ebene**

3848 Der Authorization Server MUSS die gebundene E-Mail (F1) auf Identitätsebene führen
3849 sowie den Instanzschlüssel (F2) und die berechtigten Audiences je Client-Instanz auf
3850 Client-Ebene führen (Datenhaltung vgl. A_26585-02). [<=]

3851 **A_29902 -Authorization Server - Verwaltung nur durch eingebuchten Client**

3852 Der Authorization Server MUSS für identitäts- und clientbezogene
3853 Verwaltungsoperationen (Änderung der E-Mail, Schlüssel-Rollover, Löschung von Clients)
3854 den Nachweis eines an diesem Fachdienst eingebuchten, gültigen Clients verlangen. [<=]

3855 **A_29903 -Authorization Server - Ableitung der Identitätsbindung aus der**
3856 **Nutzerauthentisierung**

3857 Der Authorization Server MUSS die Bindung einer Registrierung an die TI-Identität aus
3858 der OIDC-Nutzerauthentisierung ableiten (Statusmodell [pending_user_binding](#) gemäß

3859 A_29658 [/\[dcr-response-202.yaml\]](#)). Ein realm-/mandantenweites Initial Access Token
3860 ohne Identitätsbezug DARF NICHT als hinreichend für die Bindung der Identität verwendet
3861 werden. [**<=**]

3862 5.5.6 Erstregistrierung (First Use)

3863 **A_29905 -Authorization Server - Freischaltung erst nach E-Mail-Verifikation**
3864 Klarstellend zum Statusmodell nach A_29658 ([pending_verification/pending_user_binding](#))
3865 DARF der Authorization Server den vollen Berechtigungsumfang (Scope/Audiences) NICHT
3866 freigeben, bevor die E-Mail-Verifikation erfolgreich abgeschlossen ist. [**<=**]

3867 **A_29907 -Authorization Server - Instanzschlüssel im Besitz der Client-Instanz**
3868 Der Authorization Server MUSS den Instanzschlüssel (F2) als öffentlichen Schlüssel
3869 der Client-Instanz führen. Der zugehörige private Schlüssel DARF NICHT an den
3870 Authorization Server übertragen werden (vgl. A_25769). [**<=**]

3871 Hinweis: Erstregistrierung und E-Mail-OTP-TOFU nutzen die **bereits definierte** Client-
3872 Registration-API [POST /register](#) (Body [\[dcr-request.yaml\]](#)) → [202 Accepted](#) ([\[dcr-response-](#)
3873 [202.yaml\]](#), [challenge_type: email_otp](#), [transaction_id](#)) →
3874 [POST /register/verify](#) (OTP) → [201 Created](#) mit [client_id](#) (Status
3875 [pending_user_binding](#)); vgl. A_29658, A_25432-01.

3876 5.5.7 Folgeregistrierung und E-Mail-Bindung

3877 **A_29909 -Authorization Server - Eine E-Mail je Identität; Verwaltung nur durch**
3878 **registrierten Client**

3879 Der Authorization Server MUSS einer Identität genau eine verifizierte E-Mail-Adresse
3880 zuordnen, gebunden bei der Erstnutzung (TOFU). Der Authorization Server DARF NICHT
3881 das Hinzufügen einer weiteren E-Mail-Adresse nach der Erstnutzung zulassen. Nach
3882 erfolgreicher Erstregistrierung MUSS der Authorization Server sicherstellen, dass die
3883 Anwendungsfälle *Client löschen*, *Client umbenennen* und *E-Mail-Adresse aktualisieren* nur
3884 durch einen registrierten, gültigen (mobilen) Client durchgeführt werden. [**<=**]

3885 **A_30005 -Authorization Server - Beschränkung der Metadatenaktualisierung**
3886 **über RFC 7592**

3887 Der Authorization Server MUSS die Metadatenaktualisierung über PUT
3888 [/register/{client_id}](#) mit Content-Type: application/json (autorisiert allein durch das
3889 Registration Access Token nach [RFC7592]) ausschließlich auf die Umbenennung des
3890 Clients ([client_name](#)) beschränken. Der Authorization Server DARF NICHT über diesen
3891 Pfad eine Änderung des Instanzschlüssels (F2), der gebundenen E-Mail (F1), der
3892 berechtigten Audiences oder von DCR-Parametern ([jwks](#), [redirect_uris](#), [grant_types](#))
3893 zulassen; entsprechende Felder MUSS er ablehnen. Der Schlüssel-Rollover erfolgt
3894 ausschließlich über Content-Type: application/zeta-rollover+jws (A_29921 ff.), die
3895 E-Mail-Änderung über [POST /zeta/identity/email](#) (A_29911).
3896 [**<=**]

3897 **A_29910 -Authorization Server - Verknüpfung weiterer Clients mit bestehender**
3898 **Identität**

3899 Der Authorization Server MUSS bei jeder weiteren Registrierung einer bereits bekannten
3900 Identität die gebundene E-Mail erneut verifizieren, DARF NICHT eine abweichende E-Mail-
3901 Adresse zulassen und MUSS den neuen Client mit dem bestehenden Identitätsdatensatz
3902 verknüpfen sowie seinen eigenen Instanzschlüssel (F2) binden. [**<=**]

3903 **A_29911 -Authorization Server - E-Mail-Änderung mit Step-up und**
3904 **überlebendem Faktor**

3905 Der Authorization Server MUSS für eine Änderung der gebundenen E-Mail eine erneute
3906 (Step-up-)Authentisierung gegenüber dem IDP UND den Nachweis eines überlebenden

3907 Faktors (Besitz des Instanzschlüssels F2, Nachweis der Kontrolle über die bisherige E-Mail
3908 F1) verlangen sowie die neue Adresse verifizieren. Für die Step-up-Authentisierung MUSS
3909 der Authorization Server den Mechanismus nach [RFC9470] verwenden: Bei fehlender
3910 oder nicht ausreichender Nutzerauthentisierung MUSS er die Anfrage mit 401 und einem
3911 WWW-Authenticate-Header (error="insufficient_user_authentication", geforderte
3912 acr_values, optional max_age) ablehnen. Der ZETA Client MUSS daraufhin einmalig den
3913 regulären Authorization Code Flow (PAR → GET /authorize → Nutzerauthentisierung am
3914 sektoralen IDP → POST /token) mit den geforderten acr_values durchlaufen. Als Nachweis
3915 der Step-up-Authentisierung MUSS der Authorization Server das dabei von ihm selbst
3916 ausgestellte, DPoP-gebundene Access Token akzeptieren und dessen Signatur, die gemäß
3917 A_29994 aus dem ID-Token des sektoralen IDP übernommenen Claims acr/amr, ein
3918 aktuelles auth_time sowie die DPoP-Bindung ([RFC9449]) prüfen. [≤]

3919 **A_29912 -Authorization Server - Identitätsweite Wirkung der E-Mail-Änderung**

3920 Der Authorization Server MUSS eine erfolgreiche E-Mail-Änderung identitätsweit für alle
3921 Clients dieser Identität an diesem Fachdienst wirksam machen. [≤]

3922 Hinweis: Die Folgeregistrierung (A_29910 und A_29911) erfolgt über die definierte API
3923 [POST /register +POST /register/verify](#) (erneute OTP-Verifikation der gebundenen E-Mail;
3924 A_29658). Die identitätsweite E-Mail-Änderung (vgl. A_29911 und A_29912) erfolgt über
3925 die Erweiterung [POST /zeta/identity/email](#) (Annex [zeta-guard-client-management]). Der
3926 Step-up-Nachweis wird dabei als vom Authorization Server ausgestelltes, DPoP-
3927 gebundenes Access Token im Feld idp_step_up übergeben; der zugehörige DPoP-Proof
3928 ([RFC9449], gebunden an Methode und Ziel-URI) im HTTP-Header DPoP. Die Verifikation
3929 der neuen Adresse erfolgt über [POST /zeta/identity/email/verify](#) (Annex [zeta-guard-client-
3930 management]); Challenge und Verify-Payload entsprechen dem definierten E-Mail-OTP-
3931 Mechanismus der Registrierung ([dcr-response-202.yaml], [verify-request.yaml],
3932 verify_type: email_otp). Erst nach erfolgreicher Verifikation wird die Änderung
3933 identitätsweit wirksam (A_29912).

3934 Die Registrierung eines neuen Clients an einem *weiteren* ZETA Guard soll komfortabel
3935 und reibungsarm sein. Statt einer erneuten E-Mail-OTP-Erstnutzung KANN eine bereits
3936 bestehende, gültige und faktorgestützte Registrierung derselben Identität an einem
3937 anderen Guard als Grundlage dienen. Der Pfad ist ausschließlich clientvermittelt (der
3938 Client trägt seinen ZETA Attestation Token vom Quell- zum Ziel-Guard); die Guards
3939 tauschen keinen Zustand direkt aus (vgl. A_29898 und A_29899). Da der Pfad über den
3940 Instanzschlüssel (F2) verankert ist — der aus einer IDP-Kompromittierung nicht ableitbar
3941 ist —, ist er mindestens so stark wie die E-Mail-OTP-Erstnutzung. E-Mail (F1) entwickelt
3942 sich anschließend je Guard eigenständig; eine Divergenz zwischen Fachdiensten ist
3943 ausdrücklich zulässig.

3944 **A_29913 -Authorization Server - Fachdienstübergreifende Fast-Path- 3945 Registrierung (Guard-Vouching)**

3946 Der Authorization Server MUSS als Alternative zur E-Mail-OTP-Erstnutzung
3947 eine fachdienstübergreifende Fast-Path-Registrierung zulassen
3948 (attestation_type=zeta_attestation_token), bei der ein Client eine
3949 bestehende, gültige und faktorgestützte Registrierung derselben Identität an einem
3950 anderen ZETA Guard nachweist. Der Ziel-Guard MUSS in diesem Fall die Identitätsbindung
3951 und die verifizierte E-Mail (F1) aus einer geprüften, signierten ZETA Attestation Token des
3952 Quell-Guards übernehmen und DARF für diesen Pfad KEINE erneute E-Mail-OTP-
3953 Verifikation verlangen. Die Einlösung erfolgt als zusätzliches Credential auf der
3954 definierten API [POST /register](#) (vgl. A_29658). [≤]

3955 **A_29914 -Authorization Server - ZETA Attestation Token als Cross-Guard- 3956 Credential: Inhalt und Vertrauensanker**

3957 Der Authorization Server MUSS für die fachdienstübergreifende Fast-Path-Registrierung
3958 (A_29913) einen ZETA Attestation Token (attestation_type=zeta_attestation_token) als
3959 Credential akzeptieren, der von einem von der gematik zugelassenen ZETA Guard
3960 signiert ist und für diesen Anwendungsfall mindestens enthält: das Identitätssubjekt

3961 (KVNR/Telematik-ID), die verifizierte E-Mail nebst Verifikationsstatus (F1), die Bestätigung
 3962 des attestierten Schlüssels (cnf nach [RFC7800]/[RFC7638]), eine eindeutige Kennung
 3963 (jti), eine Gültigkeitsdauer (exp) sowie eine Einschränkung des Empfängerkreises (aud)
 3964 auf ZETA Guards. [<=]

3965 **A_29915 -Authorization Server - Prüfung des ZETA Attestation Token durch den** 3966 **Ziel-Guard**

3967 Der Authorization Server MUSS einen vorgelegten ZETA Attestation Token vollständig
 3968 prüfen: (a) Signatur gegen das Zertifikat eines von der gematik zugelassenen ZETA
 3969 Guards (TI-Vertrauensraum/Trust-Liste), (b) Empfängerkreis (aud) sowie (c) den DCR-
 3970 Besitznachweis (signed_hash_puk_client_sig), der den neu vorgelegten Instanzschlüssel
 3971 (F2) an den im Token bestätigten (attestierten) Schlüssel bindet. Schlägt eine dieser
 3972 Prüfungen fehl, MUSS der Ziel-Guard die Fast-Path-Registrierung ablehnen. Der ZETA
 3973 Attestation Token ist mehrfach und an mehreren Guards einlösbar. Da jede Registrierung
 3974 an einen frischen, nur mit dem attestierten Schlüssel signierbaren Instanzschlüssel
 3975 gebunden ist, ist eine Wiedereinspielung ohne den zugehörigen privaten Schlüssel
 3976 nutzlos; der Besitznachweis ist NICHT challenge-gebunden. Der Schutz gegen
 3977 missbräuchliche Einlösung ergibt sich aus der Pflicht-Benachrichtigung nach A_29920
 3978 (akzeptiertes Restrisiko). [<=]

3979 **A_29916 -Authorization Server - Faktorverankerung statt IDP-Ableitung**

3980 Der Authorization Server MUSS sicherstellen, dass der Fast-Path durch den Besitz
 3981 des Instanzschlüssels (F2) autorisiert wird, der aus einer IDP-Kompromittierung nicht
 3982 ableitbar ist. Eine erfolgreiche IDP-Authentisierung allein DARF NICHT ausreichen, um den
 3983 Fast-Path (zeta_attestation_token) einzulösen (vgl. A_29896). [<=]

3984 **A_29918 -ZETA Guard - Eigenständige Weiterentwicklung nach** 3985 **Fast-Path-Registrierung (akzeptierte Divergenz)**

3986 Der ZETA Guard MUSS E-Mail (F1) nach einer Fast-Path-Registrierung eigenständig
 3987 führen. Eine spätere E-Mail-Änderung (A_29911) an einem Guard DARF NICHT automatisch
 3988 auf andere Guards übertragen werden; eine Divergenz der E-Mails zwischen den Guards
 3989 ist zulässig und erwartbar. [<=]

3990 **A_29919 -ZETA Guard - Keine persistente fachdienstübergreifende Korrelation**

3991 Der ZETA Guard MUSS die die im ZETA Attestation Token offengelegte Herkunft (Quell-
 3992 Guard) ausschließlich transient zur Prüfung verwenden und DARF NICHT eine
 3993 fachdienstübergreifende Korrelation über das nach A_29899 zulässige Maß hinaus
 3994 persistieren; insbesondere DARF NICHT die Identität des Quell-Guards als verknüpfbares
 3995 Attribut der neuen Registrierung gespeichert werden. [<=]

3996 **A_29920 -Authorization Server - Pflicht-Benachrichtigung bei Cross-Guard-** 3997 **Onboarding**

3998 Der Authorization Server MUSS bei jeder erfolgreichen fachdienstübergreifenden Fast-
 3999 Path-Registrierung (vgl. A_29913 und A_29915) die aus dem ZETA Attestation Token
 4000 übernommene E-Mail (F1) unverzüglich über den Notification Service benachrichtigen
 4001 (vgl. A_25652, A_25735-01, A_25750). Die Benachrichtigung MUSS den neu registrierten
 4002 Client erkennbar machen und auf die Widerrufsmöglichkeiten (Löschung nach
 4003 A_29934, OOB-Löschung nach A_29940 ff.) hinweisen. [<=]

4004 Hinweis: Da jeder zugelassene Guard einen ZETA Attestation Token für den
 4005 Cross-Guard-Fast-Path ausstellen kann, ist die Benachrichtigung des Identitätsinhabers
 4006 die maßgebliche Erkennungs- und Widerspruchsmöglichkeit gegen ein missbräuchliches
 4007 Vouching eines kompromittierten Quell-Guards.

4008 **5.5.8 Schlüssel-Rollover (Proof-of-Possession)**

4009 **A_29921 -Authorization Server - Schlüssel-Rollover mit Proof-of-Possession**

4010 Der ZETA Client MUSS seinen Instanzschlüssel regelmäßig wechseln (Intervall durch
 4011 gematik festgelegt). Der Wechsel MUSS als In-Place-Rollover der bestehenden

4012 Registrierung erfolgen und durch eine Signatur mit dem bisherigen (aktiven)
4013 Instanzschlüssel autorisiert werden (Proof-of-Possession). Der Authorization Server DARF
4014 NICHT einen Schlüsselwechsel allein auf Basis des Registration Access Token nach
4015 [RFC7592] zulassen. [\leq]

4016 Hinweis: Der Client Instance Key soll initial 2 Jahre gültig sein.

4017 **A_29922 -Authorization Server - Verschachtelte Signatur (Autorisierung und** 4018 **Anti-Substitution)**

4019 Der Authorization Server MUSS für den Rollover eine verschachtelte JWS-Struktur
4020 verlangen, bei der die äußere Signatur mit dem bisherigen Schlüssel (Autorisierung) und
4021 die innere Signatur mit dem neuen Schlüssel (Nachweis der Kontrolle über den neuen
4022 Schlüssel) erfolgt. [\leq]

4023 **A_29923 -Authorization Server - Replay- und Fehlleitungsschutz**

4024 Der Authorization Server MUSS den Rollover gegen Wiedereinspielung und Fehlleitung
4025 absichern durch (a) einen serverseitig ausgestellten Einmal-Nonce, (b) eine Bindung an
4026 die Identität des Guard (Audience) sowie (c) eine Bindung an HTTP-Methode und Ziel-URI
4027 der Anfrage. [\leq]

4028 **A_29924 -Authorization Server - Overlap-Fenster: Ressourcenzugriff vs.** 4029 **Verwaltung**

4030 Der Authorization Server MUSS nach erfolgreichem Rollover den bisherigen Schlüssel für
4031 einen begrenzten Übergangszeitraum ausschließlich für den Ressourcenzugriff weiter
4032 akzeptieren. Der bisherige Schlüssel DARF NICHT nach dem Rollover weitere
4033 Verwaltungsoperationen (insbesondere einen erneuten Rollover) autorisieren. [\leq]

4034 **A_29925 -Authorization Server - Invalidierung des bisherigen Schlüssels**

4035 Der Authorization Server MUSS den bisherigen Schlüssel nach Ablauf des
4036 Übergangszeitraums vollständig invalidieren und aus dem Schlüsselmaterial des Clients
4037 entfernen. [\leq]

4038 **A_29926 -Authorization Server - Keine Kopplung von Rollover und** 4039 **Faktoränderung**

4040 Der Authorization Server DARF NICHT einen Schlüssel-Rollover mit einer Änderung von E-
4041 Mail (F1) in einer untrennbaren Operation zusammenfassen. [\leq]

4042 **A_29927 -Authorization Server - Abgrenzung zum Schlüsselverlust**

4043 Der In-Place-Rollover setzt den Besitz des bisherigen Instanzschlüssels voraus. Bei Verlust
4044 DARF der Rollover-Pfad NICHT verwendet werden; verfügt der Nutzer noch über ein
4045 weiteres eingebuchtes Gerät (F2) oder die Kontrolle über die gebundene E-Mail (F1),
4046 erfolgt die (Neu-)Einbuchung als Folgeregistrierung mit erneuter E-Mail-Verifikation
4047 (A_29910). Sind alle Faktoren {F1, F2} verloren, ist keine Selbst-Wiederherstellung
4048 möglich; die Identität kann nur über die außerordentliche Löschung (OOB, A_29940 ff.)
4049 mit anschließender Erstnutzung neu etabliert werden. [\leq]

4050

4051 **A_29933 -ZETA Client - Ersatzgerät bei E-Mail-Verlust**

4052 Der ZETA Client SOLL einen Nutzer, der den Zugriff auf die gebundene E-Mail verloren hat
4053 und über ein weiteres registriertes Gerät verfügt, anleiten, zunächst von diesem Gerät die
4054 E-Mail zu ändern (A_29911) und erst danach das Ersatzgerät zu registrieren (vgl. Nutzer-
4055 unterstützung A_25732). [\leq]

4056 **5.5.9 Verwaltung von Clients/Geräten**

4057 **A_29934 -Authorization Server - Löschung anderer Clients nur auf Client-Ebene**

4058 Der Authorization Server MUSS einem eingebuchten Client die Löschung eines anderen
4059 Clients derselben Identität erlauben und DARF NICHT bei dieser Löschung
4060 identitätsgebundenen Faktor (F1) verändern oder entfernen. Die clientseitige Darstellung

4061 und Auslösung erfolgt gemäß A_25733; die Höchstzahl registrierbarer Clients bleibt nach
4062 A_25748-02 begrenzt. [**<=**]

4063 **A_29935 -Authorization Server - Benachrichtigung bei Client-Löschung**

4064 Der Authorization Server MUSS bei der Löschung eines Clients die gebundene E-Mail (F1)
4065 der Identität benachrichtigen (vgl. Notification Service A_25652, Push A_25735-01,
4066 Nutzerinformation A_25750). [**<=**]

4067 **A_29936 -Authorization Server - Erhöhte Anforderung bei Löschung des** 4068 **letzten/aller Clients**

4069 Der Authorization Server MUSS für die Löschung des letzten verbleibenden Clients einer
4070 Identität eine Step-up-Authentisierung verlangen ODER die Löschung mit einer
4071 Einspruchsfrist (Veto-Fenster) ausführen. Das Veto-Fenster wird analog A_29942
4072 ausgestaltet (geplante Löschung mit Benachrichtigung, Abbruch durch einen
4073 überlebenden Faktor gemäß A_29943); die Einlegung des Vetos erfolgt über POST
4074 /zeta/deletions/{deletion_id}/veto. Eine Sammellöschung mehrerer Clients ist KEINE
4075 eigene Serveroperation; sie erfolgt clientseitig als wiederholte Einzel-Löschung (DELETE
4076 /zeta/clients/{target_client_id}). Die erhöhte Anforderung greift, sobald dabei der letzte
4077 verbleibende Client gelöscht würde. Die Step-up-Authentisierung erfolgt nach demselben
4078 Mechanismus wie bei der E-Mail-Änderung (A_29911, [RFC9470]): 401 mit
4079 error="insufficient_user_authentication" und geforderten acr_values, einmaliger regulärer
4080 Authorization Code Flow, Nachweis durch das vom Authorization Server ausgestellte,
4081 DPoP-gebundene Access Token. [**<=**]

4082 **A_29937 -Authorization Server - Fachdienstbezogener Geltungsbereich der** 4083 **Löschung**

4084 In Anwendung des Per-Guard-Prinzips (vgl. A_29898) MUSS der Authorization
4085 Server Client-Löschungen ausschließlich auf den eigenen Fachdienst beziehen;
4086 eine fachdienstübergreifende Löschung DARF NICHT serverseitig erfolgen. [**<=**]

4087 **A_29938 -Authorization Server - Fortbestand der Identität nach Löschung des** 4088 **letzten Clients**

4089 Der Authorization Server MUSS den Identitätsdatensatz (gebundene E-Mail F1) nach
4090 Löschung des letzten Clients erhalten (unbeschadet der Inaktivitäts-Löschfristen
4091 nach A_28808). Eine erneute Einbuchung MUSS als Folgeregistrierung mit erneuter
4092 Verifikation der bestehenden E-Mail behandelt werden. [**<=**]

4093 **A_29939 -Authorization Server - Autorisierungsprüfung bei Client-** 4094 **übergreifender Löschung**

4095 Der Authorization Server MUSS bei der Löschung eines anderen Clients ([DELETE](#)
4096 [/zeta/clients/{target_client_id}](#)) prüfen, dass der Ziel-Client tatsächlich zur Identität des
4097 aufrufenden Registration Access Token gehört, und einen nicht zur Identität gehörenden
4098 Ziel-Client ablehnen ([403](#)). Der Besitz einer Objektreferenz ([target_client_id](#)) allein DARF
4099 NICHT als Autorisierung gelten (Schutz gegen Broken Object Level Authorization). [**<=**]

4100 Genutzte Schnittstellen (Annex [zeta-guard-client-management]): [GET /zeta/clients](#) ,
4101 [DELETE /zeta/clients/{target_client_id}](#) , [DELETE /register/{client_id}](#) (Selbst-
4102 Deregistrierung, RFC 7592).

4103 **5.5.10 Außerordentliche Löschung (Out-of-Band) und** 4104 **Protokollierung**

4105 **A_29940 -Authorization Server - OOB-Löschung als Notfallpfad**

4106 Der Authorization Server MUSS einen außerordentlichen Löschpfad bereitstellen, wenn
4107 ein Nutzer alle online verfügbaren Faktoren (F1, F2) verloren hat und eine Selbst-
4108 Recovery nicht mehr möglich ist. Der Prozess DARF NICHT Zugriff gewähren, ein Token
4109 ausstellen oder einen Faktor neu binden. [**<=**]

- 4110 **A_29941 -Authorization Server - IDP-unabhängiger, hochassuranter**
4111 **Identitätsnachweis**
4112 Der Authorization Server MUSS für die OOB-Löschung einen vom sektoralen IDP
4113 unabhängigen Identitätsnachweis auf hohem Vertrauensniveau verlangen. Ein Nachweis
4114 auf niedrigerem Niveau DARF NICHT akzeptiert werden. [<=]
- 4115 **A_29942 -Authorization Server - Verzögerte Ausführung mit Veto-Fenster**
4116 Der Authorization Server MUSS eine OOB-Löschung mit einer Einspruchsfrist planen und
4117 DARF sie NICHT sofort ausführen. Der Authorization Server MUSS während der Frist alle
4118 überlebenden, gebundenen Kanäle benachrichtigen (vgl. A_25750). Die Dauer der
4119 Einspruchsfrist MUSS über Policy definiert und dokumentiert sein und MUSS mindestens
4120 so bemessen sein, dass die Benachrichtigung zugestellt und ein überlebender Faktor
4121 rechtzeitig ein Veto einlegen kann. [<=]
- 4122 **A_29943 -Authorization Server - Veto durch überlebenden Faktor**
4123 Der Authorization Server MUSS während der Einspruchsfrist den Abbruch der geplanten
4124 Löschung durch den Proof-of-Possession eines noch eingebuchten Clients (F2) zulassen.
4125 Ein Veto allein durch Kontrolle der gebundenen E-Mail (F1) ist NICHT vorgesehen. Verfügt
4126 der Nutzer über keinen eingebuchten Client mehr, MUSS er zunächst ein Gerät neu
4127 registrieren (Folgerregistrierung mit erneuter Verifikation der gebundenen E-Mail F1,
4128 A_29910) und kann anschließend mit diesem Client das Veto einlegen. [<=]
- 4129 **A_29944 -Authorization Server - Wirkung der Ausführung: Tombstone, kein**
4130 **Zugriff**
4131 Der Authorization Server MUSS bei Ausführung der OOB-Löschung den
4132 Identitätsdatensatz in einen Tombstone-Zustand überführen und alle zugehörigen Client-
4133 Registrierungen entfernen. Die OOB-Löschung DARF NICHT Zugriff gewähren oder einen
4134 Faktor neu binden. [<=]
- 4135 **A_29945 -Authorization Server - Zweite, unabhängige OOB-Bestätigung für**
4136 **Wiederregistrierung**
4137 Der Authorization Server MUSS nach einem Tombstone die Erstnutzung der betroffenen
4138 Identität sperren und diese Sperre erst mit dem Erfassen einer zweiten, unabhängigen
4139 OOB-Bestätigung aufheben. Diese Bestätigung MUSS durch einen anderen Operator
4140 erfolgen als die Ausführung der Löschung (Funktionstrennung). Diese Funktionstrennung
4141 ist ein zweiter, eigenständiger Vier-Augen-Schritt in einer späteren Lebenszyklus-Phase
4142 (Wiederregistrierung nach dem
4143 Tombstone) und NICHT identisch mit dem Vier-Augen-Prinzip bei der Auslösung der
4144 Löschung (A_29946). Die Sperre ist ereignisgesteuert: Sie endet ausschließlich mit dieser
4145 Bestätigung und DARF NICHT allein durch Zeitablauf aufgehoben werden (bleibt ohne
4146 Bestätigung unbegrenzt bestehen). [<=]
- 4147 **A_29946 -Authorization Server - Funktionstrennung bei der Löschung (Vier-**
4148 **Augen-Prinzip)**
4149 Der Authorization Server SOLL die Auslösung einer OOB-Löschung einem Vier-Augen-
4150 Prinzip unterwerfen, bei dem die Freigabe der Löschung durch einen anderen Operator
4151 erfolgt als deren Initiierung. Diese Funktionstrennung wirkt vor der Ausführung
4152 (Tombstone, A_29944) und ist ein von der zweiten, unabhängigen Bestätigung bei der
4153 Wiederregistrierung (A_29945) getrennter, früherer Vier-Augen-Schritt: A_29943 sichert
4154 das *Veto der Löschung*, A_29945 die *spätere Wiederregistrierung* nach dem Tombstone.
4155 [<=]
- 4156 **A_29947 -Authorization Server - Vorrang der Selbst-Recovery**
4157 Der Authorization Server SOLL vor einer OOB-Löschung prüfen, ob noch ein überlebender
4158 Faktor vorliegt, und in diesem Fall auf die Selbst-Recovery verweisen. [<=]
- 4159 **A_29948 -Authorization Server - Revisions sichere Protokollierung**
4160 Der Authorization Server MUSS alle sicherheitsrelevanten Lebenszyklus-
4161 Ereignisse (Erst-/Folgerregistrierung, Bindung/Verifikation von Faktoren, Rollover,
4162 Recovery, Client-Löschung, OOB-Planung/Veto/Ausführung, Wiederregistrierungs-

4163 Bestätigung) reversionssicher protokollieren, jeweils mit Zeitstempel und — bei
4164 operatorgetriebenen Aktionen — der Operatoridentität und Begründung (erweitert
4165 A_25738, A_25749).[<=]

4166 **A_29949 -ZETA Guard - Absicherung des OOB-Administrationskanals**

4167 Der ZETA Guard MUSS den Administrationskanal für OOB-Operationen über
4168 gegenseitige TLS-Authentisierung (mTLS nach [RFC8705]) absichern und für die
4169 Operatorauthentisierung eine vom sektoralen IDP unabhängige Identitätsquelle
4170 verwenden.[<=]

4171 Hinweis: Genutzte Schnittstellen Operator-Pfad — Annex [zeta-guard-admin-oob.yaml](#)
4172 (POST /deletions , POST /deletions/{deletion_id}/approve, POST
4173 /identities/{identity_ref}/reregister-confirm , mTLS +
4174 IDP-unabhängige Operator-Auth nach [RFC8705]). Clientseitiges Veto — Annex
4175 [zeta-guard-client-management]POST /zeta/deletions/{deletion_id}/veto.

4176 **5.6 ZETA Guard**

4177 Die Software des ZETA Guards wird im Auftrag der gematik entwickelt und alle
4178 Komponenten als signierte OCI Images und als signiertes Helm Chart in einer gematik
4179 Artifact Registry bereitgestellt, sodass die Anbieter von TI 2.0 Diensten ihren spezifischen
4180 ZETA Guard darauf aufbauend in ihrer Kubernetes Umgebung konfigurieren und
4181 betreiben können.

4182

4183 **A_28798 -ZETA Guard - Ausführung in einem Kubernetes Cluster**

4184 Der Anbieter eines TI 2.0-Dienstes MUSS ZETA Guard grundsätzlich in einem Kubernetes
4185 Cluster betreiben.

4186 Wenn z. B. in einer VAU die Nutzung von Kubernetes nicht sinnvoll möglich ist, dann
4187 MUSS der Anbieter des TI 2.0-Dienstes nachweisen, dass seine Lösung ohne Kubernetes
4188 eine hinreichende Verfügbarkeit, Skalierbarkeit und Betriebsstabilität ermöglicht.[<=]

4189 **A_26519 -ZETA Guard, Unterstützung von Service-Mesh Lösungen**

4190 Die Komponenten des ZETA Guard MÜSSEN es ermöglichen, dass Service-Mesh Lösungen
4191 zur Verwaltung der Komponenten eingesetzt werden können.[<=]

4192 **A_26521 -ZETA Guard, Unterstützung von Canary Releases**

4193 Die Komponenten des ZETA Guard MÜSSEN Canary Releases unterstützen.[<=]

4194 **A_28851 -ZETA Guard - Architektur der TLS-Terminierung**

4195 Der Anbieter des TI 2.0-Dienstes MUSS sicherstellen, dass alle von außen eingehenden
4196 Verbindungen zum PEP HTTP Proxy und zum PDP Authorization Server über TLS
4197 abgesichert sind.

4198 Die TLS-Terminierung KANN dabei flexibel erfolgen:

- 4199 • durch ein dem ZETA Guard vorgeschaltetes System (z. B. CDN, WAF oder externer
4200 Ingress),
- 4201 • durch den ZETA-internen Ingress / API-Gateway,
- 4202 • oder direkt an den Komponenten PEP HTTP Proxy und PDP Authorization Server.

4203 [=<=]

4204 *Hinweis: Die TLS-Terminierung für den PDP Authorization Server und den PEP HTTP Proxy*
4205 *kann über eine gemeinsame TLS-Verbindung (gleicher FQDN, gleicher Port) abgebildet*
4206 *werden.*

4207 **A_29868 -ZETA Guard - Dual-Stack bei TLS-Terminierung**

4208 Der Anbieter des TI 2.0-Dienstes MUSS sicherstellen, dass die gemäß A_28851
4209 vorgesehenen TLS-Terminierungspunkte über IPv4 und IPv6 erreichbar sind. Die
4210 Sicherheitsmechanismen (z. B. Zertifikatsprüfung, Rate-Limiting, Protokollierung) MÜSSEN
4211 für beide IP-Versionen gleichwertig umgesetzt werden. [≤]

4212 **A_28852 -ZETA Guard - TLS-Terminierung und ASL bei Einsatz einer VAU**

4213 Falls der ZETA Guard Daten mit dem Schutzbedarf „sehr hoch“ verarbeitet und in einer
4214 VAU (Vertrauenswürdige Ausführungsumgebung) betrieben wird, MUSS der Anbieter des
4215 TI 2.0-Dienstes sicherstellen, dass die TLS-Verbindung des Clients erst innerhalb der VAU
4216 terminiert wird (z. B. am PEP und PDP oder Ingress innerhalb der VAU).

4217 Wird die TLS-Verbindung stattdessen an einer Komponente außerhalb der VAU terminiert
4218 (z. B. durch ein externes CDN oder eine WAF), MUSS zwingend der ZETA/ASL-Kanal
4219 (Application Security Layer) für PEP und PDP Endpunkte verwendet werden, um die
4220 Vertraulichkeit und Integrität der Daten vom Client bis in die VAU sicherzustellen.

4221 [≤]

4222 **A_25666-01 -ZETA Guard - TLS-Terminierung**

4223 Die Komponente Ingress MUSS TLS für von außen eingehende Verbindungen terminieren
4224 können.

4225 [≤]

4226 *Hinweis: TLS-Verbindungen von ZETA Clients können dort terminiert werden, wo es aus*
4227 *Sicht des Anbieters des TI 2.0-Dienstes am sinnvollsten ist (z. B. an einem CDN). Um*
4228 *dennoch eine verschlüsselte Verbindung bis zum ZETA Guard zu haben, kann ZETA/ASL*
4229 *verwendet werden.*

4230 **A_26964-01 -ZETA Guard - OCSP Stapling**

4231 Komponenten innerhalb des ZETA Guards (inkl. Ingress), die von außen kommende TLS-
4232 Verbindungen terminieren, SOLLEN OCSP-Stapling [RFC6066] verwenden.

4233 [≤]

4234 **A_26639 -ZETA Guard - Unterstützung Websocket**

4235 Die Komponente Ingress MUSS das WebSocket-Protokoll unterstützen. [≤]

4236 **A_26640 -ZETA Guard - HTTP Protokoll-Versionen**

4237 Die Komponente Ingress MUSS das HTTP Protokoll in den Versionen HTTP/1.1, HTTP/2 und
4238 SOLL HTTP/3 unterstützen. [≤]

4239 **A_25652 -ZETA Guard - Notification Service**

4240 Der ZETA Guard Notification Service MUSS Push-Notifications über die von App-Anbietern
4241 bereitgestellten Push-Gateways unterstützen, um die Notifications an bestimmte oder alle
4242 registrierte Clients eines Anwenders verschicken zu können.

4243 Der Notification Service MUSS gemäß [gemF_PushNotification] die Push Konfigurationen
4244 von ZETA Clients registrieren und verwalten können und MUSS die vom Resource Server
4245 empfangenen Notification Events an die dafür registrierten Push Gateways der Clients
4246 weiterleiten. [≤]

4247 **A_25737 -ZETA Guard - Push Notification**

4248 Der ZETA Guard MUSS eine Push Benachrichtigung an alle registrierten Clients des
4249 Nutzers, für die eine Push Notification aktiviert ist, verschicken, sobald sich Änderungen
4250 an der Liste der registrierten Clients dieses Nutzers ergibt. [≤]

4251 **A_26988 -Telemetriedaten Service - Fehlermeldungen**

4252 Alle Komponenten des ZETA Guard MÜSSEN ihre Fehlermeldungen so bereitstellen, dass
4253 der Telemetriedaten Service die Fehlermeldungen sammeln und an einen Monitoring
4254 Service weiterleiten kann. [≤]

4255 **A_26661 -ZETA Guard - HTTP Statuscodes**

4256 Der ZETA Guard MUSS die HTTP Statuscodes gemäß Tabelle "ZT_HTTP_Statuscodes"
4257 unterstützen. [≤]

4258 **A_26662 -ZETA Guard, HTTP Fehlerdetails**

4259 Der ZETA Guard MUSS bei Beantwortung eines Requests mit einem HTTPFehler ein JSON
4260 Object ergänzen, das den Fehler beschreibt. Das JSON Object MUSS gemäß [zeta-
4261 error.yaml] aufgebaut sein. [≤]

4262 **Beispiel für eine sinnvolle Ergänzung der Fehlerdetails:**

4263 Bei der Authentifizierung mit Client Assertion JWT fehlt im JWT eine Angabe
4264 product_version oder dieproduct_version ist nicht in der Policy Engine bekannt, dann
4265 antwortet der Authorization Server mit HTTP Status 400 Bad Request sowie einer
4266 Beschreibung, dass dieproduct_version fehlt oder nicht bekannt ist.

4267 **A_27802-03 -ZETA Guard, JWT Prüfung**

4268 Der ZETA Guard MUSS bei der Prüfung von JWT die Prüfschritte gemäß [https://www.rfc-
4269 editor.org/rfc/rfc7519.html#section-7.2](https://www.rfc-editor.org/rfc/rfc7519.html#section-7.2) und
4270 <https://www.rfc-editor.org/rfc/rfc7515.html#section-5.2> durchführen.

4271 [≤]

4272 **A_29046 -ZETA Guard, spezifische Prüfungen für TI-Zertifikate**

4273 Der ZETA Guard MUSS bei der Prüfung von Signaturen mit TI-Zertifikaten zusätzlich
4274 folgende Prüfungen durchführen:

- 4275 • Integrität der Nachricht: Hash-Überprüfung: Sicherstellen, dass die Nachricht nicht
4276 manipuliert wurde.
- 4277 • Ablaufdatum des Zertifikats: Sicherstellen, dass das Zertifikat noch gültig ist.
- 4278 • Widerrufsstatus des Zertifikats (OCSP): Prüfen, ob das Zertifikat nicht widerrufen
4279 wurde. Die OCSP Response wird für eine konfigurierbare Dauer zwischengespeichert,
4280 um zu häufige OCSP Anfragen zu verhindern. Die Dauer soll der Gültigkeitsdauer des
4281 Refresh Token entsprechen.
- 4282 • Vertrauenswürdigkeit der Zertifikatskette: Sicherstellen, dass die Kette zu einer
4283 vertrauenswürdigen Root-CA führt.

4284 [≤]

4285 **A_28963 -ZETA Guard, DPoP Prüfung**

4286 Der ZETA Guard MUSS bei der Prüfung von JWT die Prüfschritte gemäß [\[RFC9449\]#name-
4287 checking-dpop-proofs](#) durchführen.

4288 [≤]

4289 **A_26668-02 -ZETA Guard - Rate Limit**

4290 Die Komponenten des ZETA Guard MÜSSEN für ihre durch ZETA Clients erreichbaren
4291 Endpunkte ein Rate Limit konfigurierbar einstellen können. Wenn ein Rate Limit
4292 konfiguriert ist, dann MUSS der Client über folgende Response Header informiert werden:

- 4293 • ((das erlaubte Limit),
- 4294 • (verbleibende Anfragen) und
- 4295 • (Zeitpunkt, an dem das Limit zurückgesetzt wird)).

4296 [≤]

4297 Hinweis: Es gibt einen RFC Draft, der Rate Limits neu spezifiziert
4298 (<https://datatracker.ietf.org/doc/draft-ietf-httpapi-ratelimit-headers/>). Es ist geplant, dass
4299 nach Veröffentlichung des RFC ZETA Guard angepasst wird, um die neuen Rate Limit
4300 Festlegungen zu unterstützen.

4301 **A_27864-02 -ZETA Guard - Egress**

4302 Der ZETA Guard MUSS eine Egress Funktion implementieren, die durchsetzt, dass nur
4303 erlaubte ZETA Guard Verbindungen, zu Endpunkten außerhalb des Kubernetes Clusters,
4304 aufgebaut werden.

4305 Zu den erlaubten ZETA Guard Verbindungen gehören mindestens:

- 4306 • Telemetriedaten-Empfänger der gematik
- 4307 • SIEM der gematik
- 4308 • Sektorale IDPs (wenn Versicherte zur Nutzergruppe gehören)
- 4309 • Federation Master der TI
- 4310 • ZETA Artifact Registry
- 4311 • DNS Resolver
- 4312 • OCSP Responder oder CRL (TLS TSPs nach CAB Forum)
- 4313 • SMC-B TSP OCSP Responder
- 4314 • OCSP Responder des TSP der Komponenten PKI der TI
- 4315 • Notification Services der App Hersteller
- 4316 • Mail Server des TI 2.0 Dienst Anbieters (für TOFU OTP Codes; wenn Versicherte zur
- 4317 Nutzergruppe gehören)
- 4318 • PoPP Service (Download des JWKS, wenn PoPP Token durch den TI 2.0 Dienst
- 4319 verwendet werden)

4320 Weitere erlaubte Verbindungen zu Endpunkten des Anbieters, zu Endpunkten von
4321 Clientsystem Notification Services oder für Dienst-zu-Dienst Kommunikation können
4322 hinzugefügt werden. [<=]

4323 **A_29869 -ZETA Guard - Dual-Stack in Egress- und Zielauflösung**

4324 Die Egress-Funktion MUSS Verbindungen zu erlaubten Zielen über IPv4 und IPv6
4325 unterstützen. Bei FQDN-basierten Zielen MÜSSEN A- und AAAA-Auflösungen verarbeitet
4326 werden können [<=]

4327 **A_28435 -ZETA Guard, Ingress - Unterstützung Forwarded-Header**

4328 Die Komponente Ingress MUSS in jeder empfangene HTTP-Anfrage für die nachgelagerten
4329 Komponenten PDP Authorization Server und PEP HTTP Proxy den Forwarded-Header
4330 gemäß [RFC 7239] in der weitergeleiteten Anfrage hinzufügen oder aktualisieren. [<=]

4331 **A_29870 -ZETA Guard - Verarbeitung von IPv6 in Forwarded-Headern**

4332 Die Komponenten des ZETA Guard MÜSSEN IPv6-Adressen in Forwarded-Headern gemäß
4333 RFC 7239 korrekt verarbeiten und an nachgelagerte Komponenten konsistent
4334 weiterreichen. [<=]

4335 **A_28526-01 -ZETA Guard - Bereitstellung der lokalen Artifact Registry**

4336 Der Anbieter eines TI 2.0 Dienstes MUSS für seinen ZETA Guard eine lokale Artifact
4337 Registry bereitstellen, die alle benötigten Container Images aus der gematik Artifact
4338 Registry enthält.

4339 Die lokale Artifact Registry MUSS so konfiguriert werden, dass
4340

- 4341 • OCI Container Images für die ZETA Guard Komponenten in der benötigten Version
- 4342 vorhanden sind,
- 4343 • das ZETA Guard Provisioning Container Image alle 60 Minuten aktualisiert wird.

4344 [<=]

4345 *Hinweis: Die Verfügbarkeit von ZETA Guard soll durch die lokale Bereitstellung der*
4346 *Artifact Registry erhöht werden. Die Policies und Daten werden von der Policy Engine*
4347 *direkt aus der ZETA Artifact Registry geladen und aktualisiert.*

4348 **5.6.1 Klassifizierung der ZETA Guard Komponenten**

4349 Der ZETA Guard ist ein logisches Bundle das in der Laufzeitumgebung des TI-2.0-
4350 Dienstansbieters (in der Regel ein Kubernetes-Cluster) betrieben wird. Die Komponenten
4351 des ZETA Guard werden nach ihrem Funktionsumfang, ihrer Austauschbarkeit und der
4352 Verantwortung für die Bereitstellung klassifiziert. Diese Klassifizierung bestimmt die
4353 Möglichkeiten bei der Integration von ZETA Guard in die individuelle Laufzeitumgebung
4354 des TI-2.0-Dienstansbieters und die Sicherheitsvorgaben z. B. beim Betrieb in einer VAU.

4355 **5.6.1.1 Unveränderliche Kernkomponenten**

4356 Diese Komponenten bilden den funktionalen Kern des ZETA Guard. Sie werden von der
4357 gematik als signierte OCI-Images in der ZETA Artifact Registry bereitgestellt.

4358 **A_28789 -ZETA Guard - Integrität der Kernkomponenten**

4359 Der Anbieter eines TI 2.0-Dienstes MUSS die folgenden Komponenten zwingend als
4360 unveränderte Images der gematik beziehen und deren Signatur vor dem Deployment
4361 validieren:

- 4362 • PEP HTTP Proxy
- 4363 • PDP Authorization Server
- 4364 • PDP Policy Engine (OPA)
- 4365 • Telemetriedaten-Service
- 4366 • Notification Service (nur für mobile Clients)

4367 Wenn z. B. in einer VAU-Umsetzung die Images der gematik nicht unverändert
4368 übernommen werden können, dann MUSS per Attestation nachgewiesen werden, dass
4369 das verwendete Image auf dem unveränderten Source-Code der gematik basiert.
4370 [**<=**]

4371 **5.6.1.2 Austauschbare Kernkomponenten**

4372 Diese Komponenten sind für den Betrieb des ZETA Guard notwendig. Der Anbieter des TI
4373 2.0-Dienstes kann wählen, ob die von gematik bereitgestellten Images verwendet werden
4374 oder ob eigene Lösungen eingesetzt werden.

4375 **A_28790 -ZETA Guard - Kernkomponenten, Verwendung von eigenen Lösungen**

4376 Der Anbieter eines TI 2.0-Dienstes MUSS bei Verwendung eigener Lösungen der
4377 folgenden Infrastruktur-Komponenten sicherstellen, dass die Anforderungen an die
4378 Komponenten erfüllt werden und dass die Schnittstellen zur Kernkomponenten-Schicht
4379 kompatibel sind.

4380 Folgende ZETA Guard Komponenten können durch eigene Lösungen ersetzt werden:

- 4382 • PDP Datenbank
- 4383 • HSM Proxy
- 4384 • Local Artifact Registry Cache
- 4385 • Gateway oder Ingress und Egress

4386 [**<=**]

4387 *Hinweis: Bei der Verwendung von eigenen Lösungen für ZETA Guard Kernkomponenten*
4388 *muss die Umsetzung und Einhaltung der zu den Komponenten gehörenden*
4389 *Anforderungen gemäß Anbietertypsteckbrief nachgewiesen werden. Im ZETA Guard*
4390 *Produkthandbuch ist beschrieben, was hinsichtlich Kompatibilität zu den ZETA Guard*
4391 *Kernkomponenten zu beachten ist.*

4392 A_28432 -ZETA Guard, Komponenten Ingress optional

4393 Der Ingress MUSS als optionale Komponente im ZETA Guard angeboten werden. [**<=**]

4394 Es ist möglich auf die Komponente Ingress im ZETA Guard zu verzichten. Dadurch wird
4395 der Hersteller des TI 2.0-Dienstes in die Lage versetzt seinen eigenen externen Ingress zu
4396 verwenden.

4397 A_28433 -ZETA Guard, Bereitstellung externer Ingress

4398 Der Hersteller des TI 2.0-Dienstes MUSS entweder den Kubernetes Ingress des ZETA
4399 Guard oder einen eigenen Ingress verwenden. [**<=**]

4400 5.6.1.3 Hilfskomponenten und Funktionen

4401 Diese Komponenten und Funktionen bieten zusätzliche Funktionen zur Verbesserung der
4402 Sicherheit und zur Vereinfachung des Betriebs. Sie können durch eigene Lösungen
4403 umgesetzt werden. ZETA Guard enthält keine fertigen Images, sondern ausschließlich
4404 unterstützende Konfigurationen (z. B. Policies für einen Admission Controller).

4405 A_28792 -ZETA Guard - Hilfskomponenten und Funktionen

4406 Der Anbieter eines TI 2.0-Dienstes MUSS bei Verwendung eigener Lösungen der
4407 Hilfskomponenten sicherstellen, dass die Anforderungen an die Komponenten erfüllt
4408 werden und dass die Schnittstellen zur Kernkomponenten-Schicht kompatibel sind.

4409

4410 Die folgenden Komponenten zählen zu den Hilfskomponenten:

4411 • Admission Controller

4412 • Service Mesh

4413 [**<=**]

4414 *Hinweis: Bei der Verwendung von eigenen Lösungen für ZETA Guard Hilfskomponenten*
4415 *und Funktionen muss die Umsetzung und Einhaltung der zu den Komponenten*
4416 *gehörenden Anforderungen gemäß Anbietertypsteckbrief nachgewiesen werden. Für die*
4417 *Hilfskomponenten und Funktionen werden keine Images durch die gematik bereitgestellt.*

4418 5.6.2 Kommunikation mit Diensten der gematik

4419 ZETA Guard Instanzen benötigen Zugriff auf Dienste der gematik, um PIP und PAP OCI
4420 Container Images zu laden, um Telemetriedaten an den gematik Empfänger zu senden
4421 und um Security Events an das gematik SIEM zu senden. Für den Zugriff auf diese Dienste
4422 ist ein gültiges Access Token erforderlich.

4423 Um ein gültiges Access Token zu erhalten wird Workload Identity Federation eingesetzt.
4424 Voraussetzung dafür ist, dass

4425 • der Kubernetes Cluster, in dem ZETA Guard ausgeführt wird, seine OpenID
4426 Konfiguration und seine JWKS URI (`/.well-known/openid-configuration` und bei
4427 Kubernetes IDP/openid/v1/jwks) öffentlich erreichbar bereitstellt (alternativ kann ein
4428 Schlüssel (JWK) im Workload Identity Federation Pool der gematik registriert werden),

4429 • ein ServiceAccount innerhalb des Clusters existiert, der die benötigten Secrets
4430 (Access Token) für die Workloads bereitstellt und

4431 • die Issuer URI des Kubernetes Cluster bei der gematik registriert ist.

4432 Siehe auch Kapitel [5.13.8- Prozesse zur Inbetriebnahme eines ZETA Guard](#).

4433 *Hinweis: Der Kubernetes IDP wird vom kube-apiserver umgesetzt. Der API-Server ist die*
4434 *Instanz, die die Identitätsnachweise (Tokens) signiert und die notwendigen Metadaten für*
4435 *externe Dienste bereitstellt. Damit dies funktioniert, muss er mit bestimmten Flags*

4436 konfiguriert sein (--service-account-issuer, -service-account-signing-key-file, --service-
4437 account-jwks-uri).

4438 Für die Workload Identity Federation (WIF) nutzt Kubernetes die TokenRequest API. Diese
4439 API ermöglicht es, kurzlebige, zielgruppenspezifische (Audience-bound) JSON Web Tokens
4440 (JWT) zu erstellen. Ein externer Dienst (gematik Telemetriedaten Empfänger) akzeptiert
4441 nur Tokens,
4442 - deren Issuer im gematik WIF Pool registriert sind (per Onboarding Prozess für den TI 2.0
4443 Dienst Anbieter) und
4444 - die eine für ihn definierte aud (Audience) enthalten.

4445 Damit die gematik Dienste die Token validieren können, muss entweder der Token
4446 Signatur-Schlüssel bei der gematik registriert sein oder es werden die standardisierten
4447 Endpunkte des api-servers öffentlich bereitgestellt:

- 4449 • /.well-known/openid-configuration: Das Discovery-Dokument, das die unterstützten
4450 Funktionen und die URL zum Schlüsselsatz enthält.
- 4451 • /openid/v1/jwks: Der JSON Web Key Set (JWKS), der die öffentlichen Schlüssel zur
4452 Verifizierung der Signatur enthält.

4453 5.6.3 Deployment Szenarien

4454 5.6.3.1 Geo-Redundanz

4455 Der Betrieb von Kubernetes in einer Multi-Cluster-Umgebung bietet Unternehmen
4456 erhebliche Vorteile in Bezug auf Skalierbarkeit, Ausfallsicherheit und Flexibilität, bringt
4457 jedoch auch eine erhöhte Komplexität in Verwaltung, Netzwerk und Sicherheit mit sich
4458 und es ergeben sich Anforderungen an Anbieter von TI 2.0-Diensten.

4459 **A_28438-01 -ZETA Guard, geo-redundanter Betrieb**

4460 Der Anbieter eines TI 2.0-Dienstes MUSS beim geo-redundanten Betrieb des ZETA Guard
4461 in einer Kubernetes Multi-Cluster-Umgebung folgende Bedingung erfüllen:

- 4462 • Die Authorization Server Instanzen müssen über einen globalen Load Balancer als ein
4463 logischer Authorization Server bereitgestellt werden.

4464 [**<=**]

4465 **A_28464 -ZETA Guard, genau ein Authorization Server**

4466 Die Komponente PEP HTTP Proxy MUSS das OAuth Protected Resource Well-known so
4467 bereitstellen, dass für ZETA Clients der ZETA Guard Authorization Server als genau eine
4468 logische Komponente bereitgestellt wird (nur ein Eintrag `authorization_servers` im
4469 OAuth Protected Resource Well-known). [**<=**]

4470 5.6.4 Provisioning Image für den ZETA Guard

4471 Das Provisioning Image dient der sicheren Bereitstellung und Verteilung von
4472 kryptografischen Trust Anchors (Vertrauensankern) und Konfigurationsdaten, die der
4473 ZETA Guard zur Verifikation von Signaturen, Attestierungen und Zertifikaten benötigt. Es
4474 handelt sich um ein rein datenführendes OCI Image, welches von der gematik signiert
4475 bereitgestellt wird.

4476 **Pfad für die Produktivumgebung:**

4477 `europa-west3-docker.pkg.dev/gematik-pt-zeta-prod/zeta-provisioning/zeta-`
4478 `guard-provisioning:latest`

4479 **Pfad für die Testumgebung:**

4480 europe-west3-docker.pkg.dev/gematik-pt-zeta-test/zeta-provisioning/zeta-
4481 guard-provisioning:latest

4482 5.6.4.1 Struktur und Inhalt

4483 Das Provisioning Image enthält folgende Daten:

```
4484 .
4485 |-- .manifest
4486 |-- .revision
4487 |-- ECC-RSA_TSL-test.xml
4488 |-- TrustedTpm.cab
4489 |-- android-roots
4490 |   |-- roots.json
4491 |-- apple-roots
4492 |   |-- apple-root.pem
4493 |-- federation-master
4494 |   |-- federation-master.yaml
4495 |-- federation-master.yaml
4496 |-- policy-engine-bundle-keys
4497 |   |-- ca
4498 |   |   |-- GEM.KOMP-CA8.pem
4499 |   |   |-- GEM.RCA7.pem
4500 |   |-- signers
4501 |       |-- ZETA_PIPPAP_Policies_03.06.2026_16_31.pem
4502 |-- roots.json
4503 |-- ti-roots
4504 |   |-- roots.json
4505 |-- trusted-tpm
4506 |   |-- TrustedTpm.cab
4507 |-- tsl
4508 |   |-- ECC_PU_TSL_10322.xml
```

4509 Das Provisioning Image wird als OCI-konformes Image in der oben dargestellten
4510 Verzeichnisstruktur bereitgestellt.

4511 Das Provisioning Image enthält im Root-Verzeichnis die Dateien `.manifest` und `.revision`.

4512 **.manifest:** Diese Datei enthält die SHA-256-Prüfsummen aller im Image enthaltenen
4513 Dateien im Format:

4514 <SHA-256-Hex-Hash> <Relativer-Pfad>.

4515 **.revision:** Diese Datei enthält die eindeutige Version des Bereitstellungsstands als Git-
4516 Commit-Hash oder als Versionsstring.

4518 *Hinweis: Die Dateien im Root-Verzeichnis (außer `.manifest` und `.revision`), namentlich:*

- 4519 • `ECC-RSA_TSL-test.xml`
- 4520 • `TrustedTpm.cab`
- 4521 • `federation-master.yaml`
- 4522 • `roots.json`

4523 *werden aus Gründen der Abwärtskompatibilität weiterhin mitgeführt.*

4524 **A_29737 -ZETA Guard, Verwendung der Daten aus den Unterverzeichnissen des** 4525 **Provisioning Images**

4526 ZETA Guard Komponenten MÜSSEN die in den Unterverzeichnissen des Provisioning
4527 Images strukturierten Dateien verwenden. [**<=**]

4528 **A_29738 -ZETA Guard, Dynamische Dateinamererkennung in**
4529 **Unterverzeichnissen des Provisioning Images**

4530 Die Komponenten des ZETA Guard DÜRFEN NICHT auf fest vorgegebene (hardcodierte)
4531 Dateinamen für die Nutzdaten in den Unterverzeichnissen des Provisioning Images
4532 vertrauen, da sich diese im Rahmen von Aktualisierungen ändern können. [**<=**]

4533 *Hinweis: Insbesondere bei Dateien mit versions- oder laufzeitabhängigen*
4534 *Namensbestandteilen – wie der Trust Service Status List (TSL) im Verzeichnis /tsl/ (z. B.*
4535 *ECC_PU_TSL_10322.xml) oder den Richtlinien-Signaturen – MÜSSEN die ZETA Guard*
4536 *Komponenten die zu ladenden Dateien dynamisch ermitteln. Dies kann beispielsweise*
4537 *durch das Auslesen des Verzeichnisinhalts (Directory Listing), das Filtern nach*
4538 *Dateiendungen (z. B. *.xml, *.pem, *.json) oder die Auswertung der in der .manifest-Datei*
4539 *gelisteten Pfade gelöst werden.*

4540 **5.6.4.2 Integritätsschutz und Verifikation**

4541 Das von der gematik bereitgestellte Provisioning Image ist kryptografisch signiert. Das für
4542 die Verifikation genutzte Signaturzertifikat der gematik wird im offiziellen GitHub-
4543 Repository unter:

4544 <https://github.com/gematik/zeta/tree/main/zeta-guard-prv-signing-key>

4545 in den umgebungsspezifischen Verzeichnissen (prod und test) veröffentlicht.

4546 **A_29739 -ZETA Guard, Verifikation des Provisioning Images**

4547 Der ZETA Guard MUSS vor dem Laden und Verarbeiten der Inhalte des Provisioning
4548 Images folgende Prüfschritte erfolgreich durchführen:

- 4549 • Verifikation der Image-Signatur: Die Signatur des OCI-Images MUSS gegen das für die
4550 jeweilige Umgebung (prod/test) gültige Signaturzertifikat der gematik erfolgreich
4551 validiert werden.

4552 [**<=**]

4553 **5.6.4.3 Verarbeitung durch ZETA Guard Komponenten**

4554 **A_29740 -PDP Authorization Server, Verwendung Trust Anchor des Provisioning**
4555 **Images**

4556 Der ZETA Guard Authorization Server MUSS die im Provisioning Image enthaltenen Trust
4557 Anchor für folgende Validierungsschritte nutzen:

- 4558 • TI-Zertifikate (TSL): Die Trust Service Status List (TSL) unter /tsl/ sowie die TI-Roots
4559 unter /ti-roots/roots.json MÜSSEN zur Validierung von SMC-B-Zertifikatsketten
4560 herangezogen werden.
- 4561 • Plattform-Attestierung:
 - 4562 • TPM-Attestierung: Zur Verifikation von hardwarebasierten Attestierungsdaten der
4563 Clients MÜSSEN die TPM-Stammzertifikate aus /trusted-tpm/TrustedTpm.cab
4564 verwendet werden.
 - 4565 • Android-Attestierung: Zur Verifikation von Android-Clients MÜSSEN die Trust
4566 Anchors aus /android-roots/roots.json verwendet werden.
 - 4567 • Apple-Attestierung: Zur Verifikation von iOS-/macOS-Clients MÜSSEN die Apple-
4568 Wurzelzertifikate aus /apple-roots/apple-root.pem verwendet werden

4569 [**<=**]

4570 **A_29741 -PDP Policy Engine, Verwendung der Signer-Keys des Provisioning**
4571 **Images**

4572 Die ZETA Guard Policy Engine MUSS die im Verzeichnis /policy-engine-bundle-keys/
4573 bereitgestellten Schlüssel nutzen, um die Signaturen geladener Policy Bundles zu

4574 verifizieren:
4575 Unter /policy-engine-bundle-keys/ca/ abgelegte CA-Zertifikate dienen als
4576 Vertrauensanker für die ausstellenden Instanzen.
4577 Unter /policy-engine-bundle-keys/signers/ abgelegte Zertifikate definieren die
4578 explizit autorisierten Signaturschlüssel für die Freigabe von OPA-Policies.

4579 [**<=**]

4580 **A_29742 -HTTP Proxy - Verwendung von Trust Anchors und TSL beim ZETA/ASL-** 4581 **Verbindungsaufbau**

4582 Der ZETA Guard HTTP-Proxy MUSS für die Etablierung des ZETA/ASL-Kanals die im
4583 Provisioning Image bereitgestellten TI-Root-Zertifikate (/ti-roots/roots.json) sowie die
4584 Trust Service Status List (/tsl/*.xml) verwenden.

4585 Diese Daten MÜSSEN genutzt werden, um die aktuellen Status- und Sperrinformationen
4586 der beteiligten Zertifikate im Zuge des Handshakes zu ermitteln.

4587 [**<=**]

4588 **5.6.4.4 Lifecycle und Update-Regeln**

4589 Das Provisioning Image wird durch die gematik neu erstellt und publiziert, sobald sich
4590 enthaltene Vertrauensanker oder Konfigurationsdateien ändern.

4591 Bei regulären TSL-Updates erfolgt die Bereitstellung zyklisch (in der Regel alle 14 Tage).
4592 Bei ad-hoc Änderungen (z. B. Sperrung oder Austausch von CA-/Signaturschlüsseln)
4593 erfolgt die Bereitstellung unverzüglich.

4594 **A_29743 -ZETA Guard, Automatische und unterbrechungsfreie Aktualisierung** 4595 **der Provisioning Daten**

4596 Der ZETA Guard MUSS einen Mechanismus implementieren, um das Provisioning Image
4597 im laufenden Betrieb zu aktualisieren:

- 4598 • **Erkennung neuer Versionen:** Der ZETA Guard MUSS zyklisch auf das Vorhandensein
4599 eines neuen Images (identifiziert über das Tag latest) in der Registry prüfen.
- 4600 • **Unterbrechungsfreies Laden (Hot-Reload):** Nach erfolgreicher Signaturprüfung
4601 MÜSSEN die neuen Trust Anchor und Konfigurationen unterbrechungsfrei (Zero-
4602 Downtime / Hot-Reload) in die aktiven Konfigurationen der betroffenen Komponenten
4603 eingespielt werden. Laufende Validierungsprozesse und bestehende Sitzungen
4604 DÜRFEN hierbei nicht gestört werden

4605 [**<=**]

4606 **A_29950 -Anbieter TI 2.0 Dienst, Betriebsanforderung beim Root-CA-Wechsel**

4607 Der Anbieter des TI 2.0 Dienstes MUSS sicherstellen, dass Provisioning-Updates im Root-
4608 CA-Wechselzeitraum ohne Betriebsunterbrechung in die produktiven ZETA Guard
4609 Instanzen übernommen werden.

4610 *Hinweis: Beim Wechsel auf eine neue TI-Root-CA wird durch die gematik ein Provisioning*
4611 *Image bereitgestellt, das für einen definierten Übergangszeitraum sowohl die bisherige*
4612 *als auch die neue TI-Root-CA sowie die zugehörigen TSL-Informationen enthält. Nach*
4613 *Ende des Übergangszeitraums wird durch die gematik ein aktualisiertes Provisioning*
4614 *Image bereitgestellt, in dem die bisherige TI-Root-CA nicht mehr als gültiger*
4615 *Vertrauensanker enthalten ist.* [**<=**]

4617 **5.6.5 Laufzeitüberwachung**

4618 Um die Integrität und Vertraulichkeit der ZETA Guard Microservices innerhalb einer
4619 Kubernetes-Infrastruktur zu gewährleisten, werden verschiedene sich ergänzende
4620 Sicherheitsmechanismen eingesetzt. Diese dienen der Härtung der Laufzeitumgebung,
4621 der Isolation von Workloads und der Erkennung von Anomalien. Diese werden

4622 nachfolgend beschrieben. Die Mechanismen sind in aktuell eingesetzte und geplante
4623 Verfahren unterteilt; darüber hinaus werden weitere empfohlene Maßnahmen zur
4624 Vertiefung der Cluster-Sicherheit aufgeführt.

4625 **5.6.5.1 Aktuell eingesetzte Verfahren**

4626 *5.6.5.1.1 Pod Security Standards (PSS)*

4627 Kubernetes Pod Security Standards definieren drei Sicherheitsstufen für Pods (Privileged,
4628 Baseline, Restricted). Im ZETA Guard-Cluster soll die Stufe Restricted für alle ZETA Guard
4629 Namespaces per Namespace-Label durchgesetzt werden. Damit werden unter anderem
4630 folgende Eigenschaften erzwungen:

- 4631 • Ausführung als Nicht-Root-Benutzer ()
- 4632 • Keine privilegierten Container ()
- 4633 • Keine Fähigkeiten über die Allowlist hinaus (,)
- 4634 • Nur schreibgeschützte Root-Filesysteme ()
- 4635 • Ausschluss von Host-Netzwerk, Host-PID und Host-IPC

4636 *5.6.5.1.2 Admission Controller*

4637 Admission Controller sind Kubernetes-Webhooks, die API-Server-Anfragen vor der
4638 persistenten Speicherung validieren oder mutieren. Im ZETA Guard Cluster sollen
4639 folgende Admission Controller eingesetzt werden:

4640 Validating Admission Webhooks prüfen eingehende Ressourcen gegen Richtlinien und
4641 lehnen nicht-konforme Anfragen ab. Typische Prüfungen umfassen:

- 4642 • Sicherstellung, dass nur Images mit der gematik Signatur produktiv eingesetzt
4643 werden. Bei jedem Deployment wird geprüft, ob das Image eine gültige gematik-
4644 Signatur trägt.
- 4645 • Ablehnung von Deployments ohne definierte und

4646 Als Policy-Engine für Admission Control sollen Kyverno oder OPA Gatekeeper eingesetzt
4647 werden, um Richtlinien deklarativ als Kubernetes-Custom- Resources zu verwalten.

4648 *5.6.5.1.3 Network Policies*

4649 Kubernetes Network Policies steuern den zulässigen Netzwerkverkehr zwischen Pods auf
4650 Basis von Label-Selektoren, Namespaces und Ports. Im ZETA Guard Cluster gilt das
4651 Default-Deny-Prinzip: Jeder Namespace verfügt über eine Network Policy, die
4652 eingehenden und ausgehenden Verkehr standardmäßig verbietet. Explizite Freigaben
4653 erfolgen nur für notwendige Kommunikationsbeziehungen, insbesondere:

- 4654 • Eingehender Verkehr zum PEP (HTTP Proxy) ausschließlich vom Ingress Controller
- 4655 • Kommunikation zwischen PEP, PDP Authorization Server und Policy Engine (OPA) nur
4656 innerhalb dedizierter ZETA Guard Namespaces
- 4657 • Ausgehender Verkehr zu gematik-Diensten (PIP/PAP Artifact Registry, Telemetrie-
4658 Service) nur über definierte Egress-Regeln
- 4659 • Vollständige Isolation zwischen ZETA Guard Namespace und Resource-Server-
4660 Namespace auf Netzwerkebene

4661 *Hinweis: Native Kubernetes Network Policies sind Layer-3/4-Konstrukte und operieren auf*
4662 *IP-Adressen und Ports. Für eine weitergehende Layer-7-Kontrolle wird auf das Service*
4663 *Mesh (s. u.) verwiesen.*

4664 5.6.5.1.4 Service Mesh

4665 Ein Service Mesh (z. B. Istio, Linkerd oder Cilium) ergänzt Kubernetes um transparente
4666 mTLS-Verschlüsselung, feingranulare Traffic-Kontrolle und Observability zwischen
4667 Microservices. Das Service Mesh SOLL Kommunikationspfade zwischen ZETA Guard
4668 Komponenten absichern und überwachen. Im ZETA Guard Cluster übernimmt das Service
4669 Mesh folgende Aufgaben:

4670 **Mutual TLS (mTLS):**

4671 Alle Kommunikationsverbindungen zwischen ZETA Guard Komponenten werden durch
4672 das Service Mesh mit mTLS verschlüsselt und wechselseitig authentifiziert. Zertifikate
4673 werden automatisch rotiert.

4674 **Autorisierungsrichtlinien:**

4675 Mit AuthorizationPolicy-Ressourcen (Istio) oder Server-Ressourcen (Linkerd) wird die
4676 dienstspezifische Kommunikation auf Layer 7 eingeschränkt. So darf z. B. nur der PEP den
4677 Resource Server aufrufen.

4678 **Traffic-Observability:**

4679 Das Service Mesh liefert metrische Daten (Latenz, Fehlerrate, Throughput) und Traces für
4680 jede Dienstkommunikation, die in das Monitoring einfließen.

4681 5.6.5.1.5 Ingress und Egress / Gateway

4682 **Ingress Controller:**

4683 Externer eingehender Verkehr zum ZETA Guard wird über einen Ingress Controller (z. B.
4684 NGINX oder Envoy-basiert) terminiert. Der Ingress Controller übernimmt TLS-
4685 Terminierung, Rate Limiting und ggf. WAF-Funktionen (Web Application Firewall). Es wird
4686 ausschließlich HTTPS mit TLS 1.2 oder höher akzeptiert und die Ingress-Kommunikation
4687 wird für IPv4 und IPv6 bereitgestellt.

4688 **Egress Gateway:**

4689 Ausgehender Verkehr aus dem ZETA Guard Cluster (z. B. zu PIP/PAP-Registry,
4690 Telemetriedaten-Service, Identity Providern) wird über ein dediziertes Egress Gateway
4691 geleitet. Durch Egress-Policies auf dem Gateway wird sichergestellt, dass kein
4692 unkontrollierter Datenabfluss stattfindet. Zulässige externe Ziele werden in einer Allowlist
4693 gepflegt.

4694 *Hinweis: Am Anfang wird ZETA Guard nur Network-Policies als Egress Funktion*
4695 *verwenden.*

4696 5.6.5.2 Geplante Verfahren

4697 5.6.5.2.1 Pod-Überwachung gemäß Cilium Tetragon

4698 Cilium Tetragon ist ein eBPF-basiertes Security-Observability- und Enforcement-
4699 Framework für Kubernetes. Im Unterschied zu netzwerkorientierten Schutzmechanismen
4700 operiert Tetragon im Linux-Kernel und ermöglicht **Laufzeit-Sicherheitsüberwachung**
4701 auf Systemaufruf-Ebene (syscall-Ebene) innerhalb von Pods und Containern.

4702 *Hinweis: Da Tetragon tief in den Linux-Kernel (via eBPF) eingreift und clusterweite*
4703 *Ressourcen erstellt, benötigt das Service-Konto, das die Installation durchführt, cluster-*
4704 *admin-Rechte. Daher wird Tetragon nicht durch ZETA Guard, sondern durch den Anbieter*

4705 *installiert. Für den laufenden Betrieb und das Erstellen von Regeln reicht ein dediziertes*
4706 *RBAC-Profil für die Cilium-CRDs aus.*

4707 Im ZETA Guard Cluster ist der Einsatz von Cilium Tetragon geplant für:

4708 **Process Execution Monitoring:**

4709 Jede Prozessausführung innerhalb eines ZETA Guard Pods wird erfasst. Unerwartete
4710 Prozesse (z. B. Shell-Aufruf in einem Applikations-Container, Ausführung von
4711 Netzwerktools) lösen Sicherheitswarnungen aus und können direkt blockiert werden
4712 (Enforcement Mode). Dies erkennt insbesondere Post-Exploitation-Aktivitäten nach einer
4713 Kompromittierung.

4714 **File Integrity Monitoring (FIM):**

4715 Schreibzugriffe auf kritische Dateisystempfade innerhalb von Pods (z. B.
4716 Konfigurationsdateien, Zertifikate, Binary-Pfade) werden überwacht. Unzulässiger Zugriff
4717 auf Dateien wird unterbunden (Enforcement Mode) und gemeldet.

4718 **Network Observability auf Systemaufruf-Ebene:**

4719 Netzwerkverbindungen werden auf Basis der tatsächlichen Systemaufrufe (, , etc.)
4720 beobachtet, nicht nur auf Paketebene. Dies ermöglicht die Attribution jeder
4721 Netzwerkverbindung auf den auslösenden Prozess und Container.

4722 **TracingPolicy und Enforcement:**

4723 Über -Custom-Resources können deklarative Sicherheitsregeln definiert werden, die
4724 Tetragon im Kernel durchsetzt. Im Enforcement Mode werden unzulässige Aktionen (z. B.
4725 eines nicht-erlaubten Binaries, Verbindung zu einer nicht-erlaubten IP) direkt durch den
4726 Kernel blockiert, noch bevor eine Reaktion auf Anwendungsebene erfolgen kann.

4727 *5.6.5.2.2 RBAC-Härtung (Role-Based Access Control)*

4728 Kubernetes RBAC steuert, welche Subjekte (ServiceAccounts, User, Groups) welche API-
4729 Ressourcen lesen oder verändern dürfen. Für den ZETA Guard Cluster gilt:

- 4730 • Jeder ZETA Guard Microservice erhält einen **dedizierten ServiceAccount** mit
4731 minimalen Berechtigungen (Least Privilege).
- 4732 • Die Verwendung des default-ServiceAccounts in Pods wird durch eine Admission
4733 Policy verboten.
- 4734 • ClusterAdmin- und ClusterRole-Bindings werden auf das absolute Minimum reduziert
4735 und regelmäßig überprüft.
- 4736 • Der Kubernetes API-Server-Zugriff vom Produktivsystem wird auf dedizierte Operator-
4737 Identitäten beschränkt; interaktive Zugriffe erfordern MFA und werden auditiert.

4738 **5.7 Policy Enforcement Points**

4739 Der Policy Enforcement Point (PEP) stellt die zentrale Sicherheitskomponente einer Zero
4740 Trust-Architektur dar, da in dieser alle Zugriffsentscheidungen durchgesetzt (engl.:
4741 enforce) werden.

4742 **5.7.1 PEP HTTP Proxy**

4743 Die Komponente HTTP Proxy ist die "letzte" vor das Resource Backend geschaltete Zero
4744 Trust-Komponente und prüft das Access Token im Authorization Header des Requests. Ist
4745 das Access Token gültig, wird der Zugriff gewährt. Zudem wird der Request um
4746 zusätzliche HTTP-Header angereichert, um ein Tracing zu ermöglichen. Wenn ein PoPP
4747 Token im popp Header vorhanden ist, wird es geprüft.

- 4748 **A_26666-01 -PEP HTTP Proxy - TLS Terminierung**
4749 Die Komponente PEP HTTP Proxy MUSS TLS für eingehende Verbindungen terminieren
4750 können.[<=]
- 4751 **A_26195 -PEP HTTP Proxy - Unterstützung Websocket**
4752 Die Komponente HTTP Proxy MUSS das WebSocket-Protokoll unterstützen.[<=]
- 4753 **A_26641 -PEP HTTP Proxy - HTTP Protokoll-Versionen**
4754 Die Komponente HTTP Proxy MUSS das HTTP Protokoll in den Versionen HTTP/1.1, HTTP/2
4755 und SOLL HTTP/3 unterstützen.[<=]
- 4756 **A_25667 -PEP HTTP Proxy - Verifikation Access Token Binding**
4757 Die Komponente HTTP Proxy MUSS das Access Token Binding über den Mechanismus
4758 OAuth 2.0 Demonstrating Proof of Possession (DPoP) gemäß [RFC9449] verifizieren; d. h.
4759 der Claim "jkt" im Access Token MUSS eindeutig der Angabe im DPoP-Token
4760 entsprechen.[<=]
- 4761 **A_30003 -PEP HTTP Proxy, keine Mehrfachnutzung der DPoP Proof Token**
4762 Der HTTP Proxy MUSS anhand des claims jti im DPoP Proof durchsetzen, dass schon
4763 verwendete DPoP Proofs nicht noch einmal genutzt werden können.[<=]
- 4764 **A_25668-01 -PEP HTTP Proxy - Access Token Validierung**
4765 Die Komponente HTTP Proxy MUSS das übergebene Access Token validieren.
4766 Insbesondere MÜSSEN
- 4767 • die Signatur des Authorization Servers gültig,
 - 4768 • die Angaben zur zeitlichen Gültigkeit (Felder:iat, exp) valide,
 - 4769 • der Claim aud für das Resource Backend korrekt eingetragen, d.h. exakt
4770 übereinstimmend mit der für den aufgerufenen Request-Pfad konfigurierten logischen
4771 Audience, und
- 4773 • die Angabe scope passend zur Request url
- 4774 sein. Die Bindung des Tokens an die konkret aufgerufene URL erfolgt nicht über aud
4775 sondern über den DPoP-Claim htu (siehe A_29676).
- 4776 Der HTTP Proxy MUSS dabei sowohl Access Tokens mit ver : 1 (Legacy; aud kann ein
4777 verbatim übernommener Wert sein) als auch mit ver : 2 (von der Policy Engine
4778 vergebener logischer aud) verarbeiten.
- 4779
- 4780 Die Signatur des Access Token ist gültig, wenn sie mathematisch gültig ist und die
4781 Signatur vom Authorization Server im gleichen ZETA Guard erstellt wurde (default
4782 Einstellung) oder von einem Authorization Server, der in der Konfiguration des HTTP
4783 Proxy angegeben und im Entity Statement des Federation Master aufgeführt ist. Es MUSS
4784 möglich sein, mehrere Authorization Server in die Konfiguration des HTTP Proxy
4785 einzutragen.
- 4786 Wenn das Access Token ungültig ist, dann MUSS der Request mit HTTP Code 401
4787 Unauthorized beantwortet werden.[<=]
- 4788 *Hinweis: Jeder Authorization Server, der zur Föderation des Federation Masters gehört,*
4789 *veröffentlicht ein eigenes Entity Statement, in dem die Schlüssel enthalten sind, mit*
4790 *denen die Signatur der Access Token geprüft werden kann.*
4791 *Hinweis: Einige TI 2.0-Dienste benötigen ein PoPP Token. Diese werden im Request*
4792 *Header PoPP übertragen und vom HTTP Proxy geprüft.*
- 4793
- 4794 **A_26477-01 -PEP HTTP Proxy - PoPP Token Validierung**
4795 Die Komponente HTTP Proxy MUSS so konfiguriert werden können, dass pro Endpunkt des
4796 Resource Servers der Request Header PoPP verlangt und das PoPP Token validiert wird.

4797 Zusätzlich MUSS die Konfiguration pro Endpunkt unterstützen, dass die Dauer der
4798 Gültigkeit des PoPP Token in Sekunden seit Ausstellung und nach Ausstellungszeitpunkt
4799 und Prüfzeitpunkt innerhalb des gleichen Quartals eingestellt werden kann. Bei einem
4800 Prüfzeitpunkt innerhalb des gleichen Quartals MUSS eine grace-period von 5 Minuten
4801 beim Quartalswechsel akzeptiert werden.
4802 Wenn ein Request für einen Endpunkt des Resource Servers empfangen wird, der kein
4803 PoPP Header enthält und das Vorhandensein des PoPP Headers verlangt wird, dann MUSS
4804 der HTTP Proxy den Request mit HTTP Code 400 Bad Request beantworten.
4805 Insbesondere MUSS die Signatur des PoPP Servers gültig sein. Die Signatur des PoPP
4806 Token ist gültig, wenn sie mathematisch gültig ist und die Signatur vom PoPP Server
4807 erstellt wurde. Der HTTP Proxy MUSS ein vorhandenes und noch gültiges JWKS des PoPP
4808 Servers verwenden oder das JWKS des PoPP Servers herunterladen, um anhand der im
4809 JWKS enthaltenen Schlüssel die Signatur des PoPP Token zu prüfen.
4810 Der claimactorId des PoPP Token MUSS mit dem Attribut sub aus den zum Access Token
4811 gehörenden Nutzer-Daten übereinstimmen.
4812 Vor Ablauf der Gültigkeit MUSS der HTTP Proxy ein neues JWKS des PoPP Servers
4813 herunterladen.
4814 Wenn die Signatur des PoPP Token ungültig ist oder eine der anderen Prüfungen nicht
4815 erfolgreich war, dann MUSS der Request mit HTTP Code 403 Forbidden beantwortet
4816 werden.
4817 **[<=]**

4818 *Hinweis: Wenn ein ZETA/ASL-Kanal am HTTP Proxy terminiert wird, dann wird das PoPP*
4819 *Token durch den ZETA/ASL-Kanal geschützt transportiert.*

4820 **A_28525-02 -PEP HTTP Proxy - Step-up-Bedingung**

4821 Die Komponente HTTP Proxy MUSS dem Client Step-up-Bedarf mit HTTP-Statuscode 401
4822 (Unauthorized) und einem WWW-Authenticate-Header gemäß [RFC9470] signalisieren,
4823 wenn eine der folgenden, durch eine erneute (Step-up-)Authentisierung behebbaren
4824 Bedingungen vorliegt:

- 4825 • das im gültigen Access Token (Claim acr) nachgewiesene Authentifizierungsniveau
4826 erreicht nicht das für die aufgerufene Route konfigurierte Mindest-acr, oder
- 4827 • der im gültigen Access Token (Claim scope) enthaltene Scope deckt den für die
4828 aufgerufene Route konfigurierten erforderlichen Scope nicht ab, dieser ist aber über
4829 eine erneute (Step-up-)Authentisierung erlangbar.

4830 Der WWW-Authenticate-Header MUSS den zutreffenden Fehlercode enthalten:

- 4831 • bei unzureichendem Niveau error="insufficient_user_authentication" mit den
4832 Parametern acr_values (und ggf. max_age);
- 4833 • bei unzureichendem Scope error="insufficient_scope" mit dem Parameter scope.

4834 Sind beide Bedingungen erfüllt, MUSS error="insufficient_user_authentication"
4835 verwendet werden (das höhere Niveau schließt die Scope-Erlangung ein).

4836 Die angefragte URL MUSS existieren und einer konfigurierten Route (mit hinterlegter
4837 logischer Audience, erforderlichem acr und erforderlichem Scope) zugeordnet sein.

4838 **[<=]**

4839 *Hinweis: Ein Nicht-Übereinstimmen des aud-Claims oder des DPoP-htu-Claims ist KEIN*
4840 *Step-up-Fall und wird gemäß A_29676 mit 403 (Forbidden) abgewiesen.*

4841 **A_26493-01 -PEP HTTP Proxy - Umgang mit JWKS des Popp Servers**

4842 Die Komponente HTTP Proxy MUSS, falls der Header Parameter "kid" im PoPP Token JWT
4843 nicht zu einem Key im JWKS passt, ein neues JWKS des PoPP Servers herunterladen. **[<=]**

4844 **A_26480 -PEP HTTP Proxy - Umsetzen eines ZETA/ASL-Kanals**

4845 Die Komponente HTTP Proxy MUSS einen ZETA/ASL-Kanal (Server-Seite) umsetzen
 4846 können. Die Verwendung des ZETA/ASL-Kanals MUSS durch Konfiguration ein- und
 4847 ausschaltbar sein. In der Default-Einstellung ist der ZETA/ASL-Kanal ausgeschaltet. [\leq]

4848 *Hinweis: Ob ein ZETA/ASL Kanal zu verwenden ist, wird in der Spezifikation des TI 2.0-*
 4849 *Dienstes festgelegt. Die Anforderungen für den ZETA/ASL-Kanal sind in*
 4850 *[gemSpec_Krypt#8] zu finden.*

4851 *Wenn der ASL Kanal am HTTP Proxy terminiert, dann enthält der äußere Request kein*
 4852 *Access und kein DPoP Token. Der HTTP Proxy terminiert den Kanal und prüft das Access*
 4853 *und das DPoP Token im inneren Request.*

4854 *Wenn der Resource Server den ASL Kanal terminiert, dann enthält der äußere Request ein*
 4855 *Access Token und ein DPoP Token passend zum ASL Endpunkt am HTTP Proxy. Der HTTP*
 4856 *Proxy leitet den Request nach erfolgreicher Token-Prüfung an den Resource Server*
 4857 *weiter. Der Resource Server terminiert den ASL Kanal und findet im inneren Request ein*
 4858 *Access und ein DPoP Token passend zum Endpunkt des Resource Servers. Es wird*
 4859 *empfohlen, dass der Resource Server das Access und das DPoP Token prüft.*

4860
 4861 **A_26492-02 -PEP HTTP Proxy - Weiterleitung von Client-Daten**

4862 Die Komponente HTTP Proxy MUSS so konfiguriert werden können, dass pro Endpunkt des
 4863 Resource Servers die Weiterleitung der Client-Daten durch den HTTP Proxy ein- und
 4864 ausgeschaltet werden kann. Die default-Einstellung ist keine Weiterleitung der Client-
 4865 Daten.

4866 [\leq]

4868 **A_25669-01 -PEP HTTP Proxy - Zusätzliche HTTP-Header**

4869 Die Komponente HTTP Proxy MUSS die HTTP Requests mit allen HTTP-Headern an das
 4870 Resource Backend weiterleiten und dabei die folgenden zusätzlichen HTTP-Header
 4871 einsetzen.

4872 **Tabelle 14: PEP HTTP Proxy - Zusätzliche HTTP-Header**

HTTP-Header	Format	Schema	Größe (max. geschätzt)
zeta-user-info	Base64-URL kodierte JSON Struktur des User-Info Inhalts	[zeta-user-info.yaml]	250 Byte
zeta-popp-token-content	Base64-URL kodierte JSON Struktur des PoPP Token Inhalts. Der PoPP Header enthält das PoPP Token. Optional: Wird nur gesetzt, wenn im Request ein PoPP Header enthalten ist.	PoPP Token Payload	450 Byte
zeta-client-data	Base64-URL kodierte JSON Struktur der Client-Daten Optional: Wird nur gesetzt, wenn die Konfiguration des HTTP Proxy für die URL des Requests die Weiterleitung der Client-Daten festlegt.	[client-data.yaml]	250 Byte

4873 Gleichnamige HTTP-Header aus dem ursprünglichen HTTP-Request MÜSSEN entfernt bzw.
4874 überschrieben werden. [≤]
4875

4876 *Hinweis: Die Schema-Dateien sind in [GitHub ZETA Schemas] festgelegt.*

4877 **A_26560-01 -PEP HTTP Proxy - Weiterleitungskonfiguration**

4878 Der PEP HTTP Proxy MUSS eine Konfigurationsschnittstelle bereitstellen, über die ein
4879 statisches Mapping von externen Request-URLs (FQDNs und Request-Pfade) auf die
4880 internen Ziel-URLs der Resource Server konfiguriert werden kann.

4881 Der PEP HTTP Proxy MUSS es ermöglichen, für jede dieser statisch konfigurierten Routen
4882 zwingend eine erwartete logische Audience zu hinterlegen. Diese logische Audience
4883 entspricht dem aud-Wert, den die Policy Engine für die zugehörige Zielressource
4884 vergibt. Das Routing an die internen Resource Server DARF NICHT rein dynamisch auf
4885 Basis des ausgelesenen aud-Claims des Access Tokens ohne vordefinierte Pfad-
4886 Zuordnung erfolgen. [≤]

4887 **A_29675 -Anbieter TI 2.0 Dienst - Konfiguration der internen Endpunkte am PEP**

4888 Der Anbieter des TI 2.0 Dienstes MUSS das statische Routing des PEP HTTP Proxys so
4889 konfigurieren, dass externe Aufrufe der Clients ausschließlich auf die für den jeweiligen
4890 Dienst vorgesehenen, internen Resource Server Endpunkte des Anbieters geleitet werden
4891 (Whitelist-Prinzip).

4892 Der Anbieter MUSS dabei für jeden konfigurierten Pfad oder virtuellen Host die exakte
4893 logische Audience festlegen, deren Vorhandensein im aud-Claim des Access Tokens
4894 durch den PEP für den Zugriff erzwungen wird (Strict Audience Enforcement). Diese
4895 logische Audience MUSS mit dem Wert übereinstimmen, den die Policy Engine des ZETA
4896 Guard für die betreffende Zielressource im aud-Claim vergibt. [≤]

4897 **A_29676 -PEP HTTP Proxy - Sicherheitsprüfungen vor Weiterleitung**

4898 Zusätzlich zur Tokenvalidierung MUSS der PEP HTTP Proxy vor der Weiterleitung an einen
4899 internen Resource Server die folgenden Übereinstimmungen prüfen:

- 4900 • Strict Audience-Matching (nur wenn im PEP konfiguriert): Der aud-Claim des Access
4901 Tokens MUSS exakt mit der für den aufgerufenen Resource-Pfad konfigurierten
4902 logischen Audience übereinstimmen. Der PEP MUSS dabei Tokens mit ver: 1 und ver:
4903 2 unterstützen.
- 4904 • Strict acr-Matching (nur wenn im PEP konfiguriert): Der acr-Claim des Access Tokens
4905 MUSS mindestens dem geforderten Niveau gemäß LoA-Ordnung mit der für den
4906 aufgerufenen Resource-Pfad konfigurierten acr entsprechen.
- 4907 • DPoP Target URI-Abgleich: Die im DPoP-Proof enthaltene Target URI (Claim htu) MUSS
4908 exakt der extern aufgerufenen URL des PEP HTTP Proxys entsprechen.

4909 Sind diese Übereinstimmungen nicht gegeben, MUSS der PEP HTTP Proxy den Request
4910 mit dem HTTP-Fehlercode 403 Forbidden abweisen und DARF ihn NICHT an den internen
4911 Resource Server weiterleiten.

4912 [≤]

4913 **A_27265 -PEP HTTP Proxy - unveränderte Weiterleitung von Host Header und Request Zeile**

4914 Der PEP HTTP Proxy MUSS den Host Header und die Request Zeile unverändert an den
4915 Resource Server weiterleiten. [≤]
4916

4917 **A_26589-01 -PEP HTTP Proxy - Nutzer-Daten**

4918 Die Komponente HTTP Proxy MUSS für jeden Request mit gültigem Access Token die
4919 Nutzer-Daten aus dem Access Token als neuen HTTP Header zeta-user-info in den
4920 Request eintragen, bevor der Request an den Resource Server weitergeleitet wird.
4921 Folgende claims des Access Token gehören zu den Nutzer-Daten und werden gemäß
4922 Tabelle in den zeta-user-info Header übernommen.

4923 **Tabelle 15: zeta-user-info-Header**

Access Token claim	zeta-user-info Header claim
identifizier	identifizier
profession_oid	professionOID
common_name	commonName
organization_name	organizationName

4924 **[<=]**

4925 Beispiel für den zeta-user-info Header:

```
4926 {"commonName":"Arztpraxis Walter","identifizier":"1-
4927 234567890123","organizationName":"Arztpraxis
4928 Walter","professionOID":"1.2.276.0.76.4.50"}
```

4929 **A_26590-02 -PEP HTTP Proxy - Client-Daten**

4930 Wenn die Request-Weiterleitung mit Client-Daten konfiguriert wurde und der Request ein
 4931 gültiges Access Token hat, dann MUSS die Komponente HTTP Proxy die Client-Daten aus
 4932 dem Access Token als neuen HTTP Header zeta-client-data in den Request eintragen,
 4933 bevor der Request an den Resource Server weitergeleitet wird.

4934 Folgende claims des Access Token gehören zu den Client-Daten:

- 4935 • client_id
- 4936 • product_id
- 4937 • product_version

4938 **[<=]**4939 **A_26974-01 -PEP HTTP Proxy - Fehler vom Resource Server**

4940 Die Komponente HTTP Proxy MUSS die Response vom Resource Server als Fehler des
 4941 HTTP Proxy werten, wenn der Resource Server den Response Header zeta-cause: Proxy
 4942 gesetzt hat (der Resource Server hat einen Fehler im Request festgestellt und vermutet
 4943 die Ursache beim HTTP Proxy) und DARF diese Response nicht an den Client weiterleiten.
 4944 Der entsprechende Request des Clients MUSS in diesem Fall mit HTTP 500 beantwortet
 4945 werden. **[<=]**

4946 **A_27266 -PEP HTTP Proxy - Protected Resource Metadata Well-known**

4947 Die Komponente HTTP Proxy MUSS gemäß [RFC9728] und [opr-well-known.yaml] ein
 4948 Well-known JSON Dokument wie folgt bereitstellen, damit Clients die notwendigen
 4949 Informationen zur Interaktion mit dem Resource Server finden können.

4950 GET /.well-known/oauth-protected-resource

4951 Host: <FQDN des Resource Servers>

4952 Das Well-known JSON Dokument MUSS mit dem Schema [opr-well-known.yaml] validiert
 4953 werden können.

4954 **[<=]**

4955 *Hinweis: Es ist geplant, die Protected Resource Metadata Well-known JSON Dokumente in*
 4956 *einer folgenden ZETA Ausbaustufe durch den Federation Master bereitzustellen. Der*
 4957 *Federation Master signiert die Protected Resource Metadata Dokumente und stellt so*
 4958 *sicher, dass alle Services zur Föderation und zur gleichen Umgebung gehören (dev, prod,*
 4959 *ref, etc.). Dadurch verbessert sich für ZETA Clients die Authentizität der TI 2.0-Services.*

4960

4961 **A_28439 -PEP HTTP Proxy - Unterstützung Forwarded-Header**
4962 Die Komponente PEP HTTP Proxy MUSS in jeder empfangene HTTP-Anfrage den
4963 Forwarded-Header gemäß [RFC 7239] in der weitergeleiteten Anfragen aktualisieren. [<=]

4964 **A_29850 -PEP HTTP Proxy, Kennzeichnung von PEP-Fehlern im HTTP-Header**
4965 Tritt im PEP HTTP Proxy ein Fehler auf, der zu den HTTP-Statuscodes 401 oder 403 führt,
4966 MUSS der HTTP Proxy der HTTP-Antwort an den ZETA-Client den folgenden HTTP-Header
4967 hinzufügen:
4968 Header-Name: zeta-error-origin
4969 Header-Wert: pep [<=]

4970 **A_29851 -PEP HTTP Proxy, Ausschluss von Nicht-PEP-Fehlern**
4971 Fehler, die vom nachgelagerten Resource Server (Fachdienst) generiert und durch das
4972 PEP lediglich durchgereicht werden, sowie PEP-Fehler mit anderen HTTP-Statuscodes (wie
4973 z. B. 404), DÜRFEN den Header zeta-error-origin NICHT enthalten. [<=]

4974 **A_29860 -PEP HTTP Proxy, Erzeugung der Step-Up-Fehlermeldung**
4975 Stellt der PEP HTTP Proxy fest, dass für einen Request ein höheres
4976 Authentifizierungsniveau erforderlich ist, als das im gültigen Access Token des ZETA-
4977 Clients hinterlegte Niveau aufweist (Step-Up-Bedarf), MUSS er den Request abweisen und
4978 eine Fehlerantwort mit folgenden Parametern erzeugen:

- 4979 • HTTP-Statuscode: 401 Unauthorized
- 4980 • HTTP-Response-Header: WWW-Authenticate mit
4981 error=insufficient_user_authentication, acr_values, ggf. max_age (gemäß
4982 [RFC9470]); zeta-error-origin: pep
- 4983 • Response-Body (Content-Type: application/json): Ein JSON-Objekt entsprechend dem
4984 Schema [zeta-error.yaml] mit folgendem Inhalt:
 - 4985 • error: MUSS auf den Wert insufficient_user_authentication gesetzt werden.
 - 4986 • error_description: MUSS eine für Entwickler verständliche Beschreibung des
4987 Fehlers enthalten (z. B. eine Erklärung, dass das aktuelle Sicherheitsniveau
4988 unzureichend ist).
 - 4989 • error_uri: KANN eine URL enthalten, die zusätzliche Dokumentation zur
4990 Fehlerbehebung bereitstellt.

4991 [<=]

4992 5.7.2 Sicherheits- und Datenschutz-Anforderungen an den PEP

4993 **A_25445 -PEP - Zugriffsentscheidung nur über PDP**
4994 Der PEP MUSS sicherstellen, dass Zugriffe auf den Resource Server nur durch eine
4995 positive Zugriffsentscheidung vom PDP möglich sind. [<=]

4996 *Hinweis: Die positive Zugriffsentscheidung auf den Resource Server ist gegeben, wenn der*
4997 *Client im Request Authorization Header ein gültiges Access Token mit passendem scope -*
4998 *ausgestellt von einem Authorization Server, zu dem eine Vertrauensbeziehung besteht -*
4999 *vorweisen kann. Durch das gültige Access Token ist sichergestellt, dass der Zugriff von*
5000 *dem PDP freigegeben wird.*

5001 5.8 Policy Decision Point

5002 Der Policy Decision Point (PDP) setzt sich aus den Komponenten

- 5003 • Authorization Server

- 5004 • Policy Engine und
- 5005 • PDP Datenbank
- 5006 zusammen.

5007 **5.8.1 Policy Engine**

5008 Der PDP implementiert die Policy Engine als [Open Policy Agent] (OPA). Die Policies und
5009 die zugehörigen Daten erhält die Policy Engine per OCI Protokoll von der ZETA Artifact
5010 Registry. Aus den Input-Daten vom Authorization Server, den Daten vom PIP und den
5011 Policies vom PAP ermittelt die Policy Engine eine Entscheidung und gibt diese zurück an
5012 den Authorization Server.

5013 Neben der OPA Instanz, die die Entscheidung für den Authorization Server trifft (aktive
5014 Instanz), ob eine Kommunikation zulässig ist, implementiert die Policy Engine noch eine
5015 zweite OPA Instanz, die mit einem zweiten OPA Bundle vom PIP und PAP Service arbeitet,
5016 aber die getroffenen Entscheidungen nicht an den Authorization Server zurückgibt. Diese
5017 Instanz wird Simulations-Instanz genannt und dient dazu in produktiven Umgebungen die
5018 weiterentwickelte Policies und Daten zu evaluieren.

5019
5020

5021 **A_25739-02 -PDP Policy Engine, OPA Instanzen**

5022 Der PDP MUSS als Policy Engine zwei Open Policy Agent (OPA) Instanzen bereitstellen,
5023 wobei eine Instanz die Entscheidung für den Authorization Server trifft (aktive Instanz),
5024 und eine Instanz eine Entscheidung trifft, diese aber nicht an den Authorization Server
5025 sendet (Simulations-Instanz).

5026 Die OPA Instanzen MÜSSEN so konfiguriert sein, dass nur signierte OPA Bundles mit
5027 gematik Signatur akzeptiert werden. Das Polling Intervall zur Aktualisierung des OPA
5028 Bundles über die lokale Artifact Registry MUSS 60 Sekunden sein. [≤]

5029 **A_27401 -PDP Policy Engine - Decision Eigenschaften**

5030 Die PDP Policy Engine MUSS Decision-Anfragen mit einem JSON Objekt nach dem Schema
5031 [pdp-decision.yaml] beantworten. [≤]

5032 **A_25490-03 -PDP - Sicherheitsmeldung bei Änderungen und Aktualisierung**

5033 Der PDP MUSS sicherstellen, dass bei Aktualisierung und Änderungen der Policies oder
5034 PIP-Daten eine Sicherheitsmeldung inklusive der OPA Bundle revision an das Security
5035 Monitoring automatisiert übermittelt wird. [≤]

5036 **A_29699 -PDP Policy Engine, Verwendung der korrekten Policies und Daten**

5037 Der Anbieter des TI 2.0 Dienstes MUSS die PDP Policy Engine so konfigurieren, dass das
5038 korrekte, dem TI2-0 Dienst zugewiesene OPA Bundle Image aus der ZETA Artifact
5039 Registry geladen wird. [≤]

5040 *Hinweis: Die Policy Engine muss das OPA Bundle Image nicht direkt aus der ZETA Artifact*
5041 *Registry laden. Ein lokaler Artifact Registry Cache kann verwendet werden.*

5042 **5.8.2 PDP Authorization Server**

5043 **A_25760 -PDP Authorization Server - OAuth2 Schnittstellen**

5044 Der PDP Authorization Server MUSS eine OAuth2 Schnittstelle gemäß [RFC6749] und
5045 [RFC7636] implementieren.

5046 Der Authorization Server MUSS am Token Endpunkt REFRESH_TOKEN entsprechend
5047 [RFC6749] ausstellen können. [≤]

5048 **A_26669-01 -PDP Authorization Server - TLS Terminierung**

5049 Die Komponente PDP Authorization Server MUSS eingehende TLS Verbindungen von
5050 außerhalb des ZETA Guards terminieren können. [≤]

5051 **A_25659 -PDP Authorization Server - Check Client-Registrierung**

5052 Der PDP Authorization Server MUSS die Client Instanzen über den Mechanismus JSON
5053 Web Token Client Authentication gemäß [RFC7523] mit DPoP gemäß [RFC9449]
5054 authentifizieren und Anfragen, die den Mechanismus nicht verwenden, ablehnen. [≤]

5055 **A_25660 -PDP Authorization Server - Session Management mittels Access Token 5056 und Refresh Token**

5057 Die Komponente Authorization Server MUSS ein Session Management mittels OAuth2 und
5058 Ausgabe, Verwaltung und Entzug von Access und Refresh Token gemäß [RFC6749#1.5]
5059 unterstützen. [≤]

5060 **A_29854 -PDP Authorization Server - Aktive Session-Termination**

5061 Die Komponente Authorization Server MUSS eine aktive Session-Termination anhand
5062 einer Session-Id unterstützen.

5063 Für die Session-Termination MÜSSEN die Eingaben Trace-Id, Session-Id, Reason-Code und
5064 Trigger-Source verarbeitet werden. [≤]

5065 **A_29855 -PDP Authorization Server - Entzug der Refresh Token nach Session- 5066 Termination**

5067 Die Komponente Authorization Server MUSS bei erfolgreicher Session-Termination alle
5068 der Session zugeordneten Refresh Token unverzüglich ungültig machen. Die
5069 Ungültigmachung MUSS auch rotierte Refresh Token derselben Session umfassen. [≤]

5070 **A_29856 -PDP Authorization Server - Verhalten bei Token Requests nach 5071 Session-Termination**

5072 Die Komponente Authorization Server MUSS vor der Ausstellung eines neuen Token-Sets
5073 bei Token Requests mit grant_type=refresh_token prüfen, ob die referenzierte Session
5074 terminiert wurde. Ist die Session terminiert, MUSS der Request mit einem
5075 standardisierten Fehler gemäß Kapitel 5.4.6 abgelehnt werden. [≤]

5076 **A_29857 -PDP Authorization Server - Protokollierung der Session-Termination**

5077 Die Komponente Authorization Server MUSS jede Session-Termination revisionssicher
5078 protokollieren.

5079 Das Protokoll MUSS mindestens Trace-Id, Session-Id, Trigger-Source, Reason-Code und
5080 Zeitpunkt enthalten. [≤]

5081 **A_26944 -PDP Authorization Server - Access Token Inhalt**

5082 Die Komponente Authorization Server MUSS Access Token mit Attributen gemäß [access-
5083 token.yaml] ausstellen. [≤]

5084 **A_25661-01 -PDP Authorization Server - Umsetzung der Policy Decision**

5085 Die Komponente Authorization Server MUSS die positive Zugriffs-Entscheidung eines PDP
5086 mittels Ausstellung eines Access- und Refresh-Token (200 OK) umsetzen. Fehler- und
5087 Ablehnungsfälle MUSS der Authorization Server gemäß [RFC6749]#section-5.2 (für Token
5088 Exchange [RFC8693] und Refresh) mit dem zutreffenden HTTP-Statuscode und error-Code
5089 beantworten, wobei der Response-Body dem Schema [zeta-error.yaml] entspricht:

- 5090 • 400 bei fehlerhaftem Request oder ungültigem/abgelaufenem/wiederverwendetem
5091 Grant (invalid_request, invalid_grant, invalid_scope, invalid_target) sowie DPoP-
5092 Fehlern (invalid_dpop_proof, use_dpop_nonce gemäß [RFC9449]);
- 5093 • 401 bei fehlgeschlagener Client-Authentifizierung (invalid_client) sowie bei Step-up-
5094 Bedarf (insufficient_user_authentication gemäß [RFC9470], siehe A_28525-02);
- 5095 • 403 ausschließlich bei einer negativen Policy-Entscheidung trotz technisch gültiger
5096 Credentials (access_denied) sowie bei session_terminated / refresh_token_revoked.

5097 Die Claims im Access Token MUSS zur Entscheidung der Policy Engine passen. [≤]

5098 *Hinweis: RFC 6749 §5.2 definiert für den Token-Endpunkt keinen Fehlercode für eine*
 5099 *Policy-basierte Ablehnung bei gültigen Credentials; ZETA verwendet hierfür bewusst 403*
 5100 *access_denied, um technische Grant-Fehler (400/401) von der autorisierungsbezogenen*
 5101 *Ablehnung zu unterscheiden.*

5102 **A_28837 -PDP Authorization Server, Policy Prüfung bei Ausstellung Token-Set**
 5103 Die Komponente Authorization Server MUSS vor der Ausstellung des neuen Token-Sets
 5104 (Access Token und Refresh Token) eine Autorisierungsanfrage an die Policy Engine
 5105 stellen.[<=]

5106 **A_28527-01 -PDP Authorization Server - Gültigkeitsdauer Access und Refresh**
 5107 **Token**
 5108 Die Komponente Authorization Server MUSS beim Ausstellen von Access und Refresh
 5109 Token die Gültigkeitsdauer aus der Policy Decision übernehmen.
 5110 Dabei MUSS eine feste Höchstgrenze der Gültigkeit vom Authorization Server eingehalten
 5111 werden.

- 5112 • Access Token: maximal 3600 Sekunden
- 5113 • Refresh Token (Versicherte, mobile Clients): maximal 1 Tag

5114 [<=]

5115 **A_25662 -PDP Authorization Server - Refresh Token Rotation**
 5116 Die Komponente Authorization Server MUSS eine Refresh Token Rotation gemäß
 5117 [RFC6749#10.4] erzwingen und MUSS sicherstellen, dass ein Refresh Token nur einmal
 5118 gegen ein Access Token und ein Refresh Token getauscht werden kann.
 5119 Die Komponente Authorization Server MUSS erzwingen, dass der Nutzer eine
 5120 Authentisierung durchführen muss, wenn seit der letzten Authentisierung die Zeit der
 5121 Gültigkeitsdauer des Refresh Token abgelaufen ist.[<=]

5122 **A_25663 -PDP Authorization Server - Token-Binding an Client-Registrierung**
 5123 Die Komponente Authorization Server MUSS auszugebende Access Token und Refresh
 5124 Token über den Mechanismus OAuth 2.0 Demonstrating Proof of Possession (DPoP)
 5125 gemäß [RFC9449] an die registrierte Client-Instanz binden, indem im Token-Binding-
 5126 Claim die Angabe der Clientidentifikation als "jkt" eindeutig referenziert wird.[<=]

5127 **A_25665-01 -PDP Authorization Server - Plugin-Schnittstelle Application**
 5128 **Authorization Backend**
 5129 Die Komponente Authorization Server MUSS eine Plug-In Schnittstelle (per Webhook) zu
 5130 einem anwendungsspezifischen Authorization Backend implementieren und dabei die
 5131 folgenden Signale und Informationen aus der erhaltenen Zugriffsanfrage weiterreichen.

5132 **Tabelle 16: PDP Authorization Server - Plugin-Schnittstelle Application Authorization**
 5133 **Backend**

Operation	Operation Kennung	Input	Output
Benachrichtigung über die Ablehnung des Zugriffs durch PDP	notifyAccessDenied	Trace-Id Subject-Information Client-Information PDP-Decision	-
Anwendungsspezifische Autorisierung	authorizeAccess	Trace-Id Session-Id Authorization-Scopes Authorization-	Zugriff erlauben Ja/Nein Zusätzliche Authorization Scopes

		Details Subject- Information Client- Information PDP-Decision	Zusätzliche Authorization Details Zusätzliche Claims
Benachrichtigung über abgelaufene oder terminierte Sessions	notifySessionTermination	Trace-Id Session-Id	-
Aktive Beendigung einer Session durch Anwendung	terminateSession	Trace-Id, Session- Id, Reason-Code, Trigger-Source	Session- Termination akzeptiert Ja/Nein, Fehlercode

5134 [**<=**]

5135 **A_29858 -PDP Authorization Server - Idempotenz der Session-Termination**

5136 Die Operation terminateSession MUSS idempotent ausgeführt werden. Wiederholte
5137 Aufrufe mit derselben Session-Id DÜRFEN keinen fachlichen Fehler verursachen.

5138 [**<=**]

5139 **A_26586-01 -PDP Authorization Server - Session-Daten**

5140 Die Komponente Authorization Server MUSS nach jeder vollständigen und erfolgreichen
5141 Authentifizierung Session-Daten in der Datenbank des PDP speichern.

5142 Zu den Session Daten gehören die Client-, User, und Request-Daten gemäß [policy-
5143 engine-input.yaml].

5144 Die Session-Daten MÜSSEN bei Anfragen des Authorization-Servers an die Policy Engine
5145 im Request mit übergeben werden. [**<=**]

5146 **A_26972-03 -PDP Authorization Server - Nutzer-Daten aus der SM(C)-B**

5147 Die Komponente Authorization Server MUSS im Falle der SM(C)-B Authentifizierung die
5148 folgenden Daten aus dem SM(C)-B Zertifikat auslesen und als Nutzer-Daten gemäß [zeta-
5149 user-info.yaml] in der PDP Datenbank speichern:

5150 **Tabelle 17: SM(C)-B_Nutzer-Daten**

SM(C)-B Daten (C.HCI.AUT gemäß [gemSpec_PKI])	Nutzer-Daten gemäß [zeta- user-info.yaml]	Beschreibung
Extension Admission,registrationNumber	identifier	Telematik-ID Eindeutige ID der Organisation des Gesundheitswesens
subject,commonName	common_name	Wird aus dem Zertifikat übernommen. „Kurzname“ der Institution, so wie sie sich auf dem Anschriftenfeld findet.
Extension Admission,professionOID	profession_oid	OID der Institution gemäß [gemSpec_OID#GS-A_4443-*]
subject,organizationName	organization_nam	Wird übernommen, wenn im

	e	Zertifikat vorhanden. Name der Organisation/Einrichtung des Gesundheitswesens
--	---	---

5151 [**<=**]

5152 **A_26973-02 -PDP Authorization Server - Nutzer-Daten aus id_token**

5153 Die Komponente Authorization Server MUSS im Falle der OIDC Authentifizierung für
 5154 Versicherte alle Claims aus dem id_token gemäß[gemSpec_IDP_Sek] auslesen und als
 5155 Nutzer-Daten gemäß [zeta-user-info.yaml] in der PDP Datenbank speichern.
 5156 Dabei gilt das folgende Mapping für die Pflicht-Nutzer-Daten.

5157 **Tabelle 18: id_token_Nutzer-Daten**

id_token claims (gemäß [gemSpec_IDP_Sek])	Nutzer-Daten gemäß [zeta-user-info.yaml]	Beschreibung
urn:telematik:claims:id	identifizier	für Versicherte der unveränderliche Anteil der KVNR
urn:telematik:claims:profession	profession_oid	OID für Versicherte
urn:telematik:claims:organization	organization_name	ID oder Name der attributsbestätigenden Stelle (IK-Nummer der Kasse)
urn:telematik:claims:id	common_name	für Versicherte der unveränderliche Anteil der KVNR

5158 [**<=**]

5159 **A_28144 -PDP Authorisation Server - Länge der Nonce**

5160 Der ZETA Guard MUSS am Endpunkt GET /nonce einen 128 Bit langen base64url
 5161 kodierten Zufallswert ausgeben. [**<=**]

5162 **A_28440 -PDP Authorization Server - Auswertung Forwarded-Header**

5163 Die Komponenten PDP Authorization Server MUSS HTTP-Anfragen mit einem vorhandenen
 5164 Forwarded-Header auswerten, um die Client-IP Adresse zu ermitteln. Bei der Auswertung
 5165 ist die Semantik gemäß [RFC 7239] und die Reihenfolge der Parameter zu beachten. [**<=**]

5166 **A_25644-01 -PDP Client-Registrierung mit Attestation**

5167 Die Komponente Authorization Server MUSS die Clients bei der Registrierung über
 5168 folgende Mechanismen attestieren:

- 5169 • Android Key and ID Attestation (für Google-Android Clients)
- 5170 • Apple DCAppAttest
- 5171 • TPM Attestation für Windows und Linux Clients
- 5172 • Software Attestation

5173 [**<=**]

5174 **A_25645-01 -PDP Authentifizierung mittels TI-Smartcard**

5175 Die Komponente Authorization Server MUSS die Authentifizierung per Token Exchange
 5176 eines Subject Token gemäß [subject-token-smb.yaml] mit Signatur einer SM(C)-B
 5177 unterstützen. [**<=**]

5178 **A_25649 -PDP Authorization Server - Regelmäßige Wiederholung der**
5179 **Attestation**

5180 Die Komponente Authorization Server SOLL die Client-Attestierung für jede neue Session
5181 verlangen. [≤]

5182 *Hinweis: Eine Session des Authorization Servers ist gültig vom Zeitpunkt $t_1 =$*
5183 *(Ausstellungszeitpunkt des ersten Refresh Token nach vollständiger Authentifizierung) bis*
5184 *zum Zeitpunkt $t_2 = t_1 + (\text{Gültigkeitsdauer des ersten Refresh Token})$. Bei mobilen Clients*
5185 *werden für die Attestation Services des mobilen Betriebssystems verwendet, die ein Rate*
5186 *Limit haben können. In diesem Fall kann die Attestation eventuell nicht für jede neue*
5187 *Session wiederholt werden oder es werden bei Folge-Attestations nur lokale Verfahren*
5188 *angewendet.*

5189 **A_25650 -PDP Client-Registrierung - TI-Identität in Attestation**

5190 Die Komponente Authorization Server MUSS den registrierten Client, wenn möglich, an
5191 eine TI-Identität (mit KVNR oder TelematikID binden. [≤]

5192 *Hinweis: In Kapitel 5.3.3- Authentifizierung ohne Nutzer-Identität ist eine Autorisierung*
5193 *vorgesehen, bei der keine Nutzer-Identität durch ZETA Guard festgestellt wird. In diesem*
5194 *Fall erfolgt eine Client-Registrierung ohne Bindung an eine TI-Identität.*

5195 **A_25651 -PDP Client-Registrierung - Offband Nutzer Verification**

5196 Die Komponente Authorization Server MUSS einen Offband Prozess (per E-Mail) für die
5197 Kommunikation mit diesem Nutzer unterstützen (Trust on First Use), wobei der Nutzer
5198 seine E-Mail Adresse eigenverantwortlich vergibt. [≤]

5199 *Hinweis: TOFU wird nur für mobile Clients genutzt.*

5200 **A_25752-01 -PDP Policy Engine - Client über Hintergrund der Ablehnung**
5201 **informieren**

5202 Falls ein Client beim Token Request nicht die geforderten Parameter der Policy
5203 unterstützt bzw. das geforderte Niveau nicht erreicht, MUSS der Authorization Server die
5204 von der Policy Engine in der Decision-Response aufgeführten Gründe (Attribut reasons) in
5205 der Response an den Client übergeben. [≤]

5206 **A_25738 -PDP Client-Registrierung - Telemetrie**

5207 Die Komponente Authorization Server MUSS in den Telemetriedaten zu jeder versuchten
5208 Client-Registrierung folgende Parameter ohne einen Nutzerbezug protokollieren:

- 5209 • Clientparameter (Betriebssystem(-version), Patchlevel, Geolocation etc.) gemäß
- 5210 Clientattestierung
- 5211 • verwendeter Faktor für Offband-Verifikation (E-Mail)
- 5212 • Zeitstempel Registrierung, Zeitpunkt Offband-Bestätigung
- 5213 • verwendeter Faktor der Nutzerauthentifizierung (SmartCard, Digitale Identität)
- 5214 • Status/Ergebnis des Registrierungsversuchs

5215 [≤]

5216 **A_26585-02 -PDP Client-Registrierung - Client-Daten**

5217 Die Komponente Authorization Server MUSS die Client-Daten gemäß [policy-engine-client-
5218 data.yaml] in der PDP Datenbank verwalten.

5219 Die Client-Daten MÜSSEN bei Anfragen des Authorization-Servers an die Policy Engine im
5220 Request mit übergeben werden. [≤]

5221 **5.8.2.1 PDP Relying Party**

5222 **A_25655 -PDP - Relying Party**

5223 Der PDP Authorization Server MUSS in der TI-Föderation als Relying Party registriert sein.
5224 [≤]

5225 **A_25656 -PDP - Entity Statement**
 5226 Der PDP Authorization Server MUSS die Redirect-URLs aller zulässigen Clients als erlaubte
 5227 Redirect-URLs im Entity Statement ausweisen.[<=]

5228 **A_25657 -PDP - Authentication über sektoralen IDP**
 5229 Der PDP Authorization Server MUSS die Nutzer über sektorale IDPs authentifizieren
 5230 können.[<=]

5231 **5.8.2.2 Claims amr und acr bei Token Request**

5232 **A_29820 -Authorization Server, Attribute amr und acr bei Token Exchange mit**
 5233 **SMC-B signiertem Subject-Token**
 5234 Der PDP Authorization Server MUSS beim Token Exchange mit SMC-B signiertem Subject-
 5235 Token im Request an die Policy Engine gemäß [policy-engine-input.yaml] die Attributeamr
 5236 und acr wie folgt verwenden:
 5237 amr: urn:telematik:auth:sc
 5238 acr: gematik-ehealth-loa-substantial
 5239 [<=]

5240 **A_29994 -Authorization Server, Attribute amr und acr bei OIDC**
 5241 **Authentifizierung mit Sek IDP**
 5242 Der Authorization Server MUSS bei derOIDC Authentifizierung mit Sek IDP
 5243 die Attributeamr und acr aus dem ID Token übernehmen und im Request an die Policy
 5244 Engine gemäß [policy-engine-input.yaml] verwenden.[<=]

5245 **A_30004 -Authorization Server, Attribute amr und acr bei Authentifizierung**
 5246 **ohne Nutzer-Identität**
 5247 Der Authorization Server MUSS bei derAuthentifizierung ohne Nutzer-Identität das
 5248 Attributacr wie folgt setzen und im Request an die Policy Engine gemäß [policy-engine-
 5249 input.yaml] verwenden.
 5250 acr: gematik-ehealth-loa-low
 5251 Das Attribut amr wird nicht verwendet.[<=]

5252 **A_29993 -Authorization Server, Attribut acr im Access Token**
 5253 Der PDP Authorization Server MUSS das nachgewiesene Authentifizierungsniveau als
 5254 Claim acr (Wertebereich gemäß gematik-LoA) in das Access Token übernehmen, damit
 5255 der PEP die Step-up-Bedingung auswerten kann.[<=]

5256 *Hinweis: Die bei ZETA zulässigengematik-LoA Werte sindgematik-ehealth-loa-high und*
 5257 *gematik-ehealth-loa-substantial sowie gematik-ehealth-loa-low.*

5259

5260 **5.8.2.3 Token-Ausstellung**

5261 Wenn der ZETA/ASL Kanal am Resource Server terminiert, dann benötigt der PEP HTTP
 5262 Proxy im äußerenPOST /ASL Request ein passendes Access Token und ein DPoP Token.
 5263 Die Ausstellung dieses zusätzlichen Access Tokens folgt dem versionierten Token-
 5264 Ausstellungsverfahren (Contract v1/v2, siehe 5.4.2.5): Deraud-Claim benennt die
 5265 Zielressource über ihren logischen Bezeichner, während die Endpunkt-URL ausschließlich
 5266 über den DPoP-Claimhtu gebunden wird.

5267 Tabelle 19 Tab-ZETA-Token-Ausstellung-ASL-am-Resource-Server

Token	claim	Wert
Access Token	aud	<logischer Bezeichner des ZETA/ASL Endpunkts> (von der Policy Engine vergeben, siehe 5.4.2.4)

	scope	asl
	ver	2 (Contract v2) bzw. 1 (Contract v1, Legacy)
DPoP Token	htm	POST
	htu	<a href="https://<FQDN>/ASL">https://<FQDN>/ASL

5268 *Hinweis: Der <FQDN> entspricht dem <FQDN> aus dem Attribut resource des OAuth*
 5269 *Protected Resource Well-known JSON Dokuments. Diese Endpunkt-*
 5270 *URL <https://<FQDN>/ASL> wird nicht in den aud-Claim übernommen, sondern*
 5271 *ausschließlich über den DPoP-Claim htu an den konkreten Request gebunden*
 5272 *([RFC9449]). Der aud-Claim des Access Tokens enthält den logischen Bezeichner des*
 5273 *ZETA/ASL Endpunkts, den die Policy Engine aus dem angegebenen resource-Wert ableitet*
 5274 *(Contract v2, ver: 2); bzw bei einem Legacy-Request (ver: 1) wird der Wert verbatim aus*
 5275 *audience übernommen.*

5276

5277 **A_29861 -PDP Authorization Server, Token-Ausstellung bei ZETA/ASL**

5278 **Terminierung am Resource Server**

5279 Der Authorization Server MUSS, wenn:

5280 • die Policy Engine die Token-Ausstellung erlaubt hat und

5281 • der ZETA/ASL Kanal am Resource Server terminiert,

5282 ein zusätzliches Access Token für den Zugriff auf den /ASL Endpunkt ausstellen
 5283 gemäß Tab-ZETA-Token-Ausstellung-ASL-am-Resource-Server.

5284 Das zusätzliche Access Token wird in der Response auf den Token Request im Parameter
 5285 asl_access_token gesendet.【<=】

5286 **A_29951 -PDP Authorization Server, Token Ausstellung**

5287 Der Authorization Server MUSS die Token Response nach dem Schema [token-
 5288 response.yaml] erstellen.【<=】

5289 **A_29953 -PDP Authorization Server, Abfrage der Policy Engine Decision vor der**

5290 **Token Ausstellung**

5291 Der Authorization Server MUSS vor der Token Ausstellung die Decision der Policy Engine
 5292 abfragen und den Request Body gemäß [policy-engine-input.yaml] erstellen.

5293 Wenn die Response der Policy Engine gemäß [pdp-decision.yaml] allow: true enthält,
 5294 dann MUSS der Authorization Server die Token mit den Gültigkeitsdauern aus der
 5295 Response erstellen.

5296 Wenn die Response allow: false enthält, dann MUSS der Authorization Server in der
 5297 Token Response gemäß [token-response.yaml] die reasons aus der Response der Policy
 5298 Engine übernehmen.

5299 【<=】

5300

5301 **5.8.2.4 Token Revocation (RFC 7009)**

5302 **A_29996 -PDP Authorization Server, Revocation-Endpunkt**

5303 Der Authorization Server MUSS einen Endpunkt POST /revoke gemäß [RFC7009]

5304 bereitstellen, über den ein Client seine eigenen Refresh Token widerrufen kann. Der

5305 Request MUSS den Parameter token enthalten und KANN den Parameter token_type_hint
 5306 (refresh_token) enthalten.

5307 Die URL des Revocation-Endpunkts MUSS im [as-well-known.yaml] im Attribut

5308 revocation_endpoint angegeben sein.【<=】

5309 **A_29997 -PDP Authorization Server - Client-Authentifizierung am Revocation-**
5310 **Endpunkt**

5311 Der Authorization Server MUSS Requests an POST /revoke in derselben Weise wie am
5312 Token-Endpunkt authentifizieren (private_key_jwt Client Assertion [RFC7523] mit DPoP
5313 [RFC9449]) und MUSS sicherstellen, dass ein Client ausschließlich eigene Token
5314 widerrufen kann. Schlägt die Client-Authentifizierung fehl, MUSS der Endpunkt mit 401
5315 (Unauthorized) und error=invalid_client gemäß [zeta-error.yaml] antworten. [`<=`]

5316 **A_29998 -PDP Authorization Server - Wirkung des Widerrufs**

5317 Bei erfolgreicher Client-Authentifizierung MUSS der Authorization Server das im
5318 Parameter token übergebene Refresh Token sowie alle der zugehörigen Session
5319 zugeordneten - auch rotierten - Refresh Token unverzüglich ungültig machen (analog
5320 A_29855) und die Session terminieren. Nachfolgende Refresh-Requests MÜSSEN mit
5321 refresh_token_revoked abgelehnt werden. Der Widerruf MUSS gemäß A_29857
5322 revisionsicher protokolliert werden. [`<=`]

5323 **A_29999 -PDP Authorization Server - Antwortverhalten ohne Token-**
5324 **Enumeration**

5325 Der Authorization Server MUSS auf einen erfolgreich client-authentifizierten Revocation-
5326 Request unabhängig davon, ob das übergebene Token gültig, bereits ungültig oder
5327 unbekannt ist, mit 200 OK antworten (RFC 7009 §2.2), um Rückschlüsse auf die Existenz
5328 von Token zu verhindern. [`<=`]

5329 **A_30000 -PDP Authorization Server - Access Token außerhalb des**
5330 **Geltungsbereichs**

5331 Wird ein Access Token übergeben, MUSS der Authorization Server mit 200 OK ohne
5332 serverseitige Wirkung antworten und KANN alternativ error=unsupported_token_type
5333 (400) gemäß RFC 7009 §2.2.1 zurückgeben. [`<=`]

5334 *Hinweis: Access Token sind kurzlebig, DPoP-gebunden und werden vom Resource Server*
5335 *zustandslos geprüft; ihr Widerruf ist nicht Zweck dieses Endpunkts.*

5336 **5.8.3 PDP Datenbank**

5337 Die Datenbank speichert Session-, Nutzer- und Client-Daten. Die Struktur und Verwaltung
5338 der Daten wird vom Authorization Server bestimmt.

5339 **A_26587-01 -PDP Datenbank - Kompatibilität**

5340 Die Komponente PDP Datenbank MUSS kompatibel zum Authorization Server sein. [`<=`]

5341 *Hinweis: Die vom ZETA Guard Authorization Server Keycloak unterstützten Datenbanken*
5342 *werden im ZETA Guard Produkthandbuch aufgelistet (siehe auch*
5343 <https://www.keycloak.org/server/db>*).*

5344 **5.8.4 Sicherheits- und Datenschutzerfordernungen an den PDP**

5345 **A_28832 -PDP Policy Engine - Integritätsprüfung der Bundle-Signaturen**

5346 Die PDP Policy Engine MUSS sicherstellen, dass beim Import des Bundles mit den Policies
5347 und Daten die folgenden Signaturen erfolgreich validiert werden, bevor Bundle in die
5348 Policy Engine übernommen wird:

- 5349
- Autor-Signatur
 - Freigabe-Signatur
- 5350

5351 [`<=`]

5352 **A_25449 -PDP- Nutzeridentität nur von einem zugelassenem IDP**

5353 Der PDP MUSS sicherstellen, dass nur Nutzeridentitäten von einem zugelassenen IDP
5354 akzeptiert werden. [`<=`]

5355 A_26281-01 -PDP Authorization Server - Schlüssel für Token-Signatur

5356 Der PDP Authorization Server MUSS für die Signatur von Token asymmetrische
5357 Schlüsselpaare (PrK.AuthS.Sig, PuK.AuthS.Sig) verwenden. Die Gültigkeitsdauer
5358 (Rotationsintervall) dieser Schlüssel MUSS in den ZETA Guard-Manifest-Dateien
5359 konfigurierbar sein.

5360 [**<=**]

5361 A_25748-02 -PDP Client-Registrierung - Maximale Anzahl von Clients

5362 Die Komponente Authorization Server MUSS sicherstellen, dass ein Nutzer maximal
5363 256 Clients registrieren kann (konfigurierbares Limit).

5364 Um eine Überlastung durch fehlerhafte Clients zu verhindern, MUSS der Authorization
5365 Server das Limit für die maximale Anzahl an Client-Registrierungen pro User durchsetzen.
5366 Wird dieses Limit erreicht, MUSS der Authorization Server bei der Anlage einer neuen
5367 Registrierung automatisch die am längsten inaktive Client-Registrierung dieses Users
5368 löschen, um Platz für die neue Registrierung zu schaffen. [**<=**]

5369 A_28808 -PDP Authorization Server - Löschfristen

5370 Der Authorization Server (PDP) MUSS konfigurierbare Inaktivitäts-Löschfristen für User-
5371 und Client-Daten durchsetzen und diese Daten nach Ablauf der Fristen löschen (Default-
5372 Wert: 1 Jahr).

5373 Der Authorization Server MUSS Session-Daten automatisch löschen, wenn die Session
5374 expired ist. [**<=**]

5375 A_25749 -PDP Client-Registrierung - Nutzer Protokollierung

5376 Die Komponente Authorization Server MUSS ein Nutzerprotokoll führen und die folgenden
5377 Anwendungsfälle für den Nutzer protokollieren:

- 5378 • Client hinzufügen
- 5379 • Client löschen
- 5380 • Client umbenennen
- 5381 • E-Mail-Adresse hinzufügen
- 5382 • E-Mail-Adresse aktualisieren

5383 [**<=**]

**5384 A_25750 -PDP Client-Registrierung - Nutzer über sicherheitsrelevante
5385 Ereignisse informieren**

5386 Die Komponente Authorization Server MUSS sicherstellen, dass der Nutzer bei folgenden
5387 Anwendungsfällen informiert wird:

- 5388 • Client hinzufügen
- 5389 • Client löschen
- 5390 • Client umbenennen
- 5391 • E-Mail-Adresse aktualisieren

5392 [**<=**]

5393 *Hinweis: Der Nutzer kann z.B. durch eine geeignete E-Mail oder App-Notifikation über die*
5394 *sicherheitsrelevanten Ereignisse informiert werden.*

5395 5.9 Policies und Daten

5396 Die Policies und Daten werden in der ZETA Artifact Registry als OPA Bundles im OCI
5397 Format für die PDP Policy Engines der ZETA Guard Instanzen bereitgestellt. Die Bundles
5398 werden von der gematik in einem CI Prozess mit Quality Gates entwickelt und in die ZETA

5399 Artifact Registry deployed. Für jeden Fachdienst-Typ werden spezifische OPA Bundles
5400 erstellt. Falls es erforderlich ist, können auch pro ZETA Guard Instanz spezifische OPA
5401 Bundles erstellt werden.

5402 Es gibt umgebungsspezifische Repositories in der ZETA Artifact Registry, sodass für Test
5403 und Entwicklung andere OPA Bundles verwendet werden können als für die
5404 Produktionsumgebung.

5405 Unter dem label "latest" werden Bundles für die aktive Policy Engine bereitgestellt. Unter
5406 dem label "latest-sim" werden Bundles für die Simulations-Policy Engine bereitgestellt.

5407 Im folgenden werden die Anforderungen an den Prozess zur Erstellung und Verteilung der
5408 OPA Bundles, den Policies und den Daten des CI/CD-Prozess, festgelegt.

5409 **A_25464-02 -Policies und Daten CI/CD-Prozess - Tamper-Proof Protokollierung** 5410 **von Administrationsaktivitäten**

5411 Der Policies und Daten CI/CD-Prozess MUSS ein "Tamper-Proof" Audit-Log von allen
5412 administrativen Vorgängen umsetzen. [≤]

5413 *Hinweis: Die Tamper-Proof Protokollierung von Administrationsaktivitäten betrifft*
5414 *insbesondere Änderungen an der ZETA Artifact Registry.*

5415 **A_25777-02 -Policies und Daten CI/CD-Prozess - Löschrufen Auditeinträge des** 5416 **Admin Audit-Logs**

5417 Der Policies und Daten CI/CD-Prozess MUSS sicherstellen, dass die Löschung eines
5418 Auditeintrags den gesetzlichen Vorgaben entspricht und frühestens nach 12 Monaten
5419 erfolgt. [≤]

5420 **A_25778-01 -Policies und Daten CI/CD-Prozess - Kontrolle des Audit-Logs**

5421 Der Policies und Daten CI/CD-Prozess MUSS so umgesetzt werden, dass das Audit-Log
5422 mindestens alle 3 Monate im Vieraugenprinzip kontrolliert wird. Die Rolle des Prüfers
5423 DARF NICHT an der Administration der Infrastruktur zur Bereitstellung der Policies und
5424 Daten teilnehmen. Bei der Kontrolle ist insbesondere auf ungewöhnliche, nicht
5425 nachvollziehbare oder maliziöse Administratoraktivitäten zu achten. [≤]

5426 **A_25465-01 -Policies und Daten CI/CD-Prozess - Änderungen nur durch** 5427 **berechtigte Nutzer**

5428 Der Policies und Daten CI/CD-Prozess MUSS sicherstellen, dass nur berechtigte Nutzer
5429 Änderungen von Policies oder PIP-Daten durchführen können. [≤]

5430 **A_25466-01 -ZETA Artifact Registry - Sicherheitsmeldung bei Aktualisierung von** 5431 **Policies oder PIP-Daten**

5432 Die ZETA Artifact Registry MUSS sicherstellen, dass bei der Aktualisierung der Policies
5433 oder der PIP-Daten eine Sicherheitsmeldung automatisiert über die von der gematik
5434 angebotene Schnittstelle an das TI SIEM-System übermittelt wird. [≤]

5435 **A_25467-01 -Policies und Daten CI/CD-Prozess - Änderungen nur unter 4 Augen**

5436 Der Policies und Daten CI/CD-Prozess MUSS sicherstellen, dass Änderungen in Policies
5437 oder PIP-Daten nur im Vieraugenprinzip durchgeführt werden können. [≤]

5438 **5.10 Telemetriedaten-Service**

5439 Der Telemetriedaten-Service ist eine Implementierung des OpenTelemetry-Frameworks
5440 der die Telemetriedaten aller ZETA Guard Komponenten einsammelt (Collector) und für
5441 die Weitergabe (Exporter) an externe Systeme (Monitoring des Anbieters, gematik
5442 Telemetriedaten Schnittstelle und gematik SIEM) aufbereitet.

5443 Zusätzlich nimmt der Telemetriedaten-Service Fehlermeldungen (Traces), Bestandsdaten
5444 (Metriken) und Selbstauskunftsdaten (Logs) des Resource Servers entgegen und leitet
5445 sie an die gematik Telemetriedaten Schnittstelle weiter.

5446 **A_27260 -ZETA Guard, Telemetriedaten-Service, Weitergabe der Daten ohne**
 5447 **Profilbildung**
 5448 Der Telemetriedaten-Service im ZETA Guard MUSS die von den Komponenten des ZETA
 5449 Guard gesammelten Telemetriedaten so verändert an das Monitoring System des
 5450 Anbieters und an die gematik Telemetriedaten Schnittstelle sowie SIEM der gematik
 5451 weitergeben, dass eine Profilbildung nicht mehr möglich ist.
 5452 [**<=**]

5453 **A_27261 -ZETA Guard, Telemetriedaten-Service, Protokoll**
 5454 Der Telemetriedaten-Service im ZETA Guard MUSS zur Weitergabe der Daten das
 5455 OpenTelemetry Protokoll (OTLP) über gRPC oder über HTTP/JSON verwenden. [**<=**]

5456 **A_27492-02 -ZETA Guard, OpenTelemetry Unterstützung**
 5457 Die ZETA Guard Komponenten HTTP Proxy, Authorization Server, Policy Engine und
 5458 Notification Service MÜSSEN für alle durchgeführten Operationen Telemetriedaten in
 5459 Form von Traces an den Telemetriedaten-Service senden. [**<=**]

5460
 5461 **A_27727-02 -ZETA Guard, Telemetriedaten-Service, Lieferung**
 5462 Der Telemetriedaten-Service im ZETA Guard SOLL Telemetriedaten asynchron an den
 5463 gematik Telemetriedaten-Service liefern. Dafür MUSS der Telemetriedaten-Service des
 5464 ZETA Guard für die Telemetriedatenlieferung von Traces an den gematik
 5465 Telemetriedaten-Service den OTLP-Exporter mit eingeschaltetem Batching verwenden.
 5466 [**<=**]

5467 **A_27725-01 -ZETA Guard, Telemetriedaten Service, Status Codes**
 5468 Der Telemetriedaten Service im ZETA Guard MUSS im Parameter
 5469 `http.response.status_code` einen HTTP-Statuscode gemäß
 5470 `Tab_gemSpec_ZETA_Telemetriedaten_HTTP_Statuscodes` übermitteln.

5472 **Tabelle 20: Tab_gemSpec_ZETA_Telemetriedaten_HTTP_Statuscodes**

HTTP-Statuscodes	Name der Statuscodegruppe	Beschreibung
1xx	INFORMATIONAL	Der Server hat die Anfrage erhalten und befindet sich in der Bearbeitung.
2xx	SUCCESSFUL	Die Operation wurde erfolgreich durchgeführt.
3xx	REDIRECTION	Der Client muss zusätzliche Maßnahmen ergreifen, um die Anfrage abzuschließen.
4xx	CLIENT_ERROR	Ein Client-seitiger Fehler verhindert die erfolgreiche Durchführung der Operation.
5xx	SERVER_ERROR	Ein Server-seitiger Fehler verhindert die erfolgreiche Durchführung der Operation.

5473 [**<=**]
 5474 *Hinweis: Es sind vom Anbieter, anstatt der Status Code Klassen (first digit of status code),*
 5475 *die konkreten 3-stelligen HTTP-Statuscodes gemäß [RFC9110] zu verwenden.*

5476

A_27494-02 -Telemetriedaten-Service, Custom Collector für Selbstauskunft

5477 Der Telemetriedaten-Service MUSS einen OTEL Collector für die Selbstauskunft des
 5478 Resource Servers bereitstellen. Die Selbstauskunft des Resource Servers erfolgt für jede
 5479 eigenständige Instanz als OTLP Log Record.
 5480

5481 Hinweis: Der OTLP Log Record für die Selbstauskunft ist in gemSpec_Perf
 5482 (Tab_gemSpec_Perf_Telemetriedaten_product_info) definiert. [\leq]
 5483

5484 Beispiel: Selbstauskunft OTLP-Log-Nachricht (JSON) vom Resource Server an den
 5485 Telemetriedaten-Service. Die Log-Nachricht wird unverändert an die gematik
 5486 Telemetriedaten-Schnittstelle weitergeleitet.

5487

```

5488 {
5489   "resourceLogs": [
5490     {
5491       "scopeLogs": [
5492         {
5493           "logRecords": [
5494             {
5495               "timestamp": "1678886400000000000",
5496               "body": { "stringValue": "Selbstauskunft" },
5497               "attributes": [
5498                 { "key": "product_name", "value": { "stringValue": "Backend-
5499 Service" } },
5500                 { "key": "product_version", "value": { "stringValue": "1.2.0"
5501 } },
5502                 { "key": "producttype_version", "value": { "stringValue":
5503 "1.2.3" } },
5504                 { "key": "configuration_version", "value": { "stringValue":
5505 "cfg-rev-45" } },
5506                 { "key": "pod_name", "value": { "stringValue": "my-app-pod-1"
5507 } }
5508             ]
5509           }
5510         ]
5511       }
5512     ]
5513   }
5514 ]
5515 }

```

5.11 HSM Proxy

5517 Der HSM Proxy dient als Abstraktionsschicht zwischen den ZETA Guard Kernkomponenten
 5518 (insbesondere dem AuthS und dem HTTP Proxy) und den physischen oder cloudbasierten
 5519 Hardware Security Modulen (HSM). Er stellt sicher, dass kryptographisches
 5520 Schlüsselmaterial (Private Keys) die sichere Umgebung des HSM niemals verlässt. Die
 5521 Kommunikation mit dem HSM Proxy erfolgt zustandslos über das gRPC-Protokoll.

5522 Neben der Erzeugung von Signaturen ermöglicht der HSM Proxy ein sicheres
 5523 Schlüsselmanagement durch Key Wrapping. Hierbei werden Schlüssel innerhalb der
 5524 sicheren HSM-Umgebung durch Key Encryption Keys (KEK) verschlüsselt, sodass sie
 5525 außerhalb des HSM gespeichert werden können, ohne die Vertraulichkeit zu gefährden.

5526 *Hinweis: Aktuell wird der HSM Proxy immer durch den Anbieter/Hersteller des TI 2.0*
5527 *Dienstes bereitgestellt.*

5528 **A_28829 -HSM Proxy, Schnittstelle zu den Komponenten**

5529 Der ZETA Guard HSM Proxy MUSS die Schnittstelle zu den Komponenten gemäß [hsm-
5530 proxy.proto] umsetzen.[<=]

5531 **A_28830 -HSM Proxy, Attribute Based Access Control**

5532 Der ZETA Guard HSM Proxy MUSS den Zugriff auf die Operationen der Schnittstelle nach
5533 berechtigter Komponente und key_id einschränken können.

5534 Die berechtigten Komponenten müssen mit mTLS authentifiziert werden können.[<=]

5535 **A_28831 -HSM Proxy, key_id**

5536 Der ZETA Guard HSM Proxy MUSS die key_id nach dem Schema[Komponente] - [Zweck] -
5537 [Algorithmus] - [Version/Datum] umsetzen.[<=]

5538 Beispiel für eine key_id: zeta-guard-keycloak-jwt-es256-v1 -> Darf nur von Keycloak
5539 genutzt werden.

5540 **5.11.1 Sicherheits- und Datenschutzerfordernungen an den HSM** 5541 **Proxy**

5542 **A_28746 -ZETA Guard - HSM Proxy in einer VAU**

5543 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der Hersteller des TI 2.0-
5544 Dienstes sicherstellen dass der HSM-Proxy in einer VAU läuft.[<=]

5545 **A_28748 -HSM-Proxy - Zugriff auf HSM Proxy durch attestierte ZETA-** 5546 **Komponenten**

5547 Falls der ZETA Guard in einer VAU umgesetzt wird, MUSS der HSM Proxy sicherstellen,
5548 dass den Zugriff auf den ZETA relevanten Schlüssel ausschließlich durch attestierte ZETA-
5549 Komponenten erfolgen kann.[<=]

5550 **A_28747 -HSM-Proxy - Zugriff auf ZETA-Schlüssel über HSM-Proxy**

5551 Falls der HSM-Proxy in einer VAU umgesetzt wird, MUSS der Anbieter des TI 2.0-Dienstes
5552 sicherstellen, dass Zugriffe auf die ZETA-relevanten Schlüssel oder die Schnittstellen zur
5553 Nutzung der Schlüssel im HSM ausschließlich durch einen HSM-Proxy erfolgen können,
5554 der entweder

- 5555 • gegenüber der HSM attestiert ist oder
- 5556 • gekoppelt (paired) ist, d. h., dass der Proxy im Rahmen einer dokumentierten
5557 Schlüsselzeremonie eindeutig mit dem HSM(-Cluster) bzw. der HSM(-Cluster)-Partition
5558 verbunden wird; die dabei erzeugten und zugewiesenen kryptographischen
5559 Zugriffscredentials sind für das HSM Proxy exklusiv).

5560 [**<=**]

5561 **A_28814 -HSM-Proxy - Sealing von Credentials**

5562 Der HSM-Proxy MUSS im Falle eines Pairings mit dem HSM seine lokal auf dem Server
5563 gehaltenen Zugriffs-Credentials mittels Sealing schützen und nach einem Neustart des
5564 Systems damit nur für die korrekt gestartete Verarbeitungskontexte des HSM-Proxy
5565 wieder verfügbar machen.[<=]

5566 **A_28797-01 -HSM-Proxy - Isolation HSM Proxy**

5567 Falls das HSM-Proxy in einer VAU umgesetzt wird, MUSS die VAU mittels eines
5568 wirksamen technischen Separationsmechanismus gewährleisten, dass
5569 Verarbeitungen in anderen Verarbeitungskontexten weder direkt noch indirekt
5570 die Integrität oder Sicherheit der Verarbeitungen einem HSM-Proxy
5571 beeinträchtigen können.[<=]

5572 **A_28816 -HSM-Proxy - Minimale Trusted-Computing-Base**

5573 Der HSM-Proxy MUSS als gehärteter Software-Stack umgesetzt sein und insbesondere
5574 dem Prinzip der minimalen Trusted-Computing-Base folgen, um die Härtung bzw. die
5575 notwendige Robustheit des Ergebnisses der Begutachtung zu erreichen. Die Trusted-
5576 Computing-Base DARF NICHT mehr Komponenten enthalten als für die Bereitstellung der
5577 HSM-Proxy-Funktionalität erforderlich. [≤]

5578 **A_28817 -HSM-Proxy - Keine Persistierung von Request/Response Daten**
5579 Der HSM-Proxy DARF Request- oder Response-Daten NICHT persistieren bzw. cachen.
5580 [≤]

5581 **A_28819 -HSM-Proxy - Integrität und Authentizität der Konfiguration und**
5582 **Rollback-Schutz**
5583 Der HSM-Proxy MUSS mit technischen Mitteln (wie Sealing, Signaturen und monotonen
5584 Versionszählern) sicherstellen, dass die Integrität und Authentizität geladener
5585 Konfigurationsdaten sichergestellt sind und dass ein Downgrade auf eine nicht mehr
5586 autorisierte Version der Konfiguration abgewehrt wird. [≤]

5587 **A_28820 -HSM-Proxy - Eingabe Validierung von Operationen**
5588 Der HSM-Proxy MUSS sicherstellen, dass alle Daten und Parameter, die über eine API
5589 kommuniziert werden, sicherheitstechnisch validiert werden. [≤]

5590 **A_28821 -HSM-Proxy - Schutzmaßnahmen gegen die OWASP Top 10 Risiken**
5591 Der HSM-Proxy MUSS geeignete technische Maßnahmen zum Schutz vor den Risiken in
5592 der aktuellen Version der [OWASP-Top-10-Risiken] umsetzen. [≤]

5593 **A_28822 -HSM-Proxy - Datenschutzkonformes Logging und Monitoring**
5594 Der HSM-Proxy MUSS die für den Betrieb des Zero Trust erforderlichen Logging- und
5595 Monitoring-Informationen in solcher Art und Weise erheben und verarbeiten, dass mit
5596 technischen Mitteln ausgeschlossen ist, dass dem Anbieter eines TI 2.0-Dienstes
5597 vertrauliche oder zur Profilbildung geeignete Daten zur Kenntnis gelangen. [≤]

5598 5.12 Notification Service

5599 Der ZETA Notification Service ist die zentrale Fassade vor den Push Gateways und
5600 übernimmt innerhalb der Telematikinfrastruktur die Aufgabe, Benachrichtigungen eines
5601 Fachdienstes (Resource Server) an die zugehörigen Endgeräte der Versicherten
5602 zuzustellen. Er baut auf dem Push-Notification-Konzept der gematik
5603 [gemF_PushNotification].

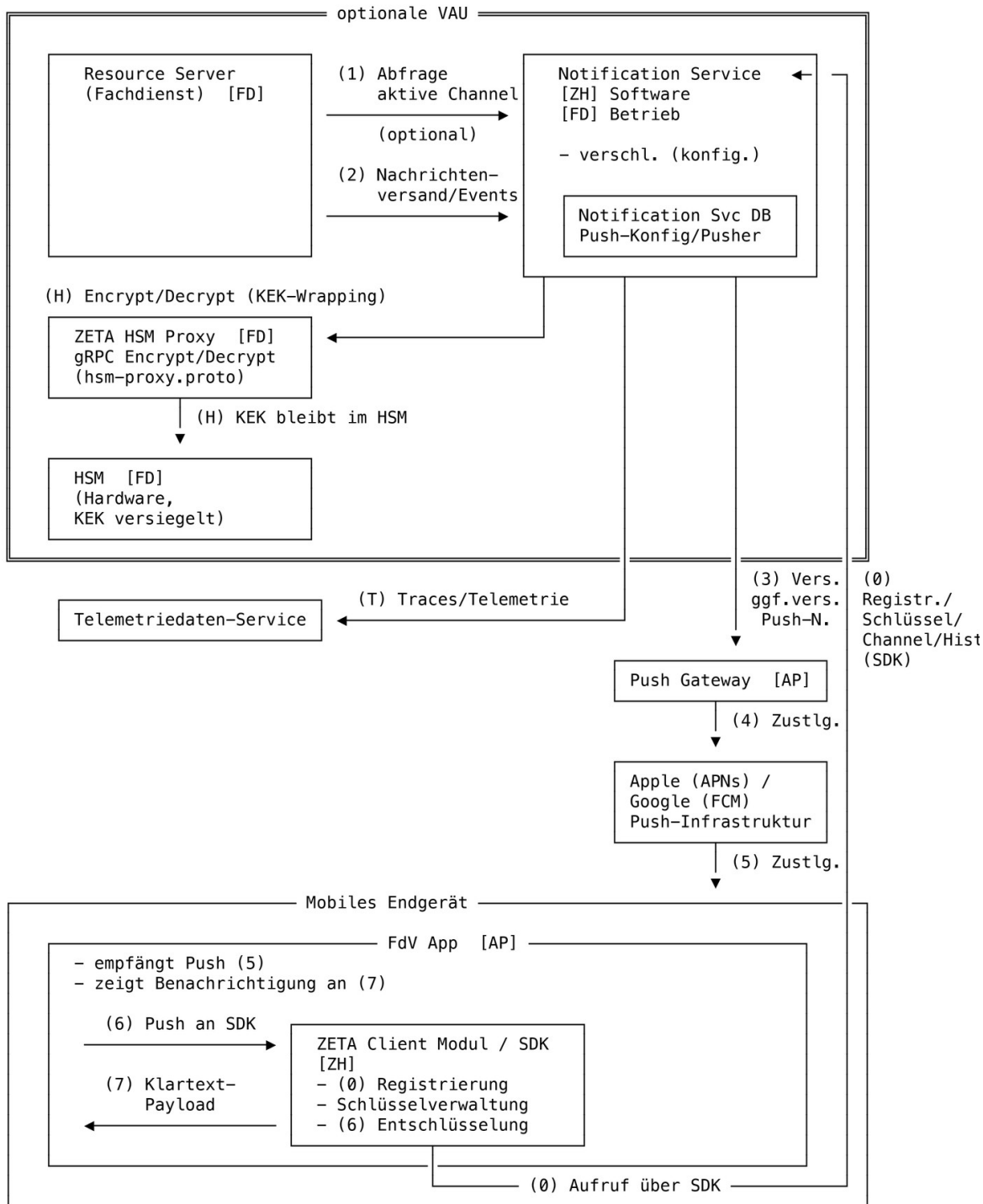
5604 Fachlich ordnet sich der Notification Service als Bindeglied zwischen dem Fachdienst und
5605 der plattformspezifischen Push-Infrastruktur der Betriebssystemhersteller (Apple APNs,
5606 Google FCM) ein. Er entkoppelt den Fachdienst von den Details der Push-Zustellung,
5607 verwaltet die Push-Konfigurationen und Pusher der Clients und stellt sicher, dass
5608 Nachrichteninhalte und
5609 Nutzeridentifikatoren geschützt verarbeitet werden.

5610 Der Dienst wird in zwei Betriebsvarianten betrieben: mit VAU, bei der Notification Service
5611 und Resource Server gemeinsam in einer Vertrauenswürdigem Ausführungsumgebung
5612 laufen und HSM-Anbindung sowie At-rest-Verschlüsselung und Pseudonymisierung aktiv
5613 sind, sowie ohne VAU, bei der diese HSM-gestützten Schutzmaßnahmen entfallen. Die
5614 Verschlüsselung der Nachrichten ist ein pro ZETA-Guard-Instanz zentral konfigurierbares
5615 Feature; ist es nicht aktiv, werden Nachrichten unverschlüsselt weitergegeben.

5616 5.12.1 Ablauf des Push-Versands

5617 Das folgende Architekturbild zeigt die am Push-Versand beteiligten Komponenten und
5618 ihr Zusammenspiel entlang der Zustellkette vom Fachdienst bis zur FdV App. Es umfasst

5619 die optionale VAU mit Resource Server, Notification Service und dessen Datenhaltung, die
 5620 HSM-Anbindung über den ZETA HSM Proxy, das Push Gateway sowie die
 5621 plattformspezifische Push-Infrastruktur von Apple und Google. Die eingeklammerten
 5622 Marker (0) bis (7) sowie (T) und (H) kennzeichnen die einzelnen Ablaufschritte, die im
 5623 Anschluss an das Bild erläutert werden.



5624
5625

Abbildung 29 - Ablauf Notification Versand

- 5626 • (0) Registrierung über das SDK: Das ZETA Client Modul (SDK) in der FdV App
5627 registriert die Pusher, verwaltet die Schlüssel und konfiguriert die Channel beim
5628 Notification Service. Optional kann der Client über das Feature Nachrichten-Historie
5629 bereits zugestellte oder ältere Benachrichtigungen abrufen, sofern der Betreiber
5630 dieses Feature aktiviert hat. Die Authentisierung erfolgt mit einem dienstspezifischen,
5631 DPoP-gebundenen ZETA-Zugriffstoken für den Notification Service.
- 5632 • (1) Optionale Channel-Abfrage: Der Resource Server (Fachdienst) fragt beim
5633 Notification Service die aktiven Channel ab. Die Kommunikation erfolgt innerhalb der
5634 VAU über einen technischen Nutzer.
- 5635 • (2) Nachrichtenversand vom Fachdienst: Der Resource Server übergibt die
5636 Notification-Events an den Notification Service. Dieser verschlüsselt die Nachrichten
5637 nur dann, wenn das zentral pro ZETA-Guard-Instanz konfigurierte Verschlüsselungs-
5638 Feature aktiv ist; andernfalls werden sie unverschlüsselt weitergegeben. Die
5639 Authentisierung erfolgt innerhalb der VAU über einen technischen Nutzer.
- 5640 • (3) Versand an das Push Gateway: Der Notification Service leitet die Push-Nachricht
5641 an das Push Gateway weiter, verschlüsselt bei aktivem Verschlüsselungs-Feature,
5642 sonst unverschlüsselt. Die Verbindung ist über mTLS abgesichert.
- 5643 • (4) Zustellung an die Push-Infrastruktur: Das Push Gateway übergibt die Nachricht an
5644 die plattformspezifische Push-Infrastruktur von Apple (APNs) beziehungsweise Google
5645 (FCM).
- 5646 • (5) Zustellung an das Endgerät: Die Push-Infrastruktur stellt die Push-Nachricht an das
5647 mobile Endgerät zu.
- 5648 • (6) Verarbeitung im SDK: Die FdV App reicht die empfangene Push-Nachricht an das
5649 integrierte ZETA Client Modul (SDK) weiter, das sie verarbeitet und entschlüsselt.
- 5650 • (7) Anzeige in der App: Das SDK übergibt den Klartext-Payload an die FdV App, die
5651 die Benachrichtigung anzeigt.
- 5652 • (T) Telemetrie: Der Notification Service liefert für alle Operationen Traces
5653 und Telemetriedaten an den Telemetriedaten-Service.
- 5654 • (H) HSM-Zugriff: Der Notification Service nutzt in der Betriebsvariante mit VAU den
5655 ZETA HSM Proxy für das KEK-Wrapping (Encrypt/Decrypt). Data Encryption Key und
5656 Pseudonymisierungs-HMAC-Schlüssel werden in der VAU erzeugt und per HSM-KEK
5657 versiegelt persistiert; der KEK verlässt das HSM nicht.

5658 5.12.2 Authentisierung

5659 Der Notification Service besitzt drei authentisierungspflichtige Schnittstellen mit
5660 unterschiedlichen Vertrauensmodellen:

- 5662 • ZETA Client Modul / SDK → Notification Service: nutzergebundener Zugriff,
5663 authentisiert über ein dienstspezifisches, DPoP-gebundenes ZETA-Zugriffstoken des
5664 ZETA Guard.
- 5665 • Resource Server → Notification Service: Service-to-Service-Zugriff innerhalb der
5666 (optionalen) VAU, authentisiert über einen technischen Nutzer per mTLS Terminierung
5667 innerhalb der VAU-Vertrauensgrenze).
- 5668 • Notification Service → Push Gateway: mTLS mit EV-Zertifikat in der Rolle des
5669 Fachdienstes gemäß [gemF_PushNotification]. Normativ: A_29982.

5670 **5.12.3 Rollen der Komponenten**

5671 Die Rollen der beteiligten Komponenten:

- 5673 • **ZetaClient Modul / SDK:** integrierter Bestandteil der FdV App; übernimmt die
5674 Registrierung der Pusher inkl. Schlüsselverwaltung und Channel-Konfiguration, sowie
5675 die Verarbeitung/Entschlüsselung der von der App weitergereichten Nachrichten und
5676 gibt den Klartext-Payload zur Anzeige an die FdV App zurück.
- 5677 • **Notification Service:** Fassade vor den Push Gateways; registriert/verwaltet Push-
5678 Konfigurationen, beantwortet die optionale Channel-Abfrage, nimmt Resource-Server-
5679 Events entgegen, verschlüsselt die Nachrichten (sofern konfiguriert) und leitet sie an
5680 die Push Gateways weiter. Teil der optionalen VAU (gemeinsam mit dem Fachdienst).
5681 Übernimmt die Push-Verarbeitungsrolle des Fachdienstes aus [gemF_PushNotification]
5682 (Pusher-/Schlüssel-/Channel-Verwaltung, Verschlüsselung, Versand).
- 5683 • **Notification Service DB:** persistiert Push-Konfigurationen/Pusher. In Variante (A)
5684 liegen alle Daten verschlüsselt at-rest (Envelope-Verschlüsselung mit HSM-
5685 versiegeltem Schlüssel); die Nutzeridentifikatoren (KVNR/Telematik-ID) werden
5686 ausschließlich pseudonymisiert (schlüsselgebundener MAC) gespeichert und indiziert.
- 5687 • **ZETA HSM Proxy:** einheitliche gRPC-Schnittstelle (vgl. Kap. 5.11) zum HSM des
5688 Anbieters. Der Notification Service nutzt ausschließlich die bestehende Schnittstelle
5689 mit Encrypt/Decrypt (KEK-Wrapping). Sowohl der Data Encryption Key (DEK) als auch
5690 der Pseudonymisierungsschlüssel werden in der VAU erzeugt und per HSM-KEK
5691 (Encrypt/Decrypt) versiegelt gespeichert; der KEK verlässt das HSM nicht. Die
5692 eigentliche Ver-/Entschlüsselung sowie die Pseudonymbildung erfolgen mit dem
5693 entwrappten Schlüssel innerhalb der VAU. Zugriff nur durch attestierte/gepaarte VAU-
5694 Instanzen.
- 5695 • **Resource Server**(Fachdienst): fragt optional die aktiven Channel ab und erzeugt
5696 bzw. versendet die Notification-Events; liegt ebenfalls in der VAU (falls genutzt).
- 5697 • **Push Gateway und Apple/Google Push-Infrastruktur:** plattformspezifische
5698 Zustellkette (4)/(5) zum Endgerät.
- 5699 • **FdV App:** empfängt die Push-Nachricht (5), reicht sie zur Entschlüsselung an das
5700 integrierte ZETA Client Modul / SDK weiter und zeigt die Benachrichtigung an.

5701 **5.12.4 Anforderungen**5702 **A_29957 -Notification Service - Betriebsvarianten und Geltungsbereich des**
5703 **HSM-Schutzes**5704 Der Notification Service MUSS in einer der beiden folgenden Betriebsvarianten
5705 betrieben werden:

- 5706 • Variante (A) – mit VAU: ZETA Guard und Notification Service werden gemeinsam in
5707 einer VAU betrieben. In dieser Variante MUSS der Notification Service die HSM-
5708 Anbindung sowie die At-rest-Verschlüsselung und Pseudonymisierung gemäß A_29958
5709 - A_29963 umsetzen.
- 5710 • Variante (B) – ohne VAU: Betrieb ohne VAU-Umgebung. In dieser Variante kommen
5711 die HSM-Anbindung und die HSM-gestützte Verschlüsselung/Pseudonymisierung nach
5712 A_29958 - A_29963 NICHT zur Anwendung; der Schutz der persistierten Daten MUSS
5713 durch die allgemeinen betrieblichen und organisatorischen Maßnahmen des
5714 Betreibers sichergestellt werden.

5715 Die Anforderungen A_29958 - A_29963 sind ausschließlich auf Variante (A) anzuwenden.
5716 Unabhängig von der gewählten Variante bleiben die Anforderungen zu

5717 Authentisierung, Autorisierung, Transportschutz sowie der Schutz personenbezogener
5718 Daten gegenüber dem Push Gateway (A_29984) uneingeschränkt gültig.
5719 [**<=**]

5720 **A_29958 -Notification Service - Schlüssel ausschließlich aus dem VAU-HSM**
5721 Der Notification Service MUSS sämtliches Schlüsselmaterial zur Verschlüsselung und
5722 Pseudonymisierung der persistierten Daten durch ein VAU-HSM schützen und hierfür
5723 ausschließlich die Schnittstelle des ZETA Guard HSM Proxy (A_28829) in
5724 ihrer bestehenden Form (Encrypt/Decrypt) verwenden. Der KEK DARF das HSM NICHT im
5725 Klartext verlassen. Der DEK und der Pseudonymisierungsschlüssel werden in der VAU
5726 erzeugt und ausschließlich HSM-KEK-versiegelt persistiert; im Klartext DÜRFEN sie nur
5727 innerhalb der VAU vorliegen. Entwrappte Schlüssel DÜRFEN ausschließlich im flüchtigen
5728 Speicher derVAU-Instanz vorgehalten (gecacht) werden, DÜRFEN NICHT persistiert
5729 werden und MÜSSEN bei Beendigung der VAU-Instanz sowie bei Schlüsselrotation
5730 (A_29963) sicher verworfen werden.[**<=**]

5731 **A_29959 -Notification Service - Verschlüsselung von Pusher- und Statusdaten**
5732 Der Notification Service MUSS Pusher-Konfigurationen, Channel-/Geräte-Zuordnungen,
5733 den Nachrichten-/Versandstatus sowie das Push-Schlüsselmaterial (initial shared secret
5734 iss, shared-secret-Jahr-Monat, AES/GCM-Schlüssel-Jahr-Monat) vor der Persistierung at-
5735 rest verschlüsseln. Das Push-Schlüsselmaterial besitzt gemäß [gemF_PushNotification]
5736 Kap. 5.5 (Schutzbedarf des iss: sehr hoch) den höchsten Schutzbedarf. Die
5737 Verschlüsselung MUSS als Envelope-Verschlüsselung umgesetzt werden: Ein in der VAU
5738 verwendeter Data Encryption Key (DEK) wird durch einen im HSM gehaltenen Key
5739 Encryption Key (KEK) über HsmProxyService.Encrypt/Decrypt gewrappt/entwrappt
5740 (Sealing gemäß Kap. 5.15); der KEK DARF das HSM NICHT verlassen. Die eingesetzten
5741 kryptographischen Verfahren, Schlüssellängen und Nutzungsgrenzen (einschließlich
5742 Nonce-/IV-Handhabung und Mengengrenzen der symmetrischen Verschlüsselung)
5743 MÜSSEN den Vorgaben aus [gemSpec_Krypt] entsprechen. Die
5744 Massen-Ver-/Entschlüsselung mit dem entwrappten DEK DARF ausschließlich innerhalb
5745 der VAU erfolgen; Erzeugung, Cache-Haltung und sicheres Verwerfen des DEK richten sich
5746 nach A_29958.[**<=**]

5747 **A_29960 -Notification Service - Pseudonymisierung von KVNR und Telematik-ID**
5748 Der Notification Service MUSS Nutzeridentifikatoren (KVNR, Telematik-ID) nicht im
5749 Klartext, sondern ausschließlich als Pseudonym persistieren und indizieren. Das
5750 PseudonymMUSS als schlüsselgebundener MAC über den Identifikator gebildet werden;
5751 das eingesetzte MAC-Verfahren und die Schlüssellänge MÜSSEN [gemSpec_Krypt]
5752 entsprechen. Der zugehörige
5753 Pseudonymisierungsschlüssel MUSS in der VAU erzeugt und ausschließlich HSM-versiegelt
5754 (per HsmProxyService.Encrypt/Decrypt) persistiert werden; die eigentliche MAC-
5755 Berechnung MUSS mit dem entwrappten Schlüssel innerhalb der VAU erfolgen
5756 (Erzeugung, Cache-Haltung und sicheres Verwerfen gemäß A_29958). Die
5757 Pseudonymbildung MUSS deterministisch sein (gleicher Identifikator → gleiches
5758 Pseudonym), damit die für (1) und (2) benötigten Lookups (Channel-/Pusher-Abfrage je
5759 Nutzer) ohne Klartext-Identifikator möglich sind. Eine Rotation des
5760 Pseudonymisierungsschlüssels verändert alle abgeleiteten Pseudonyme und ist daher nur
5761 unter den in A_29963 (Punkt 3) genannten Bedingungen (vollständige Re-
5762 Pseudonymisierung oder versioniertes Vorhalten der vorherigen Schlüsselgeneration)
5763 zulässig, damit der deterministische Lookup nicht bricht.
5764 [**<=**]

5765 **A_29961 -Notification Service - Domänentrennung der Pseudonyme**
5766 Der Notification Service MUSS für KVNR und Telematik-ID unterschiedliche
5767 Ableitungsdomänen verwenden - entweder über ein in der VAU eingebundenes
5768 Domänen-/Kontext-Label (kvnr / tid) bei der schlüsselgebundenen MAC-Berechnung oder
5769 über getrennte Pseudonymisierungsschlüssel -, sodass Pseudonyme nicht über
5770 verschiedene Identifikortypen oder

5771 über andere Dienste hinweg verkettet werden können (Domain-Separation).
5772 [**<=**]

5773 **A_29962 -Notification Service - Zugriff auf HSM nur durch attestierte VAU-** 5774 **Instanz**

5775 Der Notification Service MUSS sicherstellen, dass die HSM-Proxy-Operationen zur
5776 Envelope-Verschlüsselung (Encrypt/Decrypt) sowie das HSM-Wrapping/Entwrapping des
5777 Pseudonymisierungsschlüssels (Encrypt/Decrypt) ausschließlich durch eine attestierte,
5778 integrale VAU-Instanz ausgelöst werden können (attestierter bzw. gepairter HSM Proxy
5779 gemäß gemSpec_ZETA A_28746-A_28748; Zugriffsbeschränkung nach Komponente und
5780 key_id per
5781 mTLS/ABAC gemäß A_28830). Weder die DEK noch entschlüsselte Inhalte oder Klartext-
5782 Identifikatoren
5783 DÜRFEN die VAU-Vertrauensgrenze verlassen.
5784 [**<=**]

5785 **A_29963 -Notification Service - Schlüsselrotation und 4-Augen-Prinzip**

5786 Der Betreiber des Notification Service MUSS das Einbringen und Verwalten der HSM-KEK
5787 ausschließlich im 4-Augen-Prinzip zulassen (analog A_24612-*, A_27499). Die key_id des
5788 KEK MUSS dem Schema [Komponente]-[Zweck]-[Algorithmus]-[Version] gemäß
5789 gemSpec_ZETA A_28831 folgen (z. B. notification-service-dek-kek-aesgcm-v1).

5790 Der Notification Service MUSS eine Schlüsselrotation für sämtliches eingesetztes
5791 Schlüsselmaterial (HSM-KEK, VAU-seitig erzeugte DEK bzw. Wrapping-Keys sowie den
5792 Pseudonymisierungsschlüssel) unterstützen und implementieren. Das konkrete
5793 Rotationsverfahren ist nicht vorgegeben und liegt in der Verantwortung des
5794 Herstellers/Betreibers; es MUSS jedoch
5795 folgende Ergebnisse sicherstellen:
5796

- 5797 1. Eine Rotation MUSS sowohl regulär (geplant) als auch außerordentlich (z. B.
5798 bei Verdacht auf Schlüsselkompromittierung) ohne Betriebsunterbrechung der
5799 fachlichen Schnittstelle durchführbar sein.
- 5800 2. Über eine Rotation hinweg MÜSSEN bereits persistierte Daten weiterhin
5801 entschlüsselbar bleiben (kein Klartext-/Datenverlust). Hierzu MUSS je Datensatz die
5802 verwendete key_id/Schlüsselgeneration nachvollziehbar bleiben, und die jeweils ältere
5803 Schlüsselgeneration MUSS so lange verfügbar gehalten werden, bis der Bestand
5804 vollständig auf die neue Generation überführt ist.
- 5805 3. Eine Rotation des Pseudonymisierungsschlüssels DARF den nach A_29960
5806 geforderten deterministischen Lookup nicht brechen: Sie MUSS entweder mit einer
5807 vollständigen Re-Pseudonymisierung des Bestands einhergehen oder die vorherige
5808 Schlüsselgeneration für Lookups vorhalten (versioniertes Pseudonym), bis die Re-
5809 Pseudonymisierung abgeschlossen ist.

5810 Der entwrappte DEK bzw. Pseudonymisierungsschlüssel MUSS bei einer Rotation gemäß
5811 A_29959/A_29960 sicher aus dem VAU-Cache verworfen werden. [**<=**]

5812 Hinweis zur Abgrenzung: Die hier beschriebene Verschlüsselung/Pseudonymisierung
5813 schützt die at-rest in der Notification Service DB gespeicherten Daten. Sie ist unabhängig
5814 von der Ende-zu-Ende-Verschlüsselung der eigentlichen Push-Payload zwischen
5815 Notification Service und ZETA Client Modul/SDK sowie vom mTLS-Transportschutz
5816 gegenüber dem Push Gateway.

5817 **A_29964 -Notification Service - Realisierung der RS-Schnittstelle gemäß** 5818 **OpenAPI**

5819 Der Notification Service MUSS die Schnittstelle gegenüber dem Resource Server gemäß
5820 der OpenAPI-Spezifikation [OpenApi-Notification-Service-RS] (notification-service-rs-
5821 endpoint.yaml, Version 1.0.0) umsetzen und dabei mindestens die Operationen

5822 getActiveChannels (GET /users/{userId}/channels) und submitNotification (POST
5823 /notifications) anbieten. [<=]

5824 **A_29965 -Notification Service - Strikt asynchrone Verarbeitung**

5825 Der Notification Service MUSS eine eingehende Notification (POST /notifications) nach
5826 erfolgreicher Validierung mit 202 Accepted und einer notification_id bestätigen und die
5827 Zustellung an das Push Gateway asynchron durchführen. Die 202-Antwort DARF NICHT
5828 als Zustellbestätigung an das Endgerät interpretiert werden. [<=]

5829 **A_29966 -Notification Service - Synchrone Vorprüfung der** 5830 **Zustellvoraussetzungen**

5831 Ein Nutzer ist dem Notification Service nur bekannt, solange für ihn mindestens ein
5832 registriertes Gerät (Pusher) existiert; die Channel-/Pusher-Daten werden ausschließlich
5833 geräte-/pushkey-gebunden gehalten (kein eigenständiger Nutzer-Datensatz ohne Gerät).
5834 Ist für den adressierten Identifikator kein Gerät registriert, MUSS der Notification Service
5835 den
5836 Nutzer als unbekannt behandeln und die Operation (POST /notifications bzw. GET
5837 /users/{userId}/channels) mit 404 (USER_NOT_FOUND) beantworten; ein terminaler
5838 Status no_registered_devices wird nicht verwendet.
5839

5840 Der Notification Service MUSS bei der Annahme einer Notification synchron prüfen, ob der
5841 adressierte Channel für den Nutzer aktiv (enabled) ist. Ist der Channel auf keinem
5842 registrierten Gerät enabled, MUSS er die Notification nicht an das Push Gateway
5843 weiterleiten und den terminalen Status no_active_channel bereits in der 202-Antwort
5844 zurückgeben. In diesem
5845 terminalen Fall MUSS der Status synchron in der Antwort übermittelt werden; eine
5846 Persistierung eines Notification-Datensatzes oder ein nachgelagertes Status-Update DARF
5847 dabei NICHT erfolgen - eine ggf. zurückgegebene notification_id wird nur transient
5848 erzeugt. Ein (verschlüsselter) Notification-Datensatz DARF erst angelegt werden, wenn
5849 tatsächlich ein Versand an das Push Gateway erfolgt (Status accepted). [<=]

5850 Geltungsebene der Channel-Prüfung (Gerät vs. Nutzer): Die Channel-Konfiguration wird
5851 gemäß [gemF_PushNotification] A_27190 geräteindividuell pro pushkey geführt; ein
5852 Nutzer kann mehrere FdV-Instanzen/Geräte (Pusher) mit je eigener Channel-Auswahl
5853 registrieren. Die RS-seitige Schnittstelle GET /users/{userId}/channels bzw. die
5854 Vorprüfung beim Versand arbeitet demgegenüber auf Nutzerebene: Der Notification
5855 Service MUSS den Channel als für den Nutzer aktiv behandeln, wenn der Channel für
5856 mindestens ein registriertes Gerät (pushkey) des Nutzers den Status enabled hat
5857 (Aggregation über alle Geräte des Nutzers), und den Versand anschließend an jeden
5858 Pusher ausführen, dessen pushkey den adressierten Channel auf enabled gesetzt hat. Die
5859 gerätegenaue Auswertung bleibt damit Push-Spec-konform; die Nutzerebene ist nur eine
5860 Aggregation/Abstraktion für die RS-Schnittstelle und führt keine zweite, abweichende
5861 Channel-Haltung ein.

5862 **A_29967 -Notification Service - Nutzeridentifikation über KVNR oder Telematik-** 5863 **ID**

5864 Der Notification Service MUSS den Nutzer ausschließlich über die Kombination aus
5865 Identifikatorwert (userId/user.value) und deklariertem Typ (id_type ∈ {kvnr, telematik-
5866 id}) auflösen und die Werte gemäß den in der OpenAPI definierten Formaten/Pattern
5867 validieren. Die interne Persistierung/Indizierung erfolgt ausschließlich pseudonymisiert
5868 (siehe A_29960/A_29961). [<=]

5869 **A_29968 -Notification Service - Konfigurierbare, transparente** 5870 **Verschlüsselungsentscheidung**

5871 Die Verschlüsselung der Nachrichten-Payload MUSS ein pro ZETA Guard Instanz
5872 konfigurierbares Feature sein, das durch den Fachdienst-Anbieter bzw. ZETA-Betreiber
5873 gesetzt wird. Der Notification Service MUSS die Entscheidung, ob und wie eine
5874 Notification-Payload verschlüsselt wird, ausschließlich aus dieser eigenen
5875 (Instanz-)Konfiguration (je Fachdienst/Channel) ableiten. Der Resource Server DARF im

5876 Request keine Verschlüsselungsanweisung übergeben, und der Notification Service DARF
5877 eine solche NICHT auswerten (Transparenz gegenüber dem RS).
5878

5879 Die Konfigurierbarkeit ist durch die Spezifikation des jeweiligen Fachdienstes begrenzt:
5880 Schreibt die Fachdienst-Spezifikation die Verschlüsselung zwingend vor (z. B. ePA, E-
5881 Rezept), so MUSS das Feature in der Instanz aktiv sein und der Betreiber DARF es NICHT
5882 deaktivieren. Nur wo die Fachdienst-Spezifikation die Verschlüsselung als optional zulässt
5883 (z. B. TI-M), DARF der Betreiber sie je Instanz aktivieren oder deaktivieren. Bei aktivem
5884 Feature gelten die Schlüsselableitungs- und Verschlüsselungspflichten nach A_29970.
5885 **[<=]**

5886 **A_29969 -Resource Server - Nutzung der Notification-Service-Schnittstelle**

5887 Der Resource Server MUSS zum Versand von Push Notifications und zur optionalen
5888 Channel-Abfrage ausschließlich die Schnittstelle [OpenApi-Notification-Service-RS] des
5889 Notification Service verwenden und das reference-Feld – sofern genutzt – als
5890 unverschlüsselten Identifier ohne personenbezogene Daten (keine KVNR, kein Name, kein
5891 Nachrichteninhalte) befüllen. **[<=]**

5892 **A_29970 -Notification Service - Übernahme der 5893 Schlüsselableitungs-/Verschlüsselungspflichten**

5894 Der Notification Service MUSS, soweit er gemäß Konfiguration verschlüsselt versendet,
5895 die Verschlüsselungs- und Schlüsselableitungspflichten aus [gemF_PushNotification]
5896 übernehmen, insbesondere:

- 5897 • initiale und fortlaufende Schlüsselableitung mittels HKDF-SHA256 auf Basis shared-
5898 secret-Jahr-Monat mit Ableitungsvektor yyyy-MM (A_27157, A_27158-01, A_27160),
- 5899 • AES/GCM-Verschlüsselung des Nachrichteninhalts und Kodierung als Base64(IV ||
5900 Ciphertext || Authentication Code) (A_27161),
- 5901 • 1024-Byte-Padding zur Größenverschleierung (A_27610),
- 5902 • Einbetten von time_message_encrypted im Format yyyy-MM (A_27162),
- 5903 • Löschen überholten Schlüsselmaterials nach jeder Ableitung (A_27405),
- 5904 • UTF-8-Kodierung des Nachrichteninhalts (A_27680).

5905 **[<=]**

5906 **A_29971 -Notification Service - Übernahme der Versand- und Pusher-Pflege- 5907 Pflichten**

5908 Der Notification Service MUSS die Versandpflichten des Fachdienstes aus
5909 [gemF_PushNotification] beim Weiterleiten an das Push Gateway übernehmen,
5910 insbesondere:

- 5911 • Exponential Back-off bei Nichtverfügbarkeit/Fehlern des Push Gateways (A_27166),
- 5912 • Verarbeitung der Push-Gateway-Antwort und Löschen ungültig gewordener Pusher
5913 (A_27374),
- 5914 • Versand ausschließlich an die in der Registrierung hinterlegte URL (A_27652).

5915 Für die Übermittlung MUSS der Notification Service den zur Betriebsart passenden
5916 Endpunkt des Push Gateways nutzen:

- 5917 • unverschlüsselt: POST /notify (Einzelnachricht, Antwort { rejected[] }) oder POST
5918 /notify/batch (Stapel, Antwort BatchNotificationResponse),
- 5919 • verschlüsselt: ausschließlich POST /notifyEncrypted/batch (Stapel); ein Einzel-
5920 Endpunkt für verschlüsselte Nachrichten existiert in der Push-Spezifikation nicht –
5921 auch eine einzelne verschlüsselte Nachricht MUSS daher als Batch (mit einem
5922 Element) gesendet werden.

5923 Der Notification Service MUSS die in der Antwort zurückgemeldeten abgelehnten
 5924 Pushkeys (rejected je Notification bzw. je Batch-Item) auswerten und den Versand an
 5925 diese Pushkeys einstellen sowie die zugehörigen Pusher entfernen (Push-Spezifikation
 5926 RejectedPushKeys, [gemF_PushNotification] A_27374). Dies MUSS auch dann erfolgen,
 5927 wenn der abgelehnte Pushkey nicht aus der aktuellen, sondern aus einer früheren
 5928 Notification resultiert. [≤]

A_29972 -Notification Service - Client-Modul-Schnittstelle

5929 Der Notification Service MUSS für das ZETA Client Modul / SDK (Schritt (0)) die
 5930 Fachdienst-seitigen Schnittstellen aus [gemF_PushNotification] realisieren – mindestens
 5931 die Endpunkte zur Pusher-Registrierung/-Verwaltung (GET /pushers, POST /pushers/set,
 5932 A_27104) und zur Channel-Konfiguration (GET /channels, GET/POST /channels/{pushkey},
 5933 A_27190) . Die Channel-Konfiguration wird dabei – gemäß Push-Spezifikation –
 5934 geräteindividuell je pushkey gehalten (GET /channels/{pushkey} liefert die Konfiguration
 5935 für das Gerät mit diesem pushkey, POST /channels/{pushkey} aktualisiert sie); der Status
 5936 je Channel ist enabled, disabled oder not_set, wobei not_set wie disabled behandelt wird.
 5937 Eine
 5938 nutzerbezogene Sicht (z. B. GET /users/{userId}/channels der RS-Schnittstelle) MUSS der
 5939 Notification Service als Aggregation über alle pushkey-Konfigurationen des Nutzers
 5940 ableiten (siehe A_29966) und DARF NICHT als eigenständige, von der geräteindividuellen
 5941 Konfiguration abweichende Channel-Haltung implementiert werden.
 5942 [≤]

A_29973 -Notification Service - Umsetzung des Features Nachrichten-Historie

5944 Das Feature Nachrichten-Historie (Push-Historie) ist in [gemF_PushNotification] als
 5945 optionales Fachdienst-Feature beschrieben. In der ZETA-Notification-Service-
 5946 Implementierung ist es als Pflichtfunktion umzusetzen: Der Notification Service MUSS das
 5947 Feature Nachrichten-Historie implementieren. Dazu MUSS er

- 5949 • die fachdienstseitigen Historien-Endpunkte /history/* für authentifizierte Nutzer (ZETA
 5950 Client Modul / SDK, Authentisierung gemäß A_29975–A_29978) bereitstellen, über die
 5951 nach dem Versand auf die referenzierten versendeten Push-Nachrichten zugegriffen
 5952 werden kann, und
- 5953 • den beim Versand an das Push Gateway mitgegebenen, PII-freien identifier (Feld
 5954 reference, siehe A_29969) so verarbeiten/persistieren, dass eine vollständige,
 5955 datenschutzkonforme Liste der versendeten Notifications je Nutzer rekonstruierbar ist
 5956 (kein Personenbezug im identifier, keine Klartext-Inhalte im Push Gateway).

5957 Die Persistierung der Historiendaten MUSS den Schutzanforderungen der Pusher und
 5958 Statusdaten (vgl.. A_29959) genügen. [≤]

A_29974 -Notification Service - Optionale Aktivierung durch den Betreiber

5960 Die Nutzung/Aktivierung des Features Nachrichten-Historie KANN der Betreiber des
 5961 Notification Service (der ZETA-/FD-Anbieter) je Betrieb bzw. je Fachdienst konfigurieren;
 5962 sie ist für den Betreiber optional. Ist das Feature deaktiviert, MUSS der Notification
 5963 Service die Historien-Endpunkte /history/* deaktiviert halten (Beantwortung mit 404/501)
 5964 und DARF
 5965 keine Historiendaten persistieren. Die Implementierung des Features im Produkt bleibt
 5966 davon
 5967 unberührt gemäß A_29973 verpflichtend. [≤]

A_29975 -Notification Service - Authentisierung des Client-Moduls über ZETA-Zugriffstoken

5970 Der Notification Service MUSS Aufrufe des ZETA Client Moduls / SDK
 5971 (Registrierung/Verwaltung von Pushern, Schlüsselverwaltung, Channel-Konfiguration)
 5972 über ein vom ZETA Guard (AuthS) ausgestelltes ZETA-Zugriffstoken (OAuth 2.0 Access
 5973 Token, at+jwt gemäß [RFC9068], DPoP-gebunden gemäß [gemAPI_ZETA]) authentisieren.
 5974 Das Token MUSS spezifisch für den Notification Service ausgestellt sein (siehe A_29979).
 5975 Aufrufe ohne gültiges Token MUSS er mit 401 ablehnen. [≤]

5976 **A_29976 -Notification Service - Prüfung des ZETA-Zugriffstoken**
 5977 Der Notification Service MUSS bei jedem Aufruf durch das Client-Moduls Signatur,
 5978 Gültigkeitszeitraum exp / nbf / iat, Aussteller iss sowie die Zielgruppe aud - diese
 5979 MUSS der eigenen Resource-Kennung des Notification Service entsprechen (siehe
 5980 A_29979) - sowie die für die Operation erforderlichen Scopes/Berechtigungen des Tokens
 5981 prüfen und das Token bei einer fehlenden oder fehlgeschlagenen Prüfung zurückweisen
 5982 401 / 403. Ein Token, dessen aud den Notification Service nicht adressiert, DARF er
 5983 NICHT akzeptieren. [<=]

5984 **A_29977 -Notification Service - Sender-Constrained Token (Token-Binding)**
 5985 Der Notification Service MUSS sicherstellen, dass das ZETA-Zugriffstoken sender-
 5986 constrained ist (z. B. via DPoP oder mTLS-gebundenem Token gemäß den ZETA-
 5987 Vorgaben) und DARF ein Token NICHT akzeptieren, dessen Nachweis der Besitzbindung
 5988 (Proof-of-Possession) fehlt oder ungültig ist. Damit werden Token-Replay und Bearer-
 5989 Token-Missbrauch verhindert.
 5990 [<=]

5991 **A_29978 -Notification Service - Bindung von Operationen an den**
 5992 **authentisierten Nutzer**
 5993 Der Notification Service MUSS Schreib-/Leseoperationen des Client-Moduls (Registrierung,
 5994 Schlüssel, Channel-Konfiguration eines Pushers) ausschließlich auf die im ZETA-
 5995 Zugriffstoken authentifizierte Nutzeridentität beziehen und MUSS Operationen auf
 5996 Pusher/Daten anderer Nutzer mit 403 ablehnen. [<=]

5997 **A_29979 -Notification Service - Dienstspezifisches Access Token**
 5998 **(Audience/Resource & Discovery)**
 5999 Der Zugriff des Client-Moduls auf den Notification Service MUSS mit einem
 6000 dienstspezifischen Access Token erfolgen, das ausschließlich für den Notification Service
 6001 ausgestellt ist; ein für einen anderen Resource Server (Fachdienst) ausgestelltes Token
 6002 DARF NICHT verwendet bzw. akzeptiert werden. Hierzu MUSS:

- 6003 • der Notification Service ein eigenes Protected-Resource-Discovery-Dokument
 6004 (GET /.well-known/oauth-protected-resource) gemäß [gemAPI_ZETA] bereitstellen, das
 6005 die eigene resource-Kennung (stabile URI, z. B. https://<fd>/notification-service), die
 6006 zuständigen authorization_servers, die unterstützten scopes_supported
 6007 sowiedpop_bound_access_tokens_required: true deklariert,
- 6008 • der Client das Token beim Token Exchange (POST /token) gezielt für diese Resource
 6009 anfordern (Resource Indicator resource gemäß RFC 8707), sodass der AuthS die aud
 6010 des Tokens auf die Resource-Kennung des Notification Service setzt,
- 6011 • das Token ausschließlich die folgend normativ festgelegten Scopes tragen, die der
 6012 Notification Service je Operation nach dem Least-Privilege-Prinzip auswertet; eine
 6013 Operation ohne den ihr zugeordneten Scope MUSS er mit 403 ablehnen.

Scope	erechttigt ausschließlich zu (Endpunkte gemäß A_29972/A_29973)
notification.pusher.read	GET /pushers
notification.pusher.write	POST /pushers/set
notification.channel.read	GET /channels, GET /channels/{pushkey}
notification.channel.write	POST /channels/{pushkey}
notification.history.read	GET /history/* (nur bei aktiviertem Feature, A_29974)

--	--

- 6014
6015 Der Notification Service MUSS die aud-Bindung gemäß A_29976 durchsetzen. Ein Token
6016 mit einem nicht hier aufgeführten Scope DARF NICHT zur Autorisierung einer Operation
6017 der Client-Modul-Schnittstelle herangezogen werden. Andere als die in Tabelle X
6018 aufgeführten Scopes DARF der AuthS für die Resource-Kennung des Notification Service
6019 NICHT ausstellen.[<=]
- 6020 Hinweis (Umsetzungsvariante PEP): Die in A_29975-A_29977 und A_29979 geforderten
6021 Prüfungen des ZETA-Zugriffstokens (Signatur, exp/nbf/iat, iss, aud, Scopes, Sender-
6022 Constraining/DPoP) KANN der Betreiber statt im Notification Service selbst durch einen
6023 dem Notification Service vorgelagerten, dedizierten Policy Enforcement Point (PEP HTTP
6024 Proxy, vgl. Kap. 5.7.1) erbringen; der Ingress routet die Aufrufe des Client-Moduls dann an
6025 diesen PEP, der nur erfolgreich autorisierte Anfragen weiterleitet. Die Anforderungen
6026 gelten in diesem Fall als durch den vorgelagerten PEP erfüllt. Die objektbezogene
6027 Nutzerbindung nach A_29978 verbleibt in jedem Fall beim Notification Service.
- A_29980 -Notification Service - mTLS-Authentisierung des Resource Servers**
6028 Der Notification Service MUSS alle Aufrufe des Resource Servers (getActiveChannels,
6029 submitNotification) über gegenseitiges TLS (mTLS) mit einem technischen Nutzer
6030 authentisieren, das Client-Zertifikat des Resource Servers prüfen und Verbindungen mit
6031 fehlendem oder ungültigem Zertifikat ablehnen (401).[<=]
- A_29981 -Notification Service - mTLS-Terminierung innerhalb der VAU**
6033 Der Betreiber des Notification Service MUSS die für die mTLS-Authentisierung
6034 verwendeten Client-/Server-Zertifikate gemeinsam mit dem Betreiber des Resource
6035 Servers ausstellen und verwalten sowie eine Sperrung/Rotation kompromittierter
6036 Zertifikate ermöglichen.[<=]
- A_29982 -Notification Service - mTLS mit EV-Zertifikat zum Push Gateway**
6038 Der Notification Service MUSS die Verbindung zum Push Gateway über mTLS absichern
6039 und dabei – in der Rolle des Fachdienstes gemäß [gemF_PushNotification] – ein Extended-
6040 Validation-TLS-Zertifikat gemäß den Anforderungen des CA/Browser Forums verwenden
6041 bzw. ein solches des PushGateways prüfen (vgl. A_28267, A_28268).[<=]
- A_29983 -Notification Service - Mandantentrennung / Autorisierung je Fachdienst**
6043 Der Notification Service MUSS nach erfolgreicher Authentisierung autorisieren, dass ein
6044 Resource Server bzw. ein technischer Nutzer ausschließlich Notifications für die ihm
6045 zugeordneten Fachdienste/Channels versenden und ausschließlich die
6046 Channel-/Geräteinformationen dieser Fachdienste abfragen darf. Nicht autorisierte
6047 Zugriffe MUSS er mit 403 ablehnen.[<=]
- A_29984 -Notification Service - Keine personenbezogenen Klartextdaten Richtung Push Gateway**
6048 Der Notification Service DARF NICHT personenbezogene Daten (insbesondere KVNR,
6049 Telematik-ID, Name) oder unverschlüsselte Nachrichteninhalte an das Push Gateway
6050 übermitteln (vgl. A_27436, A_27539). Konsistent zu A_27396 DARF der Notification
6051 Service in den an das Push Gateway weitergereichten Registrierungs-/Versanddaten
6052 keine KVNR oder Telematik ID führen; die Adressierung erfolgt ausschließlich über
6053 pushkey/app_id.[<=]
- A_29985 -Notification Service - Registry zulässiger Channels**
6054 Der Notification Service MUSS eine Konfiguration/Registry der je Fachdienst zulässigen
6055 Channels führen. Eine Notification für einen dem Fachdienst nicht zugeordneten Channel
6056 MUSS er mit 403 und eine Notification für einen dem Nutzer unbekanntem Channel mit
6057 422 (UNKNOWN_CHANNEL, siehe [OpenApi-Notification-Service-RS]) zurückweisen;
6058 syntaktisch ungültige

6064 Channel-Angaben werden gemäß A_29986 mit 400 abgelehnt. Ein Channel gilt als „dem
6065 Nutzer unbekannt“ (422) nur dann, wenn er nicht in der Channel-Registry des
6066 Fachdienstes geführt wird, für den Fachdienst des Aufrufers aber grundsätzlich zulässig
6067 wäre. Ist der Channel in der Registry geführt, aber auf keinem registrierten Gerät des
6068 Nutzers enabled (einschließlich des Status not_set, der wie disabled behandelt wird, vgl.
6069 A_29972), antwortet der Notification Service mit 202 und dem terminalen Status
6070 no_active_channel (A_29966).[<=]

6071 **A_29986 -Notification Service - Eingabevalidierung**

6072 Der Notification Service MUSS alle eingehenden Requests gegen das in der OpenAPI
6073 definierte Schema validieren (Pflichtfelder, Formate/Pattern für KVNR und Telematik-ID,
6074 id_type) und ungültige Requests mit 400 ablehnen.[<=]

6075 **A_29987 -Notification Service - Schutz vor Überlast und Missbrauch**

6076 Der Notification Service MUSS Maßnahmen gegen Überlast und Missbrauch
6077 implementieren (Rate-Limiting mit 429/Retry-After, Begrenzung von Payload- und
6078 Stapelgrößen) und bei temporärer Nichtverfügbarkeit mit 503/Retry-After antworten.
6079 Überschreitet eine Notification die zulässige Payload-Größe, MUSS er sie mit 413
6080 (PAYLOAD_TOO_LARGE, siehe [OpenApi-Notification-Service-RS]) zurückweisen.[<=]

6081 **A_29988 -Notification Service - Datensparsamkeit und Aufbewahrungsfristen**

6082 Der Notification Service MUSS Notification-Status- und Verarbeitungsdaten
6083 (notification_id, Status) nur für eine kurze, definierte Aufbewahrungsfrist speichern und
6084 nach deren Ablauf löschen. Hiervon unberührt bleiben die Historiendaten des Features
6085 Nachrichten-Historie (A_29973), sofern dieses durch den Betreiber aktiviert ist; deren
6086 Aufbewahrung richtet sich nach der Betreiber-Konfiguration. Bei deaktiviertem Feature
6087 DARF der Notification Service keine Inhalts-/Versandhistorie vorhalten.[<=]

6088 **A_29989 -Notification Service - Löschung bei Deregistrierung**

6089 Der Notification Service MUSS bei Deregistrierung eines Pushers durch das Client-Modul
6090 die zugehörige Push-Konfiguration sowie das gespeicherte kryptographische
6091 Schlüsselmaterial (shared-secret-Jahr-Monat, AES/GCM-Schlüssel-Jahr-Monat) löschen
6092 (analog A_27156)[<=]

6093 **A_29990 -Notification Service - Protokollierung und Telemetrie ohne Klartext-PII**

6094
6095 Der Notification Service MUSS sicherheitsrelevante Ereignisse sowie
6096 Betriebs-/Telemetriedaten protokollieren, dabei aber keine Klartext-Identifikatoren (KVNR,
6097 Telematik-ID) und keine entschlüsselten Nachrichteninhalte protokollieren; Identifikatoren
6098 sind ausschließlich pseudonymisiert zu protokollieren. Eine etwaige Übermittlung an
6099 einen Telemetrie-/Monitoring-Dienst MUSS ebenfalls frei von Klartext-PII erfolgen.[<=]

6100 **A_29991 -Notification Service - TLS-/Krypto-Konformität**

6101 Der Notification Service MUSS für alle TLS-/mTLS-Verbindungen (RS-Schnittstelle sowie
6102 Push Gateway) die in [gemSpec_Krypt] zugelassenen TLS-Versionen, Cipher-Suiten und
6103 Schlüssellängen verwenden. Für die Verbindung zum Push Gateway gilt ergänzend
6104 A_29982 (EV-Zertifikat).[<=]

6105 **A_29992 -Notification Service - Idempotenz des Versands**

6106 Der Notification Service SOLL Retransmissionen des Resource Servers idempotent
6107 behandeln. Übermittelt der Resource Server einen Idempotency-Key, MUSS der
6108 Notification Service innerhalb der Aufbewahrungsfrist nach A_29988 wiederholte
6109 Übermittlungen mit demselben Schlüssel erkennen und dieselbe notification_id ohne
6110 erneuten Versand zurückgeben. Das reference-Feld ist hierfür NICHT geeignet, da es
6111 nicht eindeutig sein muss (A_29969).[<=]

6112 **5.13 Betrieb**

6113 Die Komponenten des ZETA Guard werden in einer Kubernetes (K8s) Umgebung des TI
6114 2.0 Dienst-Anbieters betrieben.

6115 **5.13.1 Anforderungen an Hersteller einer ZETA Komponente**6116 **A_27851 -ZETA Guard - Rückwärtskompatibilität**

6117 Der Hersteller des ZETA Guard MUSS sicher stellen, dass bei Änderungen an den
6118 öffentlichen Schnittstellen keine Breaking-Changes eingeführt werden, bei gleichzeitiger
6119 Wahrung der Rückwärtskompatibilität für alle aktiv unterstützten Releases. [<=]

6120 **5.13.2 Anforderungen an Hersteller eines TI 2.0-Dienstes**6121 **A_27818 -Unterstützung der Wartbarkeit des ZETA Guard-Dienstes**

6122 Der Hersteller eines Dienstes der TI2.0 MUSS regelmäßige Updates seines Produktes
6123 einplanen, damit die Aktualisierung der ZETA Guard gewährleistet ist. Ein Regelupdate
6124 erfolgt maximal einmal pro Quartal. Diese Anforderung gilt über den gesamten
6125 Lebenszyklus des Produktes hinweg. [<=]

6126 **5.13.3 Anforderungen an Anbieter eines TI 2.0-Dienstes**6127 **A_27792 -ZETA Guard - Verbot der Nutzung bestimmter ZETA Guard Versionen**

6128 Der Anbieter eines Dienstes der TI2.0 DARF eine zurückgezogene oder ungültige ZETA
6129 Guard Version NICHT produktiv einsetzen. [<=]

6130 **A_27793 -ZETA Guard - Reguläre Aktualisierung von ZETA Guard**

6131 Der Anbieter eines Dienstes der TI2.0 MUSS in der Lage sein, regelmäßig Patchupdates
6132 der integrierten ZETA Guard Version, die keine Auswirkung auf das Zusammenspiel mit
6133 dem Ressource Server haben, durchzuführen. Ein Regelupdate erfolgt maximal einmal
6134 pro Quartal. [<=]

6135 **A_27794 -ZETA Guard - Prüfung auf neue ZETA Guard Versionen**

6136 Der Anbieter eines Dienstes der TI2.0 MUSS regelmäßig auf neue freigegebene ZETA
6137 Guard Versionen prüfen und - wenn vorhanden - Aktualisierungen im Rahmen des
6138 Gültigkeitszeitraums einplanen. Ein Regelupdate erfolgt maximal einmal pro Quartal. [<=]

6139 **A_27795-01 -ZETA Guard - Gewährleistung der Verbindung zur ZETA Artifact
6140 Registry**

6141 Der Anbieter eines Dienstes der TI 2.0 MUSS gewährleisten, dass der eingesetzte ZETA
6142 Guard jederzeit Aktualisierungen der PIP-Daten und PAP-Policies sowie der Provisioning
6143 Daten über die lokale Artifact Registry von der ZETA Artifact Registry abrufen kann. [<=]

6144 *Hinweis: Der ausfallsichere Betrieb der lokalen Artifact Registry und die Verbindung zur*
6145 *ZETA Artifact Registry sind vom Anbieter sicherzustellen.*

6146 **A_27796 -ZETA Guard - Gewährleistung der Verbindung zur
6147 Telemetriedatenlieferung der gematik**

6148 Der Anbieter eines Dienstes der TI2.0 MUSS gewährleisten, dass der eingesetzte ZETA
6149 Guard jederzeit Datenlieferungen an die gematik übermitteln kann. [<=]

6150 *Hinweis: Notwendige Freischaltungen sind vom Anbieter zu beauftragen und deren*
6151 *Funktionsfähigkeit sicherzustellen.*

6152 **A_25773-02 -ZETA Guard - Nutzung der von der gematik bereitgestellten
6153 Container Images**

6154 Der Anbieter eines Dienstes der TI 2.0 MUSS die von der gematik bereitgestellten
6155 Container Images im ZETA Guard verwenden, um den Zugang zum TI 2.0 Dienst zu
6156 kontrollieren. [≤]

6157 *Hinweis: Ausgenommen sind die austauschbaren ZETA Guard Komponenten.*

6158 **5.13.4 Anforderungen für nahtlose Aktualisierungen**

6159 Es ist durch geeignete Maßnahmen sicherzustellen, dass ein unterbrechungsfreier
6160 Betrieb, bzw. die durchgängige Verfügbarkeit zu jeder Zeit gewährleistet ist.

6161 **A_25784 -ZETA Guard-Komponenten - Download von Aktualisierungen im Hintergrund**

6163 Die Komponenten des ZETA Guard MÜSSEN in der Lage sein, Aktualisierungen im
6164 Hintergrund herunterzuladen, ohne den laufenden Betrieb zu beeinträchtigen. [≤]

6165 **A_25785 -ZETA Guard-Komponenten - Nahtloser Übergang zu neuen Versionen**

6166 Die Komponenten des ZETA Guard MUSS einen Mechanismus bieten, der einen nahtlosen
6167 Übergang zu neuen Versionen oder Patches ermöglicht, ohne die Verfügbarkeit für
6168 Endnutzer zu unterbrechen. [≤]

6169 **A_25786 -ZETA Guard-Komponenten - Abschluss von Transaktionen vor Aktualisierung**

6171 Die Komponenten des ZETA Guard MUSS sicherstellen, dass alle aktuellen Transaktionen
6172 und Anfragen abgeschlossen oder ordnungsgemäß übernommen werden, bevor ein
6173 Update finalisiert wird. [≤]

6174 **A_25787 -ZETA Guard-Komponenten - Gewährleistung der Systemintegrität während Aktualisierungen**

6176 Die Komponenten des ZETA Guard MUSS während des gesamten
6177 Aktualisierungsprozesses die Systemintegrität und Sicherheitsrichtlinien aufrechterhalten.
6178 [≤]

6179 **A_25788 -ZETA Guard-Komponenten - Unterstützung von Rollbacks**

6180 Die Komponenten des ZETA Guard MUSS die Fähigkeit besitzen, zu einer stabilen
6181 Vorversion zurückzukehren, sollte eine Aktualisierung fehlerhaft sein oder abgebrochen
6182 werden müssen. [≤]

6183 **A_25789 -ZETA Guard-Komponenten - Schnelle Rollback-Durchführung**

6184 Die Komponenten des ZETA Guard MUSS Rollbacks schnell und ohne manuelle Eingriffe
6185 durchführen können. [≤]

6186 **5.13.5 Überwachung des Betriebsstatus**

6187 Um die Verfügbarkeit des ZETA-Guard sicherzustellen, führt die gematik ein externes
6188 Probing durch, das von außen den allgemeinen Erreichbarkeits- und Betriebsstatus
6189 überprüft. Dies erfolgt im Rahmen der überwachten TI 2.0-Dienstes, die eine konkrete
6190 ZETA Guard Instanz einsetzen.

6191 Dem Anbieter von TI 2.0-Dienstes wird empfohlen, ergänzend dazu geeignete technische
6192 Maßnahmen aufzusetzen, um interne Self-Checks der ZETA Guard Komponenten
6193 durchzuführen sowie Mechanismen zur Selbstüberwachung und -reparatur zu etablieren.
6194 Dies umfasst insbesondere automatisierte Verfahren zur Prüfung zentraler
6195 Funktionspfade, der Erreichbarkeit benötigter Abhängigkeiten sowie der Integrität
6196 interner Zustände.

6197 Durch solche Self-Checks können potenzielle Störungen frühzeitig erkannt, lokalisiert und
6198 häufig sogar ohne manuelle Eingriffe behoben werden. Dies maximiert die
6199 Betriebsstabilität, minimiert Ausfallzeiten und stärkt die Resilienz des Gesamtsystems.

6200 Bei einer Bereitstellung in Kubernetes-Umgebungen stehen hierfür erprobte
6201 Mechanismen zur Verfügung. Beispiele umfassen:

- 6202 • **Liveness-Probes**(z. B.): erkennt Hänger oder Deadlocks und startet Pods
6203 automatisch neu.
- 6204 • **Readiness-Probes**(z. B.): signalisiert, ob eine Komponente aktuell Anfragen
6205 zuverlässig bedienen kann; Pods werden erst nach erfolgreichem Probe-Ergebnis in
6206 den Service-Traffic aufgenommen.
- 6207 • **Startup-Probes**: eignen sich für Komponenten mit längerer Initialisierungsphase, um
6208 sicherzustellen, dass der Pod nicht fälschlicherweise als „defekt“ betrachtet wird.
- 6209 • **Self-Healing-Mechanismen**durch Kubernetes (z. B. RestartPolicy, ReplicaSets):
6210 ermöglichen automatische Wiederherstellung bei Ausfällen einzelner Komponenten,
6211 ohne dass ein manuelles Eingreifen nötig ist.
- 6212 • **Resource-Überwachung**mittels Kubernetes-Metriken (CPU, RAM, Netzwerk): kann
6213 frühzeitig Engpässe und Anomalien im Ressourcenverbrauch sichtbar machen.

6214 **A_27385-01 -ZETA Guard-Komponenten - Abbildung von Produkt- und** 6215 **Konfigurationsversionen**

6216 Der ZETA Guard MUSS als Subkomponente eines TI 2.0-Fachdienstes folgende
6217 Versionsangaben abrufbar vorhalten:

- 6218 • ZETA-Guard Modulversion gem.Semantic Versioning (Major-Minor-Patch)
- 6219 • Konfigurationsversion gem. [gemKPT_Betr#A_20219-01] des ZETA-Guard

6220 [**<=**]

6221 **A_27496 -Zeta Guard-Komponenten - Aufbereitung von Client Daten zum** 6222 **Monitoring**

6223 Die Komponenten des ZETA Guard MÜSSEN zu jedem Schnittstellenaufruf an den
6224 Resource-Server mindestens folgende Informationen aus der PDP Datenbank - und dem
6225 Request des Aufrufers verarbeiten und protokollieren, damit eine anschließende
6226 Zusammenführung von Monitoring-Daten aus verschiedenen Quellen (siehe [A_27494*])
6227 im Telemetriedaten-Service ermöglicht werden kann.

6228 Diese Daten umfassen mindestens:
6229

- 6230 • product_id - aus dem Token Self-Assessment des aufrufenden Clientsystems
- 6231 • product_version - aus demToken Self-Assessment des aufrufenden Clientsystems
- 6232 • professionOID - aus dem SM-B Zertifikat des aufrufenden Clientsystems

6233 [**<=**]

6234 **5.13.6 Leistungs-Anforderungen**

6235 In diesem Kapitel werden die Anforderungen an Performance, Skalierbarkeit und Last an
6236 die ZETA Guard Komponenten zusammengefasst. Die zugrunde liegenden Messungen
6237 dienen der Verifikation der Leistungsfähigkeit der Komponenten des ZETA Guard. Da der
6238 ZETA Guard in verschiedenen Szenarien genutzt werden kann, werden die Messungen
6239 einheitlich direkt am Eingang zum PEP bzw. PDP gemessen und durch das Service-Mesh
6240 vorgenommen. Die Latenzmessungen müssen über die Differenz zwischen Antwortzeiten
6241 beim Eingang in den PEP und dem Ausgang aus dem PEP gebildet werden.

6242 Die geforderten Antwortzeiten bzw. Latenzen werden dabei in Perzentilen
6243 vorgegeben. Zur Methode der Perzentile siehe u.a. auch [gemSpec_Perf, Kapitel 2.1] zur
6244 Bearbeitungszeit. Dort werden grundsätzlich Mittelwerte der Bearbeitungszeiten sowie
6245 99%-Quantile vorgegeben.

6246 **A_26486-01 -PEP HTTP Proxy - Bearbeitungszeiten**
 6247 Die Komponente PEP HTTP Proxy MUSS die Bearbeitungszeitvorgaben unter Last aus der
 6248 Tabelle "Tab_gemSpec_ZETA_PEP_HTTP_Proxy: Bearbeitungszeitvorgaben" erfüllen.
 6249

6250 Die Bearbeitungszeiten lassen sich in Latenzen und Antwortzeiten unterscheiden.
 6251 PEP Latenzen sind die Zeiten, die der PEP benötigt, um die Anfragen an den Fachdienst
 6252 weiterzuleiten, bzw. die Response eben zurück durchzuleiten. Die Benennung als Latenz
 6253 erfolgt aufgrund der Tatsache, dass der PEP hier Requests nur durchleitet, die eigentliche
 6254 Beantwortung erfolgt durch den Fachdienst.
 6255 PEP Antwortzeiten beziehen sich dabei dann auf den Austausch der Nachrichten 1-4 des
 6256 ASL Protokolls, die der PEP selbst beantwortet.

6257 **Tabelle 21: Tab_gemSpec_ZETA_PEP_HTTP_Proxy: Bearbeitungszeitvorgaben**

Request-Typ	Messung-Typ	Mittelwert	90%	95%	99%
Request an Fachdienst ohne ASL	Latenz	75ms	100ms	150ms	1s
Request an Fachdienst mit ASL	Latenz	75ms	100ms	150ms	1s
Austausch der ASL-Nachrichten 1-4	Antwortzeit	75ms	100ms	150ms	1s
Ausliefern der .well-known (1)	Antwortzeit	7,5ms	10ms	15ms	100ms

6258 (1) dies kann durch Caching mitigiert werden
 6259 **[<=]**

6260 **A_26487 -PEP HTTP Proxy -Skalierbarkeit**
 6261 Die Komponente PEP HTTP Proxy MUSS horizontal skalierbar sein, sodass mit jedem
 6262 zusätzlichen Pod mindestens 75% der Leistung eines einzigen Pods verfügbar werden.
 6263 **[<=]**

6264 **A_26488 -PEP HTTP Proxy - Last**
 6265 Die Komponente PEP HTTP Proxy MUSS pro Pod mehr als 300 Websocket Verbindungen
 6266 und mehr als 300 Requests pro Sekunde unterstützen können.**[<=]**

6267 **A_26489-02 -PDP Authorization Server - Bearbeitungszeiten**
 6268 Die Komponente PDP Authorization Server MUSS die Bearbeitungszeitvorgaben unter Last
 6269 aus der Tabelle "Tab_gemSpec_ZETA_PDP_Auth_Server: Bearbeitungszeitvorgaben"
 6270 erfüllen.
 6271

6272 Für den PDP gibt es verschiedene Typen von Requests, die hier analog zum PEP in
 6273 Perzentilen dargestellt werden. Alle Typen von Requests sind Antwortzeiten, daher ist
 6274 hier kein besonderer Messung-Typ aufgeführt.

6275 **Tabelle 22: Tab_gemSpec_ZETA_PDP_Auth_Server: Bearbeitungszeitvorgaben**

Request-Typ	Mittelwert	90%	95%	99%
/nonce	33ms	50ms	75ms	500ms
/register	75ms	100ms	150ms	1s

/token	75ms	100ms	150ms	1s
/revoke	75ms	100ms	150ms	1s

6276 [\leq]

6277 **A_26490 -PDP Authorization Server - Skalierbarkeit**

6278 Die Komponente PDP Authorization Server MUSS horizontal skalierbar sein, sodass mit
6279 jedem zusätzlichen Pod mindestens 75% der Leistung eines einzigen Pods verfügbar
6280 werden. [\leq]

6281 **A_26491 -PDP Authorization Server - Last**

6282 Die Komponente PDP Authorization Server MUSS pro Pod über alle Endpunkte zusammen
6283 mehr als 300 Requests pro Sekunde unterstützen können. [\leq]

6284 *Hinweis: Anfragen an abhängige Dienste im Hintergrund, um einen Request vollständig*
6285 *zu bearbeiten (z.B. OCSP), werden bei den Bearbeitungszeiten mit berücksichtigt, jedoch*
6286 *nicht dem abfragenden Dienst zur Last gelegt.*

6287 **5.13.7 Betriebliche Schnittstellendefinition**

6288 Die Komponenten des ZETA Guard stellen Endpunkte zur Verfügung, um die
6289 grundlegende Funktionalität, eingebettet in einen Service, zu gewährleisten. Jeder Dienst,
6290 der die ZETA Guard Komponenten betreibt, stellt damit folgende Endpunkte für einen
6291 Nutzer zur Verfügung.

6292 **A_28436 -ZETA Guard, Endpunkte im Internet**

6293 Der Anbieter eines TI 2.0 Dienstes MUSS die ZETA Guard und Kubernetes Cluster
6294 Endpunkte gemäß Tab_gemSpec_ZETA_Schnittstellendefinition_ZETA_Guard im Internet
6295 bereitstellen. [\leq]

6296 **Tabelle 23: Tab_gemSpec_ZETA_Schnittstellendefinition_ZETA_Guard**

Endpunkt / Anwendungsfall	Beschreibung
GET /.well-known/oauth-protected-resource/<resource>	Abruf des Resource Server Well-known JSON Dokuments. Default ist GET /.well-known/oauth-protected-resource/. Wenn mehrere Ressourcen vom Resource Server bereitgestellt werden, dann werden die Well-known Dokumente über den Subpath <resource> bereitgestellt. Beispiel: GET /.well-known/oauth-protected-resource/resource1 (https://www.rfc-editor.org/rfc/rfc9728.html)
GET /.well-known/oauth-authorization-server	Abruf des Autorisierungsserver Well-known JSON Dokuments
GET /.well-known/openid-federation HOST <ZETA Guard Authorization Server>	Abruf des Entity Statement Well-known nach OpenID Federation 1.0
GET /.well-known/openid-configuration HOST <ZETA Guard Authorization Server>	Abruf des OpenID Well-known JSON Dokuments. Der ZETA Guard Authorization Server stellt

	Subject Token für Workloads im ZETA Guard aus, für den Zugriff auf das SIEM und den Telemetriedaten Empfänger der gematik.
GET /openid/v1/jwks HOST <ZETA Guard Authorization Server>	JWKS des ZETA Guard Authorization Server Enthält die Signatur-Zertifikate des ZETA Guard Authorization Server
GET /openid/v1/jwks HOST <Kubernetes Cluster IDP>	JWKS des Kubernetes Cluster IDP Enthält die Signatur-Zertifikate des IDP <i>Hinweis: Die Bereitstellung erfolgt nicht durch de ZETA Guard, sondern durch den Kubernetes Cluster.</i>
GET /nonce	Nonce abrufen
POST /register	Dynamic Client Registration
GET /authorize	Für die Autorisierung nach OAuth Authorization Code Flow
POST /token	Es werden verschiedene Token Requests unterstützt: - Token Request mit Authorization Code - Token Request mit Subject Token (Token Exchange) - Token Request mit Refresh Token
POST /revoke	Widerruf (Revocation) eines Refresh Token gemäß RFC 7009

6297

6298 Zusätzlich wird die ZETA Artifact Registry bereitgestellt, die über die lokale Artifact
6299 Registry von ZETA Guard abgefragt wird, um beispielsweise aktualisierte Policy-
6300 Informationen abzuholen. Folgende Repositories sind in der ZETA Artifact
6301 Registry definiert.

6302 **Tabelle 24: Tab_gemSpec_ZETA_ZETA_Artifact_Registry_Repositories**

ZETA Artifact Registry Repository	Beschreibung
europa-west3-docker.pkg.dev/gematik-pt-zeta-prod/zeta-policies/{application}:{label}	Abruf der Policy eines Dienstes
europa-west3-docker.pkg.dev/gematik-pt-zeta-prod/zeta-helm	ZETA Guard Helm Charts
europa-west3-docker.pkg.dev/gematik-pt-zeta-prod/zeta-dcr	ZETA Guard Container Images, ZETA Guard Provisioning Container Image und ZETA Guard Sperrlisten

6303

6304 5.13.8 Prozesse zur Inbetriebnahme eines ZETA Guard

6305 Für den Betrieb eines ZETA Guard ist es erforderlich, dass ein Registrierungsprozess der
6306 gematik durchlaufen wird. In diesem Prozess werden Informationen über den Kubernetes
6307 Cluster und den Authorization Server des ZETA Guard bereitgestellt, die eine sichere
6308 Kommunikation mit Diensten der gematik (Artifact Registry, SIEM und Telemetriedaten
6309 Empfänger) und die Integration in den Federation Master der TI ermöglichen.

6310 **A_28437-01 -ZETA Guard, Registrierung bei der gematik**

6311 Der Anbieter des TI 2.0 Dienstes MUSS den ZETA Guard Authorization Server und den
6312 Issuer des Kubernetes Cluster IDP (oder alternativ einen JWK, der für die Signatur von
6313 Subject Token im Workload Identity Federation Authorization Flow verwendet wird) bei
6314 der gematik registrieren. [<=]

6315 5.14 Anforderungen an Dienste der TI

6316 Dienste der TI (Resource Server), die durch einen ZETA Guard geschützt sind, erhalten
6317 nur Requests von Clients oder anderen Diensten, wenn der PDP des ZETA Guard den
6318 Zugriff gewährt und ein Access Token ausgestellt hat. Der PEP des ZETA Guards setzt
6319 durch, dass nur Requests mit gültigem Access Token zum Resource Server gelangen.

6320 5.15 Sicherheitsleistungen des ZETA Guard für Resource Server

6321 Der ZETA Guard bietet umfassende Sicherheitsleistungen für Resource Server in der TI
6322 2.0, indem er das Zero Trust-Paradigma umsetzt. Seine wichtigsten Sicherheitsleistungen
6323 lassen sich wie folgt zusammenfassen:

- 6324 • **Zugriffskontrolle:** Der Policy Enforcement Point (PEP) fungiert als HTTP Proxy und
6325 kontrolliert den gesamten Datenverkehr zwischen Client-Anwendungen und dem
6326 Resource Server. Er lässt nur Anfragen mit gültigem Access Token durch, das vom
6327 Policy Decision Point (PDP) ausgestellt wurde. Zusätzliche Prüfungen, gesteuert durch
6328 Attribute im Access Token, können integriert werden. Optional kann konfiguriert
6329 werden, dass im Request Header PoPP ein PoPP Token vorhanden sein muss.
- 6330 • **Client-Registrierung:** Der ZETA Guard setzt eine sichere Client-Registrierung durch,
6331 bei der Clients durch verschiedene Mechanismen attestiert werden (Android Key and
6332 ID Attestation, Apple DCAppAttest, TPM Attestation und Software Attestation). Der
6333 Client wird in der Regel an eine TI-Identität gebunden (gID, SM(C)-B oder HBA).
- 6334 • **Autorisierung und Authentifizierung:** Der ZETA Guard OAuth2 Authorization
6335 Server steuert die Authentifizierung des Nutzers. Die Entscheidung zur Ausstellung
6336 des Access Token, nach erfolgreicher Client-Registrierung und Authentifizierung, wird
6337 durch die ZETA Guard Policy Engine basierend auf definierten Richtlinien (Policies)
6338 getroffen. Dies beinhaltet die Überprüfung von Nutzer- und Client-Registrierungsdaten
6339 inkl. Client-Attestierung. Unterstützte Authentifizierungsverfahren sind Token
6340 Exchange mit SM(C)-B signiertem Subject-Token sowie Authentifizierung über
6341 sektorale IDPs mit OIDC Flow.
- 6342 • **Policy-Based-Access-Control:** Der Policy Information Point (PIP) und der Policy
6343 Administration Point (PAP) verwalten und liefern die freigegebenen Policies an die
6344 ZETA Guard Policy Engine. Die Policies sind maschinenlesbar und definieren die
6345 Zugriffsregeln, nach denen die Entscheidung zur Ausstellung von Access und Refresh
6346 Token erfolgt. Die Integrität und Authentizität der Policies wird durch Signaturen
6347 sichergestellt.

- 6348
- 6349
- 6350
- **Schutz vor Token-Theft:** Durch den Einsatz von DPoP wird verhindert, dass gestohlene Access und Refresh Token durch Angreifer verwendet werden können, um Zugriff auf den Resource Server zu erhalten.
- 6351
- 6352
- 6353
- **TLS Transportverschlüsselung:** Alle Komponenten des ZETA Guards stellen die Vertraulichkeit und Integrität der transportierten Daten sicher, indem sie TLS an allen Endpunkten verwenden und clientseitig mTLS unterstützen.
- 6354
- 6355
- 6356
- 6357
- 6358
- 6359
- **Mehrschichtige Transport-Sicherheit:** ZETA/ASL bietet neben TLS eine zweite Sicherungsschicht beim Transport der Daten vom Client zum ZETA Guard, um Schwachstellen in der TLS-Schicht abzufangen. Dies schützt vor bekannten und zukünftigen Schwachstellen in TLS-Protokoll und -Implementierungen. Der Betrieb der Anwendung kann fortgeführt werden, selbst wenn Schwachstellen im TLS-Protokoll entdeckt werden. Diese Funktion ist optional nutzbar.
- 6360
- 6361
- 6362
- 6363
- 6364
- **Datenverschlüsselung:** Die ZETA Guard Daten (ZETA/ASL-Daten, Session-, User- und Client-Daten) werden bei der Persistenz zusätzlich verschlüsselt (SymK.DB.Enc). Der ZETA Guard kann so konfiguriert werden, dass private Schlüssel für die Datenverschlüsselung mit einem Key Encryption Key geschützt werden, der in einem Hardware Security Module (HSM) gespeichert ist.
- 6365
- 6366
- **VAU Unterstützung:** Der ZETA Guard kann optional in einer Vertrauenswürdig ausgeführten Umgebung (VAU) ausgeführt werden.
- 6367
- 6368
- 6369
- 6370
- 6371
- 6372
- 6373
- **Monitoring und Logging:** Der ZETA Guard sammelt Telemetriedaten und sicherheitsrelevante Ereignisse von PEP und PDP, um die Sicherheit kontinuierlich zu überwachen. Dies beinhaltet die Erkennung von Anomalien und potenziellen Bedrohungen. ZETA Guard sendet fachdienstspezifische Telemetriedaten und sicherheitsrelevante Ereignisse an die gematik (TI SIEM und gematik Telemetriedaten-Schnittstelle) sowie an den Anbieter des TI 2.0 Dienstes in einer anonymisierten Form, um eine Profilbildung zu verhindern.
- 6374
- 6375
- **Sicherheits- und Produktgutachten:** Für die ZETA Guard Implementierung werden ein Sicherheits- und ein Produktgutachten bereitgestellt.
- 6376
- 6377
- 6378
- 6379
- 6380
- 6381
- 6382
- **Produkthandbuch sowie Sicherheits- und Datenschutzkonzept:** Im Produkthandbuch sind die Anforderungen an den Betrieb sowie die Konfiguration des ZETA Guard beschrieben. Im Sicherheits- und Datenschutzkonzept sind die Bedrohungen und umgesetzten Maßnahmen gegen Bedrohungen beschrieben. Dadurch wird für den Anbieter des TI 2.0 Dienstes erkennbar, welche zusätzlichen Sicherheitsleistungen zum Schutz des Dienstes und der Daten vom Anbieter erbracht werden müssen.

6383 5.16 Weitere Leistungen des ZETA Guard für Resource Server

6384 Der ZETA Guard übernimmt für Resource Server weitere Leistungen:

- 6385
- 6386
- 6387
- 6388
- **gematik Telemetriedaten Lieferung:** Der Telemetriedaten-Service sammelt von allen Komponenten des ZETA Guard Metriken, Traces und Logdaten und bereitet diese für den Versand an das Monitoring des TI 2.0 Dienst-Anbieters und an den gematik Telemetriedaten Empfänger in anonymisierter Form auf.
- 6389
- 6390
- **SIEM Daten Lieferung:** Der Telemetriedaten Service sammelt vom Monitoring des TI 2.0 Dienst-Anbieters SIEM Daten und leitet sie an das SIEM der gematik weiter.
- 6391
- 6392
- 6393
- 6394
- **Notification Service:** Der ZETA Guard implementiert eine API um Notification-Konfigurationen für Nutzer und ihre Clients zu speichern und um Notification-Events vom Resource Server entgegenzunehmen und an die Clientsystem Notification Services weiterzuleiten.

- 6395
6396
6397
6398
- **Protected Resource Metadata:** Über den HTTP Proxy können Clients das OAuth 2.0 Protected Resource Metadata Well-known JSON-Dokument ([RFC9728]) abfragen, um die notwendigen Informationen für die Interaktion mit diesem Resource Server zu erhalten.

6399

6 Beispiele und Referenzimplementierungen

6400 Die gematik stellt API-Spezifikationen und Proof-of-Concept-Implementierungen im
6401 Internet zur freien Verfügung.

6402 Das Projekt <https://dsr.gematik.solutions> demonstriert eine Attestation mobiler
6403 Anwendungen auf gängigen mobilen Betriebssystemplattformen.

6404 Im GitHub-Projekt [gemAPI_ZETA] werden die Schnittstellenspezifikationen der hier
6405 spezifizierten Zero Trust-Komponenten veröffentlicht.

6406

7 Anhang A - Verzeichnisse

6407

7.1 Abkürzungen

6408

Tabelle 25: Im Dokument verwendete Abkürzungen

Kürzel	Erläuterung
API	Application Programming Interface
ASL	Application Security Level
CD	Continuous Delivery
CN	Common Name
DCR	Dynamic Client Registration
DEK	Data Encryption Key
DPoP	Demonstrating Proof-of-Possession
DSR	Device Security Rating
eGK	elektronische Gesundheitskarte
ePA	elektronische Patientenakte
FdV	Fachdienst der Versicherten
FIPS	Federal Information Processing Standards
GesundheitsID	Digitale Identität
HSM	Hardware Security Module
IDP	Identity Provider
IDS	Intrusion Detection System
ISMS	Informationssicherheitsmanagementsystem
JWT	JSON Web Token
k8s	Kubernetes

KEK	Key Encryption Keys
KVNR	Krankenversicherungsnummer
mTLS	Mutual Transport Layer Security
OCSP	Online Certificate Status Protocol
OIDC	OpenID Connect, https://de.wikipedia.org/wiki/OpenID_Connect
OPA	Open Policy Agent
OTLP	Open Telemetry Protocol
OTP	One Time Password
PAP	Policy Administration Point
PAR	Pushed Authorization Request
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKCE	Proof Key for Code Exchange
PoPP	Proof of Patient Presence
PU	Produktivumgebung
SAN	Subject Alternative Name
SIEM	Security Information and Event Management
SMC-B	Security Module Card Typ B, (Institutionskarte, Praxiskarte)
SPIFFE	Secure Production Identity Framework for Everyone
SPIRE	SPIFFE Runtime Environment
TI	Telematikinfrastruktur
TOFU	Trust On First Use
TPM	Trusted Platform Module
VAU	Vertrauenswürdige Ausführungsumgebung

ZAS	ZETA Attestation Service
ZETA	Zero Trust Access

6409 **7.2 Glossar**

6410 **Tabelle 26: Glossar der explizit im Dokument verwendeten Begriffe**

Begriff	Erläuterung
Authorization Server	Ein Server, der Zugriffstoken ausgibt, nachdem er die Identität eines Nutzers authentifiziert und die Berechtigungen überprüft hat. In OAuth 2.0-basierten Systemen ist der Authorization Server eine zentrale Komponente zur Verwaltung von Zugriffsrechten.
Google Remote Procedure Call (gRPC)	Framework für die Kommunikation zwischen verteilten Systemen. Die Daten werden in einem effizienten binären Datenformat (Protocol Buffers alias Protobuf) serialisiert und per HTTP/2 übertragen. Es ermöglicht Anwendungen, welche direkt auf Funktionen anderer Anwendungen zuzugreifen, als wären diese lokal verfügbar, unabhängig von der zugrunde liegenden Plattform oder Programmiersprache.
Demonstrating Proof of Possession (DPoP)	Ein DPoP Token ist ein Sicherheitsmechanismus im OAuth 2.0-Protokoll, der den Besitz eines kryptografischen Schlüssels nachweist. Es stellt sicher, dass der Client, der ein Access Token verwendet, auch den zugehörigen privaten Schlüssel besitzt, um Missbrauch durch Dritte zu verhindern.
Identity and Access Management (IAM)	Prozesse und Technologien zur Verwaltung digitaler Identitäten und deren Zugriff auf Unternehmensressourcen.
Management Service	Ein Dienst zur Verwaltung und Orchestrierung von ZETA Guard Ressourcen, insbesondere in containerisierten Umgebungen wie Kubernetes. Er ermöglicht die Verwaltung von Services, Pods und anderen Ressourcen innerhalb Kubernetes, um die Verfügbarkeit, Skalierbarkeit und Sicherheit der Anwendungen zu gewährleisten.
Open Policy Agent (OPA)	Eine Open Source Policy Engine, die es ermöglicht, Richtlinien als Code zu definieren und durchzusetzen, um Entscheidungslogik zentralisiert und flexibel in verschiedensten Software-Systemen und Anwendungen zu implementieren.
Policy Administration Point (PAP)	Eine Komponente, die Sicherheitsrichtlinien erstellt, verwaltet und verteilt. Der PAP definiert und verwaltet die Richtlinien, die von der PDP Policy Engine bei der Entscheidungsfindung verwendet werden.
Policy Decision Point (PDP)	Der PDP trifft die Entscheidung, ob ein Access Token ausgestellt werden darf, basierend auf den definierten Richtlinien und Informationen über den Anfragenden und den Client.
Policy Enforcement	Ein Punkt in einem Netzwerk, an dem Sicherheitsrichtlinien durchgesetzt werden. Der PEP überwacht und kontrolliert den Zugriff auf Ressourcen

Point (PEP)	basierend auf den Entscheidungen, die vom Policy Decision Point (PDP) getroffen werden.
Policy Information Point (PIP)	Eine Quelle von Attributen oder Kontextinformationen, die für die Entscheidungsfindung des Policy Decision Point (PDP) erforderlich sind. Der PIP stellt die notwendigen Daten zur Verfügung, um Zugriffsanfragen entsprechend den festgelegten Richtlinien zu bewerten.
Security Information and Event Management (SIEM)	Technologien und Prozesse zur Sammlung, Analyse und Korrelation von Sicherheitsdaten aus verschiedenen Quellen, um Sicherheitsvorfälle zu erkennen und darauf zu reagieren.
Telemetriedaten	<p>Telemetriedaten sind Daten, die von entfernten oder verteilten Systemen, Geräten oder Anwendungen gesammelt und an ein zentrales System zur Überwachung, Analyse und Verwaltung übertragen werden. Im Kontext der IT spielen Telemetriedaten eine entscheidende Rolle bei der Überwachung und Sicherung von Netzwerken und Systemen.</p> <p>Merkmale und Arten von Telemetriedaten:</p> <ul style="list-style-type: none"> - System- und Leistungsmetriken: Informationen über die Leistung und den Zustand von Hardware und Software, wie CPU-Auslastung, Speichernutzung, Netzwerkbandbreite und Festplattenkapazität. - Nutzeraktivitätsdaten: Protokolle und Aufzeichnungen über Nutzeraktionen und -verhalten, einschließlich Anmeldungen, Dateizugriffe, Anwendungsnutzung und andere Interaktionen. - Sicherheitsereignisse: Daten über sicherheitsrelevante Vorfälle, wie fehlgeschlagene Anmeldeversuche, erkannte Malware, unerlaubte Zugriffsversuche und andere sicherheitsbezogene Anomalien. - Netzwerkverkehrsdaten: Informationen über den Datenfluss im Netzwerk, einschließlich IP-Adressen, Ports, Protokolle, Datenmengen und Verbindungen zwischen verschiedenen Systemen und Diensten. - Konfigurationsdaten: Details zu den aktuellen Einstellungen und Konfigurationen von Systemen und Anwendungen, einschließlich Softwareversionen, installierte Patches und Sicherheitsrichtlinien.
Telemetriedaten-Service	Ein Dienst, der Telemetriedaten sammelt, verarbeitet und analysiert. Telemetriedaten umfassen Informationen über die Nutzung, Leistung und Zustände von Systemen und Anwendungen. Der Dienst hilft dabei, Einblicke in das Verhalten und die Gesundheit der Infrastruktur zu gewinnen, um proaktive Maßnahmen zur Optimierung und Sicherheit zu ergreifen.
Zero Trust (ZT)	Ein Sicherheitskonzept, das davon ausgeht, dass keine Entität (intern oder extern) automatisch vertraut wird. Alle Zugriffsanfragen werden überprüft, unabhängig von ihrem Ursprung.
Zero Trust/Application Security Layer (ZETA/ASL)	Eine auf HTTP basierende zusätzliche Verschlüsselung der Daten zwischen ZETA Client und ZETA Guard PEP oder zwischen ZETA Client und Resource Server. Die verschlüsselte Verbindung wird auch ZETA/ASL-Kanal genannt.

6411 **7.3 Abbildungsverzeichnis**

6412 Abbildung 1: NIST Zero Trust-Referenzarchitektur, Quelle [NIST_SP1800-35_FIG1].....15

6413 Abbildung 2: Abb-ZETA-Architektur.....16

6414 Abbildung 3 : Abb-Attestierungsablauf-nach-Betriebssystem.....51

6415 Abbildung 4 : Abb-ZETA-Schlüsselgenerierung-Windows-und-Linux.....52

6416 Abbildung 5 Abb-ZETA-Client-Start-mit-TPM-und-ZAS.....54

6417 Abbildung 6 Abb-ZETA-Service-Discovery.....55

6418 Abbildung 7 Abb-ZETA-TPM-Attestation-Key.....56

6419 Abbildung 8 Abb-ZETA-SE-Attestation-Key.....57

6420 Abbildung 9 Abb-ZETA-DCR-für-stationäre-Clients.....58

6421 Abbildung 10 Abb-ZETA-Client-Statement-mit-TPM-Attestation.....61

6422 Abbildung 11 Abb-ZETA-Client-Statement-mit-Apple-AppAttest.....62

6423 Abbildung 12 Abb-ZETA-Token-Exchange-mit-Attestation.....64

6424 Abbildung 13 Abb-ZETA-Token-Request-mit-Refresh-Token.....67

6425 Abbildung 14 Abb-ZETA-Zugriff-auf-RS-mit-ASL.....70

6426 Abbildung 15 Abb-ZETA-Zugriff-auf-RS-ohne-ASL.....72

6427 Abbildung 16 Abb-ZETA-Schlüsselgenerierung-Android.....74

6428 Abbildung 17 Abb-ZETA-Schlüsselgenerierung-Apple.....77

6429 Abbildung 18 Abb-ZETA-DCR-für-mobile-Clients.....79

6430 Abbildung 19 Abb-ZETA-Client-Statement-mit-Android-Attestation.....82

6431 Abbildung 20 Abb-ZETA-Client-Statement-mit-Apple-AppAttest.....84

6432 Abbildung 21 Abb-ZETA-OIDC-Authentifizierung-mobiler-Clients.....87

6433 Abbildung 22 Abb-ZETA-OIDC-Authorization-Request-mit-äußerem-und-innerem-PAR.....88

6434 Abbildung 23 Abb-ZETA-OIDC-Nutzerauthentisierung.....90

6435 Abbildung 24 Abb-ZETA-OIDC-Token-Bezug.....91

6436 Abbildung 25 Abb-ZETA-OAuth-Client-Authentifizierung-ohne-Nutzer.....94

6437 Abbildung 26 : Abb-ZETA-Dienst-zu-Dienst-Kommunikation.....97

6438 Abbildung 27 Abb-ZETA-Dienst-zu-Dienst-Kommunikation-mit-ZG-Client.....100

6439 Abbildung 28 Abb-ZETA-Token-Revocation.....103

6440 Abbildung 29 - Ablauf Notification Versand.....168

6441

6442 **7.4 Tabellenverzeichnis**

6443 Tabelle 1: Security Telemetriedaten PEP und PDP.....33

6444 Tabelle 2: Telemetriedaten Policy Entscheidungen.....34

6445 Tabelle 3: Telemetriedaten Angriffserkennung.....38

6446 Tabelle 4: Tab-ZETA-Guard-Keys.....44

6447 Tabelle 5: Tab-ZETA-Client-Keys.....46

6448 Tabelle 6: Tab-Gematik-Keys.....47

6449 Tabelle 7: ZT_HTTP_Statuscodes.....115

6450 Tabelle 8: Tab-Client-Assertion-JWT..... 120

6451 Tabelle 9: Tab-Client-Statement..... 120

6452 Tabelle 10: Tab-Posture-TPM..... 121

6453 Tabelle 11: Tab-Posture-Software..... 121

6454 Tabelle 12: Tab-Posture-Apple..... 122

6455 Tabelle 13: Tab-Posture-Android..... 123

6456 Tabelle 14: PEP HTTP Proxy - Zusätzliche HTTP-Header..... 148

6457 Tabelle 15: zeta-user-info-Header..... 149

6458 Tabelle 16: PDP Authorization Server - Plugin-Schnittstelle Application Authorization

6459 Backend..... 154

6460 Tabelle 17: SM(C)-B_Nutzer-Daten..... 155

6461 Tabelle 18: id_token_Nutzer-Daten..... 156

6462 Tabelle 19 Tab-ZETA-Token-Ausstellung-ASL-am-Resource-Server..... 159

6463 Tabelle 20: Tab_gemSpec_ZETA_Telemetriedaten_HTTP_Statuscodes..... 164

6464 Tabelle 21: Tab_gemSpec_ZETA_PEP_HTTP_Proxy: Bearbeitungszeitvorgaben..... 182

6465 Tabelle 22: Tab_gemSpec_ZETA_PDP_Auth_Server: Bearbeitungszeitvorgaben..... 183

6466 Tabelle 23: Tab_gemSpec_ZETA_Schnittstellendefinition_ZETA_Guard..... 184

6467 Tabelle 24: Tab_gemSpec_ZETA_ZETA_Artifact_Registry_Repositories..... 185

6468 Tabelle 25: Im Dokument verwendete Abkürzungen..... 189

6469 Tabelle 26: Glossar der explizit im Dokument verwendeten Begriffe..... 191

6470 Tabelle 27: Referenzierte Dokumente der gematik..... 194

6471 Tabelle 28: Weitere Referenzen..... 198

6472

6473 **7.5 Referenzierte Dokumente**

6474 **7.5.1 Dokumente der gematik**

6475 Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument

6476 referenzierten Dokumente der gematik zur Telematikinfrastruktur.

6477 **Tabelle 27: Referenzierte Dokumente der gematik**

posture-android.yaml[Quelle]	Herausgeber: Titel
[access-token.yaml]	Schema access-token.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/access-token.yaml
[API-ZETA-Attestation-]	API des ZETA Attestation Service https://github.com/gematik/ZETA/blob/main/docs/api/v1/index.md#zeta-

Service]	attestation-service-endpunkte
[as-well-known.yaml]	Schema für das OAuth Authorization Server Well-known JSON Dokument https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/as-well-known.yaml
[client-assertion-jwt.yaml]	Schema client-assertion-jwt https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/client-assertion-jwt.yaml
[client-data.yaml]	Schema client-data.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/client-data.yaml
[client-statement.yaml]	Schema client-statement.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/client-statement.yaml
[dcr-request.yaml]	Schema dcrrequest.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/dcr-request.yaml
[federation-master.yaml]	Schemafederation-master.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/federation-master.yaml
[gemAPI_ZETA]	gematik: OpenAPI Schnittstellen- und Schemaspezifikation ZETA https://github.com/gematik/zeta
[gemGlossar]	gematik: Glossar der Telematikinfrastruktur
[gemF_PushNotification]	gematik: Feature: Anwendungsübergreifende Push Notification https://gemspec.gematik.de/docs/gemF/gemF_PushNotification/latest/
[gemKPT_Betr]	gematik: Betriebskonzept Online-Produktivbetrieb https://gemspec.gematik.de/docs/gemKPT/gemKPT_Betr/latest/
[gemSpec_DS_Hersteller]	gematik: Spezifikation Datenschutz- u. Sicherheitsanforderungen der TI an Hersteller https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Hersteller/latest/
[gemSpec_IDP_Sek]	gematik: Spezifikation Sektoraler Identity Provider https://gemspec.gematik.de/docs/gemSpec/gemSpec_IDP_Sek/latest/
[gemSpec_Krypt]	gematik: Übergreifende Spezifikation Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur https://gemspec.gematik.de/docs/gemSpec/gemSpec_Krypt/latest/

[gemSpec_OID]	Spezifikation Festlegung von OIDs https://gemspec.gematik.de/docs/gemSpec/gemSpec_OID/latest/
[gemSpec_OM]	gematik: Übergreifende Spezifikation Operations und Maintenance https://gemspec.gematik.de/docs/gemSpec/gemSpec_OM/latest/
[gemSpec_Perf]	gematik: Übergreifende Spezifikation Performance und Mengengerüst TI-Plattform https://gemspec.gematik.de/docs/gemSpec/gemSpec_Perf/latest/
[gemSpec_PKI]	Übergreifende Spezifikation Spezifikation PKI https://gemspec.gematik.de/docs/gemSpec/gemSpec_PKI/latest/
[GitHub ZETA Schemas]	Schemas für zusätzliche HTTP-Header https://github.com/gematik/zeta/tree/main/src/schemas
[hsm-proxy.proto]	Protobuf Spezifikation der HSM Proxy Schnittstelle für Komponenten https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/gRPC/hsm-proxy.proto
[ISMS]	Spezifikation Datenschutz- und Sicherheitsanforderungen der TI an Anbieter (Abschnitt 3.3) https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Anbieter/latest/#3.3
[notification-service-rs-endpoint]	OpenApi Spezifikation der Schnittstelle zwischen ZETA Guard Notification Service und Resource Server https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/openapi/notification-service-rs-endpoint.yaml
[opr-well-known.yaml]	Schema für das OAuth Protected Resource Metadata Well-known JSON Dokument https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/opr-well-known.yaml
[pdp-decision.yaml]	Schema pdp-decision.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/pdp-decision.yaml
[policy-engine-client-data.yaml]	Schema policy-engine-client-data.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/policy-engine-client-data.yaml
[policy-engine-input.yaml]	Schemapolicy-engine-input.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/policy-engine-input.yaml
[posture-android.yaml]	Schemaposture-android.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/posture-android.yaml
[posture-	Schemaposture-apple.yaml

apple.yaml]	https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/posture-apple.yaml
[posture-software.yaml]	Schema posture-software.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/posture-software.yaml
[posture-tpm.yaml]	Schema posture-tpm.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/posture-tpm.yaml
[posture.yaml]	Schema posture.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/posture.yaml
[dcr-response-202.yaml]	Schema dcr-response-202.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/dcr-response-202.yaml
[TrustedTPM_Root CA]	Liste der vertrauenswürdigen TPM Stamm- und SubCA-Zertifikate https://learn.microsoft.com/de-at/windows-server/security/guarded-fabric-shielded-vm/guarded-fabric-install-trusted-tpm-root-certificates
[subject-token-smb.yaml]	Schema subject-token-smb.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/subject-token-smb.yaml
[token-response.yaml]	Schema token-response.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/token-response.yaml
[verify-request.yaml]	Schema verify-request.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/verify-request.yaml
[zeta-attestation-token.yaml]	Schemazeta-attestation-token.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/zeta-attestation-token.yaml
[zeta-user-info.yaml]	Schema zeta-user-info.yaml https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/zeta-user-info.yaml
[zeta-error.yaml]	Schema zeta-error.yaml https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/zeta-error.yaml
[zeta-guard-client-management]	OpenAPI Spezifikation der ZETA Guard Client Management API https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/openapi/zeta-guard-client-management.yaml
[zeta-guard-admin-oob]	OpenAPI Spezifikation der ZETA Guard Out Of Band Admin API https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/

	openapi/zeta-guard-admin-oob.yaml
--	---

6478

7.5.2 Weitere Referenzen

6479

Tabelle 28: Weitere Referenzen

[Quelle]	Herausgeber (Erscheinungsdatum) Titel
[Android Platform Security Model]	The Android Platform Security Model (2023) https://research.google/pubs/the-android-platform-security-model/ (Abruf 01/2025)
[Apple Platform Security Guide]	Einführung in die Sicherheit der Apple-Plattformen https://support.apple.com/de-de/guide/security/seccd5016d31/web (Abruf 01/2025)
[BSI-Grundschriftz]	IT-Grundschriftz - Informationssicherheit mit System https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschriftz/it-grundschriftz_node.html (Abruf 01/2025)
[CAB-Forum]	Certification Authority Browser Forum (CA/Browser Forum) https://cabforum.org/ (Abruf 01/2025)
[CAPEC OWASP]	CAPEC: OWASP Related Patterns CAPEC - CAPEC-659: OWASP Related Patterns (Version 3.9) (mitre.org) (Abruf 01/2025)
[ExpBack]	Exponential Backoff https://en.wikipedia.org/wiki/Exponential_backoff (Abruf 01/2025)
[JOSE]	JSON Object Signing and Encryption (JOSE) https://www.iana.org/assignments/jose/jose.xhtml
[NIST_SP1800-35_FIG1]	National Institute of Standards and Technology (NIST) NIST SP 1800-35 Publication (Figure 1 - General ZTA Reference Architecture) https://pages.nist.gov/zero-trust-architecture/VolumeB/architecture.html (Abruf 01/2025)
[OPA Bundle]	Open Policy Agent, Bundles https://www.openpolicyagent.org/docs/latest/management-bundles/ (Abruf 01/2025)
[Open Policy Agent]	Open Policy Agent https://www.openpolicyagent.org/docs/latest/ (Abruf 01/2025)
[OWASP-]	OWASP Top 10

Top-10-Risiken]	https://owasp.org/www-project-top-ten/ (Abruf 01/2025)
[OWASP-Top-Ten-Kubernete s]	OWASP Kubernetes Top 10 https://owasp.org/www-project-kubernetes-top-ten/ (Abruf 04/2026)
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels https://datatracker.ietf.org/doc/html/rfc2119 (Abruf 01/2025)
[RFC2986]	PKCS #10: Certification Request Syntax Specification https://datatracker.ietf.org/doc/html/rfc2986 (Abruf 01/2025)
[RFC6066]	Transport Layer Security (TLS) Extensions: Extension Definitions https://datatracker.ietf.org/doc/html/rfc6066 (Abruf 01/2025)
[RFC6749]	The OAuth 2.0 Authorization Framework https://datatracker.ietf.org/doc/html/rfc6749 (Abruf 01/2025)
[RFC7009]	OAuth 2.0 Token Revocation https://www.rfc-editor.org/info/rfc7009/
[RFC7231]	Hypertext Transfer Protocol (HTTP/1.1) Semantics and Content https://datatracker.ietf.org/doc/html/rfc7231 (Abruf 01/2025)
[RFC7232]	Hypertext Transfer Protocol (HTTP/1.1) Conditional Requests https://datatracker.ietf.org/doc/html/rfc7232 (Abruf 01/2025)
[RFC7239]	Forwarded HTTP Extension https://datatracker.ietf.org/doc/html/rfc7239
[RFC7515]	JSON Web Signature (JWS) https://datatracker.ietf.org/doc/html/rfc7515
[RFC7519]	JSON Web Token (JWT) https://datatracker.ietf.org/doc/html/rfc7519
[RFC7521]	Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants https://datatracker.ietf.org/doc/html/rfc7521 (Abruf 01/2025)
[RFC7523]	JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants https://datatracker.ietf.org/doc/html/rfc7523
[RFC7591]	OAuth 2.0 Dynamic Client Registration Protocol https://datatracker.ietf.org/doc/html/rfc7591
[RFC7592]	OAuth 2.0 Dynamic Client Registration Management Protocol https://datatracker.ietf.org/doc/html/rfc7592

[RFC7636]	Proof Key for Code Exchange by OAuth Public Clients https://datatracker.ietf.org/doc/html/rfc7636
[RFC7638]	JSON Web Key (JWK) Thumbprint https://datatracker.ietf.org/doc/html/rfc7638
[RFC8414]	OAuth 2.0 Authorization Server Metadata https://datatracker.ietf.org/doc/html/rfc8414
[RFC8555]	Automatic Certificate Management Environment (ACME) https://datatracker.ietf.org/doc/html/rfc8555#section-6.5.1
[RFC8693]	OAuth 2.0 Token Exchange https://datatracker.ietf.org/doc/html/rfc8693
[RFC8705]	OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Token https://datatracker.ietf.org/doc/html/rfc8705
[RFC8707]	Resource Indicators for OAuth 2.0 https://www.rfc-editor.org/rfc/rfc8707.html
[RFC9068]	JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens https://www.rfc-editor.org/info/rfc9068/
[RFC9110]	HTTP Semantics https://datatracker.ietf.org/doc/html/rfc9110
[RFC9126]	OAuth 2.0 Pushed Authorization Request https://datatracker.ietf.org/doc/html/rfc9126
[RFC9334]	Remote Attestation procedureS (RATS) Architecture https://www.rfc-editor.org/rfc/rfc9334.html
[RFC9449]	OAuth 2.0 Demonstrating Proof of Possession (DPoP) https://datatracker.ietf.org/doc/html/rfc9449
[RFC9470]	OAuth 2.0 Step Up Authentication Challenge Protocol https://datatracker.ietf.org/doc/html/rfc9470
[RFC9728]	OAuth 2.0 Protected Resource Metadata https://www.rfc-editor.org/rfc/rfc9728.html
[Shared Signals]	OpenID Shared Signals and Events Framework https://openid.net/specs/openid-sse-framework-1_0.html (Abruf 01/2025)
[SPIFFE und SPIRE]	Universal identity control plane for distributed systems https://spiffe.io/ (Abruf 01/2025)
[TR-03107-1]	BSI TR-03107 Elektronische Identitäten und Vertrauensdienste im E-Government https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/

	TechnischeRichtlinien/TR03107/TR-03107-1.pdf (Abruf 01/2025)
[TR-03161]	BSI TR-03161 Anforderungen an Anwendungen im Gesundheitswesen https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03161/tr-03161.html (Abruf 01/2025)
[TR-03161-1]	Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen Teil 1: Mobile Anwendungen; Version 3.0 https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-1.pdf?__blob=publicationFile&v=13 (Abruf 01/2025)
[VerifiedBoot]	Verifizierter Start https://source.android.com/docs/security/features/verifiedboot?hl=de (Abruf 01/2025)

6480

6481