

**Telematikinfrastruktur 2.0**

# Spezifikation Zero Trust Access (ZETA)

Version:	1. <del>1</del> 2.0 <u>CC</u>
Revision:	<del>13244881402548</del>
Stand:	<del>06-08</del> 15.10.2025
Status:	<u>zur Abstimmung</u> freigegeben
Klassifizierung:	öffentlich <u>Entwurf</u>
Referenzierung:	gemSpec_ZETA

---

## Dokumentinformationen

---

### Änderungen zur Vorversion

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

### Dokumentenhistorie

Version	Stand	Kap./ Seite	Grund der Änderung, besondere Hinweise	Bearbeitung
1.0.0	28.02.2025		initiale Erstellung	gematik
1.1.0	06.08.2025		Einarbeitung ZETA_25_2	gematik
<u>1.2.0 CC</u>	<u>15.10.2025</u>		<u>Einarbeitung ZETA 25 3</u>	<u>gematik</u>

---

## Inhaltsverzeichnis

---

<b>1 Einordnung des Dokuments .....</b>	<b>8</b>
<b>1.1 Zielsetzung .....</b>	<b>8</b>
<b>1.2 Zielgruppe .....</b>	<b>8</b>
<b>1.3 Abgrenzungen .....</b>	<b>8</b>
<b>1.4 Methodik .....</b>	<b>9</b>
1.4.1 Anforderungen .....	9
<b>2 Features und Epics .....</b>	<b>10</b>
<b>2.1 Clientregistrierung .....</b>	<b>10</b>
2.1.1 Wiedererkennung bekannter Clients .....	10
2.1.2 Device Security Rating .....	10
<b>2.2 Policy Enforcement .....</b>	<b>10</b>
2.2.1 Zugriffsschutz .....	10
2.2.2 HTTP Proxy .....	11
<b>2.3 Decide from Policies .....</b>	<b>11</b>
2.3.1 Maschinenlesbare Zugriffsregeln .....	11
2.3.2 Ein reproduzierbares Ja/Nein .....	11
2.3.3 Policies nach Betroffenheit .....	11
<b>2.4 Policy Information und Administration .....</b>	<b>11</b>
2.4.1 Policy Verwaltung .....	11
2.4.2 Monitoring .....	12
<b>2.5 Client Authorization .....</b>	<b>12</b>
2.5.1 Autorisierung auf Basis von Policy Entscheidungen .....	12
2.5.2 Client Authentication .....	12
<b>3 Einordnung in die TI 2.0 .....</b>	<b>13</b>
<b>4 Technisches Konzept .....</b>	<b>17</b>
<b>4.1 ZETA Guard .....</b>	<b>17</b>
<b>4.2 Policy Enforcement Point (PEP) .....</b>	<b>17</b>
<b>4.3 Policy Decision Point (PDP) .....</b>	<b>18</b>
<b>4.4 ZETA Client .....</b>	<b>18</b>
<b>4.5 Policy Information und Administration .....</b>	<b>19</b>
4.5.1 Policy Information Point (PIP) .....	19
4.5.2 Policy Administration Point (PAP) .....	19
<b>4.6 Clientregistrierung .....</b>	<b>20</b>
<b>4.7 Monitoring .....</b>	<b>24</b>
4.7.1 Security Information and Event Management (SIEM): .....	24
4.7.2 Shared Signals .....	24
4.7.3 Telemetrie, Monitoring und Logging .....	25
<b>4.8 Zusammenspiel mit Identity Provider .....</b>	<b>25</b>

<b>4.9 TI 2.0 Dienst Backend .....</b>	<b>26</b>
<b>5 Spezifikation .....</b>	<b>27</b>
<b>5.1 Übergreifende Anforderungen für Datenschutz und Sicherheit .....</b>	<b>27</b>
5.1.1 Sicherheits- und Datenschutzanforderungen an Logging und Monitoring .....	29
5.1.2 Sicherheits- und Datenschutz Anforderungen an das Security Monitoring .....	30
5.1.3 Sicherheits- und Datenschutz Anforderungen an die Verarbeitung von Daten mit dem Schutzbedarf "sehr hoch" .....	32
5.1.4 Sicherheits- und Datenschutz Anforderungen an dem ZETA Client in FdVs .....	34
<b>5.2 Clientsystem und ZETA Client .....</b>	<b>34</b>
5.2.1 Hersteller .....	35
5.2.2 Verbindungsaufbau .....	35
5.2.3 Clientregistrierung .....	37
5.2.4 Nutzerauthentifizierung .....	39
5.2.5 Session Management .....	39
5.2.6 Liste der HTTP Statuscodes .....	40
5.2.7 ZETA Attestation Service .....	42
<b>5.3 ZETA Guard .....</b>	<b>43</b>
<b>5.4 Policy Enforcement Points .....</b>	<b>48</b>
5.4.1 PEP HTTP Proxy .....	48
5.4.2 Sicherheits- und Datenschutz Anforderungen an den PEP .....	51
<b>5.5 Policy Decision Point .....</b>	<b>52</b>
5.5.1 Policy Engine .....	52
5.5.2 PDP Authorization Server .....	55
5.5.2.1 PDP Relying Party .....	60
5.5.2.2 Ablauf für den Zugriff auf einen Resource Server .....	60
5.5.2.3 Service Discovery .....	61
5.5.2.4 Client Registrierung für stationäre Clients .....	63
5.5.2.5 Authentifizierung und Autorisierung für stationäre Clients .....	65
5.5.2.5.1 Pfad A: Token Austausch mit Attestierung .....	67
5.5.2.5.2 Pfad B: Token Erneuerung via Refresh Token .....	68
5.5.2.5.3 Gemeinsame nachfolgende Schritte .....	68
5.5.2.6 Ablauf der Authentifizierung bei Dienst zu Dienst Kommunikation .....	69
5.5.3 PDP Datenbank .....	70
5.5.4 Sicherheits- und Datenschutzanforderungen an den PDP .....	72
5.5.5 Konfiguration .....	73
<b>5.6 PIP und PAP Service .....</b>	<b>74</b>
<b>5.7 Telemetrie Daten Service .....</b>	<b>76</b>
<b>5.8 Betrieb .....</b>	<b>78</b>
5.8.1 Anforderungen an Hersteller einer ZETA Komponente .....	78
5.8.2 Anforderungen an Hersteller eines TI2.0 Dienstes .....	79
5.8.3 Anforderungen an Anbieter eines TI2.0 Dienstes .....	79
5.8.4 Anforderungen für nahtlose Aktualisierungen .....	80
5.8.5 Anforderungen für Steuerung durch Feature Flags .....	80
5.8.6 Anforderungen zur Überwachung des Betriebsstatus .....	81
5.8.7 Leistungs Anforderungen .....	82
5.8.8 Betriebliche Schnittstellendefinition .....	82
<b>5.9 Anforderungen an Dienste der TI .....</b>	<b>84</b>

<del>5.10 Anforderungen an den Test der Zero-Trust-Komponenten .....</del>	<del>85</del>
<del>5.11 Sicherheitsleistungen des ZETA-Guard für Resource-Server .....</del>	<del>85</del>
<del>5.12 Weitere Leistungen des ZETA-Guard für Resource-Server .....</del>	<del>86</del>
<del>6 Beispiele und Referenzimplementierungen .....</del>	<del>88</del>
<del>7 Anhang A – Verzeichnisse .....</del>	<del>89</del>
<del>7.1 Abkürzungen .....</del>	<del>89</del>
<del>7.2 Glossar .....</del>	<del>91</del>
<del>7.3 Abbildungsverzeichnis .....</del>	<del>93</del>
<del>7.4 Tabellenverzeichnis .....</del>	<del>93</del>
<del>7.5 Referenzierte Dokumente .....</del>	<del>95</del>
<del>7.5.1 Dokumente der gematik .....</del>	<del>95</del>
<del>7.5.2 Weitere Referenzen .....</del>	<del>97</del>
<b>1 Einordnung des Dokuments .....</b>	<b>8</b>
<b>1.1 Zielsetzung .....</b>	<b>8</b>
<b>1.2 Zielgruppe .....</b>	<b>8</b>
<b>1.3 Abgrenzungen .....</b>	<b>8</b>
<b>1.4 Methodik .....</b>	<b>9</b>
1.4.1 Anforderungen .....	9
<b>2 Features und Epics .....</b>	<b>10</b>
<b>2.1 Clientregistrierung .....</b>	<b>10</b>
2.1.1 Wiedererkennung bekannter Clients .....	10
2.1.2 Device Security Rating .....	10
<b>2.2 Policy Enforcement .....</b>	<b>10</b>
2.2.1 Zugriffsschutz .....	10
2.2.2 HTTP Proxy .....	11
<b>2.3 Decide from Policies .....</b>	<b>11</b>
2.3.1 Maschinenlesbare Zugriffsregeln .....	11
2.3.2 Ein reproduzierbares Ja/Nein .....	11
2.3.3 Policies nach Betroffenheit .....	11
<b>2.4 Policy-Information und -Administration .....</b>	<b>11</b>
2.4.1 Policy-Verwaltung .....	11
2.4.2 Monitoring .....	12
<b>2.5 Client Authorization .....</b>	<b>12</b>
2.5.1 Autorisierung auf Basis von Policy-Entscheidungen .....	12
2.5.2 Client Authentication .....	12
<b>3 Einordnung in die TI 2.0 .....</b>	<b>13</b>
<b>4 Technisches Konzept .....</b>	<b>17</b>
<b>4.1 ZETA Guard .....</b>	<b>17</b>

<b>4.2 Policy Enforcement Point (PEP)</b>	<b>17</b>
<b>4.3 Policy Decision Point (PDP)</b>	<b>18</b>
<b>4.4 ZETA Client</b>	<b>18</b>
<b>4.5 Policy-Information und -Administration</b>	<b>19</b>
4.5.1 Policy Information Point (PIP)	19
4.5.2 Policy Administration Point (PAP)	19
4.5.3 PIP und PAP Ausprägung in ZETA	20
<b>4.6 Clientregistrierung</b>	<b>20</b>
<b>4.7 Monitoring</b>	<b>24</b>
4.7.1 Security Information and Event Management (SIEM):	24
4.7.2 Shared Signals	24
4.7.3 Telemetrie, Monitoring und Logging	25
<b>4.8 Zusammenspiel mit Identity Provider</b>	<b>25</b>
<b>4.9 TI 2.0 Dienst-Backend</b>	<b>26</b>
<b>5 Spezifikation</b>	<b>27</b>
<b>5.1 Übergreifende Anforderungen für Datenschutz und Sicherheit</b>	<b>27</b>
5.1.1 Sicherheits- und Datenschutzanforderungen an Logging und Monitoring	29
5.1.2 Sicherheits- und Datenschutz-Anforderungen an das Security Monitoring	30
5.1.3 Sicherheits- und Datenschutz-Anforderungen an die Verarbeitung von Daten mit dem Schutzbedarf "sehr hoch"	32
5.1.4 Sicherheits- und Datenschutz Anforderungen an dem ZETA Client in FdVs	34
<b>5.2 Clientsystem und ZETA Client</b>	<b>34</b>
5.2.1 Hersteller	35
5.2.2 Verbindungsaufbau	35
5.2.3 Clientregistrierung	37
5.2.4 Nutzerauthentifizierung	39
5.2.5 Session Management	39
5.2.6 Liste der HTTP-Statuscodes	40
5.2.7 ZETA Attestation Service	42
<b>5.3 ZETA Guard</b>	<b>43</b>
5.3.1 Optionale Komponenten	46
5.3.2 Kommunikation mit Diensten der gematik	46
5.3.3 Deployment Szenarien	47
5.3.3.1 Geo-Redundanz	47
5.3.4 Laufzeitüberwachung	47
<b>5.4 Policy Enforcement Points</b>	<b>48</b>
5.4.1 PEP HTTP Proxy	48
5.4.2 Sicherheits- und Datenschutz-Anforderungen an den PEP	51
<b>5.5 Policy Decision Point</b>	<b>52</b>
5.5.1 Policy Engine	52
5.5.2 PDP Authorization Server	55
5.5.2.1 PDP Relying Party	60
5.5.2.2 Ablauf für den Zugriff auf einen Resource Server	60
5.5.2.3 Service Discovery	61
5.5.2.4 Client-Registrierung für stationäre Clients	63
5.5.2.5 Authentifizierung und Autorisierung für stationäre Clients	65
5.5.2.5.1 Pfad A: Token-Austausch mit Attestierung	67

5.5.2.5.2 Pfad B: Token-Erneuerung via Refresh Token .....	68
5.5.2.5.3 Gemeinsame nachfolgende Schritte .....	68
5.5.2.6 Ablauf der Authentifizierung bei Dienst-zu-Dienst Kommunikation .....	69
5.5.3 PDP Datenbank .....	70
5.5.4 Sicherheits- und Datenschutzanforderungen an den PDP .....	72
<b>5.6 PIP und PAP Service .....</b>	<b>73</b>
<b>5.7 Telemetrie-Daten Service .....</b>	<b>76</b>
<b>5.8 Betrieb .....</b>	<b>78</b>
5.8.1 Anforderungen an Hersteller einer ZETA Komponente .....	78
5.8.2 Anforderungen an Hersteller eines TI2.0 Dienstes .....	79
5.8.3 Anforderungen an Anbieter eines TI2.0 Dienstes .....	79
5.8.4 Anforderungen für nahtlose Aktualisierungen .....	80
5.8.5 Anforderungen zur Überwachung des Betriebsstatus .....	81
5.8.6 Leistungs-Anforderungen .....	82
5.8.7 Betriebliche Schnittstellendefinition .....	82
5.8.8 Prozesse zur Inbetriebnahme eines ZETA Guard .....	84
<b>5.9 Anforderungen an Dienste der TI .....</b>	<b>84</b>
<b>5.10 Anforderungen an den Test der Zero Trust-Komponenten .....</b>	<b>85</b>
<b>5.11 Sicherheitsleistungen des ZETA Guard für Resource Server .....</b>	<b>85</b>
<b>5.12 Weitere Leistungen des ZETA Guard für Resource Server .....</b>	<b>86</b>
<b>6 Beispiele und Referenzimplementierungen .....</b>	<b>88</b>
<b>7 Anhang A – Verzeichnisse .....</b>	<b>89</b>
7.1 Abkürzungen .....	89
7.2 Glossar .....	91
7.3 Abbildungsverzeichnis .....	93
7.4 Tabellenverzeichnis .....	93
<b>7.5 Referenzierte Dokumente .....</b>	<b>95</b>
7.5.1 Dokumente der gematik .....	95
7.5.2 Weitere Referenzen .....	97

---

## 1 Einordnung des Dokuments

---

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

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

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

### 1.1 Zielsetzung

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

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

### 1.2 Zielgruppe

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

### 1.3 Abgrenzungen

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



## 1.4 Methodik

### 1.4.1 Anforderungen

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

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

Anforderungen werden im Dokument wie folgt dargestellt:

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

Text / Beschreibung

[<=]

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

---

## 2 Features und Epics

---

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

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

### 2.1 Clientregistrierung

Gemäß des Zero Trust-Ansatzes ist jeder Schnittstellenaufruf potentiell gefährlich, soweit nicht anders festgestellt. Dazu zählt auch das Vertrauen in bekannte bzw. Misstrauen in unbekannte Clients. Um Clients wiedererkennbar zu machen, muss eine Registrierung dieser erfolgen. Sind in der Registrierung zusätzliche Sicherheitsmerkmale über das Client und den Aufrufkontext feststellbar, stärken diese das Vertrauen in nachfolgenden Aufrufen fachlicher Schnittstellen.

#### 2.1.1 Wiedererkennung bekannter Clients

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

#### 2.1.2 Device Security Rating

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

### 2.2 Policy Enforcement

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

#### 2.2.1 Zugriffsschutz

Das Policy Enforcement soll als eine Art Gatekeeper bzw. Türsteher den Zugriff auf Schnittstellen von Backendservices durch beliebige Clients durchsetzen. Grundlage ist

das Vertrauen in eine Policy-Entscheidung durch eine Komponente zur Auswertung eines Regelwerks.

### 2.2.2 HTTP Proxy

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

## 2.3 Decide from Policies

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

### 2.3.1 Maschinenlesbare Zugriffsregeln

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

### 2.3.2 Ein reproduzierbares Ja/Nein

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

### 2.3.3 Policies nach Betroffenheit

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

## 2.4 Policy-Information und -Administration

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

### 2.4.1 Policy-Verwaltung

Eine Policy-Entscheidung kann Eingangsinformation für andere Policies sein, ebenso kann das Ändern von Rahmenbedingungen oder eine Anomalieerkennung zur Beeinflussung von Policies führen. Aus diesem Grund führen Beobachtungen über Policy-Entscheidungen zu Informationen über das Gesamtsystem, die als Eingangsdaten für

nachfolgende Policy-Entscheidungen herangezogen werden. Daneben ist es erforderlich, Anpassungen am Regelwerk dem System über authentizitäts- und integritätsgeschützte Wege bekannt zu machen.

### 2.4.2 Monitoring

Durch ein Monitoring von Betriebsparametern und Telemetrie-Daten wird die Durchsetzung von Policies sowie die Auswirkung möglicher Policy-Änderungen transparent.

## 2.5 Client Authorization

Menschen benutzen Clients (Kombination aus Gerät und App). Jeder Zugriff auf Daten oder Schnittstellen wird auf eine menschliche Interaktion (Authentisierung) zurückgeführt. Nach Stand der Technik erfolgt die sichere Authentifizierung meist über 2 Faktoren. Zur Wiedererkennung und sicheren Identifikation werden Menschen und Clients Authentifizierungsmerkmale ausgestellt. Die sichere Identifikation und Authentifizierung ist eine wichtige Eingangsgröße für Zugriffsentscheidungen (s. o.).

Für die Dienst zu Dienst Kommunikation ist es ebenso erforderlich den anfragenden Dienst zu identifizieren, um eine Zugriffsentscheidung treffen zu können.

### 2.5.1 Autorisierung auf Basis von Policy-Entscheidungen

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

### 2.5.2 Client Authentication

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

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

### 3 Einordnung in die TI 2.0

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

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

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

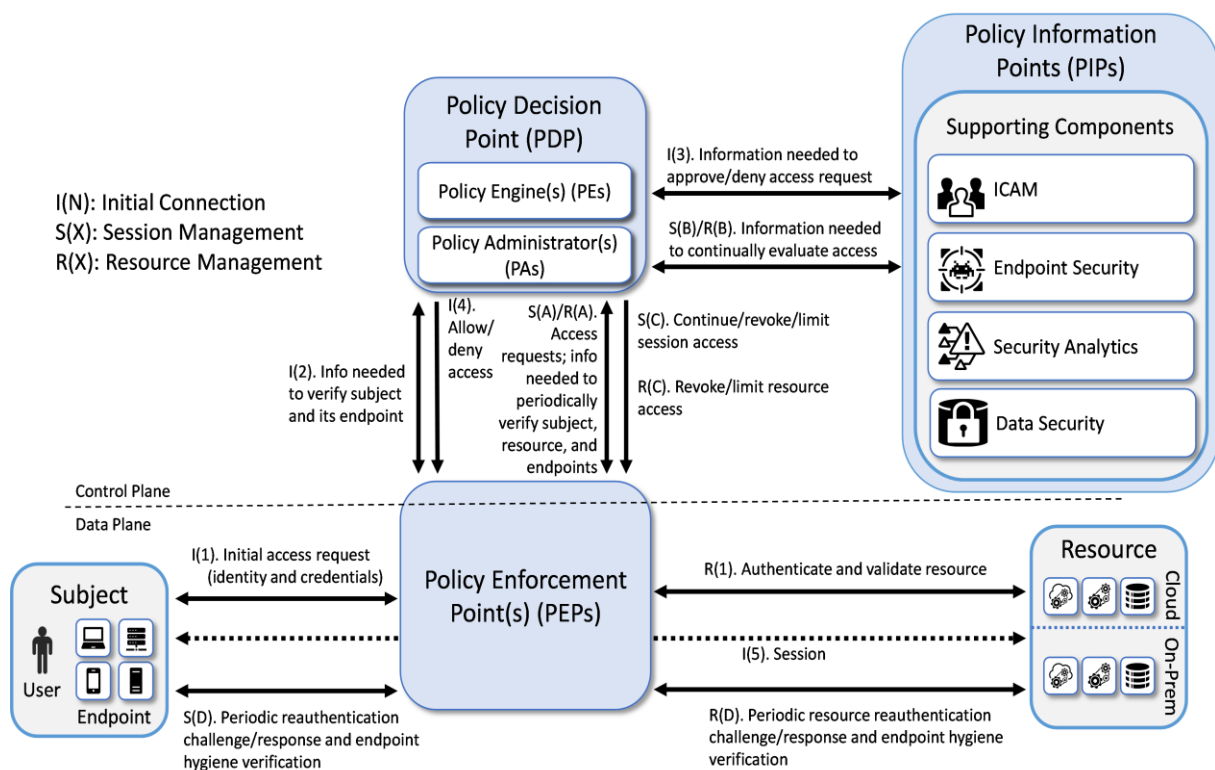
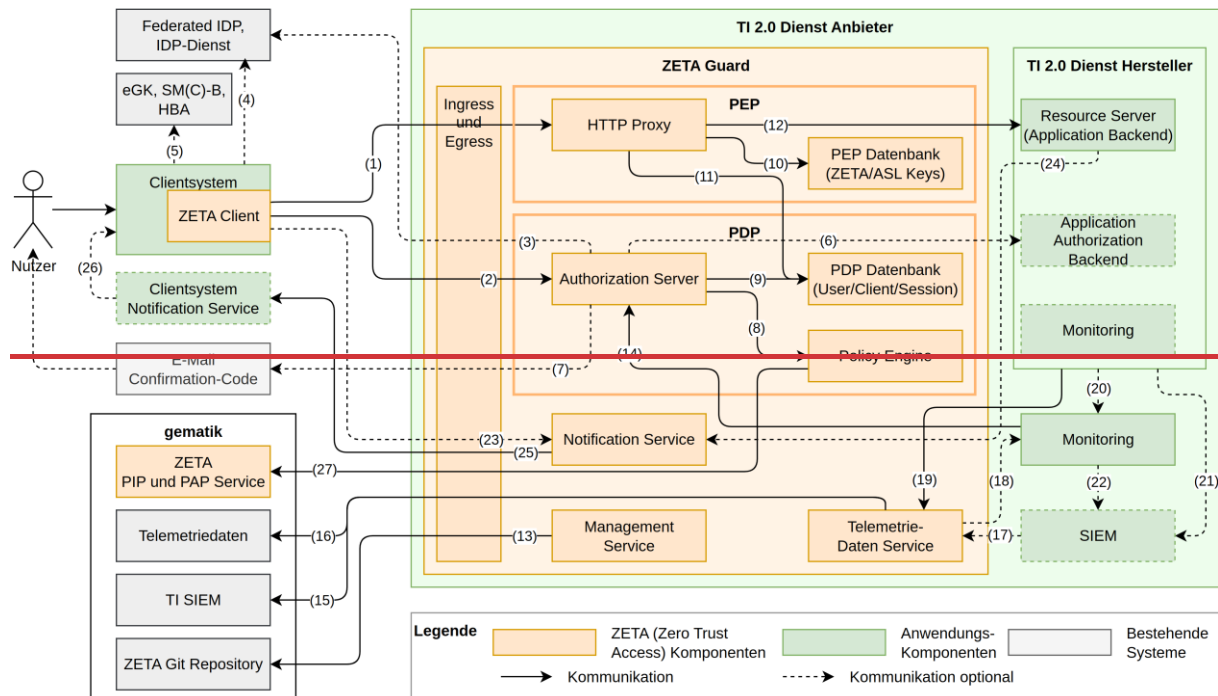


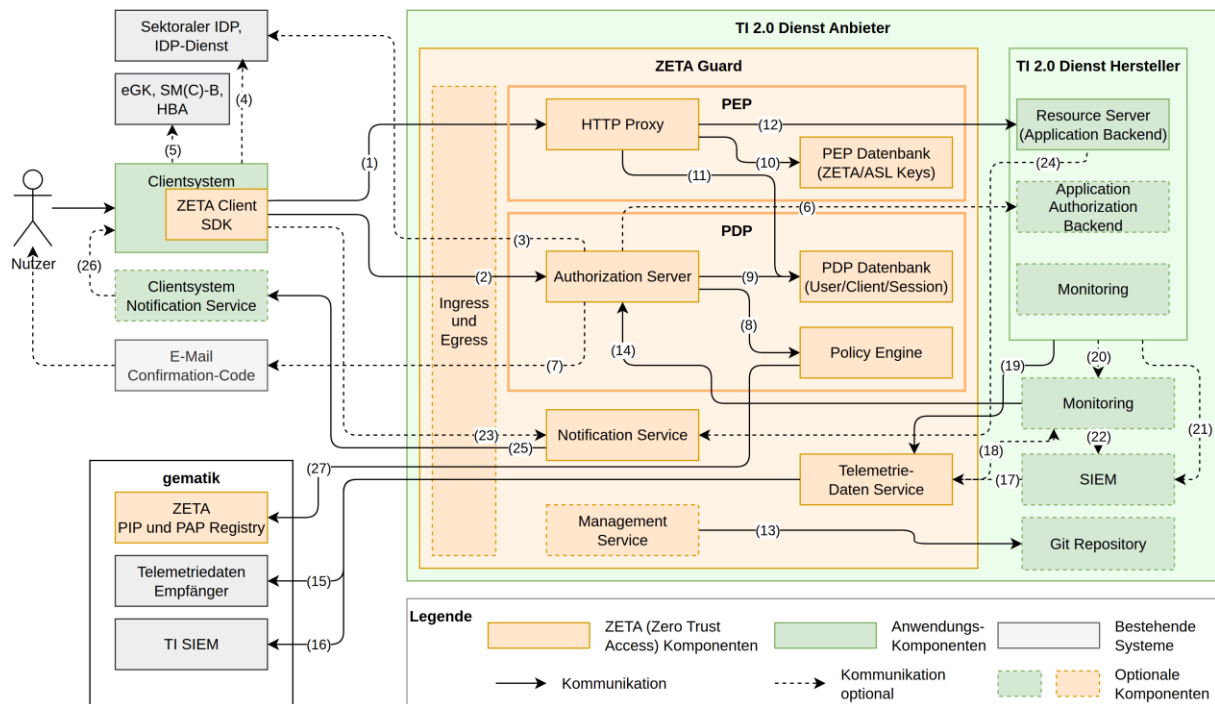
Abbildung 1: NIST Zero Trust-Referenzarchitektur, Quelle [NIST\_SP1800-35\_FIG1]

Im Architekturkonzept der TI 1.0 werden konkrete Umgebungsannahmen zu Consumer Zonen, Secure Consumer Zonen, Plattformzonen, Personal Zonen usw. getroffen, in denen kein (Personal Zone) bzw. ein gewisses Sicherheitsniveau (überall sonst) axiomatisch angenommen wird. Das Zero Trust-Konzept löst sich von der Aufteilung in verschiedene Zonen, insbesondere, da ~~weniger (teilweise gar keine mehr)~~ TI-Plattform-Produkttypen in der Kommunikation zwischen ~~den Datenaustauschen unter~~ Clients mit

Backendservices Diensten involviert werden. Im Folgenden ist eine generische Produkttypzerlegung für die Umsetzung der NIST Zero Trust-Referenzarchitektur einer TI2.0 Fachanwendung dargestellt. Die Zero Trust Ausprägung der Telematikinfrastruktur wird Zero Trust Access (ZETA) genannt.

In diesem Pattern greift ein Nutzer über ein Clientsystem inkl. ZETA Client auf Daten eines TI 2.0 Dienstes zu. Ein TI 2.0 Dienst setzt sich zusammen aus einem ZETA Guard und mindestens einem Resource Server. Der ZETA-Guard wird als eingebettetes Modul engmaschig mit dem Produkttypen verbunden und vom Anbieter des TI2.0 Dienstes zwingend mit betrieben. Der Anbieter implementiert zusätzliche Komponenten, um seine Infrastruktur vor Angriffen aus dem Internet zu schützen und um die Zugriffe zu optimieren (z. B. Content Delivery Network, eigener Ingress und Load Balancer, Web Application Firewall). Das folgende Bild zeigt eine Übersicht der beteiligten Komponenten in der Vernetzung zwischen einem Clientsystem (links grün) und einem Backendservice (rechts grün: Resource Server).





**Abbildung 2: Abbildung Zero Trust-Architektur der TI 2.0**

Die obige Abbildung zeigt die Einbettung von ZETA bezogenen, logischen Komponenten (orange) in die Aufrufkette zwischen einem Clientsystem und einem Resource Server (grün). Dargestellt sind zusätzlich heute bereits vorhandene und genutzte Komponenten und Dienste, die für die Nutzerauthentifizierung (z. B. eGK und IDP) bzw. die Betriebsüberwachung (z. B. mittels gematik Telemetriedaten Empfänger) in Anwendungsfällen der TI 2.0 weitergenutzt werden können (grau). In diese Abbildung sind diverse Architekturentscheidungen eingearbeitet, die im Kapitel 4 und 5 erläutert bzw. spezifiziert werden.

## Kurzbeschreibung der Komponenten und Schnittstellen

(1): Der HTTP Proxy erlaubt den Zugriff auf Daten des Resource Servers, wenn ein gültiges Access Token im Authorization Header enthalten ist. Der ZETA Client muss am HTTP Proxy das OAuth Protected Resource Well-known JSON Document abfragen, um herauszufinden, von welchem Authorization Server die Access Token für diesen Resource Server ausgestellt werden können.

(2): Um ein Access Token vom Authorization Server zu erhalten, ist eine Registrierung des ZETA Clients und eine Authentifizierung des Nutzers erforderlich. Die Nutzerauthentifizierung erfolgt per OIDC (4) mit Pushed Authorization Request (3) und OAuth Authorization Code Flow (2) oder per OAuth Token Exchange (5 und 2) mit vorheriger Client-Registrierung am Authorization Server.

(6): Es ist vorgesehen, dass Fachanwendungen eine zusätzliche Autorisierung mittels der Komponente Application Authorization Backend nutzen können.

(7): Wenn die Authentisierung per OIDC erfolgt, dann ist im Rahmen der ersten Client-Registrierung eine E-Mail mit einem Confirmation Code erforderlich, mit dem der Nutzer in dem Client bestätigt, dass es sich um seinen Client handelt: (Trust On First Use). Die



E-Mail Confirmation ist nur einmalig notwendig, falls die App mehrere TI 2.0 Resource Server nutzt.

~~(27) Die Policy Engine bezieht die aktuellen Policies und Daten vom ZETA PIP und PAP Service.~~

(8): Der Authorization Server stellt nur ein Access Token aus, wenn die Policy Engine durch Anwendung der Policies die Erlaubnis gegeben hat. Für die Entscheidung der Erlaubnis werden die Nutzer-Daten und Client-Daten der aktuellen Session (Input) anhand von Policies mit den vorgegebenen Daten der Policy Engine verglichen.

(27) Die Policy Engine bezieht die aktuellen Policies und Daten aus der ZETA PIP und PAP Registry als OCI Container.

(9): Die Session-, User- und Client-Daten werden vom Authorization Server in der PDP Datenbank gespeichert. Der Authorization Server ist daher grundsätzlich stateless.

(10): Der HTTP Proxy kann optional eine zusätzliche Verschlüsselung der Daten per ZETA/ASL terminieren. Die Schlüssel verwaltet der HTTP Proxy in der PEP Datenbank. Dadurch ist der HTTP Proxy stateless.

(11): Der HTTP Proxy fragt Nutzer- und optional Client-Daten anhand einesdes client\_id Claims aus dem Access Token aus der PDP Datenbank ab, um diese Daten per Custom Header im Request (1) dem Resource Server zu übergeben (12).

(13): Der optionale Management Service überwacht die Konfiguration der ZETA Guard-Komponenten und setzt durch, dass die im Git Repository der ZETA Guard Instanz gespeicherte Konfiguration ausgeführt wird. Dadurch wird ein Configuration-Drift unterbunden.

(14): Mittels Shared Signals kann das Monitoring (oder eine andere Komponente des TI 2.0 Dienst-Anbieters) sicherheitsrelevante Informationen von den ZETA-Guard Komponenten erhalten und ebenfalls mittels Shared Signals darauf reagieren, indem z. B. das Access und das Refresh Token eines Nutzers gesperrt werden und somit eine neue Authentifizierung erzwungen wird.

(15) und (16): Der Telemetrie-Daten Service sammelt Telemetrie-Daten von den ZETA Guard-Komponenten ein und bereitet sie für den Versand an die gematik Telemetriedaten Schnittstelle und das TI SIEM der gematik auf. Es wird das OTLP Protokoll über gRPC verwendet.

(17, 18, 19): Der Telemetrie-Daten Service empfängt die ~~Selbstauskunft-Daten-(und nach Bedarf weitere)~~ Selbstauskunfts- und Tracing-Daten vom TI 2.0 Dienst und kann Daten vom SIEM und Monitoring des TI 2.0 Dienst-Anbieters entgegennehmen und an die gematik Telemetriedaten Schnittstelle sowie das TI SIEM der gematik weiterleiten. Ebenso kann der Telemetrie-Daten Service Monitoring-Daten der ZETA Guard-Komponenten an das Monitoring des TI 2.0 Dienst-Anbieters senden. Zum Einsatz kommt OpenTelemetry. Alle von OpenTelemetry unterstützten Protokolle können vom Anbieter durch Konfiguration des Telemetrie-Daten Service verwendet werden, um Daten an den Telemetrie-Daten Service zu senden und vom Telemetrie-Daten Service zu empfangen. Empfohlen wird das native OTLP Protokoll über gRPC.

(20), (21) und (22): Die Komponenten des TI 2.0 Dienstes werden im Monitoring überwacht und es werden Daten für das SIEM erzeugt.

(23): Das Clientsystem kann über den ZETA Client eine Push-Konfiguration im ZETA Guard Notification Service einrichten. Wenn Ereignisse eintreten, für die der Resource Server oder der ZETA Guard eine Push Nachricht für den Benutzer generiert (24), dann werden entsprechend der vorhandenen Push-Konfigurationen des Nutzers im Notification Service, Benachrichtigungen an den Client gesendet (25 und 26).



---

## 4 Technisches Konzept

---

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

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

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

### 4.1 ZETA Guard

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

### 4.2 Policy Enforcement Point (PEP)

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

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

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

### 4.3 Policy Decision Point (PDP)

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

- Der PDP fungiert als OAuth2 Authorization Server und verwaltet die Autorisierung von Benutzeranfragen auf geschützte Ressourcen. Zudem überwacht der Authorization Server die Benutzersessions, um sicherzustellen, dass sie gültig sind und den Sicherheitsrichtlinien entsprechen. Der Authorization Server ist als vertrauenswürdige Relying Party im föderierten Identitätsmanagement registriert. Dadurch kann der Authorization Server Identitätsinformationen von Benutzern sicher und vertrauenswürdig beziehen und bei Bedarf eine (erneute) Nutzerauthentifizierung an die IDPs delegieren. **Dadurch stellt** Der Authorization Server **stellt** sicher, dass nur authentifizierte Benutzer **mit registrierten Clients** Zugriff auf die geschützten Ressourcen erhalten.
- Der PDP ermöglicht am Authorization Server die dynamische Registrierung von Clients, die auf geschützte Ressourcen zugreifen möchten. Dies umfasst auch die Offband-Bestätigung, bei der zusätzliche Sicherheitsmechanismen (Verifikation via E-Mail ~~oder SMS~~) verwendet werden, um die Identität und Integrität (plattformabhängig) der registrierten Clients zu überprüfen.
- Der PDP analysiert und interpretiert in der Policy Engine die Sicherheitsrichtlinien, die im Rahmen des Zero Trust-Modells definiert sind. Diese Policies können Kriterien wie Benutzeridentität, Gerätetyp, Standort, Zeitpunkt der Anfrage und andere Kontextinformationen ("Signale") enthalten, die relevant für die Zugriffsentscheidung sind. Basierend auf der Interpretation der Policies trifft die Policy Engine Entscheidungen darüber, ob eine Zugriffsanfrage auf eine bestimmte Ressource genehmigt oder abgelehnt wird. Diese Entscheidungen erfolgen auf Plattformebene, was bedeutet, dass die Policy Engine die Zugriffsanfragen im Kontext der gesamten Plattform oder des Netzwerks bewertet, und nicht isoliert betrachtet. Die Zugriffsentscheidung resultiert dann in der Ausstellung eines Access Tokens, das für den konkret angefragten Zugriff verwendet wird (siehe Policy Enforcement).
- Die Policy Engine verwendet dabei die Informationen, die **ihr übermittelt** **werdensie von der ZETA PIP und PAP Artifact Registry bezieht und die Daten der Zugriffsanfrage vom Authorization Server**, um die Zugriffsentscheidung zu treffen. Dazu gehören nicht nur die Policies selbst, sondern auch Echtzeitinformationen über den Zustand von Benutzeridentitäten, Clients und andere Kontextinformationen, die für die Bewertung der Zugriffsanfrage relevant sind.

Am PDP werden Betriebsdaten erhoben, verarbeitet und dem Telemetrie-Daten Service im ZETA Guard zur Verfügung gestellt.

### 4.4 ZETA Client

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

Ein ZETA Client im Zero Trust-Modell wird nicht als vertrauenswürdig angesehen, sondern muss - genauso wie alle Komponenten im Netzwerk - kontinuierlich

authentifiziert und autorisiert werden. Die Zugriffsentscheidungen werden basierend auf aktuellen Richtlinien, Kontextinformationen, Bedrohungsinformationen und insbesondere in Kenntnis des diesen Client benutzenden Benutzers getroffen.

Die Aufgaben des ZETA Clients sind:

- Erzeugung, sichere Speicherung und Prüfung der kryptographischen App/Geräte Identität
- Erzeugung der App/Gerät-Attestierung und Ermittlung und Übertragung der Eigenschaften der Laufzeitumgebung (Betriebssystem, Betriebssystem Version, etc.)
- Implementierung des OAuth Flows
- Management der Sessions inkl. Verwaltung der Access und Refresh Token.
- Management der Clientregistrierungen

### 4.5 Policy-Information und -Administration

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

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

#### 4.5.1 Policy Information Point (PIP)

Der PIP ist für die Bereitstellung von Informationen über Sicherheitsrichtlinien zuständig. Er dient als zentraler Informationsdienst, der anderen Systemen und Komponenten im Zero Trust-Netzwerk Zugriff auf aktuelle Sicherheitsrichtlinien ermöglicht. Der PIP kann Attribute wie Benutzerrollen, Zugriffsrechte, ~~Clientezustände~~Clientzustände und andere Kontextinformationen bereitstellen, die von anderen Komponenten für die Zugriffsentscheidung benötigt werden. Der PIP kann Daten aus verschiedenen Quellen beziehen, einschließlich einer zentralen Richtliniendatenbank, externen Identitätsanbietern, Sicherheitsinformationen von Clients und anderen Quellen.

#### 4.5.2 Policy Administration Point (PAP)

Der PAP ist für die Verwaltung und Konfiguration von Sicherheitsrichtlinien verantwortlich. Er bietet eine Schnittstelle oder eine Konsole, über die Richtlinien in hoheitlicher Verantwortung definiert, geändert und gelöscht werden können. Policy-Administratoren können im PAP Zugriffsregeln, Autorisierungsniveaus, Bedrohungsabwehrmaßnahmen und andere Sicherheitsrichtlinien festlegen. Der PAP ermöglicht es Policy-Administratoren, Richtlinien - basierend auf verschiedenen Kriterien wie Benutzerrollen, Gruppenzugehörigkeit, Standorten und Clientattributen - zu differenzieren. Änderungen an den Sicherheitsrichtlinien, die im PAP vorgenommen werden, werden an den PIP weitergegeben, damit andere Komponenten im Zero Trust-Netzwerk auf die aktualisierten Richtlinien zugreifen können. Das Vertrauen in

bereitgestellte und angepasste Policies wird über Signaturen für die Sicherstellung von Integrität und Authentizität jeder Policy sichergestellt.

### 4.5.3 PIP und PAP Ausprägung in ZETA

In ZETA Guard wird als Policy Engine der Open Policy Agent (OPA) verwendet. OPA bezieht Policies (PAP) und Daten (Wertebereiche für die Policies; PIP) zusammengefasst (OPA Bundle) über einen OCI Container aus der ZETA PIP und PAP Artifact Registry der gematik. Die einzelnen Dateien im OPA Bundle und der OCI Container sind signiert mit einer Identität der gematik.

Dadurch reduziert sich die Bereitstellung der PIP und PAP Daten zu einem Download in einer OCI Registry. Die Anforderungen an den PIP und PAP Service (siehe 5.6- PIP und PAP Service) beziehen sich daher auf den CI/CD-Prozess zur Entwicklung der PIP und PAP Daten.

## 4.6 Clientregistrierung

Alle Clients, die mit Diensten der TI2.0 kommunizieren, sind zur Laufzeit bekannt. Mit einer Attestierung in Abhängigkeit der verfügbaren Mechanismen der Laufzeitumgebung (Client-Features, Betriebssystem) kann ein Vertrauen und eine Wiedererkennung von Clients aufgebaut werden.

Hersteller von Clients, die mit ZETA Guard geschützten Diensten kommunizieren, müssen ihre Clients bei der gematik registrieren. Nicht registrierte Clients erhalten keinen Zugriff.

Die folgenden statischen Client-Eigenschaften werden im Rahmen der Bereitstellung von Clientanwendungen dynamischen Clientregistrierung am ZETA Guard erfasst und sind unabhängig von der Nutzung durch einen werden einem konkreten Benutzer zugeordnet.

**Tabelle 1: Statische Eigenschaften Clientsysteme auf Hersteller-/Anbiiterebene**

Client Eigenschaft	Beschreibung
Produkt-Id	Eindeutige ID der Client-Software, vergeben durch gematik
Produkt-Name	Produkt-Name, vergeben durch den Hersteller
Hersteller-Id	Kennung des Herstellers aus TI-ITSM
Hersteller-Name	Name des Herstellers aus TI-ITSM
Produkt-Plattform	<p>Zunächst werden zwei Plattformen gesondert behandelt: Android und Apple (iOS, macOS etc.). Diese beiden Plattformen bieten Mechanismen für die Attestation der Client-Instanzen und der Umgebung, in welcher diese Clients ausgeführt werden.</p> <p>Alle andere Clients werden zunächst als generische Software-Produkte eingestuft.</p>

Client Eigenschaft	Beschreibung
Produkt-Plattform-Id	Plattformspezifisch eindeutige Kennung der Client-Software  Android: Package-Name und Signer-Zertifikat Fingerprint Apple: Bundle-ID und Apple-ID Software: Registriert durch gematik , analog zu ClientIds in E-Rezept
Attestation-Methode	Die Methode, nach welcher die Client-Software und die Ablaufumgebung attestiert werden kann.  Zunächst werden Android und Apple (iOS) unterstützt, weil diese Plattformen entsprechende Mechanismen zur Remote-Attestation anbieten.  Windows und Linux verwenden zur Attestation der Clients TPM 2.0. Apple (MacOS) verwendet die Secure Enclave.

Der Nutzer muss sich ~~vor der Clientregistrierung~~ mit einer TI-Identität authentifizieren, z. B. GesundheitsID oder SM(C)-B. Bei der Client-Registrierung werden die statischen Eigenschaften eines ~~Client-Systems~~ Clientsystems für jede Client-Instanz mit einem vom Client-Nutzer signierten Softwarestatement bekannt gemacht. Der Client erzeugt dabei eine kryptographische Identität (DPoPClient Instance Key-[RFC9449]-Pair), die Serverseitig an den Nutzer und Client gebunden wird und die für die Client-Authentifizierung am Authorization Server verwendet wird (Client Assertion JWT).

~~Bei der SM(C)-B-Authentisierung mit DPoP erfolgt die Clientregistrierung implizit (wenn die SM(C)-B-freigeschaltet ist). Das vom ZETA-Client automatisch erstellte Client Assertion JWT enthält die Clientregistrierungsdaten und wird per SM(C)-B signiert.~~

Zur Laufzeit werden die Client-Eigenschaften durch Client-Instanz-Eigenschaften ergänzt. Sie sind spezifisch für eine konkrete Installation auf einem bestimmten Client eines Benutzers. Sie sind insofern dynamisch, als dass sich der Patchlevel des Betriebssystems oder ~~sich~~ die Version der Clientinstanz Client-Instanz durch Updates verändern kann.

**Tabelle 2: Eigenschaften Clientsysteme auf Instanzebene (pro Installation)**

Client-Instanz-Eigenschaften	Beschreibung
Produkt-Version	Aktuelle Version der Client-Software
Client-Nutzer (Owner)	Informationen über den Client-Nutzer (aus dem gID id_token oder aus dem SM(C)-B Zertifikat)
Client-Eigenschaften (Posture)	Aktuelle Eigenschaften der Ablaufumgebung des Clients, insbesondere: <ul style="list-style-type: none"> <li>• Betriebssystem</li> <li>• Betriebssystem-Version</li> </ul>

Client-Instanz-Eigenschaften	Beschreibung
Client-Attestation	<p>Falls die Plattform die Attestation des Clients ermöglicht, wird hier die plattformspezifische Attestation angegeben.</p> <ul style="list-style-type: none"> <li>• Google-Android: Key and ID Attestation</li> <li>• Apple: DCAAppAttest</li> <li>• <u>Windows und Linux: TPM Attestation</u></li> <li>• Software: <u>vom Client selbst erstellte Attestation (keine Attestation, Sicherheitsfunktion; dient nur Nutzer-Bindung betrieblichen Zwecken)</u></li> </ul>

*Hinweis: Für andere mobile Betriebssysteme wird die Attestation unterstützt, wenn sie verfügbar ist.*

Die Auskünfte bzw. Attestation von Clientsoftware und Geräten werden ~~von einer Backendschnittstelle für die Clientregistrierung vom ZETA Guard Authorization Server~~ geprüft. Zusätzlich wird ~~über diese Schnittstelle bei mobilen Clients~~ eine Offband-Verifikation des Benutzers durchgeführt, beispielsweise über mit Bestätigungscode via E-Mail. ~~Ist die Clientregistrierung erfolgreich, wird der Client-Instanz(!) ein Nachweis über die attestierten Client- und Client-Instanz-Eigenschaften ausgestellt. Dieser Nachweis ist in folgenden Aufrufen von Schnittstellen der TI Teil der Zugangsprüfung durchgeführt.~~

Die Clientattribute werden von den Plattformen der Endgeräte geliefert. Ihre Erhebung erfolgt im TrustClientZETA-Client des Endgeräts mittels plattformspezifischer Attestierungs- und Erhebungsmechanismen (siehe [Apple Platform Security Guide] und [Android Platform Security Model]). Die Attribute sind ~~daher~~ für die jeweilige Plattform und ihr Sicherheitsmodell spezifisch. Die für die Zugriffsentscheidung verwendeten Attribute werden daher im Folgenden für iOS-Geräte ~~separat von denen für~~ Android-Geräte sowie Windows- und Linux-Geräte separat aufgeführt.

## Android

**Tabelle 3: Verwendete ~~Device~~-Claims für Android-Clients**

<del>Attribute</del> <u>Attribut</u>	Beschreibung
aktuelle Android Version	aktuell auf dem Gerät laufende Android Version bzw. API Level / SDK Version
<b>Android Version bei Veröffentlichung</b>	Android Version (API level) mit welchem das Gerät veröffentlicht / CTS durchlief
<b>Patchlevel</b>	verschiedene Patch-Level-Angaben für OS & Co
<b>FDE / FBE</b>	Gibt an, ob Geräteverschlüsselung unterstützt wird und ob diese aktiviert ist.

<b><u>Attribute</u></b> <b><u>Attribut</u></b>	<b>Beschreibung</b>
<b>System PIN / Password / Pattern gesetzt</b>	Gibt an, ob ein PIN/Pattern/Passwort für den Sperrbildschirm gesetzt ist.
<b>System PIN / Password / Pattern Qualität</b>	Über den Device Policy Manager kann abgefragt werden, ob aktuell bestimmte Passwort-Komplexitätslevel erfüllt werden.
<b>VerifiedBoot verfügbar</b>	Gibt an, ob VerifiedBoot auf dem Gerät zur Verfügung steht (siehe [VerifiedBoot]).
<b>Mainline Patchlevel</b>	Gibt an, wann der letzte Mainline Patch installiert wurde.
<b>Gerätehersteller / -modell</b>	Gibt Informationen zu Hersteller, Model usw. zurück.
<b>Biometric Class</b>	Gibt Informationen zur Güte der vorhandenen biometrischen Sensoren zurück.

## iOS

**Tabelle 4: Verwendete ~~Device~~-Claims für iOS-Clients**

<b><u>Attribute</u></b> <b><u>Attribut</u></b>	<b><u>Description</u></b> <b><u>Beschreibung</u></b>
System Name	Name des Betriebssystems auf dem Gerät
System Version	Version des Betriebssystems auf dem Gerät
utsname.machine	Art und Version des Geräts, z. B. "iPhone 15"
identifierForVendor	eindeutige Kennung des Geräts für den App-Anbieter
App Version	Version der App auf dem Gerät

## Windows und Linux

**Tabelle 5: Verwendete Claims für Windows und Linux Clients**

<b><u>Attribut</u></b>	<b><u>Beschreibung</u></b>
<u>System Name</u>	<u>Name des Betriebssystems auf dem Gerät</u>



<u>Attribut</u>	<u>Beschreibung</u>
<u>System Version</u>	<u>Version des Betriebssystems auf dem Gerät</u>
<u>Platform</u>	<u>Windows oder Linux</u>
<u>Platform Produkt-ID</u>	<u>ID im Windows Store oder Packaging Typ Applikations-ID bei Linux</u>
<u>Architektur</u>	<u>Prozessor-Architektur</u>
<u>Hersteller Name</u>	<u>Hersteller Name des Geräts</u>
<u>Hersteller ID</u>	<u>Hersteller ID des Geräts</u>

## 4.7 Monitoring

Das Monitoring im Kontext von Zero Trust ist ein entscheidender Aspekt, um die Sicherheit des Netzwerks und der Ressourcen kontinuierlich zu überwachen und potenzielle Bedrohungen oder Anomalien zu identifizieren. Dieses bedient sich auch eines Security Information and Event Management (SIEM) und Shared Signals, die zukünftige Policy-Entscheidungen beeinflussen, in dem Erkenntnisse des Monitorings über den Policy Information Point den Policy Decision Points der verschiedenen Fachanwendungen verfügbar gemacht werden.

Insgesamt ermöglicht das Monitoring im Kontext von Zero Trust eine kontinuierliche Überwachung der Sicherheitslage, indem es aktuelle Sicherheitsrichtlinien berücksichtigt, potenzielle Bedrohungen identifiziert und auf Shared Signals zurückgreift, um umfassende Sicherheitseinblicke zu erhalten.

### 4.7.1 Security Information and Event Management (SIEM):

SIEM-Systeme spielen eine zentrale Rolle im Monitoring im Zero Trust-Paradigma. Sie sammeln Daten aus verschiedenen Quellen wie Protokollen, Ereignissen und Alarmen von Sicherheitskomponenten im Netzwerk. Durch die Analyse dieser Daten in Echtzeit können SIEM-Systeme potenzielle Sicherheitsvorfälle erkennen und Anomalien identifizieren. SIEM-Systeme können auf die vom PIP bereitgestellten Sicherheitsrichtlinien zugreifen und sicherstellen, dass die Überwachung entsprechend den aktuellen Richtlinien erfolgt. Anbieter von Betriebsleistungen mittels Produkttypen der TI1.0 erhalten durch eine Anbieterzulassung die Auflage, Anforderungen an ein [ISMS] zu erfüllen. Hierfür können sie bspw. SIEM-Systeme oder Intrusion Detection Systeme (IDS) verwenden. In der Weiterentwicklung zur TI 2.0 wird dieses Konzept fortgeführt, und finden die so gesammelten Informationen über den Sicherheitszustand eines Systems wieder Eingang in Zugriffsentscheidungen eines Policy Decision Points.

### 4.7.2 Shared Signals

Shared Signals [Shared Signals] sind Hinweise oder Indikatoren für Sicherheitsvorfälle, die von verschiedenen Systemen und Quellen im Netzwerk gemeinsam genutzt



werden. Diese Signale können von verschiedenen Sicherheitskomponenten wie Firewalls, Endpunktschutzsystemen, Intrusion Detection Systems (IDS) und anderen generiert werden.

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

## 4.7.3 Telemetrie, Monitoring und Logging

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

## 4.8 Zusammenspiel mit Identity Provider

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

1. **Nutzer stellt über den ZETA Client eine Anfrage:**Der Nutzer versucht auf eine "besser geschützte Ressource" auf dem Resource Server zuzugreifen.
2. **PEP fängt Anfrage ab:**Der PEP fängt die Anfrage ab.
3. **PEP validiert Access Token:**Der PEP prüft:
  - Ist das Access Token gültig (Signatur, Ablaufdatum etc.)?
  - Hat das Access Token den **erforderlichen Scope und die passende Audience** für die angeforderte Ressource?
4. **Zugriff verweigert:**Wenn der erforderliche ~~Scope~~**nicht Scope oder die Audience**~~nicht~~ im Access Token enthalten ist, verweigert der PEP den Zugriff.
5. **PEP antwortet mit Step-up-Anforderung:**~~Der PEP~~ Wenn die angefragte Ressource auf dem Resource Server existiert, informiert der PEP den Nutzer, dass für den Zugriff auf die Ressource ein Access Token mit einem bestimmten Scope und einer bestimmten Audience benötigt wird.
6. **Nutzer initiiert die Step-up-Authentifizierung:**Der ZETA Client ~~wird zum~~ fragt beim an Authorization Server ~~weitergeleitet,~~ um die ~~zusätzliche~~ Authentifizierung für den benötigten Scope und die Audience durchzuführen.
7. **Authorization Server authentifiziert und gewährt neues Access Token:**Der Authorization Server:
  - Steuert die Durchführung der Step-up-Authentifizierung.

- Erstellt ein **neues Access Token** mit ~~einem erweiterten Scope, der den Zugriff auf die angeforderte Ressource erlaubt~~ passendem Scope und passender Audience.
  - Gibt das neue Access Token an den ZETA Client zurück.
8. **Nutzer stellt eine erneute Anfrage:** Der Nutzer sendet die Anfrage mit dem **neuen** Access Token an den PEP.
  9. **PEP prüft ~~neues~~ das neue Token:** Der PEP prüft wieder:
    - Ist das **neue** Access Token gültig?
    - Hat das **neue** Access Token ~~den erforderlichen Scope~~ den Scope und die Audience für die angeforderte Ressource?
  10. **Zugriff gewährt:** Wenn der erforderliche Scope und die Audience 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.

## 4.9 TI 2.0 Dienst-Backend

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

---

## 5 Spezifikation

---

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

Die Festlegungen zu Schemas, OpenAPI Schnittstellen und Abbildungen sind in [gemAPI\_ZETA] enthalten.

*~~Hinweis: Details in diesem Kapitel werden im Rahmen der Implementierung zwischen gematik und dem Zero Trust Hersteller festgelegt und in einer Folgeversion veröffentlicht.~~*

*~~Offener Punkt: Details in diesem Kapitel werden im Rahmen der Implementierung zwischen gematik und dem Zero Trust Hersteller festgelegt und in einer Folgeversion veröffentlicht.~~*

### 5.1 Übergreifende Anforderungen für Datenschutz und Sicherheit

#### **A\_25400 -ZETA Guard - Umsetzung Sicherer Softwareentwicklungsprozess**

Der Hersteller des ZETA Guards MUSS einen sicheren Softwareentwicklungsprozess umsetzen (siehe [gemSpec\_DS\_Hersteller#Kapitel 2.2 Sicherer Softwareentwicklungsprozess]).[<=]

#### **~~A\_25401 -ZETA Guard - Darstellung der Voraussetzungen für sicheren Betrieb des Produkts im Betriebshandbuch~~ZETA Guard - Darstellung der Voraussetzungen für sicheren Betrieb des Produkts im Produkthandbuch**

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

#### **A\_28459 -ZETA Guard - Informationsobjekte im Produkthandbuch**

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

#### **A\_28463 -ZETA Guard - Informationsobjekte des ZETA Clients im Produkthandbuch**

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

#### **A\_28460 -ZETA Guard - Datenschutzrechtliche Bewertung durch den Dienstanbieter**

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

## **A\_28461 - Informationspflicht des Client-Herstellers gegenüber Nutzern**

Der Hersteller eines TI 2.0-Clients MUSS die seine Anwendung betreffenden Informationsobjekte aus dem Produkthandbuch des ZETA Guard informieren und seine Nutzer datenschutzkonform über deren Verarbeitung informieren. [≤]

## **A\_25402 -ZETA Guard - Schutz der transportierten Daten**

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

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

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

## **A\_26517 -ZETA Guard - Unterstützung von mTLS**

ZETA Guard MUSS die Konfiguration und Nutzung von mTLS unterstützen. [≤]

## **A\_25403 -ZETA Guard - Schutzmaßnahmen gegen die OWASP Top 10 Risiken**

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

## **A\_25404 -ZETA Guard - Angriffe erkennen**

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

## **A\_25405 -ZETA Guard - Angriffen entgegenwirken**

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

## **A\_25406 -ZETA Guard - Eingabe Validierung von Operationen**

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

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

## **A\_25407 -ZETA Guard - Sicherheitstechnische Validierung von Policy und Konfigurationen**

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

## **A\_25408-01 -ZETA Guard - Verbot Profilbildung**

Der Anbieter eines TI2.0 Dienstes DARF Profile - außer zum Zweck des Security Monitorings - NICHT bilden.

[≤]

## **A\_25409 -ZETA Guard - Privacy by Design**

ZETA Guard MUSS sicherstellen, dass bei Konfigurationsmöglichkeiten die datenschutzfreundlichere Option vorausgewählt ist. [≤]

## **A\_25410 -ZETA Guard - Verbot von Werbe- und Usability-Tracking**

ZETA Guard DARF im Produktivbetrieb ein Werbe- und Usability-Tracking NICHT verwenden. [≤]

## **A\_25411 -ZETA Guard - Verbot vom dynamischen Inhalt**

ZETA Guard DARF dynamischen Inhalt von Drittanbietern NICHT herunterladen und verwenden. [≤]

#### **A\_25412 -ZETA Guard - Zusätzliche Verschlüsselung bei der Persistierung**

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

#### **A\_25413-01 -ZETA Guard - Ordnungsgemäße IT-Administration**

Der Anbieter eines TI2.0 Dienstes MUSS die Maßnahmen für erhöhten Schutzbedarf aus dem BSI-Bausteins „OPS.1.1.2 Ordnungsgemäße IT-Administration“ [BSI-Grundschutz] während des gesamten Betriebs des ZETA Guards umsetzen. [≤]

#### **~~A\_26479-02~~ ~~A\_26479-01~~ -ZETA Guard - Ordnungsgemäße Änderung von Konfigurationen**

Der Anbieter eines TI2.0 Dienstes MUSS durch technische und organisatorische Mittel sicherstellen, dass eine Änderung der Konfiguration des ZETA Guards ~~und die Änderungen von Feature Flags des ZETA Guards~~ nur unter 4-Augen erfolgen kann. [≤]

*~~Hinweis: Feature Flags sind Konfigurationseinstellungen, die Funktionen ein- oder ausschalten.~~*

#### **A\_25718 -ZETA Guard - Bereitstellung Security-KPIs**

ZETA Guard MUSS sicherstellen, dass die Security-KPIs in A\_25484-~~01~~-\* automatisch bereitgestellt werden.

[≤]

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

#### **A\_28406 -ZETA Guard – Verification des ZETA-Images**

Der Hersteller des TI2.0-Dienstes MUSS vor der Aktualisierung von ZETA Guard innerhalb seines Build-Systems die Authentizität und Aktualität der zu aktualisierenden Komponenten auf der Grundlage der gematik-Signatur und der Versionshistorie der Komponenten verifizieren und bei Fehlschlägen der Verifikation die Aktualisierung abbrechen und gematik umgehend informieren. [≤]

#### **A\_28407 -ZETA Guard – Nachweisbarkeit verwendete Version des ZETA-Images**

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

### **5.1.1 Sicherheits- und Datenschutzanforderungen an Logging und Monitoring**

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

#### **A\_25744 -ZETA Guard - Datenschutzkonformes Logging und Monitoring**

ZETA Guard MUSS die für den Betrieb des Zero Trust erforderlichen Logging- und Monitoring-Informationen in solcher Art und Weise erheben und verarbeiten, dass mit technischen Mitteln ausgeschlossen ist, dass dem Anbieter eines TI2.0 Dienstes vertrauliche oder zur Profilbildung geeignete Daten zur Kenntnis gelangen. [≤]

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

### **A\_25745 -ZETA Guard - Keine medizinischen Informationen in Logging und Monitoring**

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

### **A\_25746 -ZETA Guard - Keine sicherheitsrelevanten Daten in Logging und Monitoring**

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

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

### **A\_25747-01 -ZETA Guard - Löschfristen Protokolle**

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

[<=]

### **A\_27806 -ZETA Guard – Löschfrist Security Monitoring Protokoll**

ZETA Guard muss sicherstellen, dass die zum Zwecke des Security Monitorings erhobenen Protokolldaten nach spätestens 6 Monaten gelöscht werden. Die Aufbewahrungsdauer muss dabei konfigurierbar sein.[<=]

## **5.1.2 Sicherheits- und Datenschutz-Anforderungen an das Security Monitoring**

Das SIEM, Plattform-Monitoring und Shared Signals bilden das Security Monitoring von Zero Trust ab. Die folgenden Anforderungen beschreiben die Fähigkeiten des Security Monitorings und welche Ereignisse erkannt werden sollen.

### **A\_25419-01 -Security Monitoring - Erkennungsfähigkeit**

Der Anbieter eines TI2.0 Dienstes MUSS sicherstellen, dass das Monitoring System die folgenden Merkmale der Kommunikation erkennen und per Shared Signals kommunizieren kann:

- Geolokation - Land und Ort
- Impossible Travel
- Zugriffe über TOR Netzwerke
- Zugriff von VPN-Provider
- Zugriffe über Proxies
- Zugriffe über Botnetze
- Traffic-Volumen-Anomalien
- Network-Protokoll Anomalien

[<=]

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

*Hinweis: Falls der Anbieter aufgrund einer Netzwerk-Sicherheitsrichtlinie Anfragen mit bestimmten Kommunikationsmerkmalen (z. B. Zugriffe aus einem NOR-Netzwerk oder Botnetz) am Netzwerk-Perimeter blockiert, hat diese Richtlinie Vorrang. Es wird nicht erwartet, dass entsprechende Requests dennoch an den ZETA Guard weitergeleitet oder ein Shared Signal (A\_25420-\*) an den ZETA Guard übermittelt wird, da diese Anfragen bereits durch den Schutzmechanismus am Netzwerk-Perimeter abgefangen werden.*

Der Resource Server kann eine Missbrauchserkennung implementieren. Dabei werden mögliche Angriffe und Anomalien innerhalb der Anwendungslogik erkannt (z. B. falsche/manipulierte Metadaten für Dokumente in der elektronischen Patientenakte (ePA)) und an das SIEM-System gemeldet.

## **A\_25421 -Security Monitoring - Empfang von Missbrauchserkennung auf Resource Server-Ebene**

Falls der Resource Server eine Missbrauchserkennung durchführt, MUSS ZETA Guard sicherstellen, dass das Security Monitoring solche Missbrauchssignale von dem Resource Server empfangen und verarbeitet werden kann. [ <= ]

## **A\_25420-01 -Security Monitoring - Kommunikationsmerkmale signalisieren**

Der Anbieter des TI 2.0 Dienstes MUSS in seinem Monitoring System sicherstellen, dass bei Erkennung folgender Kommunikationsmerkmale die erforderlichen Informationen automatisiert an ZETA Guard per Shared Signals gesendet werden:

- Impossible Travel
- Zugriffe über TOR Netzwerke
- Zugriffe über Proxies
- Zugriffe über Botnetze
- Zugriff von VPN-Provider
- Traffic-Volumen-Anomalien
- Network-Protokoll Anomalien
- Missbrauchssignale von dem Resource Server (falls implementiert)

[ <= ]

*Hinweis: Mit dem Signal erhält der Authorization Server die Information, dass sich eine Eigenschaft der aktuellen Session geändert hat. Der Authorization Server sperrt automatisch das aktuelle Access Token, sodass der Client ein neues Access Token beim Authorization Server abfragen muss. Die Abfrage des neuen Access Token beinhaltet immer eine Entscheidung durch die Policy Engine.*

*Hinweis: Shared Signals ist ein neuer Standard, der nicht von allen Netzwerk-Produkten direkt unterstützt wird. In solchen Fällen kann das Netzwerk-Produkt ein Log erzeugen, das vom SIEM-System des Anbieters ausgewertet wird. Das SIEM-System kann daraufhin eigenständig ein entsprechendes Shared Signal generieren und übermitteln.*

## **A\_25484-01 -Security Monitoring - Security KPIs**

ZETA Guard MUSS einmal täglich mittels dem Telemetrie-Daten-Service die folgende Sicherheits-KPIs automatisiert über die von der gematik angebotene Schnittstelle an das TI SIEM-System übermitteln:

- Anzahl versuchter Zugriffe von nicht registrierten Clients (hier muss die KPIs zwischen Resource Server APIs und Clientregistrierung APIs unterscheiden)
- Anzahl von Netzwerk-Protokoll-Anomalien
- Anzahl von Zugriffen von Botnetzen



- Anzahl von Zugriffen aus jedem Land gezählt plus weitere Zugriffe, die separat in Versicherte und LE ausgewiesen werden
- Anzahl fehlerhafter Clientfreischaltungen plus weitere breakdown in Versicherte und LE
- Anzahl von Impossible travel Zugriffen (inkl. Land- und Ortsdaten) plus weitere breakdown in Versicherte und LE
- Anzahl von Zugriffen über TOR Netzwerke plus weitere breakdown in Versicherte und LE
- Anzahl von Zugriffen über Proxies plus weitere breakdown in Versicherte und LE
- Anzahl von Zugriffen über VPNs plus weitere breakdown in Versicherte und LE
- Traffic Volumes plus weitere breakdown in Versicherte und LE
- Anzahl erkannte Angriffe in Kategorie (siehe A\_25404-\*) plus weitere breakdown in Versicherte und LE
- Anzahl fehlerhafte Authorization Codes vom IDP.

[<=]

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

*Hinweis: Netzwerk-Protokoll-Anomalien sind z.B. ungewöhnliche Netzwerk-Aktivitäten, Netzwerk-Protokoll-Aktivitäten oder die Manipulation von Netzwerk-Paketen.*

### **A\_25485-01 -Security Monitoring - Sicherheitsmeldung bei Aktualisierung von PIP-Daten oder PDP-Policies**

ZETA Guard MUSS bei der erfolgreiche Aktualisierung der vom PIP und PAP Service heruntergeladenen Daten eine Sicherheitsmeldung automatisiert über die von der gematik angebotene Schnittstelle an das TI SIEM-System übermitteln.[<=]

### **A\_25606-01 -Security Monitoring - Fehlermeldung bei Aktualisierung von PIP-Daten oder PDP-Policies**

ZETA Guard MUSS beim folgenden Fehler während der Aktualisierung der vom PIP und PAP Service heruntergeladenen Daten eine Fehlermeldung automatisiert über die von der gematik angebotene Schnittstelle an das TI SIEM-System ~~übermitteln~~übermitteln:

- Policy Download Fehler ((HTTP Fehlercode: 400, 404)
- Fehler bei der Integritätsprüfung der Policy-Signatur

[<=]

## **5.1.3 Sicherheits- und Datenschutz-Anforderungen an die Verarbeitung von Daten mit dem Schutzbedarf "sehr hoch"**

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



*Offener Punkt: Der ZETA Guard muss in einer VAU laufen können. Deswegen muss der ZETA Guard die Speicherung und Verwendung der Privatschlüssel von Komponente-Identitäten in einem HSM unterstützen. Details in diesem Kapitel werden im Rahmen der Implementierung zwischen gematik und dem Zero Trust Hersteller festgelegt und in einer Folgeversion veröffentlicht.*

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

#### **A\_25608-01 -ZETA Guard - Verarbeitung von Daten mit Schutzbedarf "sehr hoch"**

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

[<=]

#### **A\_25763-01 -Zero Trust-Komponenten - Private Schlüssel der Komponenten-Identitäten in einem HSM**

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

[<=]

#### **A\_25764 -Zero Trust-Komponenten - Sicherer Betrieb und Nutzung eines HSMs**

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

#### **A\_25765 -Zero Trust-Komponenten - Einsatz zertifizierter HSM**

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

Die Prüftiefe MUSS mindestens:

1. FIPS 140-2 Level 3 oder
2. FIPS 140-3 Level 3 oder
3. Common Criteria EAL 4+ (mit AVA\_VAN.5)

entsprechen.[<=]

#### **A\_26065-01 -Nur zugelassene Images in Produktion**

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

*Offener Punkt: Für den Betrieb des ZETA Guards in einer VAU werden von der gematik signierte und für den Einsatz in der PU vorgesehene Produktimages verwendet, aber für die Ausführung in der VAU angepasst. Über die Attestation des VAU-Produktimages wird der Nachweis erbracht, dass das Produktimage der gematik verwendet wurde. Wie der Nachweis genau erfolgt, dass bei Ausführung von ZETA Guard Komponenten in einer VAU die für den Einsatz in der PU vorgesehene Produktimages verwendet werden, wird in einer Folgeversion des Dokuments festgelegt.*

Das ZETA Guard-Image ist unabhängig von einer spezifischen VAU-Architektur. Eine VAU ist allerdings in der Regel auf eine bestimmte Architektur (wie z. B. SGX, TDX, SEV usw.) ausgelegt. Der Hersteller des TI2.0-Dienstes muss das ZETA Guard-Image entsprechend für die jeweils eingesetzte VAU-Architektur umwandeln.

#### **A\_28405 -ZETA Guard – Umwandlung für Ziel-VAU-Architektur**

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

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

**[<=]**

### **5.1.4 Sicherheits- und Datenschutz Anforderungen an dem ZETA Client in FdVs**

#### **A\_25802 -ZETA Client - Einhaltung der BSI [TR-03161-1]**

Der ZETA Client eines FdVs MUSS die BSI [TR-03161-1] erfüllen, sofern sie für den ZETA Client anwendbar ist. [**<=**]

*Hinweis: Nicht anwendbar können zum Beispiel sein: O.Paid .. Die Anwendbarkeit ist zwischen Hersteller des ZETA Clients und dem Gutachter zu klären. Der Gutachter gibt sein Votum über die Erfüllung der BSI [TR-03161-1] in Form der Bewertung der Erfüllung der A\_25802 ab, wobei die A\_25802 als „umgesetzt“ bewertet werden kann, wenn die anwendbaren Abschnitte der BSI [TR-03161-1] aus Sicht des Gutachters erfüllt sind.*

## **5.2 Clientsystem und ZETA Client**

Ein Clientsystem ist eine Softwarekomponente in der Verwendung eines Benutzers zum Ausführen fachlicher Anwendungsfälle z.B. als Primärsystem (PVS, AVS, LIS, KIS etc.) oder als Frontend des Versicherten (ePA-App, E-Rezept-App, TI-Messenger etc.). Dieses Clientsystem wird dem Benutzer durch einen Hersteller zur Verfügung gestellt.

Ein ZETA Client ist ein Teil des Clientsystems, der die Kommunikation mit dem PDP und dem PEP eines Dienstes übernimmt.

Mobile iOS und Android Apps werden unterstützt und setzen ebenfalls einen ZETA Client um. Anforderungen, die nur für diese Clientsysteme und ZETA Clients gelten werden, verwenden die Bezeichnung "mobiler Client" oder "mobiles Clientsystem" und "mobiler ZETA Client".

## 5.2.1 Hersteller

### A\_25335 -Hersteller Clientsystem - Hinweise und Maßnahmen sicherer Betrieb

Der Hersteller eines Clientsystems MUSS den Benutzer über Maßnahmen zum sicheren Betrieb seines Clientsystems vor der Inbetriebnahme informieren und während des Betriebs stets zum Abruf durch den Benutzer bereithalten. [≤]

### A\_25336 -Hersteller Clientsystem - Regelmäßige Updates

Der Hersteller eines Clientsystems MUSS, solange das Produkt nicht abgekündigt ist, dem Benutzer regelmäßig (z. B. quartalsweise) Updates für das Clientsystem bereitstellen, um das Clientsystem dauerhaft auf dem Stand der Technik zu halten und Sicherheitslücken zu schließen. [≤]

### A\_25337 -Hersteller Clientsystem - Registrierung für product\_id

Der Hersteller eines Clientsystems MUSS sich über einen organisatorischen Prozess bei der gematik für die Nutzung von Diensten, für welche Token abgerufen werden sollen, registrieren. Der Hersteller eines Clientsystems bekommt dabei eine "product\_id" zugewiesen, die in jeder Instanz des Clientsystems verwendet werden MUSS. [≤]

### A\_25427 -Hersteller Clientsystem Android - Google Cloud Projekt

Der Hersteller eines Clientsystems für eine Android-basierte Betriebsumgebung MUSS ein Google Cloud Projekt führen oder eine gleichwertige alternative Plattformattestierung verwenden, um Nachweise über die Clientintegrität einer jeden Clientsysteminstallation beziehen zu können. [≤]

## 5.2.2 Verbindungsaufbau

### A\_25338 -ZETA Client - Identifikation mittels product\_id

Der ZETA Client MUSS bei der SM(C)-B Authentifizierung ~~mit DPoP~~ in der JWT-Assertion unter "urn:telematik:client-self-assessment" die Client-Daten gemäß [client-instance.yaml] eintragen. Dabei MUSS die von der gematik für das Clientsystem ausgestellte product\_id nach [A\_25337] und eine selbst festgelegte Versionskennzeichnung nach folgendem Schema verwendet werden:

- <product\_id> gemäß Registrierung bei der gematik mit Länge maximal 20 Zeichen, Zeichenvorrat [0-9a-zA-Z\\_-],
- <product\_version> gemäß Produktidentifikation mit Länge 1-20 Zeichen, Zeichenvorrat [0-9a-zA-Z\\_-].

[≤]

### A\_25339 -ZETA Client - Exponential Backoff

Der ZETA Client SOLL bei Server-seitigen Fehlermeldungen, die auf eine Überlastung des Zielsystems schließen lassen (z. B. HTTP-status 5xx, 429 - too many requests etc.), erneute Verbindungsversuche nach dem Prinzip des Exponential Backoffs [ExpBack] durchführen. [≤]

*Hinweis: Die Parameter für das Verfahren des Exponential Backoffs werden vom Hersteller des ZETA Clients festgelegt.*

### A\_27378 -ZETA Client - TLS

Der ZETA Client MUSS TLS 1.3 oder höher unterstützen.

[≤]

## A\_25340 -ZETA Client- Zertifikatsprüfung

Der ZETA Client MUSS alle Zertifikate, die es aktiv verwendet (bspw. TLS-Verbindungsaufbau), auf Integrität und Authentizität prüfen. Falls ein Zertifikat ungültig ist, so MUSS der ZETA Client die von dem Zertifikat und den darin enthaltenen Attributen (bspw. öffentliche Schlüssel) abhängenden Arbeitsabläufe ablehnen.

Der ZETA Client MUSS alle öffentlichen Schlüssel, die es verwenden will, auf eine positiv verlaufene Zertifikatsprüfung zurückführen können.

Die Zertifikatsprüfung MUSS folgende Prüfungen enthalten:

- Der ZETA Client MUSS den Common Name (CN) oder Subject Alternative Name (SAN) des Zertifikats mit dem erwarteten Hostnamen vergleichen.
- Der Client MUSS die Gültigkeitsdauer des Zertifikats (Nicht vor und Nicht nach) überprüfen.
- Die Gültigkeit des Zertifikats MUSS basierend auf dem Zertifikatsvertrauenspfad (Root-CA, Intermediate-CAs) geprüft werden.

Der ZETA Client MUSS die Root-Zertifikate, die von den Mitgliedern des [CAB-Forum] ausgestellt werden, als Vertrauensanker standardmäßig unterstützen. Dies impliziert:

- Eine Standard-Liste von [CAB-Forum] Root-Zertifikaten MUSS im ZETA Client enthalten sein.
- Das Root-Zertifikat der TI Komponenten PKI MUSS in der Liste der Root-Zertifikate enthalten sein.
- Die Liste der Root-Zertifikate MUSS durch weitere Root-Zertifikate der TI erweiterbar sein.
- Der ZETA Client MUSS diese Liste automatisch und regelmäßig aktualisieren.
- Der Client MUSS eine Überprüfung dieser Root-Zertifikate auf ihre Gültigkeit und Revokation ermöglichen (z.B. über eine CRL oder OCSP).

[<=]

## A\_27379 -ZETA Client - OCSP Stapling Unterstützung

Der ZETA Client MUSS OCSP Stapling erkennen und nutzen, wenn es vom Server bereitgestellt wird.

Der ZETA Client MUSS die Gültigkeit der OCSP-Antwort (Signatur, Signierer) prüfen.

Der ZETA Client MUSS die Gültigkeitsdauer der OCSP-Antwort prüfen.

Der ZETA Client MUSS überprüfen, ob die OCSP-Antwort zum aktuellen Zertifikat passt.

OCSP-Antworten MÜSSEN im Cache für die Zeit bis ~~NextUpdate~~<sup>NextUpdate</sup> der OCSP Response gespeichert werden, um unnötige Abfragen zu vermeiden. Die Dauer der Speicherung im Cache MUSS konfigurierbar sein und mindestens 1 Stunde betragen.

Wenn ~~NextUpdate~~<sup>ThisUpdate</sup> nicht in der OCSP Response enthalten ist, dann MUSS die OCSP Response bis ~~ThisUpdate~~<sup>ThisUpdate</sup> + 24 Stunden im Cache gespeichert werden.

Wenn OCSP Stapling nicht angeboten wird, MUSS der ZETA Client entweder die CRL laden oder den OCSP Responder des Zertifikats direkt abfragen.

Wenn weder eine OCSP Response noch eine CRL verfügbar sind<sup>7.4</sup>, MUSS der ZETA Client den Verbindungsaufbau abbrechen.

[<=]

## A\_26681 -ZETA Client - Umsetzen eines ZETA/ASL-Kanals

Der ZETA Client MUSS einen ZETA/ASL-Kanal (Client-Seite) umsetzen können.

Das Access Token und das DPoP Token müssen außerhalb der Verschlüsselung des ZETA/ASL-Kanals transportiert werden.

Wenn ein ZETA/ASL-Kanal verwendet und ein PoPP Token im Request transportiert wird,

dann MUSS das PoPP Token durch den ZETA/ASL-Kanal geschützt transportiert werden. [ $\leq$ ]

*Hinweis: Ob ein ZETA/ASL Kanal zu verwenden ist, wird im OAuth Protected Resource Well-known Dokument des TI 2.0 Dienstes festgelegt (claim `zeta_asl_use`). Die Anforderungen für den ZETA/ASL-Kanal sind in [gemSpec\_Krypt#8] zu finden.*

## **A\_28426 -ZETA Client, Service Discovery**

Der ZETA Client MUSS die Well-known und JWKS JSON-Dokumente regelmäßig einmal alle 24 Stunden neu laden, wenn im HTTP-Response-Header kein Cache-Control-Header vom ZETA Guard eingefügt wurde. Der ZETA Client MUSS die Service Discovery erneut durchführen, wenn beim Kommunikationsaufbau zu den geschützten Ressourcen der HTTP-Statuscodes 404 (Not Found) empfangen wird. [ $\leq$ ]

## **A\_28425 -ZETA Client, Service Discovery – If-None-Match und ETag**

Der ZETA Client MUSS bei der Abfrage der Well-known JSON-Dokumente die HTTP-Header `If-None-Match` und `ETag` verwenden. [ $\leq$ ]

## **5.2.3 Clientregistrierung**

*Offener Punkt: Details in diesem Kapitel werden im Rahmen der Implementierung zwischen gematik und dem Zero Trust-Hersteller festgelegt und in einer Folgeversion veröffentlicht. Die Client-Registrierung wird dienstübergreifend ermöglichen, dass im Falle von Big Apps (Unterstützung mehrerer TI-Anwendungen in einem Client) der Nutzer nur einmalig aktiv werden muss, um sein Client mittels 2. Faktor zu bestätigen.*

## **A\_28465 -ZETA Client, Registrierung mit mehreren ZETA Guard**

Der ZETA Client MUSS in der Lage sein sich an jedem ZETA Guard zu registrieren, über den eine Kommunikation mit einem TI 2.0 Dienst erfolgen soll. Durch die Registrierung erhält der ZETA Client von jedem ZETA Guard eine spezifische `client_id`. Der ZETA Client MUSS die Registrierungs-Daten und Konfigurationsdaten pro TI 2.0 Dienst verwalten. [ $\leq$ ]

*Hinweis: Grundsätzlich kann über einen ZETA Guard der Zugriff auf mehrere Ressourcen geschützt werden.*

## **A\_25432 -ZETA Client - Ablauf Clientregistrierung**

Der mobile ZETA Client MUSS, sofern es an Schnittstellen der Telematikinfrastruktur wegen einer ungültigen/fehlenden Client-Registrierung abgewiesen wird, eine Registrierung am Authorization Server durchführen, in dem es

- den/die Benutzer:in mittels OpenID Connect authentifiziert,
- kryptografische Client-Credentials lokal generiert,
- die generierten Credentials sowie die Clientintegrität attestiert und
- eine zusätzliche Benutzerbestätigung mittels One-Time-Passwort (gemäß [TR-03107-1]) über einen zweiten Kommunikationsweg (z. B. E-Mailbestätigung) startet.

[ $\leq$ ]

*Hinweis: Für die Clientregistrierung muss das Vertrauensniveau hoch, nicht erreicht werden.*

**A\_25766 -ZETA Client - Client Credentials in TI Qualität**

Der ZETA Client MUSS die Client-Credentials im Form von kryptografischen Schlüsseln gemäß der Festlegungen in [gemSpec\_Krypt] (Verfahren, Algorithmen, Schlüssellängen etc.) unterstützen. [≤]

**A\_25769 -ZETA Client - Client Credentials sicher generieren und schützen**

~~Das ZETA~~Das ZETA Client auf mobilem Gerät mit Apple- oder Android-basierter Betriebsumgebung MUSS die Generierung der Client-Credentials derart generieren und speichern, dass ein Kopieren und Exportieren der Schlüssel verhindert wird. [≤]

*Hinweis: Eine Speicherung der Schlüssel in einem Hardware-Modul ist gegenüber einer Software-Lösung (z. B. Android TEE) zu bevorzugen.*

**A\_25770 -ZETA Client - Client Credentials Rotation**

Der ZETA Client MUSS seine Client-Credentials regelmäßig rotieren (erneuern und neu registrieren), wobei die Häufigkeit der Rotation durch die gematik nach einer Auswertung der initialen Benutzererfahrung festgelegt wird. [≤]

*Hinweis: Perspektivisch werden weitere Attestierungsmechanismen für Clientsysteme aufgenommen, z. B. FIDO2, TPM2.*

**A\_25767 -ZETA Client - Clientkey in JWT**

Das ZETA Client MUSS Private Key JWT [RFC7521] und [RFC7523] sowie DPoP [RFC9449] zur Authentifizierung unterstützen. [≤]

**A\_25434 -ZETA Client - Clientregistrierung mit bestätigten Umgebungseigenschaften Android**

Der ZETA Client für eine Google-Android basierte Betriebsumgebung MUSS seine Client-Credentials, App-Integrität und -Authentizität sowie OS-/FW und HW-Eigenschaften über Key and ID Attestation gegenüber PDP Client-Registrierung bestätigen. [≤]

**A\_25768 -ZETA Client - Clientregistrierung mit bestätigten Umgebungseigenschaften Apple**

Der ZETA Client für eine Apple-basierte Betriebsumgebung (iOS, macOS) MUSS die Client-Credentials, App Integrität und Authentizität über DCAppAttest gegenüber dem PDP Authorization Server bestätigen. Eigenschaften der Laufzeitumgebung MÜSSEN durch das Clientsystem über einen geprüften Prozess bestätigt werden. [≤]

**A\_25758 -ZETA Client - Erfassung Kontaktinformation für Offband-Verifikation**

Der mobile ZETA Client MUSS vom Benutzer eine strukturell valide Kontaktinformation (E-Mailadresse, Telefonnummer) abfragen und für eine Offband-Verifikation (Trust on First Use) an den Endpunkt des Authorization Servers übertragen. [≤]

**A\_25732 -ZETA Client - Unterstützung des Nutzers bei der Registrierung**

Der mobile ZETA Client MUSS den Nutzer bei der Clientregistrierung und -Verwaltung geeignet unterstützen (z. B. mittels Guide, Tutorial o. ä.). [≤]

**A\_25733 -ZETA Client - Clientverwaltung und manuelle De-Registrierung**

Der ZETA Client MUSS dem Nutzer eine Übersicht aller beim Clientregistrierungsdienst registrierten Clients darstellen und die Möglichkeit zur De-Registrierung einzelner Clients anbieten. [≤]

**A\_25734 -ZETA Client - Zugriffsprotokoll Clientregistrierung**

Der ZETA Client MUSS dem Nutzer einen Einblick in das Zugriffsprotokoll der Schnittstellen des Clientregistrierungsdienstes für genutzte Clients dieses Nutzers geben. [≤]



**A\_25735 -Clientsystem - Aktivierung Push-Benachrichtigung**

Das Clientsystem MUSS dem Nutzer die Möglichkeit geben, Push-Benachrichtigungen für Aktivitäten über registrierte Clients und Neuregistrierungen für diesen Nutzer zu aktivieren. [≤]

**5.2.4 Nutzerauthentifizierung****A\_25761 -ZETA Client - Nutzerauthentifizierung mittels etablierter Standards**

Der ZETA Client MUSS die Mechanismen OAuth Authorization Code Flow mit OpenID Connect und OpenID Federation (Auswahl des zuständigen sektoralen IDP) oder OAuth2 Token Exchange mit `private_key_jwt` Client Authentifizierung unterstützen. [≤]

*Hinweis: Perspektivisch sollen ZETA Clients auch OpenID for Verifiable Credentials (OIDC4VC) unterstützen. OAuth2 Token Exchange wird für stationäre Clients mit SM(C)-B Authentisierung verwendet. OAuth Authorization Code Flow, OpenID Connect und OpenID Federation wird grundsätzlich von mobilen Clients verwendet, kann aber auch von stationären Clients verwendet werden.*

**A\_25762 -ZETA Client - Nutzerauthentifizierung - Unterstützung etablierter Identitäten und Dienste**

Der ZETA Client MUSS zur Authentifizierung des Nutzers mindestens eines der folgenden Verfahren unterstützen:

- Authentifizierung des Nutzers gegenüber einem Sektoralen IDP der IDP Föderation gemäß [gemSpec\_IDP\_Sek] (GesundheitsID)
- Authentifizierung des Nutzers gegenüber dem zentralen IDP-Dienst der TI gemäß [gemSpec\_IDP\_Dienst] (SmartCardIDP für kartengebundene Identitäten).
- Authentifizierung des Nutzers mittels SM(C)-B signiertem ~~Client Assertion JWT und DPoP gemäß [RFC7523] und [RFC9449]-ID Token.~~

[≤]

**5.2.5 Session Management****A\_25781 -ZETA Clients - OAuth2 Autorisierung**

Der ZETA Client MUSS die Rolle eines OAuth2 Clients [RFC6749] übernehmen und eine Autorisierung vom Authorization Server einholen. Dabei MUSS PKCE Flow [RFC7636] verwendet werden.

[≤]

**A\_25782 -ZETA Client - OAuth2 Session Management**

Der ZETA Client MUSS

- die vom Authorisation Server ausgestellten Access- und Refresh Token gemäß [RFC6749#1.5] sowie die DPoP Schlüssel gemäß [RFC9449] bis zur nächsten Aufforderung zur Autorisierung oder Authentifizierung als User-Session sicher aufbewahren,
- nach Bedarf abgelaufene Access Token über Refresh Token erneuern und
- eine Refresh Token-Rotation gemäß [RFC6749#10.4] unterstützen.

[≤]

**A\_25783 -ZETA Client - Anweisungen aus HTTP Response Status Codes und Header folgen**

Der ZETA Client MUSS die HTTP Response Status Codes und HTTP Header entsprechend der Vorgaben der Resource Server und Zero Trust-APIs auswerten und den Anweisungen daraus folgen und insbesondere

- eine Step-Up- oder erneute Authentifizierung des Nutzers,
- eine Re-Autorisierung und erneute Attestation der Client-Instanz,
- eine Anzeige der Warnungen aufgrund der Policy-Entscheidungen und
- ein Replay-Nonce

umsetzen.[<=]

**5.2.6 Liste der HTTP-Statuscodes**

Der folgende Abschnitt enthält die HTTP-Statuscodes, die ZETA Clients von Zero Trust-Komponenten erhalten können, basierend auf den spezifischen Schritten wie Authentifizierung, Clientregistrierung und HTTP Proxy.

**A\_27007 -ZETA Client - HTTP Statuscodes**

Der ZETA Client MUSS die HTTP Statuscodes gemäß Tabelle "ZT\_HTTP\_Statuscodes" unterstützen.[<=]

**Tabelle 6: ZT\_HTTP\_Statuscodes**

Endpunkte	HTTP Statuscodes
<b>Authentifizierung mit Client Assertion JWT</b>	<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). Der Client sollte die JWT-Assertion auf Fehler prüfen (z.B. Format, fehlende Claims) und die Anfrage entsprechend korrigieren oder erneut senden.</p> <p>401 Unauthorized: Authentifizierung fehlgeschlagen, z.B. aufgrund einer ungültigen Signatur oder eines abgelaufenen JWT. Der Client sollte überprüfen, ob das JWT korrekt signiert und nicht abgelaufen ist. Er kann ein neues JWT generieren und die Anfrage erneut senden.</p> <p>403 Forbidden: Der Client hat keine Berechtigung, auf die angeforderte Ressource zuzugreifen, obwohl er authentifiziert ist. Der Client darf keine weiteren Versuche unternehmen und soll den Nutzer entsprechend informieren.</p>



Endpunkte	HTTP Statuscodes
<b>Authentifizierung mit OIDC und Authorization Code Flow</b>	<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 "redirect_uri", "client_id"). Der Client sollte die Anfrageparameter überprüfen und sicherstellen, dass alle erforderlichen Parameter korrekt sind. Eine erneute Anfrage kann nötig sein.</p> <p>401 Unauthorized: Authentifizierung fehlgeschlagen, z.B. ungültiger Code oder Token. Der Client sollte den Autorisierungscode überprüfen und gegebenenfalls einen neuen Code anfordern oder den Flow neu starten.</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.</p>
<b>Clientregistrierung</b>	<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. Der Client sollte die übermittelten Registrierungsdaten überprüfen und mögliche Fehler beheben, bevor er erneut versucht, sich zu registrieren.</p> <p>401 Unauthorized: Fehlende oder ungültige Authentifizierung bei der Registrierung. Der Client muss sicherstellen, dass er korrekt authentifiziert ist. Er sollte die Authentifizierungsdaten überprüfen und erneut versuchen, sich zu registrieren.</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 das Serverproblem informieren.</p>

Endpunkte	HTTP Statuscodes
<b>HTTP Proxy</b>	<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: Fehlende oder ungültige Authentifizierung.</p> <p>403 Forbidden: Zugriff auf die angeforderte Ressource durch den Proxy verweigert. Der Client darf keine weiteren Anfragen an diese Ressource senden und soll den Nutzer über die fehlende Berechtigung informieren.</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>
<b>Alle Endpunkte</b>	<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>

### 5.2.7 ZETA Attestation Service

Der ZETA Attestation Service stellt einen gRPC-Dienst zur Verfügung, der es stationären Clients (Primärsysteme auf Basis von Windows oder Linux) ermöglicht, TPM-signierte Attestierungsinformationen für den Client abzurufen. Diese Informationen basieren auf Integritätsmessungen, die in ausgewählten Platform Configuration Registers (PCRs) des Trusted Platform Module (TPM) gespeichert sind. Der ZETA Guard Authorization Server

verwendet diese Attestierungsdaten, um die Integrität und Authentizität der Softwareumgebung des Clients zu verifizieren, bevor Zugriff auf geschützte Ressourcen gewährt wird.

Der ZETA Attestation Service wird vom Hersteller des stationären Clients bereitgestellt und es muss eine Vertrauensbeziehung zwischen stationären Client und ZETA Attestation Service bestehen, um zu gewährleisten, dass die Attestation über die vorgesehenen Software-Komponenten erfolgt.

Der ZETA Attestation Service muss gemäß [API-ZETA-Attestation-Service] implementiert werden.

### 5.3 ZETA Guard

Die Software des ZETA Guards wird im Auftrag der gematik entwickelt und alle Komponenten als signierte ~~Docker Container~~OCI Images und als signiertes Helm Chart in einer gematik ~~ContainerArtifact~~ Registry sowie mit Kubernetes Manifest Dateien in einem ~~gematik git Repository~~ bereitgestellt, sodass die Anbieter von TI 2.0 Diensten ihren spezifischen ZETA Guard darauf aufbauend in ~~einer Kubernetes Umgebung konfigurieren können. Die ZETA Guard Konfiguration muss in ein git Repository der gematik verlinkt werden (z. B. als git submodule). Der ZETA Guard Management Service setzt durch, dass nur die im gematik git Repository verlinkte ZETA Guard Konfiguration ausgeführt werden kann. Dadurch ist es möglich, dass der Anbieter seine ZETA Guard Konfiguration selbständig anpassen und die gematik den korrekten Einsatz des ZETA Guards prüfen kann ihrer Kubernetes Umgebung konfigurieren können.~~

#### A\_26105 -ZETA Guard, Durchsetzung der Konfiguration

Der Anbieter eines TI 2.0 Dienstes MUSS für seinen ZETA Guard die ihm zugewiesene Konfiguration aus dem git Repository verwenden. [ $\leq$ ]

*Hinweis: Für die Anpassung und Inbetriebnahme von geänderten ZETA Guard-Konfigurationen ist ein Continuous Delivery Prozess mit Quality Gates vorgesehen, der sich noch in der Entwicklung befindet.*

#### A\_26519 -ZETA Guard, Unterstützung von Service-Mesh Lösungen

Die Komponenten des ZETA Guard MÜSSEN es ermöglichen, dass Service-Mesh Lösungen zur Verwaltung der Komponenten eingesetzt werden können. [ $\leq$ ]

#### A\_26521 -ZETA Guard, Unterstützung von Canary Releases

Die Komponenten des ZETA Guard MÜSSEN Canary Releases unterstützen. [ $\leq$ ]

#### A\_25666 -ZETA Guard - TLS Terminierung

Die Komponente Ingress MUSS TLS für von außen eingehende Verbindungen terminieren können. Alternativ wird die TLS-Verbindung am PEP HTTP Proxy terminiert.

[ $\leq$ ]

#### A\_26964 -ZETA Guard - OCSP Stapling

Komponenten innerhalb des ZETA Guards (inkl. Ingress), die von außen kommende TLS Verbindungen oder ZETA/ASL terminieren, SOLLEN OCSP-Stapling [RFC6066] verwenden.

Das OCSP Abfrageintervall MUSS das Minimum aus `dynamic_validity_period` und `max_allowed_validity_period` sein, wenn `NextUpdate` in der OCSP Response angegeben ist. Anderenfalls MUSS das OCSP Abfrageintervall

gleich `max_allowed_validity_period` sein.

Dabei gilt:

$$\text{dynamic\_validity\_period} = (\text{percentage\_of\_validity\_period} * (\text{NextUpdate} - \text{ThisUpdate})) / 100$$

Die Werte für `max_allowed_validity_period` (default Wert = 1 Tag)

und `percentage_of_validity_period` (default Wert = 60%) MÜSSEN konfigurierbar sein.

Die jeweils letzte OCSP Response MUSS im Cache gespeichert und für OCSP Stapling verwendet werden.

Es MUSS eine Warnung bereitgestellt werden, wenn eine OCSP Abfrage fehlschlägt. Sollte vom entsprechenden OCSP Responder trotz regelmäßigen Versuchs keine OCSP Response bezogen werden können, so MUSS die jüngste zur Verfügung stehende OCSP Response verwendet werden und es MUSS mit exponential Backoff weiter probiert werden. [ $\leq$ ]

## **A\_26639 -ZETA Guard - Unterstützung Websocket**

Die Komponente Ingress MUSS das WebSocket-Protokoll unterstützen. [ $\leq$ ]

## **A\_26640 -ZETA Guard - HTTP Protokoll-Versionen**

Die Komponente Ingress MUSS das HTTP Protokoll in den Versionen HTTP/1.1, HTTP/2 und SOLL HTTP/3 unterstützen. [ $\leq$ ]

## **A\_25652 -ZETA Guard - Notification Service**

Der ZETA Guard Notification Service MUSS Push-Notifications über die von App-Anbietern bereitgestellten Push-Gateways unterstützen, um die Notifications an bestimmte oder alle registrierte Clients eines Anwenders verschicken zu können.

Der Notification Service MUSS gemäß [`gemF_PushNotification`] die Push Konfigurationen von ZETA Clients registrieren und verwalten können und MUSS die vom Resource Server empfangenen Notification Events an die dafür registrierten Push Gateways der Clients weiterleiten. [ $\leq$ ]

## **A\_25737 -ZETA Guard - Push Notification**

Der ZETA Guard MUSS eine Push Benachrichtigung an alle registrierten Clients des Nutzers, für die eine Push Notification aktiviert ist, verschicken, sobald sich Änderungen an der Liste der registrierten Clients dieses Nutzers ergibt. [ $\leq$ ]

## **A\_26988 -Telemetrie-Daten Service - Fehlermeldungen**

Alle Komponenten des ZETA Guard MÜSSEN ihre Fehlermeldungen so bereitstellen, dass der Telemetrie-Daten Service die Fehlermeldungen sammeln und an einen Monitoring Service weiterleiten kann. [ $\leq$ ]

## **A\_26661 -ZETA Guard - HTTP Statuscodes**

Der ZETA Guard MUSS die HTTP Statuscodes gemäß Tabelle "ZT\_HTTP\_Statuscodes" unterstützen. [ $\leq$ ]

## **A\_26662 -ZETA Guard, HTTP Fehlerdetails**

Der ZETA Guard MUSS bei Beantwortung eines Requests mit einem HTTPFehler ein JSON Object ergänzen, das den Fehler beschreibt. Das JSON Object MUSS gemäß [`zeta-error.yaml`] aufgebaut sein. [ $\leq$ ]

## **Beispiel für eine sinnvolle Ergänzung der Fehlerdetails:**

Bei der Authentifizierung mit Client Assertion JWT fehlt im JWT eine Angabe `product_version` oder die `product_version` ist nicht in der Policy Engine bekannt, dann antwortet der Authorization Server mit HTTP Status 400 Bad Request sowie einer Beschreibung, dass die `product_version` fehlt oder nicht bekannt ist.

### **A\_27264 -ZETA Guard, Telemetrie-Daten**

Alle ZETA Guard-Komponenten mit Ausnahme der Datenbanken MÜSSEN OpenTelemetry konforme Traces, Metriken, and Logs erzeugen. [≤]

### **A\_27802 -ZETA Guard, JWT Prüfung**

Der ZETA Guard MUSS bei der Prüfung von JWT folgende Prüfschritte durchführen:

**JWT Header-Überprüfung:** Zuerst überprüfen, ob der Algorithmus akzeptabel ist, um unnötige Verarbeitung zu vermeiden. Gültige Algorithmen sind in [gemSpec\_Krypt] festgelegt.

#### **Payload-Überprüfung:**

- Ablaufdatum (exp): prüfen, um abgelaufene Tokens sofort abzulehnen.
- Audience (aud): prüfen, ob das Token für den beabsichtigten Empfänger bestimmt ist.
- Issuer (iss): Sicherstellen, dass der Aussteller dem erwarteten Aussteller entspricht.

**Integrität der Nachricht:** Hash-Überprüfung: Sicherstellen, dass die Nachricht nicht manipuliert wurde.

**Schlüsselvalidierung:** Überprüfen, ob der öffentliche Schlüssel gültig und vertrauenswürdig ist.

**Vertrauenswürdigkeit der Schlüsselkette:** Bestätigen, dass der Schlüssel von einer vertrauenswürdigen Quelle stammt.

#### **Spezifische Prüfungen für TI-Zertifikate:**

- Ablaufdatum des Zertifikats: Sicherstellen, dass das Zertifikat noch gültig ist.
- Widerrufsstatus des Zertifikats (OCSP): Prüfen, ob das Zertifikat nicht widerrufen wurde. Die OCSP Response wird für eine konfigurierbare Dauer zwischengespeichert, um zu häufige OCSP Anfragen zu verhindern. Die Dauer soll der Gültigkeitsdauer des Refresh Token entsprechen.
- Vertrauenswürdigkeit der Zertifikatskette: Sicherstellen, dass die Kette zu einer vertrauenswürdigen Root-CA führt.

#### **Spezifische Prüfungen für DPoP Token:**

- Typ: Prüfen, dass der JWT-Typ dpop+jwt ist.
- Nonce: Die Nonce muss geprüft werden.

[≤]

### **A\_27853 -ZETA Guard, Responses mit Header ZETA-API-Version**

Alle ZETA Guard Komponenten MÜSSEN an ihren Endpunkten in den Responses den Header ZETA-API-Version angeben, der die exakte SemVer-Version der ausführenden Instanz angibt. [≤]

### **A\_26668 -ZETA Guard - Rate Limit**

Die Komponenten des ZETA Guard MÜSSEN für ihre durch ZETA Clients erreichbaren Endpunkte ein Rate Limit konfigurierbar einstellen können. Wenn ein Rate Limit konfiguriert ist, dann muss der Client über Response Header-Informationen informiert werden ( (das erlaubte Limit), (verbleibende Anfragen) und (Zeitpunkt, an dem das Limit zurückgesetzt wird)). [≤]

Hinweis: Es gibt einen RFC Draft, der Rate Limits neu spezifiziert

(<https://www.ietf.org/archive/id/draft-ietf-httpapi-ratelimit-headers-09.html>). Es ist geplant, dass nach Veröffentlichung des RFC ZETA Guard angepasst wird, um die neuen Rate Limit Festlegungen zu unterstützen.

#### **A\_27864 -ZETA Guard - Egress**

Der ZETA Guard MUSS eine Funktion implementieren, die durchsetzt, dass nur erlaubte Verbindungen, zu Endpunkten außerhalb des ZETA Guard, aufgebaut werden. [ $\leq$ ]

#### **A\_28435 -ZETA Guard, Ingress - Unterstützung Forwarded-Header**

Die Komponente Ingress MUSS in jeder empfangene HTTP-Anfrage für die nachgelagerten Komponenten PDP Authorization Server und PEP HTTP Proxy den Forwarded-Header gemäß [RFC 7239] in der weitergeleiteten Anfrage hinzufügen oder aktualisieren. [ $\leq$ ]

### **5.3.1 Optionale Komponenten**

#### **A\_28432 -ZETA Guard, Komponenten Ingress optional**

Der Ingress MUSS als optionale Komponente im ZETA Guard angeboten werden. [ $\leq$ ]

Es ist möglich auf die Komponente Ingress im ZETA Guard zu verzichten. Dadurch wird der Hersteller des TI2.0 Dienstes in die Lage versetzt seinen eigenen externen Ingress zu verwenden.

#### **A\_28433 -ZETA Guard, Bereitstellung externer Ingress**

Der Hersteller des TI2.0-Dienstes MUSS entweder den Kubernetes Ingress des ZETA Guard oder einen eigenen Ingress verwenden. [ $\leq$ ]

#### **A\_28434 -ZETA Guard, Verwendung externer Ingress**

Der vom Hersteller des TI2.0-Dienstes bereitgestellte externe Ingress MUSS alle Festlegungen erfüllen, die der ZETA Guard Komponente Ingress zugewiesen sind. [ $\leq$ ]

#### **A\_28462 -ZETA Guard, externer Ingress - TLS Terminierung**

Der Hersteller des TI2.0-Dienstes MUSS TLS für eingehende Verbindungen von außerhalb Kubernetes für die Komponenten PEP HTTP Proxy und PDP Authorization Server an den jeweiligen Komponenten terminieren, wenn ein externer Ingress zum Einsatz kommt und der ZETA Guard in einer VAU bereitgestellt wird. [ $\leq$ ]

#### **A\_28480 -ZETA Guard, Integration optionaler Komponenten**

Der Anbieter des TI 2.0-Dienstes MUSS sicherstellen und nachweisen, dass alle Anforderungen an die optionalen ZETA Guard Komponenten erfüllt sind, wenn er eigene Lösungen dieser Komponenten anstelle der standardmäßigen ZETA Guard Installation verwendet. [ $\leq$ ]

### **5.3.2 Kommunikation mit Diensten der gematik**

ZETA Guard Instanzen benötigen Zugriff auf Dienste der gematik, um PIP und PAP OCI Container zu laden, um Telemetriedaten an den gematik Empfänger zu senden und um Security Events an das gematik SIEM zu senden. Für den Zugriff auf diese Dienste ist ein gültiges Access Token erforderlich.

Um ein gültiges Access Token zu erhalten wird Workload Identity Federation eingesetzt. Voraussetzung dafür ist, dass

- der Kubernetes Cluster, in dem ZETA Guard ausgeführt wird, seine OpenID Konfiguration und seine JWKS URI (`/.well-known/openid-configuration` und `/openid/v1/jwks`) öffentlich erreichbar bereitstellt,
- ein ServiceAccount innerhalb des Clusters existiert, der die benötigten Secrets (Access Token) für die Workloads bereitstellt und
- die Issuer URI des Kubernetes Cluster bei der gematik registriert ist.

### 5.3.3 Deployment Szenarien

#### 5.3.3.1 Geo-Redundanz

Der Betrieb von Kubernetes in einer Multi-Cluster-Umgebung bietet Unternehmen erhebliche Vorteile in Bezug auf Skalierbarkeit, Ausfallsicherheit und Flexibilität, bringt jedoch auch eine erhöhte Komplexität in Verwaltung, Netzwerk und Sicherheit mit sich und es ergeben sich Anforderungen an Anbieter von TI 2.0 Diensten.

##### A 28438 -ZETA Guard, geo-redundanter Betrieb

Der Anbieter eines TI 2.0 Dienstes MUSS beim geo-redundanten Betrieb des ZETA Guard in einer Kubernetes Multi-Cluster-Umgebung folgende Bedingungen erfüllen:

- Die PDP DB MUSS vom Anbieter des TI 2.0 Dienstes über alle Cluster synchronisiert bereitgestellt werden.
- Die Authorization Server Instanzen müssen über einen globalen Load Balancer als ein logischer Authorization Server bereitgestellt werden.

[<=]

##### A 28464 -ZETA Guard, genau ein Authorization Server

Die Komponente PEP HTTP Proxy MUSS das OAuth Protected Resource Well-known so bereitstellen, dass für ZETA Clients der ZETA Guard Authorization Server als genau eine logische Komponente bereitgestellt wird (nur ein Eintrag `authorization_servers` im OAuth Protected Resource Well-known).[<=]

### 5.3.4 Laufzeitüberwachung

Offener Punkt: ZETA Guard muss zur Laufzeit erkennen können,

- ob die von der gematik signierten Images ausgeführt werden,
- ob es ungewöhnliche Systemaufrufe oder Netzwerkkommunikation gibt und
- ob die Kommunikation so wie vorgesehen durch die ZETA Guard Microservices läuft.

Die Ergebnisse der Laufzeitüberwachung werden an den Telemetriedaten-Empfänger der gematik gesendet und können an das Monitoring des TI 2-0 Dienst-Anbieters gesendet werden.

ZETA Guard soll durchsetzen, dass die Konfiguration des ZETA Guard aus einem Git Repository des Dienst-Anbieters ausgeführt wird. Dafür stellt ZETA Guard den optionalen Management Service Argo CD bereit. Der Dienst-Anbieter kann eine andere Lösung verwenden (z. B. Flex).

Die Anforderungen zur Laufzeitüberwachung und Durchsetzung der Konfiguration werden in einer folgenden Version von gemSpec ZETA festgelegt.



## 5.4 Policy Enforcement Points

Der Policy Enforcement Point (PEP) stellt die zentrale Sicherheitskomponente einer Zero Trust-Architektur dar, da in dieser alle Zugriffsentscheidungen durchgesetzt (engl.: enforce) werden.

### 5.4.1 PEP HTTP Proxy

Die Komponente HTTP Proxy ist die "letzte" vor das Resource Backend geschaltete Zero Trust-Komponente und prüft das Access Token im Authorization Header des Requests. Ist das Access Token gültig, wird der Zugriff gewährt. Zudem wird der Request um zusätzliche HTTP-Header angereichert, um ein Tracing zu ermöglichen.

#### **A\_26666 -PEP HTTP Proxy - TLS Terminierung**

Die Komponente PEP HTTP Proxy MUSS TLS für eingehende Verbindungen von außerhalb Kubernetes terminieren können. Alternativ wird die TLS-Verbindung an der Komponente Ingress terminiert. [ $\leq$ ]

#### **A\_26195 -PEP HTTP Proxy - Unterstützung Websocket**

Die Komponente HTTP Proxy MUSS das WebSocket-Protokoll unterstützen. [ $\leq$ ]

#### **A\_26641 -PEP HTTP Proxy - HTTP Protokoll-Versionen**

Die Komponente HTTP Proxy MUSS das HTTP Protokoll in den Versionen HTTP/1.1, HTTP/2 und SOLL HTTP/3 unterstützen. [ $\leq$ ]

#### **A\_25667 -PEP HTTP Proxy - Verifikation Access Token Binding**

Die Komponente HTTP Proxy MUSS das Access Token Binding über den Mechanismus OAuth 2.0 Demonstrating Proof of Possession (DPoP) gemäß [RFC9449] verifizieren; d. h. der Claim "jkt" im Access Token MUSS eindeutig der Angabe im DPoP-Token entsprechen. [ $\leq$ ]

#### **A\_25668 -PEP HTTP Proxy - Access Token Validierung**

Die Komponente HTTP Proxy MUSS das übergebene Access Token validieren. Insbesondere MÜSSEN

- die Signatur des Authorization Servers gültig,
- die Angaben zur zeitlichen Gültigkeit (Felder: `iat`, `exp`) valide,
- die Angabe `aud` für das Resource Backend korrekt eingetragen und
- die Angabe `scope` und `aud` passend zur Request url

sein.

Die Signatur des Access Tokens ist gültig, wenn sie mathematisch gültig ist und die Signatur vom Authorization Server im gleichen ZETA Guard erstellt wurde (default Einstellung) oder von einem Authorization Server, der in der Konfiguration des HTTP Proxy angegeben und im Entity Statement des Federation Master aufgeführt ist. Es MUSS möglich sein, mehrere Authorization Server in die Konfiguration des HTTP Proxy einzutragen.

Wenn das Access Tokens ungültig ist, dann MUSS der Request mit HTTP Code 401 `Unauthorized` beantwortet werden. [ $\leq$ ]

*Hinweis: Jeder Authorization Server, der zur Föderation des Federation Masters gehört, veröffentlicht ein eigenes Entity Statement, in dem die Schlüssel enthalten sind, mit denen die Signatur der Access Token geprüft werden kann.*

*Hinweis: Einige TI 2.0 Dienste benötigen ein PoPP Token. Diese werden im Request Header PoPP übertragen und vom HTTP Proxy geprüft.*



**A\_26477 -PEP HTTP Proxy - PoPP Token Validierung**

Die Komponente HTTP Proxy MUSS so konfiguriert werden können, dass pro Endpunkt des Resource Servers der Request Header `PoPP` verlangt und das PoPP Token validiert wird. Zusätzlich MUSS die Konfiguration pro Endpunkt unterstützen, dass die Dauer der Gültigkeit des PoPP Token in Sekunden seit Ausstellung und nach Ausstellungszeitpunkt und Prüfzeitpunkt innerhalb des gleichen Quartals eingestellt werden kann.

Wenn ein Request für einen Endpunkt des Resource Servers empfangen wird, der kein PoPP Header enthält und das Vorhandensein des PoPP Headers wird verlangt, dann MUSS der HTTP Proxy den Request mit HTTP Code 400 Bad Request beantworten.

Insbesondere MUSS die Signatur des PoPP Servers gültig sein. Die Signatur des PoPP Tokens ist gültig, wenn sie mathematisch gültig ist und die Signatur vom PoPP Server erstellt wurde. Der HTTP Proxy MUSS ein vorhandenes und noch gültiges JWKS des PoPP Servers verwenden oder das JWKS des PoPP Servers herunterladen, um anhand der im JWKS enthaltenen Schlüssel die Signatur des PoPP Tokens zu prüfen.

Der `claimactorId` des PoPP Token MUSS mit dem Attribut `identifizier` aus den zum Access Token gehörenden Nutzer-Daten übereinstimmen.

Vor Ablauf der Gültigkeit MUSS der HTTP Proxy ein neues JWKS des PoPP Servers herunterladen.

Wenn die Signatur des PoPP Tokens ungültig ist oder eine der anderen Prüfungen nicht erfolgreich war, dann MUSS der Request mit HTTP Code 403 Forbidden beantwortet werden.

[<=]

*Hinweis: Wenn ein ZETA/ASL-Kanal am HTTP Proxy terminiert wird, dann wird das PoPP Token durch den ZETA/ASL-Kanal geschützt transportiert.*

**A\_26493 -PEP HTTP Proxy - Umgang mit JWKS des Popp Servers**

Die Komponente HTTP Proxy MUSS, falls nach Ablauf der Gültigkeitsdauer für heruntergeladene JWKS des Popp Servers, kein neues JWKS heruntergeladen werden kann,

- das bestehende JWKS des Popp Servers für eine weitere Gültigkeitsdauer nutzen und
- einen Fehler für das Monitoring-System des Anbieters generieren.

[<=]

**A\_26480 -PEP HTTP Proxy - Umsetzen eines ZETA/ASL-Kanals**

Die Komponente HTTP Proxy MUSS einen ZETA/ASL-Kanal (Server-Seite) umsetzen können. Die Verwendung des ZETA/ASL-Kanals MUSS durch Konfiguration ein- und ausschaltbar sein. In der Default-Einstellung ist der ZETA/ASL-Kanal ausgeschaltet.[<=]

*Hinweis: Ob ein ZETA/ASL Kanal zu verwenden ist, wird in der Spezifikation des TI 2.0 Dienstes festgelegt.*

**A\_26946 -PEP HTTP Proxy - Verwaltung der ZETA/ASL-Kanal Schlüssel in der PEP Datenbank**

Die Komponente HTTP Proxy MUSS die ZETA/ASL-Kanal Schlüssel so verwalten (z. B. in der PEP Datenbank), dass für die Terminierung des ZETA/ASL-Kanals mehrere HTTP Proxy Instanzen genutzt werden können.[<=]

**A\_26947 -PEP Datenbank - Zugriff nur für den PEP HTTP Proxy**

Die Komponente PEP Datenbank MUSS Zugriffe ausschließlich für den PEP HTTP Proxy gewähren.[<=]

*Hinweis: Die Anforderungen für den ZETA/ASL-Kanal sind in [gemSpec\_Krypt#8] zu finden.*

**A\_26492-01A-26492 -PEP HTTP Proxy - Weiterleitung von Client-Daten**

Die Komponente HTTP Proxy MUSS so konfiguriert werden können, dass pro Endpunkt des Resource Servers die Weiterleitung der Client-Daten durch den HTTP Proxy ein- und ausgeschaltet werden kann. Die default-Einstellung ist keine Weiterleitung der Client-Daten. Wenn die Weiterleitung der Client-Daten eingeschaltet ist, dann fragt der HTTP Proxy die Client-Daten anhand ~~des sub~~ `des client_id` claims aus dem Access Token von der ~~Client-Registry~~ `PDP-Datenbank` ab und fügt sie als zusätzlichen Header in den Request ein.

[&lt;=]

**A\_25669 -PEP HTTP Proxy - Zusätzliche HTTP-Header**

Die Komponente HTTP Proxy MUSS die HTTP Requests mit allen HTTP-Headern an das Resource Backend weiterleiten und dabei die folgenden zusätzlichen HTTP-Header einsetzen.

**Tabelle 7: 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	[user-info.yaml]	350 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-instance.yaml]	2500 Byte

Gleichnamige HTTP-Header aus dem ursprünglichen HTTP-Request MÜSSEN entfernt bzw. überschrieben werden.[<=]

*Hinweis: Die Schema-Dateien sind in [GitHub ZETA Schemas] festgelegt.*

**A\_26560 -PEP HTTP Proxy - Weiterleitungskonfiguration**

Der PEP HTTP Proxy MUSS es ermöglichen, dass Regeln für die Weiterleitung von Request URLs an die URLs von Resource Servern konfiguriert werden können.

[&lt;=]

**A\_27265 -PEP HTTP Proxy - unveränderte Weiterleitung von Host Header und Request Zeile**

Der PEP HTTP Proxy MUSS den Host Header und die Request Zeile unverändert an den Resource Server weiterleiten.[<=]

**A\_26561 -PEP HTTP Proxy - Caching**

Der PEP HTTP Proxy MUSS so konfiguriert werden können, dass Caching von Inhalten der Response möglich ist.[<=]

**A\_26589 -PEP HTTP Proxy - Nutzer-Daten**

Die Komponente HTTP Proxy MUSS für jeden Request mit gültigem Access Token die Nutzer-Daten anhand des Parameters `jti` des Access Token aus der PDP Datenbank abfragen und als neuen HTTP Header `ZETA-User-Info` in den Request eintragen, bevor der Request an den Resource Server weitergeleitet wird. [ $\leq$ ]

**A\_26590 -PEP HTTP Proxy - Client-Daten**

Wenn die Request-Weiterleitung mit Client-Daten konfiguriert wurde und der Request ein gültiges Access Token hat, dann MUSS die Komponente HTTP Proxy die Client-Daten anhand des Parameters `sub-desclient_iddes` Access Token aus der PDP Datenbank abfragen und als neuen HTTP Header `ZETA-Client-Data` in den Request eintragen, bevor der Request an den Resource Server weitergeleitet wird. [ $\leq$ ]

*Hinweis: Die Abfrage der Client-Daten kann zusammen mit der Abfrage der Nutzer-Daten in einem Request an die PDP Datenbank erfolgen.*

**A\_26974 -PEP HTTP Proxy - Fehler vom Resource Server**

Die Komponente HTTP Proxy MUSS die Response vom Resource Server als Fehler des HTTP Proxy werten, wenn der Resource Server den Response Header `ZETA-Cause: Proxy` gesetzt hat (der Resource Server hat einen Fehler im Request festgestellt und vermutet die Ursache beim HTTP Proxy) und DARF diese Response nicht an den Client weiterleiten. Der entsprechende Request des Clients MUSS in diesem Fall mit HTTP 500 beantwortet werden. [ $\leq$ ]

**A\_27266 -PEP HTTP Proxy - Protected Resource Metadata Well-known**

Die Komponente HTTP Proxy MUSS gemäß [RFC9728] und [opr-well-known.yaml] ein Well-known JSON Dokument wie folgt bereitstellen, damit Clients die notwendigen Informationen zur Interaktion mit dem Resource Server finden können.

GET /.well-known/oauth-protected-resource

Host: <FQDN des Resource Servers>

Das Well-known JSON Dokument MUSS mit dem Schema [opr-well-known.yaml] validiert werden können.

[ $\leq$ ]

*Hinweis: Es ist geplant, die Protected Resource Metadata Well-known JSON Dokumente in einer folgenden ZETA Ausbaustufe durch den Federation Master bereitzustellen. Der Federation Master signiert die Protected Resource Metadata Dokumente und stellt so sicher, dass alle Services zur Föderation und zur gleichen Umgebung gehören (dev, prod, ref, etc.). Dadurch verbessert sich für ZETA Clients die Authentizität der TI 2.0 Services.*

**A\_28439 -PEP HTTP Proxy - Unterstützung Forwarded-Header**

Die Komponente PEP HTTP Proxy MUSS in jeder empfangene HTTP-Anfrage den Forwarded-Header gemäß [RFC 7239] in der weitergeleiteten Anfragen aktualisieren. [ $\leq$ ]

**5.4.2 Sicherheits- und Datenschutz-Anforderungen an den PEP****A\_25445 -PEP - Zugriffsentscheidung nur über PDP**

Der PEP MUSS sicherstellen, dass Zugriffe auf den Resource Server nur durch eine positive Zugriffsentscheidung vom PDP möglich sind. [ $\leq$ ]

*Hinweis: Die positive Zugriffsentscheidung auf den Resource Server ist gegeben, wenn der Client im Request Authorization Header ein gültiges Access Token mit passendem scope - ausgestellt von einem Authorization Server, zu dem eine Vertrauensbeziehung*

*besteht - vorweisen kann. Durch das gültige Access Token ist sichergestellt, dass der Zugriff von dem PDP freigegeben wird.*

### 5.5 Policy Decision Point

Der Policy Decision Point (PDP) setzt sich aus den Komponenten

- Authorization Server
- Policy Engine und
- PDP Datenbank

zusammen.

#### 5.5.1 Policy Engine

Der PDP implementiert die Policy Engine als [Open Policy Agent] (OPA). Die Policies und die zugehörigen Daten erhält die Policy Engine per Download vom PIP und PAP Service. Aus den Input-Daten vom Authorization Server, den Daten vom PIP und den Policies vom PAP ermittelt die Policy Engine eine Entscheidung und gibt diese zurück an den Authorization Server.

Neben der OPA Instanz, die die Entscheidung für den Authorization Server trifft (aktive Instanz), ob eine Kommunikation zulässig ist, implementiert die Policy Engine noch eine zweite OPA Instanz, die mit einem zweiten OPA Bundle vom PIP und PAP Service arbeitet, aber die getroffenen Entscheidungen nicht an den Authorization Server zurückgibt. Diese Instanz wird Simulations-Instanz genannt.

#### **A\_25739 -PDP, Open Policy Agent Instanzen**

Der PDP MUSS als Policy Engine zwei Open Policy Agent (OPA) Instanzen bereitstellen, wobei eine Instanz die Entscheidung für den Authorization Server trifft (aktive Instanz), und eine Instanz eine Entscheidung trifft, diese aber nicht an den Authorization Server sendet (Simulations-Instanz).

Die OPA Instanzen MÜSSEN gemäß Tabelle OPA\_Konfiguration konfiguriert werden.

Tabelle 8: OPA - Konfiguration

Konfiguration	Aktive OPA Instanz	Simulations-OPA Instanz
<b>services:</b> - <b>name:</b> <service name> <b>url:</b> <PIP und PAP service>	<service name> Innerhalb der OPA Konfiguration verwendeter Service Name.  <PIP und PAP service> Download-Endpunkt des PIP und PAP Services entsprechend der verwendeten Umgebung Default: Produktions-Instanz: <a href="https://pip-pap.ti-dienste.de">https://pip-pap.ti-dienste.de</a> Referenz-Instanz: <a href="https://pip-pap-ref.ti-dienste.de">https://pip-pap-ref.ti-dienste.de</a> Test-Instanz: <a href="https://pip-pap-test.ti-dienste.de">https://pip-pap-test.ti-dienste.de</a>	wie aktive OPA Instanz
<b>bundles:</b> <b>authz:</b> <b>service:</b> <service name> <b>resource:</b> <path> <b>persist:</b> true	<path> Der Pfad wird wie in [pip-pap-service.yaml] beschrieben angegeben. Durch das label latest wird die neueste bundle.tar.gz Datei für die aktive OPA Instanz heruntergeladen. Default: /policies/<TI service>/latest	<path> Durch das label latest-sim wird die neueste bundle.tar.gz Datei für die Simulations-Instanz der Policy Engine heruntergeladen. Default: /policies/<TI service>/latest-sim
<b>bundles:</b> <b>authz:</b> <b>polling:</b> <b>min_delay_seconds:</b> <min> <b>max_delay_seconds:</b> <max>	<min> Minimale Zeit bis zum nächsten Polling. Default: 300 <max> Maximale Zeit bis zum nächsten Polling. Default: 320	wie aktive OPA Instanz
<b>bundles:</b> <b>authz:</b> <b>signing:</b> <b>keyid:</b> <PIP_and_PAP_key> <b>scope:</b> read	<PIP_and_PAP_key> Die keyid des Schlüssels, mit dem die OPA Bundles signiert sind.	wie aktive OPA Instanz

Konfiguration	Aktive OPA Instanz	Simulations-OPA Instanz
<pre> <b>decision_logs:</b>   <b>service:</b> &lt;service name&gt;   <b>resource:</b>     \${DL_REMOTE_URL}   <b>reporting:</b>     <b>min_delay_seconds:</b>       &lt;min&gt;     <b>max_delay_seconds:</b>       &lt;max&gt; </pre>	<pre> \${DL_REMOTE_URL} Die URL des Remote Servers, zu dem die decision logs gesendet werden. Dieser Parameter wird per environment Variable übergeben, sodass jeder Anbieter des ZETA Guard bei Bedarf seinen eigenen Server angeben kann, der die decision logs empfängt. &lt;min&gt; Minimale Verzögerung bis zum nächsten Versand. Default: 300 &lt;max&gt; Maximale Verzögerung bis zum nächsten Versand. Default: 360 </pre>	wie aktive OPA Instanz

**[<=]**

*Hinweis: Änderungen an der Konfiguration sind im Einvernehmen mit der gematik möglich.*

Der OPA aktualisiert seine Policies und Daten nach dem vorgegebenen Polling-Intervall. Jede Download Anfrage enthält immer ein `If-None-Match` Header mit dem hash des zuletzt heruntergeladenen bundles (aus dem `ETag` Header der Response). Wenn es keine neuen Daten zum Download gibt, dann beantwortet der PIP/PAP Service die Anfrage mit `304 Not Modified`.

Die Bundles sind immer signiert. Der OPA prüft die Signatur mit dem zur konfigurierten `keyid` passenden Schlüssel. Gültige Schlüssel und deren `keyid` werden über einen Downloadpunkt des PIP und PAP Service als JWKS geladen.

Wenn konfiguriert, dann werden die decision logs an die vom Anbieter des ZETA Guard festgelegte URL gesendet.

#### **A\_26664 -PDP - Durchsetzung der Policy-Konfiguration**

Der PDP MUSS sicherstellen, dass nur die in der Konfiguration festgelegten Eigenschaften angewendet werden, insbesondere Herkunft der Policies und Daten sowie Signaturprüfung der Policies und Daten. **[<=]**

#### **A\_25450 -PDP - Policy nur vom gematik PIP und PAP Service**

Der PDP MUSS sicherstellen, dass in den Policy Engines nur Policies und Daten vom gematik PIP und PAP Service importiert werden können. Eine unberechtigte Änderung der Konfiguration des PDP MUSS technisch ausgeschlossen werden. **[<=]**

#### **A\_27401 -PDP Policy Engine - Decision Eigenschaften**

Die PDP Policy Engine MUSS Decision-Anfragen mit einem JSON Objekt nach dem Schema `[pdp-decision.yaml]` beantworten. **[<=]**

#### **A\_25774 -PDP - Löschfristen für Auditeinträge des Admin Audit-Logs**

Der PDP MUSS sicherstellen, dass die Löschung eines Auditeintrags den gesetzlichen Vorgaben entspricht und frühestens nach 12 Monaten erfolgt. **[<=]**

**~~A\_25775-01A\_25775~~ -PDP - Kontrolle des Audit-Logs**

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

**A\_25453 -PDP - Transparenz der installierte Policies**

Der PDP MUSS sicherstellen, dass die gematik zu jeder Zeit feststellen kann, welche Policies und welche Policy-Versionen im PDP installiert sind. [≤]

**A\_25490 -PDP - Sicherheitsmeldung bei Änderungen und Aktualisierung**

Der PDP MUSS sicherstellen, dass bei Aktualisierung und Änderungen der Policies oder PIP-Daten eine Sicherheitsmeldung an das Security Monitoring automatisiert übermittelt wird. [≤]

**A\_25771 -PDP - Automatisierte Prüfung nach Policy-Aktualisierungen**

Der PDP muss alle 5 Minuten prüfen, ob Aktualisierungen der installierten und verwendeten Policy/PIP-Daten vorhanden sind. [≤]

**A\_25451 -PDP - Integritätsprüfung der Policies**

Der PDP MUSS in der Policy Engine sicherstellen, dass Policies vom gematik PIP und PAP Service nur nach einer positiven Integritätsprüfung importiert werden können. [≤]

## 5.5.2 PDP Authorization Server

**A\_25760 -PDP Authorization Server - OAuth2 Schnittstellen**

Der PDP Authorization Server MUSS eine OAuth2 Schnittstelle gemäß [RFC6749] und [RFC7636] implementieren.

Der Authorization Server MUSS am Token Endpunkt REFRESH\_TOKEN entsprechend [RFC6749] ausstellen können. [≤]

**A\_26669 -PDP Authorization Server - TLS Terminierung**

Die Komponente PDP Authorization Server MUSS eingehende TLS Verbindungen von außerhalb des ZETA Guards terminieren können. Alternativ wird die TLS-Verbindung an der Komponente Ingress terminiert. [≤]

**A\_25659 -PDP Authorization Server - Check Client Registrierung**

Der PDP Authorization Server MUSS die Client Instanzen über den Mechanismus JSON Web Token Client Authentication gemäß [RFC7523] mit DPoP gemäß [RFC9449] authentifizieren und Anfragen, die den Mechanismus nicht verwenden, ablehnen. [≤]

**A\_25660 -PDP Authorization Server - Session Management mittels Access Token und Refresh Token**

Die Komponente Authorization Server MUSS ein Session Management mittels OAuth2 und Ausgabe, Verwaltung und Entzug von Access und Refresh Token gemäß [RFC6749#1.5] unterstützen. [≤]

**A\_27867 -PDP Authorization Server - Session Management in der PDP Datenbank**

Die Komponente Authorization Server SOLL die Session-Daten gemäß [session.yaml] und die User-Daten gemäß [user-info.yaml] in der PDP Datenbank verwalten. [≤]

**A\_26944 -PDP Authorization Server - Access Token Inhalt**

Die Komponente Authorization Server MUSS Access Token mit Attributen gemäß [access-token.yaml] ausstellen. [≤]



**A\_26945 -PDP Authorization Server - Refresh Token Inhalt**

Die Komponente Authorization Server MUSS Refresh Token mit Attributen gemäß [refresh-token.yaml] ausstellen. [≤]

**A\_25661 -PDP Authorization Server - Umsetzung der Policy Decision**

Die Komponente Authorization Server MUSS die Zugriffs-Entscheidung eines PDP mittels der Ausgabe eines Access und eines Refresh Tokens umsetzen bzw. eine Zugriffsverweigerung mit einem HTTP-Statuscode 403 quittieren.

Die Claims im Access und im Refresh Token MÜSSEN zur Entscheidung der Policy Engine für die angefragten Claims passen. [≤]

**A\_25662 -PDP Authorization Server - Refresh Token Rotation**

Die Komponente Authorization Server MUSS eine Refresh Token Rotation gemäß [RFC6749#10.4] erzwingen und MUSS sicherstellen, dass ein Refresh Token nur einmal gegen ein Access Token und ein Refresh Token getauscht werden kann.

Die Komponente Authorization Server MUSS erzwingen, dass der Nutzer eine Authentisierung durchführen muss, wenn seit der letzten Authentisierung die Zeit der Gültigkeitsdauer des Refresh Tokens abgelaufen ist. [≤]

**A\_25663 -PDP Authorization Server - Token-Binding an Client-Registrierung**

Die Komponente Authorization Server MUSS auszugebende Access Token und Refresh Token über den Mechanismus OAuth 2.0 Demonstrating Proof of Possession (DPoP)

gemäß [RFC9449] an die registrierte Client-Instanz binden, indem im Token-Binding-Claim die Angabe der Clientidentifikation als "jkt" eindeutig referenziert wird. [≤]

**A\_25664 -PDP Authorization Server - Token Laufzeit gemäß Policy**

Die Komponente Authorization Server MUSS die Laufzeit der ausgegebenen Token entsprechend der Festlegungen aus der getroffenen Zugriffsentscheidung der Policy Engine setzen. [≤]

**A\_25665 -PDP Authorization Server - Plugin-Schnittstelle Application Authorization Backend**

Die Komponente Authorization Server MUSS eine Plug-In Schnittstelle (per Webhook) zu einem anwendungsspezifischen Authorization Backend implementieren und dabei die folgenden Signale und Informationen aus der erhaltenen Zugriffsanfrage weiterreichen.

**Tabelle 9: PDP Authorization Server - Plugin-Schnittstelle Application Authorization Backend**

Operation	Operation Kennung	Input	Output
Benachrichtigung über die Ablehnung des Zugriffs durch PDP	notifyAccessDenied	Trace-Id Subject-Information Client-Information PDP-Decision	-



Operation	Operation Kennung	Input	Output
Anwendungsspezifische Autorisierung	authorizeAccess	Trace-Id Session-Id Authorization-Scopes Authorization-Details Subject-Information Client-Information PDP-Decision	Zugriff erlauben Ja/Nein Zusätzliche Authorization Scopes Zusätzliche Authorization Details Zusätzliche Claims
Benachrichtigung über abgelaufene oder terminierte Sessions	notifySessionTermination	Trace-Id Session-Id	-

[&lt;=]

#### A\_26586 -PDP Authorization Server - Session- und Nutzer-Daten

Die Komponente Authorization Server MUSS nach jeder vollständigen und erfolgreichen Authentifizierung Session-Daten in der Datenbank des PDP speichern.

Die Session-Daten und Nutzer-Daten MÜSSEN bei Anfragen des Authorization-Servers an die Policy Engine im Request mit übergeben werden.[<=]

#### A\_26972 -PDP Authorization Server - Nutzer-Daten aus der SM(C)-B

Die Komponente Authorization Server MUSS im Falle der SM(C)-B Authentifizierung die folgenden Daten aus dem SM(C)-B Zertifikat auslesen und als Nutzer-Daten gemäß [user-info.yaml] in der PDP Datenbank speichern:

**Tabelle 10: SM(C)-B\_Nutzer-Daten**

SM(C)-B Daten (C.HCI.OSIG gemäß [gemSpec_PKI])	Nutzer-Daten gemäß [user-info.yaml]	Beschreibung
Extension Admission,registrationNumber	identifizier	Telematik-ID Eindeutige ID der Organisation des Gesundheitswesens
subject,commonName	commonName	Wird übernommen, wenn im Zertifikat vorhanden. „Kurzname“ der Institution, so wie sie sich auf dem Adressfeld findet.
Extension Admission,professionOID	professionOID	OID der Institution gemäß [gemSpec_OID#GS-A_4443-*]
subject,organizationName	organizationName	Wird übernommen, wenn im Zertifikat vorhanden. Name der Organisation/Einrichtung des Gesundheitswesens

[&lt;=]

**A\_26973 -PDP Authorization Server - Nutzer-Daten aus id\_token**

Die Komponente Authorization Server MUSS im Falle der OIDC Authentifizierung für Versicherte alle Claims aus dem id\_token gemäß [gemSpec\_IDP\_Sek] auslesen und als Nutzer-Daten gemäß [user-info.yaml] in der PDP Datenbank speichern. Dabei gilt das folgende Mapping für die Pflicht-Nutzer-Daten.

**Tabelle 11: id\_token\_Nutzer-Daten**

id_token claims (gemäß [gemSpec_IDP_Sek])	Nutzer-Daten gemäß [user-info.yaml]	Beschreibung
urn:telematik:claims:id	identifizier	für Versicherte der unveränderliche Anteil der KVN
urn:telematik:claims:profession	professionOID	OID für Versicherte

[&lt;=]

**A\_28144 -PDP Authorisation Server - Länge der Nonce**

Der ZETA Guard MUSS am Endpunkt GET /nonce einen 128 Bit langen base64url kodierten Zufallswert ausgeben. [<=]

**A\_28440 -PDP Authorization Server - Auswertung Forwarded-Header**

Die Komponenten PDP Authorization Server MUSS HTTP-Anfragen mit einem vorhandenen Forwarded-Header auswerten, um die Client-IP Adresse zu ermitteln. Bei der Auswertung ist die Semantik gemäß [RFC 7239] und die\* Reihenfolge der Parameter zu beachten. [<=]

**A\_25644 -PDP Client-Registrierung mit Attestation**

Die Komponente Authorization Server MUSS die Clients/Apps bei der Registrierung über folgende Mechanismen attestieren:

- Android Key ID Attestation (für Google-Android Clients)
- Apple DCAAppAttest.
- Client signiertes Client Assertion JWT (mit TPM Attestation oder Software Attestation für Windows und Linux Clients)

~~Client-Zertifikat plus Client Assertion JWT (für Dienst-zu-Dienst Kommunikation)~~ [<=] {<=}

**A\_25645 -PDP Authentifizierung mittels TI-Smartcard**

Die Komponente Authorization Server MUSS die Authentifizierung per Token Exchange eines AccessID Tokens mit Signatur einer SM(C)-B unterstützen. [<=]

**A\_25649 -PDP Authorization Server - Regelmäßige Wiederholung der Attestation**

Die Komponente Authorization Server SOLL die Client-Attestierung für jede neue Session verlangen. [<=]

*Hinweis: Eine Session des Authorization Servers ist gültig vom Zeitpunkt  $t_1$  = (Ausstellungszeitpunkt des ersten Refresh Token nach vollständiger Authentifizierung) bis zum Zeitpunkt  $t_2 = t_1 +$  (Gültigkeitsdauer des ersten Refresh Token). Bei mobilen Clients werden für die Attestation Services des mobilen Betriebssystems verwendet, die ein Rate Limit haben können. In diesem Fall kann die Attestation eventuell nicht für jede neue Session wiederholt werden.*

**A\_25650 -PDP Client Registrierung - TI-Identität in Attestation**

Die Komponente Authorization Server MUSS den registrierten Client an eine TI-Identität (KVN-R oder TelematikID, festgestellt während der Authentifizierung) binden. [≤]

*Offener Punkt: Wie im Fall der Dienst-zu-Dienst Kommunikation die Bindung an eine Dienst-Identität erfolgt, wird in einer folgenden Version des Dokuments festgelegt (z. B. über Client-Registry, JWKS und Policy Engine oder [SPIFFE und SPIRE]).*

**A\_25651 -PDP Client-Registrierung - Offband Nutzer Verification**

Die Komponente Authorization Server MUSS einen Offband Prozess (per E-Mail) für die Kommunikation mit diesem Nutzer unterstützen (Trust on First Use), wobei der Nutzer seine E-Mail Adresse eigenverantwortlich vergibt. [≤]

**A\_25653 -PDP Client-Registrierung - Umsetzung der Client Policy**

Die Komponente Authorization Server MUSS die zulässigen Clients entsprechend der Konfiguration oder Policy (über PDP-Decision) ermitteln und nur diese gemäß der festgelegten Client Policy registrieren. Clients, die die geforderten Parameter der Client Policy nicht unterstützen bzw. nicht das geforderte Niveau erreichen, MÜSSEN abgelehnt werden. [≤]

**A\_25752 -PDP Client-Registrierung - Nutzer über Hintergrund zur Ablehnung der Clientregistrierung informieren**

Falls ein Client nicht die geforderten Parameter der Client Policy unterstützt bzw. das geforderte Niveau nicht erreicht, MUSS die Komponente Authorization Server den Nutzer nutzerfreundlich darüber informieren, welche Clienteneigenschaften zu der Ablehnung geführt haben. [≤]

**A\_25738 -PDP Client-Registrierung - Telemetrie**

Die Komponente Authorization Server MUSS in den Telemetrie-Daten zu jeder versuchten Clientregistrierung folgende Parameter ohne einen Nutzerbezug protokollieren:

- Clientparameter (Betriebssystem(-version), Patchlevel, Geolocation etc.) gemäß Clientattestierung
- verwendeter Faktor für Offband-Verifikation (E-Mail)
- Zeitstempel Registrierung, Zeitpunkt Offband-Bestätigung
- verwendeter Faktor der Nutzerauthentifizierung (SmartCard, Digitale Identität)
- Status/Ergebnis des Registrierungsversuchs

[≤]

**A\_25754 -PDP Client-Registrierung - Notfall-Recovery-Prozess für Nutzer**

Die Komponente Authorization Server MUSS dem Nutzer einen Notfall-Recovery-Prozess anbieten, falls der Nutzer sein letztes Gerät verloren und keinen Zugriff mehr auf seine registrierte E-Mail-Adresse hat. [≤]

**A\_26585 -PDP Client-Registrierung - Client-Daten**

Die Komponente Authorization Server MUSS die Client-Daten gemäß [client-instance.yaml] in der PDP Datenbank verwalten.

Die Client-Daten MÜSSEN bei Anfragen des Authorization-Servers an die Policy Engine im Request mit übergeben werden. [≤]

### 5.5.2.1 PDP Relying Party

#### A\_25655 -PDP - Relying Party

Der PDP Authorization Server MUSS in der TI-Föderation als Relying Party registriert sein. [≤]

#### A\_25656 -PDP - Entity Statement

Der PDP Authorization Server MUSS die Redirect-URLs aller zulässigen Clients als erlaubte Redirect-URLs im Entity Statement ausweisen. [≤]

#### A\_25657 -PDP - Authentication über sektoralen IDP

Der PDP Authorization Server MUSS die Nutzer über sektorale IDPs authentifizieren können. [≤]

#### A\_25658 -PDP - Authentication über SmartCard IDP

Der PDP Authorization Server MUSS die Nutzer über den zentralen IDP-Dienst (SmartCard-IDP) authentifizieren können. [≤]

### 5.5.2.2 Ablauf für den Zugriff auf einen Resource Server

Um auf einen durch ZETA Guard geschützten Resource Server zugreifen zu können, müssen abhängig vom Zustand des ZETA Clients verschiedene Teilabläufe ausgeführt werden, oder können übersprungen werden. Die ZETA API besteht aus mehreren Endpunkten, die verschiedene Funktionen bereitstellen. Diese Endpunkte sind in verschiedene Unter-Abläufe aufgeteilt:

- **Konfiguration und Discovery:** Der ZETA Client muss die Konfiguration des ZETA Guards ermitteln, um die richtigen Endpunkte zu erreichen.
- **Client-Registrierung:** Jeder ZETA Client muss sich einmalig beim ZETA Guard registrieren, um eine `client\_id` zu erhalten und seinen öffentlichen Schlüssel zu hinterlegen.
- **Authentifizierung und Autorisierung:** Der Client muss sich authentifizieren und die Integrität seiner Plattform nachweisen. Zusätzlich muss sich der Nutzer oder beim Primärsystem die Organisation authentifizieren, um ein Access Token für den Zugriff auf geschützte Ressourcen zu erhalten.

Der Gesamtprozess beginnt damit, dass ein **Nutzer** auf einen Endpunkt eines Resource Servers zugreifen möchte. Dieser Zugriff wird über das Primärsystem vom **ZETA Client** im Auftrag des Nutzers ausgeführt; siehe folgende Abbildung.

#### A\_27797 -ZETA, Ablauf Zugriff auf geschützte Ressource

Der ZETA Guard und der ZETA Client MÜSSEN den Ablauf gemäß Abbildung *Abb\_ZETA\_Zugriff\_auf\_geschützte\_Ressource* unterstützen.

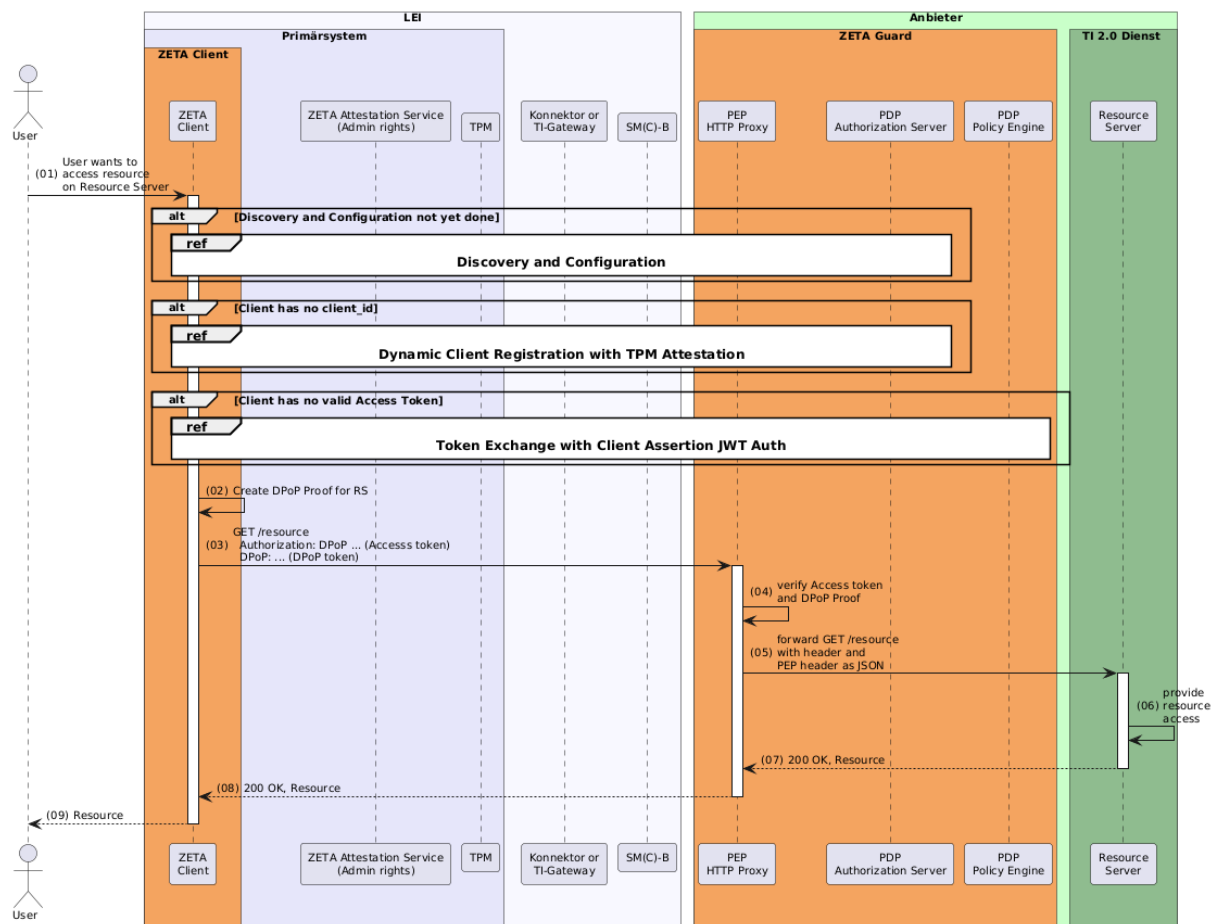


Abbildung 3: Abb\_ZETA\_Zugriff\_auf\_geschützte\_Ressource

[<=]

### 5.5.2.3 Service Discovery

Der ZETA Guard ermöglicht ZETA Clients die Ermittlung der bereitgestellten Endpunkte durch Abfrage von Well-known JSON Dokumenten.

#### A\_27798 -ZETA Guard, Service Discovery

Der ZETA Guard MUSS die Well-known JSON Dokumente für die Service Discovery gemäß Abb\_Service\_Discovery bereitstellen.

Das Protected Resource Well-known JSON Dokument MUSS mit dem Schema [opr-well-known.yaml] validiert werden können.

Das Authorization Server Well-known JSON Dokument MUSS mit dem Schema [as-well-known.yaml] validiert werden können.

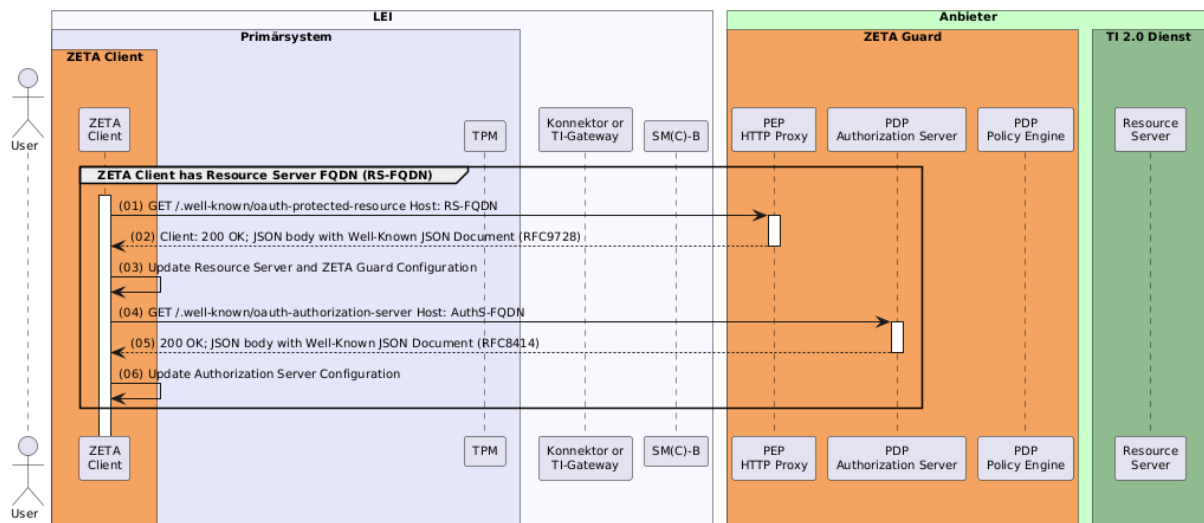


Abbildung 4: Abb\_Service\_Discovery

[&lt;=]

### A 28420 -ZETA Guard, Service Discovery – Bereitstellung ETag Header

Der ZETA Guard MUSS für jedes Well-known JSON-Dokument einen eindeutigen ETag-Header generieren und diesen gemäß [RFC 9110] in der HTTP-Antwort bei jeder Anfrage bereitstellen. Bei jeder inhaltlichen Änderung oder der Repräsentation der Well-known JSON-Dokumente MUSS der ETag sich ändern.[<=]

### A 28421 -ZETA Guard, Service Discovery – Unterstützung If-None-Match-Header

Der ZETA Guard MUSS Client-Anfragen, die den If-None-Match-Header enthalten, korrekt verarbeiten. Die nachfolgende Fallunterscheidung ist zu beachten:

- Keine Änderung des Well-known JSON Dokuments:  
Wenn der vom ZETA Client im If-None-Match -Header gesendete ETag mit dem aktuellen ETag des Dokuments auf dem Server übereinstimmt, MUSS der ZETA Guard mit einem HTTP-Statuscode 304 (Not Modified) antworten und darf keinen Nachrichtentext senden.
- Änderung des Well-known JSON Dokuments:  
Wenn der vom ZETA Client im If-None-Match-Header gesendete ETag nicht mit dem aktuellen ETag des Dokuments auf dem Server übereinstimmt (oder wenn der Header fehlt), MUSS der ZETA Guard mit einem HTTP-Statuscode 200 OK antworten und das vollständige, aktuelle Well-known JSON-Dokument im Nachrichtentext sowie den aktuellen ETag-Header senden.

[&lt;=]

### A 28422 -ZETA Guard, Service Discovery – Cache-Control-Header Direktiven

Der ZETA Guard MUSS im HTTP-Response-Header zur Anfrage der Well-Known und JWKS JSON-Dokumente einen Cache-Control-Header einfügen, der die Direktiven max-age und public setzt.

Die Dauer der Direktive max-age MUSS konfigurierbar sein (Default: 86400 s) .[<=]

Die Verwendung der Kombination der Direktiven public und max-age im Cache-Control-Header sorgt für effizientes Caching mit garantierter Aktualität der Inhalte. In Kapitel 5.8.7- Betriebliche Schnittstellendefinition sind die Well-known und JWKS Endpunkte gelistet.

### 5.5.2.4 Client-Registrierung für stationäre Clients

Jeder ZETA Client muss sich am ZETA Guard registrieren, über den er auf geschützte Ressourcen zugreifen möchte. Dieser Prozess findet einmalig pro ZETA Guard-Instanz statt. Der gesamte Prozess ist zweistufig, um die administrative Einrichtung von der technischen Inbetriebnahme zu trennen:

- **Initiale Registrierung:** Der Client erzeugt ein langlebiges kryptographisches Schlüsselpaar (Client Instance Key), sendet den öffentlichen Teil an den Authorization Server und erhält im Gegenzug eine `client_id`. Der Client ist danach im System bekannt, aber sein Status ist `pending_attestation`, d.h. er ist noch nicht für den Zugriff auf Ressourcen freigeschaltet.

- **Aktivierung (Erster Token Exchange):** Der Client wird aktiviert, indem er zum ersten Mal einen Token Exchange mit einer erfolgreichen Attestierung durchführt. Damit beweist er nicht nur den Besitz des privaten Schlüssels, sondern (bei der TPM-Attestierung) auch die Integrität der Plattform, auf der er läuft. Nach erfolgreicher Prüfung wird sein Status im ZETA Guard auf `active` gesetzt.

### A\_27799 -ZETA Guard, Client-Registrierung für stationäre Clients

Der ZETA Guard MUSS die Client-Registrierung für stationäre Clients gemäß Abbildung *Abb\_Client\_Registrierung* bereitstellen.

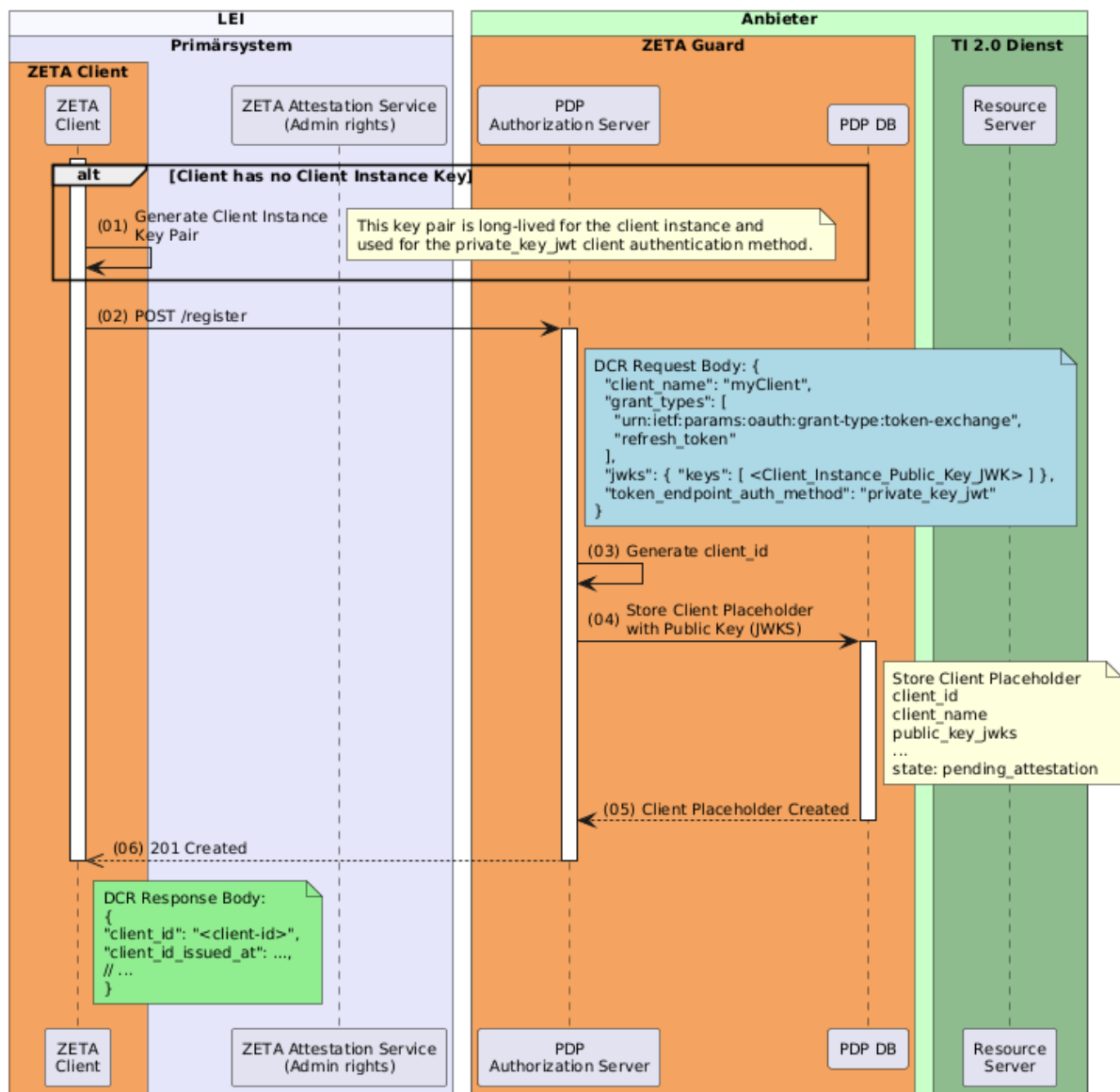


Abbildung 5-1: Abb\_Client\_Registrierung

[&lt;=]

Für die initiale Registrierung sendet der ZETA Client eine Anfrage an den Dynamic Client Registration (DCR) Endpoint. Diese Anfrage enthält alle notwendigen Metadaten, um den Client für die `private_key_jwt` Authentifizierungsmethode vorzubereiten:

- `client_name`: Ein für Menschen lesbarer Name für den Client.
- `token_endpoint_auth_method`: Die geplante Authentifizierungsmethode, hier `private_key_jwt`.
- `grant_types`: Die erlaubten Grant Types (z.B. `urn:ietf:params:oauth:grant-type:token-exchange`, `refresh_token`).
- `jwks`: Ein JSON Web Key Set, das den **öffentlichen Client Instance Key** enthält.

Dieser Schlüssel wird vom Authorization Server verwendet, um die Signatur der Client Assertions zu überprüfen.



### 5.5.2.5 Authentifizierung und Autorisierung für stationäre Clients

Nach erfolgreicher Registrierung besitzt der ZETA Client eine `client_id` und ein zugehöriges Schlüsselpaar. Um auf einen Fachdienst zugreifen zu können, benötigt der Client ein Access Token vom Authorization Server (AS). Stationäre ZETA Clients verwenden dafür den Token Exchange Flow, während mobile ZETA Clients den Authorization Code Flow mit OpenID Connect nutzen.

Die Authentifizierung und Autorisierung für stationäre Clients unterscheidet zwei Hauptfälle:

1. **Token-Austausch mit Attestierung:** Hier wird die Identität der Institution (mittels `subject_token` von der SM(C)-B) nachgewiesen und die Integrität des Clients durch eine Attestierung überprüft. Dieser aufwändigere Prozess wird zu Beginn einer neuen Session (oder zur Re-Attestierung) durchgeführt, um sicherzustellen, dass der ZETA Client und das Primärsystem vertrauenswürdig sind.

2. **Token-Erneuerung (Refresh Token):** Hier wird ein vorhandenes Refresh Token genutzt, um ein neues Access Token zu erhalten. Dieser Prozess ist performanter und verzichtet auf eine erneute Attestierung.

Diese Trennung schafft eine Balance zwischen höchster Sicherheit beim initialen Zugriff und Effizienz bei der Erneuerung bestehender Sitzungen.

Die folgende Abbildung zeigt den Ablauf des Token-Austauschs mit Client Assertion JWT Authentifizierung und DPoP.

#### **A 27800-01A\_27800 -ZETA Guard, Authentifizierung und Autorisierung für stationäre Clients**

Der ZETA Guard MUSS die Client-Registrierung für stationäre Clients gemäß Abbildung *Abb\_Authentifizierung\_und\_Autorisierung* bereitstellen.

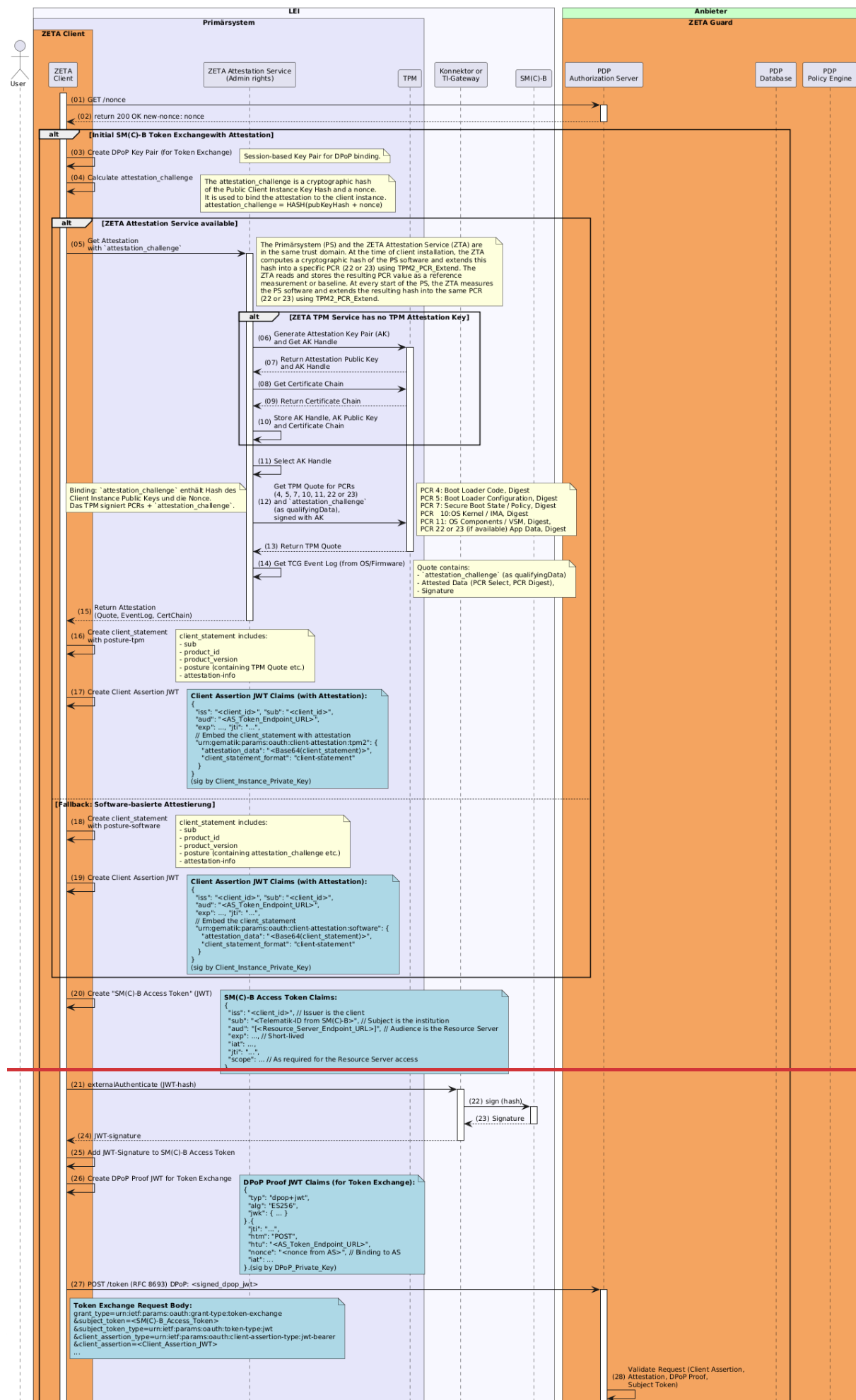




Abbildung 6+ :Abb\_Authentifizierung\_und\_Autorisierung

[&lt;=]

#### 5.5.2.5.1 Pfad A: Token-Austausch mit Attestierung

Dieser Pfad wird beschritten, wenn der Client keine bestehende Session (d.h. kein gültiges Refresh Token) hat.

##### 1. Vorbereitung:

- Der Client fordert eine frische, einmalig gültige `nonce` vom Authorization Server an (GET /nonce).
- Der Client erzeugt ein temporäres, nur für diese Session gültiges DPoP-Schlüsselpaar.

##### 2. Integritätsprüfung und kryptografische Bindung:

- Um zu beweisen, dass die Attestierung für genau diesen Client und diese Transaktion erstellt wurde, erzeugt der Client eine `attestation_challenge`. Diese bindet den Zustand des TPMs an den **öffentlichen Client Instance Key** und die `nonce` des AS:  
`attestation_challenge = HASH( HASH(Client_Instance_Public_Key_JWK) + nonce )`.

- Der Client fordert beim ZETA Attestation Service eine TPM Quote an, die diese `attestation_challenge` als `qualifyingData` enthält. Das TPM signiert somit eine Aussage, die mit der Identität des Clients verbunden ist.

**3. Erstellen des Client Statement:** Die Attestierungsartefakte (TPM Quote, Event Log, Zertifikatskette) werden in eine `client_statement`-Struktur gepackt. Im Falle des Fallbacks (Software-Attestierung) enthält diese Struktur andere, softwarebasierte Evidenz.

**4. Erstellen der Client Assertion (mit Attestierung):** Für die Authentifizierung am Token-Endpoint erstellt der Client eine **Client Assertion**. Dieses JWT, mit dem **privaten Client Instance Key** signiert, dient als "Umschlag":

- Es authentifiziert den Client gegenüber dem AS (iss und sub sind die `client_id`).
- Es enthält die `client_statement`-Struktur als Beweis für die Clientintegrität, verpackt in einem spezifischen Claim (`urn:gematik:params:oauth:client-attestation:tpm2` oder `...:software`).

```
// Client Assertion für initialen Token-Austausch (Beispiel TPM)
{
  "iss": "<client_id>",
  "sub": "<client_id>",
  "aud": "<AS_Token_Endpoint_URL>",
  "exp": "...",
  "jti": "...",
  // Kapselung des Attestierungsnachweises
  "urn:gematik:params:oauth:client-attestation:tpm2": {
    "attestation_data": "<Base64(client_statement)>",
    "client_statement_format": "client-statement"
  }
}
```

```
}
}
```

**5. Authentisierung der Institution (SM(C)-B Token):** Parallel dazu erstellt der Client das `subject_token`. Dies ist ein vom ZETA Client erzeugtes JWT, dessen Hash vom Konnektor mittels der SM(C)-B signiert wird und die Identität der Institution (z.B. Praxis) belegt. Die Audience (``aud``) dieses Tokens ist der Ziel-Fachdienst (Resource Server).

**6. Token Request:** Der Client sendet eine POST-Anfrage an den `/token`-Endpoint, die alle Teile kombiniert: `grant_type=token-exchange`, das `subject_token`, die `client_assertion` (mit der eingebetteten Attestierung) und den DPoP-Proof.

**7. Validierung durch den AS:** Der AS führt eine umfassende Prüfung durch: Validierung der Client Assertion (Signatur gegen den bei der DCR hinterlegten Public Key), des DPoP-Proofs, des Subject Tokens und insbesondere der **eingebetteten Attestierung** (Prüfung der Quote, der `attestation_challenge` und der PCR-Werte gegen die Sicherheits-Policy).

*Hinweis: Für die Signaturprüfung der Quote muss der Authorization Server die vertrauenswürdigen TPM Stamm- und Zwischensignaturzertifikate installieren, die zum Signieren des Endorsement Key in den TPMs verwendet werden. Eine Liste der CAs wird hier bereitgestellt: [TrustedTPM\_RootCA\_und\_IntermediateCA].*

#### 5.5.2.5.2 Pfad B: Token-Erneuerung via Refresh Token

Dieser effiziente Pfad wird genutzt, wenn ein gültiges Refresh Token vorhanden ist.

**1. Erstellen der Client Assertion (ohne Attestierung):** Der Client erstellt eine einfache `client_assertion`. Sie beweist durch ihre Signatur mit dem Client Instance Key die Identität des Clients. Diese Assertion enthält keine Attestierungsdaten.

```
// Client Assertion für Refresh-Token-Nutzung
{
  "iss": "<client_id>",
  "sub": "<client_id>",
  "aud": "<AS_Token_Endpoint_URL>",
  "exp": ...,
  "jti": "..."
```

**2. Token Request:** Der Client sendet eine POST-Anfrage an den `/token`-Endpoint mit `grant_type=refresh_token`, dem Refresh Token und der einfachen `client_assertion`.

**3. Validierung durch den AS:** Der AS validiert das Refresh Token, die Signatur der Client Assertion und den DPoP-Proof. Die Prüfung einer TPM-Attestierung entfällt. Bei Erfolg wird das alte Refresh Token invalidiert (Rotation).

#### 5.5.2.5.3 Gemeinsame nachfolgende Schritte

Nach erfolgreicher Validierung in einem der beiden Pfade fragt der AS bei der Policy Engine (PE) an, ob der Zugriff gewährt werden soll. Ist die Entscheidung positiv, stellt der AS ein neues Access Token (gebunden an den DPoP-Schlüssel) und ein neues Refresh Token aus.

### 5.5.2.6 Ablauf der Authentifizierung bei Dienst-zu-Dienst Kommunikation

*Offener Punkt: Grundsätzlich gilt, dass Dienste, die auf einen TI 2.0 Dienst zugreifen, sich mit mTLS (z. B. mit SPIFFE und SPIRE) oder Client Assertion JWT + DPoP (ähnlich wie SM(C)-B Authentifizierung mit DPoP.; anstatt der SM(C)-B wird ein Signaturschlüssel des Dienstes verwendet) authentifizieren müssen. Details zum Ablauf der Authentifizierung bei Dienst-zu-Dienst Kommunikation und zu den Schlüsseln werden in einer folgenden Version des Dokuments festgelegt.*

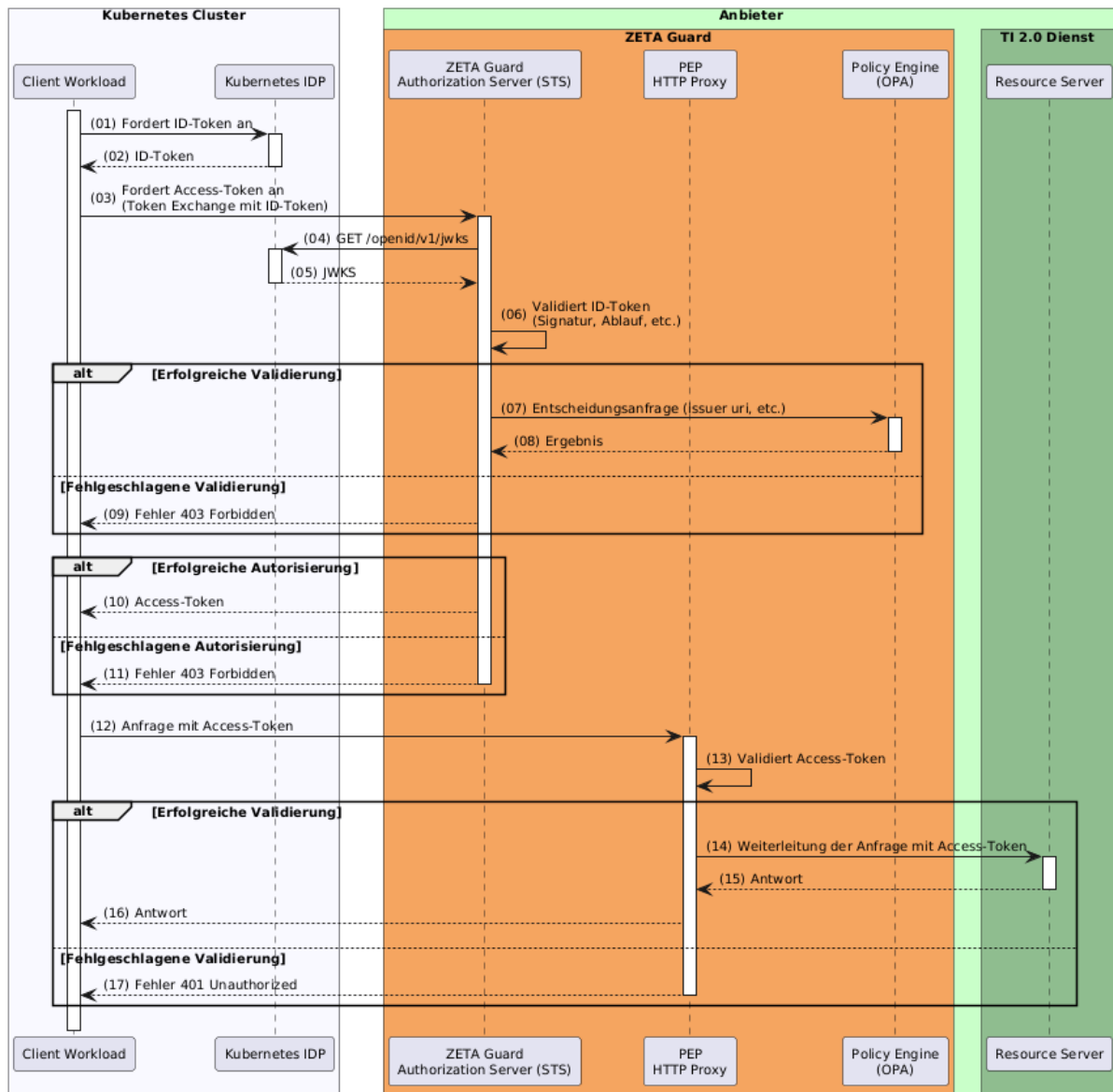
Um externen Diensten Zugriff auf den Resource Server zu ermöglichen, wird (in ZETA Stufe 2) Workload Identity Federation verwendet. Jeder Kubernetes Cluster, der ZETA Guard ausführt, stellt einen Identity Provider (IDP) bereit, der ID-Token für die im Cluster ausgeführten Workloads ausstellen kann.

Der ZETA Guard Authorization Server dient als Secure Token Service (STS) und kann per Token Exchange ein ID-Token gegen ein Access-Token austauschen. Er überprüft dabei die Gültigkeit des ID-Tokens, insbesondere ob der `issuer uri` des ausstellenden IDP registriert ist. Die Policy Engine (OPA) enthält die Liste der registrierten `issuer uri` und weiterer Attribute, die zur Autorisierung herangezogen werden.

Der folgende Ablauf beschreibt die Schritte, die für eine erfolgreiche Dienst-zu-Dienst-Kommunikation notwendig sind:

- **Registrierung des Issuers:** Die issuer URI des externen Kubernetes IDP, in dem die Client Workload läuft, wird bei der gematik registriert. Eine Policy zur Überprüfung und die issuer URI werden in die PIP und PAP Daten des Ziel-ZETA Guard aufgenommen.
- **ID-Token-Anforderung:** Die Client Workload, oder eine andere Workload im externen Kubernetes Cluster, fordert zunächst ein ID-Token von seinem Kubernetes IDP an. Dieses ID-Token enthält Informationen über die Identität der Workload.
- **Token Exchange-Anfrage an ZETA Guard:** Die externe Workload sendet das erhaltene ID-Token an den ZETA Guard Authorization Server (STS) und fordert im Austausch ein Access-Token an. Dieser Prozess wird als "Token Exchange" bezeichnet.
- **Validierung des ID-Tokens und Autorisierungsprüfung durch OPA:** Der ZETA Guard Authorization Server validiert die Signatur und die Standard-Claims des ID-Tokens. Anschließend fragt er die Policy Engine (OPA) an, um zu überprüfen, ob der Aussteller (issuer URI) des ID-Tokens vertrauenswürdig ist und ob die im Token enthaltenen Attribute den definierten Richtlinien für den Zugriff auf den angefragten Resource Server entsprechen. Die Policy Engine enthält hierfür eine Liste der zuvor registrierten issuer URI.
- **Ausstellung des Access-Tokens:** Nach erfolgreicher Überprüfung durch OPA stellt der ZETA Guard Authorization Server ein Access-Token aus. Dieses Token ist für den Zugriff auf den spezifischen Resource Server vorgesehen.
- **Zugriff auf den Resource Server:** Die Client Workload im externen Dienst verwendet das erhaltene Access-Token, um Anfragen an den Resource Server zu stellen.
- **Validierung des Access-Tokens durch den PEP HTTP Proxy:** Der PEP HTTP Proxy validiert bei jeder Anfrage das mitgesendete Access-Token durch Überprüfung der Signatur des ZETA Guard Authorization Servers sowie der `aud` und `scope claims`.

- **Zugriffsentscheidung und Antwort:** Bei gültigem Access-Token gewährt der PEP HTTP Proxy den Zugriff und verarbeitet die Anfrage. Andernfalls wird der Zugriff verweigert und eine entsprechende Fehlermeldung zurückgegeben.



**Abbildung 7 Abb ZETA-Guard-Dienst-zu-Dienst-Kommunikation**

### A 28431 -ZETA Guard, Ablauf Dienst-zu-Dienst Kommunikation

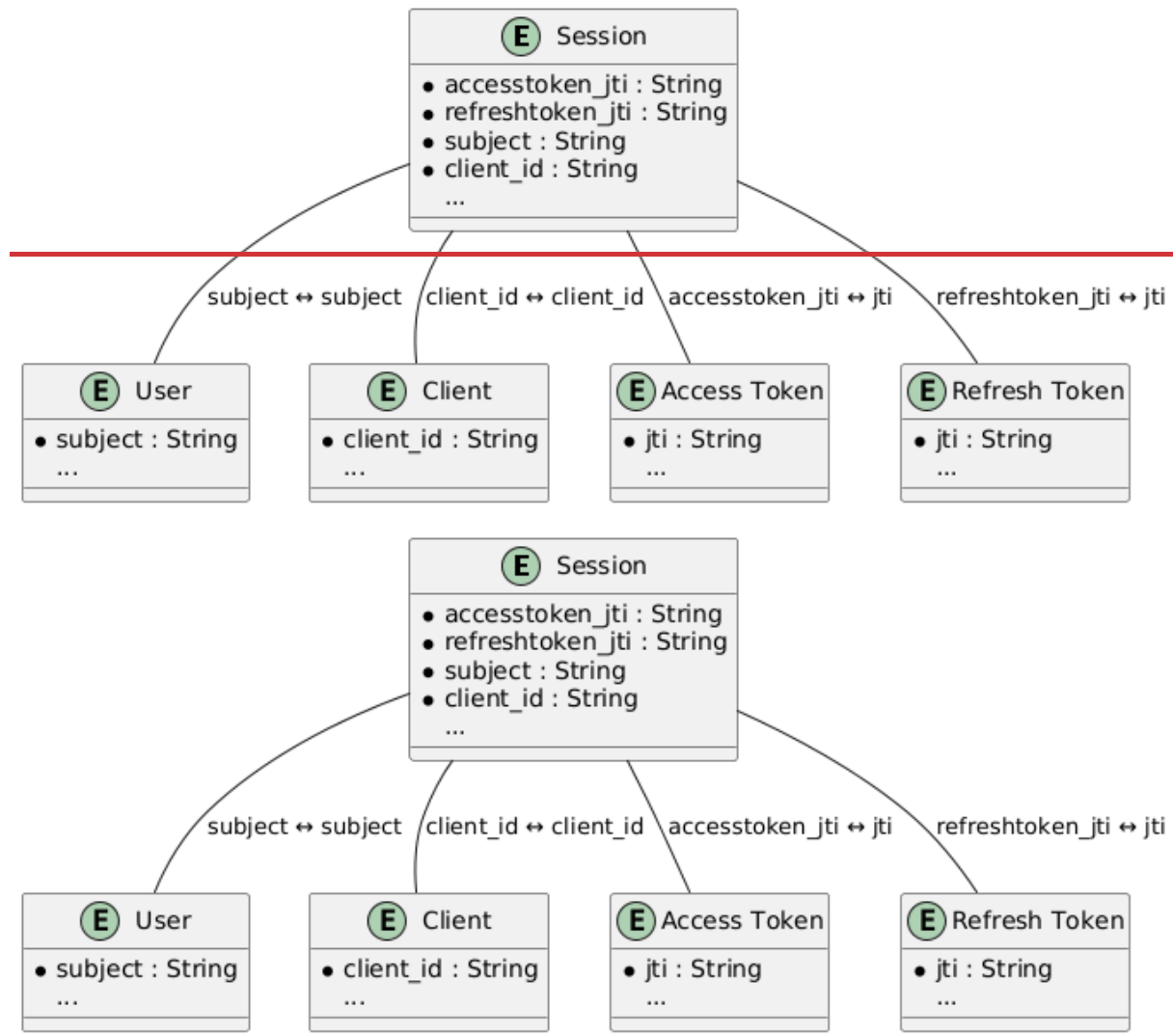
Der ZETA Guard MUSS den Ablauf gemäß Abbildung Abb ZETA-Guard-Dienst-zu-Dienst-Kommunikation unterstützen. [ $\leq$ ]

## 5.5.3 PDP Datenbank

Die Datenbank speichert Session-, Nutzer- und Client-Daten. Die Struktur der Daten ist in den Schemadateien [session.yaml], [user-info.yaml], und [client-instance.yaml] festgelegt.

Die Session Daten gemäß [session.yaml] enthalten die Attribute `Access Token_jti` zur Verlinkung der Session mit dem aktuell gültigen Access Token (Attribut `jti`) und `Refresh`

Token\_jti zur Verlinkung der Session mit dem aktuell gültigen Refresh Token (Attribut jti). Das Attribut subject enthält die Verlinkung mit den User-Daten (Attribut subject) und das Attribut client\_id enthält die Verlinkung mit den Client-Daten (Attribut client\_id).



**Abbildung 8: Beziehungen zwischen Session-, Nutzer- und Client-Daten sowie Token**

### A\_26587 -PDP Datenbank - Schnittstellen

Die Komponente PDP Datenbank MUSS Schnittstellen anbieten, sodass

- der Authorization Server die Operationen create, retrieve, update und delete für Session-Daten ausführen kann,
- der Authorization Server die Operationen create, retrieve, update und delete für Session-, User- und Client-Daten ausführen kann,
- der PEP HTTP Proxy die Operation retrieve für User-Daten und optional im gleichen Request für Client-Daten ausführen kann. Anhand des Parameters jti aus dem Access Token des Requests MUSS der HTTP Proxy die User- und Client-Daten der zugehörigen Session aus der Datenbank abfragen können.

[<=]



## **A\_26588 -PDP Datenbank - Konfiguration**

Die Komponente PDP Datenbank MUSS so konfigurierbar sein, dass für alle Attribute der Client-Daten eingestellt werden kann, welche Attribute an den PEP HTTP Proxy ausgegeben werden.

Die default Einstellung ist, dass die Attribute `platform`, `product_name` und `product_version` sowie aus dem Objekt `posture` die Attribute `system_name`, `system_version` und `device_model` ausgegeben werden. [ $\leq$ ]

## **A\_27399 -PDP Datenbank - Löschfristen**

Die Komponente PDP Datenbank MUSS konfigurierbare Inaktivitäts-Löschfristen für User- und Client-Daten haben und die Daten nach Ablauf der Fristen löschen. Der Default-Wert beträgt 1 Jahr.

Die Session-Daten MÜSSEN automatisch gelöscht werden, wenn der Zeitpunkt `session_expiry` erreicht ist.

[ $\leq$ ]

## **5.5.4 Sicherheits- und Datenschutzanforderungen an den PDP**

### **A\_25449 -PDP- Nutzeridentität nur von einem zugelassenem IDP**

Der PDP MUSS sicherstellen, dass nur Nutzeridentitäten von einem zugelassenen IDP akzeptiert werden. [ $\leq$ ]

### **A\_25447 -PDP Authorization Server - Kommunikation nur mit authentischer Policy Engine**

Der PDP Authorization Server MUSS sicherstellen, dass er mit einer authentischen und korrekt konfigurierten Policy Engine kommuniziert. [ $\leq$ ]

### **A\_26064 -Access Token bei Monitoring-Signalen sperren**

Falls das Monitoring System eine Änderung in den Kommunikationsmerkmalen signalisiert, muss der PDP das aktuelle Access Token sperren. [ $\leq$ ]

*Hinweis: der Client muss danach ein neues Access Token beim Authorization Server abfragen. Die Abfrage des neuen Access Token beinhaltet immer eine Entscheidung durch den PDP.*

### **A\_25486 -PDP - Abbruch durch Anomalie Signale**

Falls das Security Monitoring eine Anomalie beim Zugriff eines Clients per Shared Signals signalisiert, MUSS der PDP Authorization Server das Access Token des Clients annullieren und damit die aktuelle fachliche Operation abbrechen. [ $\leq$ ]

### **A\_26269 -PDP - Tamper-Proof Protokollierung von Administrationsaktivitäten**

Der PDP MUSS ein "Tamper-Proof" Audit-Log von allen administrativen Vorgängen umsetzen. [ $\leq$ ]

### **A\_26281 -PDP - kurzlebige Zertifikate**

Der PDP Authorization Server MUSS für die Signatur von Token Self-signed Zertifikate verwenden.

Die Gültigkeitsdauer der Signatur-Zertifikate MUSS in den ZETA Guard-Manifest-Dateien konfigurierbar sein. [ $\leq$ ]

### **A\_25751 -PDP Client-Registrierung - Anwendungsfälle nur für registrierte mobile Clients**

Nach der erfolgreichen Registrierung des ersten mobilen Clients (Gerät/App Kombination), MUSS die Komponente Authorization Server sicherstellen, dass die folgenden Anwendungsfälle nur von einem registrierten mobilen Client durchgeführt werden können:

- Client löschen
- Client umbenennen
- E-Mail-Adresse hinzufügen
- E-Mail-Adresse aktualisieren

[<=]

## **A\_25748 -PDP Client-Registrierung - Maximale Anzahl von Clients**

Die Komponente Authorization Server MUSS sicherstellen, dass ein Nutzer maximal 256Clients registrieren kann. Der Wert muss konfigurierbar sein.[<=]

## **A\_25749 -PDP Client-Registrierung - Nutzer Protokollierung**

Die Komponente Authorization Server MUSS ein Nutzerprotokoll führen und die folgenden Anwendungsfälle für den Nutzer protokollieren:

- Client hinzufügen
- Client löschen
- Client umbenennen
- E-Mail-Adresse hinzufügen
- E-Mail-Adresse aktualisieren

[<=]

## **A\_25750 -PDP Client-Registrierung - Nutzer über sicherheitsrelevante Ereignisse informieren**

Die Komponente Authorization Server MUSS sicherstellen, dass der Nutzer bei folgenden Anwendungsfällen informiert wird:

- Client hinzufügen
- Client löschen
- Client umbenennen
- E-Mail-Adresse aktualisieren

[<=]

*Hinweis: Der Nutzer kann z.B. durch eine geeignete E-Mail oder App-Notifikation über die sicherheitsrelevanten Ereignisse informiert werden.*

## **5.5.5 Konfiguration**

### **~~A\_26038 -PDP, Konfigurations-Parameter~~**

~~Der PDP MUSS die folgenden Konfigurations-Parameter unterstützen.~~

**~~Tabelle 11: PDP - Konfigurations-Parameter~~**

<b>Konfigurations-Parameter</b>	<b>Default Wert</b>	<b>Beschreibung</b>
<del>as-fqdn</del>	<del>n/a</del>	<del>FQDN des PDP Authorization Servers</del>

Konfigurations-Parameter	Default Wert	Beschreibung
scopes_supported		<p>vom PDP Authorization Server unterstützte scope Werte</p> <p>Minimal müssen die Zero Trust scope Werte unterstützt werden:</p> <ul style="list-style-type: none"> <li>—zero:register</li> <li>—zero:manage</li> </ul> <p>Zusätzliche scope Werte ergeben sich aus dem Bedarf des Dienstes, der durch den ZETA Guard geschützt wird.</p>

[&lt;=]

*Offener Punkt: Weitere Konfigurationsparameter werden in einer zukünftigen Version des Dokuments festgelegt und hier ergänzt.*

## 5.6 PIP und PAP Service

Der PIP und PAP Service stellt Die Policies und Daten werden in der ZETA Artifact Registry als OPA Bundles im OCI Format für die PDP Policy Engine Instanzen Engines der ZETA Guard Instanzen bereitgestellt. Die der Dienste der TI 2.0 bereit.

### ~~A\_25670 PIP und PAP Bereitstellung Download Endpunkt~~

Der PIP und PAP Service MUSS Download Endpunkte für OPA Bundles gemäß OpenAPI Spezifikation [pip-pap-service.yaml] Version 1.0.0 werden in den folgenden Instanzen bereitstellen:

Produktions-Instanz: <https://pip-pap.ti-dienste.de>

Referenz-Instanz: <https://pip-pap-ref.ti-dienste.de>

Test-Instanz: <https://pip-pap-test.ti-dienste.de>. [ <= ]

*Hinweis: Die Bereitstellung der Test-Instanz erfolgt nur während einer Testphase und kann eine andere Version der [pip-pap-service.yaml] unterstützen. Für die Entwicklung und Tests anderer Komponenten wird empfohlen, die Referenz-Instanz zu verwenden.*

### ~~A\_25671 PIP und PAP TLS am Download Endpunkt~~

Der PIP und PAP Service MUSS sich beim TLS Verbindungsaufbau am Download-Endpunkt gegenüber Clients mit einem Extended Validation TLS Zertifikat eines TSP gemäß [CAB Forum] authentisieren. [ <= ]

~~A\_27395 PIP und PAP OCSP Stapling~~ GitOps CI Prozess mit Quality Gates entwickelt und Der PIP und PAP Service SOLL für die von außen kommenden TLS Verbindungen OCSP Stapling [RFC6066] verwenden.

Das OCSP Abfrageintervall MUSS das Minimum aus dynamic\_validity\_period und max\_allowed\_validity\_period sein, wenn NextUpdate in der OCSP Response angegeben ist. Anderenfalls MUSS das OCSP Abfrageintervall gleich max\_allowed\_validity\_period sein.

Dabei gilt:

~~dynamic\_validity\_period = (percentage\_of\_validity\_period \* (NextUpdate — ThisUpdate)) / 100~~

Die Werte für max\_allowed\_validity\_period (default Wert = 1 Tag)

~~und percentage\_of\_validity\_period (default Wert = 60%) MÜSSEN konfigurierbar sein.~~

~~Die jeweils letzte OCSP-Response MUSS im Cache gespeichert und für OCSP-Stapling verwendet werden.~~

~~Es MUSS eine Warnung bereitgestellt werden, wenn eine OCSP-Abfrage fehlschlägt. Sollte vom entsprechenden OCSP-Responder trotz regelmäßigen Versuchs keine OCSP-Response bezogen werden können, so MUSS die jüngste zur Verfügung stehende OCSP-Response verwendet werden und es MUSS mit exponential Backoff weiter probiert werden. [<=]~~

die ZETA Artifact Registry deployed.

### **A\_25672 -PIP und PAP - Kompatibilität mit OPA Bundles**

Der PIP und PAP Service MUSS die Policies und Daten als [OPA Bundle] bereitstellen. [<=]

### ~~**A\_25680 -PIP und PAP - download-path**~~

~~Der PIP und PAP Service MUSS OPA Bundles mit dem filename = bundle.tar.gz unter dem Pfad /policies/{application}/{label} bereitstellen, wobei mindestens die label "latest" und "latest-sim" pro application angeboten werden. [<=]~~

~~Hinweis: Siehe [pip-pap-service.yaml]. Unter dem label "latest" werden Bundles für die aktive Policy Engine bereitgestellt. Unter dem label "latest-sim" werden Bundles für die Simulations-Policy Engine bereitgestellt.~~

~~Der PIP und PAP Service bezieht die OPA Bundles aus einem Git Repository der gematik. Die Bundles werden in einem GitOps CI Prozess mit Quality Gates entwickelt und für die Policy Engines der ZETA Guard bereitgestellt.~~

### ~~**A\_25673 -PIP und PAP - ETags für OPA Bundles**~~

~~Der PIP und PAP Service MUSS für jedes zum Download bereitgestellte OPA Bundle in der Response ein ETag Header Element gemäß [RFC7232] verwenden, das aus dem SHA-256 Hashwert der bundle.tar.gz Datei besteht. [<=]~~

~~Hinweis: Durch die Verwendung des Hashwertes der bundle.tar.gz Datei als ETag wird es möglich, den Download-Endpunkt auf mehrere Server zu verteilen. Wichtig ist nur, dass das ETag auf allen Servern gleich ist, damit bereits erhaltene OPA Bundles nicht erneut heruntergeladen werden.~~

### ~~**A\_25674 -PIP und PAP - OPA Bundle-Signaturprüfung**~~

~~Der PIP und PAP Service MUSS für alle bereitgestellten OPA Bundles prüfen, ob deren Signatur vorhanden und gültig ist. [<=]~~

~~Hinweis: In dieser Spezifikation wird der Prozess, wie die OPA Bundles sicher in den PIP und PAP Service gelangen, nicht festgelegt.~~

### **A\_25464 -PIP und PAP - Tamper-Proof Protokollierung von Administrationsaktivitäten**

Der PIP und PAP Service MUSS ein "Tamper-Proof" Audit-Log von allen administrativen Vorgängen umsetzen. [<=]

### **A\_25777 -PIP und PAP - Löschfristen Auditeinträge des Admin Audit-Logs**

Der PIP und PAP Service MUSS sicherstellen, dass die Löschung eines Auditeintrags den gesetzlichen Vorgaben entspricht und frühestens nach 12 Monaten erfolgt. [<=]

### **A\_25778 -PIP und PAP - Kontrolle des Audit-Logs**

Der Anbieter des PIP und PAP Services MUSS das Audit-Log mindestens alle 3 Monate im Vieraugenprinzip kontrollieren. Diese Rollen DÜRFEN NICHT an der Administration des PAPs oder PIPs teilnehmen. Bei der Kontrolle ist insbesondere auf ungewöhnliche, nicht nachvollziehbare oder maliziöse Administratoraktivitäten zu achten. [<=]

**~~A\_27801 - PIP und PAP - Senden von Telemetriedaten~~**

~~Der Anbieter des PIP und PAP Service MUSS OTLP Telemetriedaten an die gematik liefern, um Auskunft über Betriebsdaten und Versionsständen zu geben.~~

~~[<=]~~

**A\_25465 - PIP und PAP - Änderungen nur durch berechtigte Nutzer**

Der PIP und PAP Service MUSS sicherstellen, dass nur berechtigte Nutzer Änderungen von Policies oder PIP-Daten durchführen können. [ <= ]

**A\_25466 - PIP und PAP - Sicherheitsmeldung bei Aktualisierung von Policies oder PIP-Daten**

Der PIP und PAP Service MUSS sicherstellen, dass bei der Aktualisierung der Policies oder der PIP-Daten eine Sicherheitsmeldung automatisiert über die von der gematik angebotene Schnittstelle an das TI SIEM-System übermittelt wird. [ <= ]

**A\_25467 - PIP und PAP - Änderungen nur unter 4 Augen**

Der PIP und PAP Service MUSS sicherstellen, dass Änderungen in Policies oder PIP-Daten nur im Vieraugenprinzip durchgeführt werden können. [ <= ]

## 5.7 Telemetrie-Daten Service

Der Telemetrie-Daten Service ist eine Implementierung des OpenTelemetry-Frameworks der die Telemetrie-Daten aller ZETA Guard-Komponenten einsammelt (Collector) und für die Weitergabe (Exporter) an externe Systeme (Monitoring des Anbieters, gematik Telemetriedaten Schnittstelle und gematik SIEM) aufbereitet.

Zusätzlich nimmt der Telemetrie-Daten Service Fehlermeldungen und Selbstauskunfts-Daten des Resource Servers entgegen und leitet sie an die gematik Telemetriedaten Schnittstelle weiter.

**A\_27260 - ZETA Guard, Telemetrie-Daten Service, Weitergabe der Daten ohne Profilbildung**

Der Telemetrie-Daten Service im ZETA Guard MUSS die von den Komponenten des ZETA Guard gesammelten Telemetrie-Daten so verändert an das Monitoring System des Anbieters und an die gematik Telemetriedaten Schnittstelle sowie SIEM der gematik weitergeben, dass eine Profilbildung nicht mehr möglich ist.

[ <= ]

**A\_27261 - ZETA Guard, Telemetrie-Daten Service, Protokoll**

Der Telemetrie-Daten Service im ZETA Guard MUSS zur Weitergabe der Daten das OpenTelemetry Protokoll (OTLP) über gRPC oder über HTTP/JSON verwenden. [ <= ]

**A\_27492-01 - ZETA Guard, OpenTelemetry Unterstützung**

Die ZETA Guard Komponenten HTTP Proxy, Authorization Server, Policy Engine und Notification Service MÜSSEN für alle Endpunkte Telemetrie-Daten an den Telemetrie-Daten Service senden, sodass dort Daten zur Performance, Last und Fehlersituationen sowie Daten über die Nutzer und Clients generiert werden können. [ <= ]

**A\_27727 - ZETA Guard, Telemetrie-Daten Service, Default-Wert des Lieferintervalls**

Der Telemetrie-Daten Service im ZETA Guard MUSS standardmäßig ein Lieferintervall von einer Minute eingestellt haben.

[ <= ]

**A\_27728 - ZETA Guard, Telemetrie-Daten Service, Konfigurierbarkeit der Lieferintervalle**

Der Telemetrie-Daten Service im ZETA Guard MUSS eine Anpassung der Lieferintervalle in einer ausgelagerten Konfiguration ermöglichen. Möglich sind Intervalle von sofort bis 5

Minuten. Änderungen des Lieferintervalls MÜSSEN konfiguratativ durchgeführt werden können. [≤]

#### A\_27725 -ZETA Guard, Telemetrie-Daten Service, Status Codes

Der Telemetrie-Daten Service im ZETA Guard MUSS im Parameter *http.status* einen HTTP-Statuscode gemäß Tab\_gemSpec\_ZETA\_Telemetriedaten\_HTTP\_Statuscodes übermitteln.

**Tabelle 12: 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.

[≤]

*Hinweis:* Es sind vom Anbieter, anstatt der Status Code Klassen (first digit of status code), die konkreten 3-stelligen HTTP-Statuscodes gemäß [RFC9110] zu verwenden.

#### A\_27494-01 -Telemetrie-Daten Service, Custom Collector für Selbstauskunft

Der Telemetrie-Daten Service MUSS einen Custom Collector (oder einen Collector mit einem Custom Processor) für die Selbstauskunft des Resource Servers implementieren. Die Selbstauskunft des Resource Servers erfolgt für jede eigenständige Instanz als OTLP Log Record.

Der OTLP Log Record für die Selbstauskunft ist wie folgt definiert:

**Body:** "Selbstauskunft".

**Attributes:**

- – Name des Produkts
- – Aktuelle Produktversion
- – Version des Produkt-Typs
- – Konfigurationsversion
- – Name des Pods/der Instanz des Resource Servers
- – Der Zeitpunkt der Erstellung des Log Records (wird automatisch gesetzt).

[≤]

Beispiel: Selbstauskunft OTLP-Log-Nachricht (JSON) vom Resource Server an den Telemetrie-Daten Service. Die Log-Nachricht wird unverändert an die gematik Telemetriedaten Schnittstelle weitergeleitet.

```
{
  "resourceLogs": [
    {
      "scopeLogs": [
        {
          "logRecords": [
            {
              "timestamp": "1678886400000000000",
              "body": { "stringValue": "Selbstauskunft" },
              "attributes": [
                { "key": "product_name", "value": { "stringValue": "Backend-Service" } },
                { "key": "product_version", "value": { "stringValue": "1.2.0" } },
                { "key": "producttype_version", "value": { "stringValue": "1.2.3" } },
                { "key": "configuration_version", "value": { "stringValue": "cfg-rev-45" } },
                { "key": "pod_name", "value": { "stringValue": "my-app-pod-1" } }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

## 5.8 Betrieb

Die Komponenten des ZETA Guard werden in einer Kubernetes (K8s) Umgebung betrieben. ~~In einem von der gematik vorgegebenen Git Repository werden die Konfigurationsdateien des ZETA Guards bereitgestellt, mit denen die ZETA Guard Instanz für den TI 2.0 Dienst erstellt und ausgeführt werden kann. Im ZETA Guard ist ein Management Service (eine Continuous Delivery (CD) Komponente) enthalten, der den Betriebszustand des ZETA Guards überwacht und regelmäßig prüft, ob eine neuere Version des ZETA Guards im Git Repository verfügbar ist, und ggf. ZETA Guard automatisch aktualisiert.~~ des TI 2.0 Dienst-Anbieters betrieben.

### 5.8.1 Anforderungen an Hersteller einer ZETA Komponente

#### A\_27851 -ZETA Guard - Rückwärtskompatibilität

Der Hersteller des ZETA Guard MUSS sicher stellen, dass bei Änderungen an den öffentlichen Schnittstellen keine Breaking-Changes eingeführt werden, bei gleichzeitiger Wahrung der Rückwärtskompatibilität für alle aktiv unterstützten Releases. [**<=**]



## 5.8.2 Anforderungen an Hersteller eines TI2.0 Dienstes

### A\_27818 -Unterstützung der Wartbarkeit des ZETA Guard-Dienstes

Der Hersteller eines Dienstes der TI2.0 MUSS regelmäßige Updates seines Produktes einplanen, damit die Aktualisierung der ZETA Guard gewährleistet ist. Ein Regelupdate erfolgt maximal einmal pro Quartal. Diese Anforderung gilt über den gesamten Lebenszyklus des Produktes hinweg.[<=]

## 5.8.3 Anforderungen an Anbieter eines TI2.0 Dienstes

### A\_27792 -ZETA Guard - Verbot der Nutzung bestimmter ZETA Guard Versionen

Der Anbieter eines Dienstes der TI2.0 DARF eine zurückgezogene oder ungültige ZETA Guard Version NICHT produktiv einsetzen.[<=]

### A\_27793 -ZETA Guard - Reguläre Aktualisierung von ZETA Guard

Der Anbieter eines Dienstes der TI2.0 MUSS in der Lage sein, regelmäßig Patchupdates der integrierten ZETA Guard Version, die keine Auswirkung auf das Zusammenspiel mit dem Ressource Server haben, durchzuführen. Ein Regelupdate erfolgt maximal einmal pro Quartal.[<=]

### A\_27794 -ZETA Guard - Prüfung auf neue ZETA Guard Versionen

Der Anbieter eines Dienstes der TI2.0 MUSS regelmäßig auf neue freigegebene ZETA Guard Versionen prüfen und - wenn vorhanden - Aktualisierungen im Rahmen des Gültigkeitszeitraums einplanen. Ein Regelupdate erfolgt maximal einmal pro Quartal.[<=]

### A\_27795 -ZETA Guard - Gewährleistung der Verbindung zu PIP/PAP

Der Anbieter eines Dienstes der TI2.0 MUSS gewährleisten, dass der eingesetzte ZETA Guard jederzeit auf Aktualisierungen am PIP/PAP-Service abrufen kann.[<=]

*Hinweis: Notwendige Freischaltungen sind vom Anbieter zu beauftragen und deren Funktionsfähigkeit sicherzustellen.*

### A\_27796 -ZETA Guard - Gewährleistung der Verbindung zur Telemetriedatenlieferung der gematik

Der Anbieter eines Dienstes der TI2.0 MUSS gewährleisten, dass der eingesetzte ZETA Guard jederzeit Datenlieferungen an die gematik übermitteln kann.[<=]

*Hinweis: Notwendige Freischaltungen sind vom Anbieter zu beauftragen und deren Funktionsfähigkeit sicherzustellen.*

### A\_25773-02 -ZETA Guard - Nutzung der von der gematik bereitgestellten Container Images

~~A\_25773 -ZETA Guard - Nutzung der von der gematik bereitgestellten Zero Container Images~~ Der Anbieter eines Dienstes der TI 2.0 MUSS die von der gematik bereitgestellten Container Images im ZETA Guard verwenden, um den Zugang zum TI 2.0 Dienst zu kontrollieren.[<=]

~~Hinweis: Für TI 2.0 Dienste, die inkl. ZETA Guard in der VAU betrieben werden, wird noch festgelegt, wie sichergestellt wird, dass ZETA Guard-Komponenten eingesetzt werden.~~

### ~~A\_25776 -ZETA Guard - Änderung der Konfiguration~~

~~Der Anbieter eines Dienstes der TI 2.0 MUSS die Kubernetes-Manifeste (Konfiguration) des ZETA Guards aus dem git Repository der gematik verwenden.[<=]~~

~~Hinweis: Die Kubernetes-Manifeste (Konfiguration) des ZETA Guards werden per git submodule in das git Repository der gematik integriert, sodass der Anbieter des ZETA Guards seine Konfiguration selbständig anpassen kann.~~

## 5.8.4 Anforderungen für nahtlose Aktualisierungen

Es ist durch geeignete Maßnahmen sicherzustellen, dass ein unterbrechungsfreier Betrieb, bzw. die durchgängige Verfügbarkeit zu jeder Zeit gewährleistet ist.

### **A\_25784 -ZETA Guard-Komponenten - Download von Aktualisierungen im Hintergrund**

Die Komponenten des ZETA Guard MÜSSEN in der Lage sein, Aktualisierungen im Hintergrund herunterzuladen, ohne den laufenden Betrieb zu beeinträchtigen. [ <= ]

### **A\_25785 -ZETA Guard-Komponenten - Nahtloser Übergang zu neuen Versionen**

Die Komponenten des ZETA Guard MUSS einen Mechanismus bieten, der einen nahtlosen Übergang zu neuen Versionen oder Patches ermöglicht, ohne die Verfügbarkeit für Endbenutzer zu unterbrechen. [ <= ]

### **A\_26104 -ZETA Guard-Komponenten - Protokollieren von Änderungen**

Die Komponenten des ZETA Guard MUSS jede Änderung protokollieren, einschließlich des Zeitpunkts der Änderung und des Administrators, der die Änderung vorgenommen hat. [ <= ]

### **A\_25786 -ZETA Guard-Komponenten - Abschluss von Transaktionen vor Aktualisierung**

Die Komponenten des ZETA Guard MUSS sicherstellen, dass alle aktuellen Transaktionen und Anfragen abgeschlossen oder ordnungsgemäß übernommen werden, bevor ein Update finalisiert wird. [ <= ]

### **A\_25787 -ZETA Guard-Komponenten - Gewährleistung der Systemintegrität während Aktualisierungen**

Die Komponenten des ZETA Guard MUSS während des gesamten Aktualisierungsprozesses die Systemintegrität und Sicherheitsrichtlinien aufrechterhalten. [ <= ]

### **A\_25788 -ZETA Guard-Komponenten - Unterstützung von Rollbacks**

Die Komponenten des ZETA Guard MUSS die Fähigkeit besitzen, zu einer stabilen Vorversion zurückzukehren, sollte eine Aktualisierung fehlerhaft sein oder abgebrochen werden müssen. [ <= ]

### **A\_25789 -ZETA Guard-Komponenten - Schnelle Rollback-Durchführung**

Die Komponenten des ZETA Guard MUSS Rollbacks schnell und ohne manuelle Eingriffe durchführen können. [ <= ]

## ~~5.8.5 Anforderungen für Steuerung durch Feature-Flags~~

### ~~A\_25790 -ZETA Guard-Komponenten - Aktivierung/Deaktivierung von Funktionen in Echtzeit~~

~~Die Komponenten des ZETA Guard MUSS Funktionen oder Verhaltensweisen zur Laufzeit durch Feature-Flags aktivieren oder deaktivieren können, ohne dass ein Neustart erforderlich ist. [ <= ]~~

~~Feature-Flags dürfen nur unter 4 Augen und unter strenger Einhaltung des Change-Management-Prozesses geändert werden.~~

## 5.8-65.8.5 Anforderungen zur Überwachung des Betriebsstatus

### A\_25794 -ZETA Guard-Komponenten - Implementierung von Health Checks

Die Komponenten des ZETA Guard MUSS Health Checks implementieren, um ihren aktuellen Zustand und ihre Verfügbarkeit zu überwachen. [ $\leq$ ]

### A\_25797-01A\_25797 -ZETA Guard-Komponenten - Health Check Schnittstelle für gematik Monitoring

Der Anbieter des ~~ZETA-Guard~~**MUSSTI 2.0 Dienstes MUSS** die Schnittstellen zu Health Checks des ZETA Guard dem gematik Monitoring zur Verfügung stellen. [ $\leq$ ]

### A\_25795 -ZETA Guard-Komponenten - Automatische Antwort auf Health Check Anfragen

Die Komponenten des ZETA Guard MUSS automatisch auf Health Check Anfragen antworten können, um ihre Funktionalität und Verfügbarkeit zu bestätigen. [ $\leq$ ]

### A\_25796 -ZETA Guard-Komponenten - Bereitstellung von Zustandsinformationen

Die Komponenten des ZETA Guard MUSS detaillierte Zustandsinformationen als Teil ihrer Health Check Antworten bereitstellen, einschließlich - aber nicht beschränkt auf - Betriebszeit, letzte erfolgreiche Transaktion und eventuelle Fehlerzustände. [ $\leq$ ]

### A\_25798 -ZETA Guard-Komponenten - Regelmäßige Selbstüberprüfung

Die Komponenten des ZETA Guard MUSS in der Lage sein, regelmäßige Selbstüberprüfungen durchzuführen, um interne Funktionen und Abhängigkeiten zu verifizieren und sicherzustellen, dass sie korrekt arbeiten. [ $\leq$ ]

### A\_25799 -ZETA Guard-Komponenten - Protokollierung von Health Check Ergebnissen

Die Komponenten des ZETA Guard MUSS die Ergebnisse der Health Checks protokollieren, um eine Historie ihrer Betriebszustände und eventuell aufgetretener Probleme zu erhalten. [ $\leq$ ]

### A\_25800 -ZETA Guard-Komponenten - Benachrichtigung bei Fehlern

Die Komponenten des ZETA Guard MUSS im Falle eines fehlgeschlagenen Health Checks oder der Erkennung eines kritischen Zustandes automatisch eine Benachrichtigung an ein vordefiniertes Management- oder Monitoring-System senden. [ $\leq$ ]

*Hinweis: Im ZETA Guard ist dafür der Telemetrie-Daten Service vorgesehen, der die Daten an ein Monitoring System des Anbieters weitergibt.*

Der Hersteller der ZETA Guard-Komponenten wird im Rahmen seiner Entwicklungs- und Wartungstätigkeit die Aufgaben eines Third (3<sup>rd</sup>) Level Supports gewährleisten. First- (1<sup>st</sup>) und Second- (2<sup>nd</sup>) Level Supporttätigkeiten fallen im Rahmen der ZETA Guard-Komponenten nicht an. Diese sind bei Bedarf vom jeweiligen TI 2.0 Dienst eigenständig einzusetzen.

### A\_27385 -ZETA Guard-Komponenten - Abbildung von Produkt- und Konfigurationsversionen

Die Komponenten des ZETA Guard MUSS folgende Versionsangaben jederzeit im Betrieb vorhalten:

- Produktversion gem.[gemSpec\_OM#Tab\_ProdIdent] des ZETA-Guard
- Konfigurationsversion gem. [gemKPT\_Betr#A\_20219-01] des ZETA-Guard

[ $\leq$ ]

**A\_27496 -Zeta Guard-Komponenten - Aufbereitung von Client Daten zum Monitoring**

Die Komponenten des ZETA Guard MÜSSEN zu jedem Schnittstellenaufruf an den Resource-Server mindestens folgende Informationen aus der PDP Datenbank - und dem Request des Aufrufers verarbeiten und protokollieren, damit eine anschließende Zusammenführung von Monitoring-Daten aus verschiedenen Quellen (siehe [A\_27494\*]) im Telemetrie-Daten-Service ermöglicht werden kann.

Diese Daten umfassen mindestens:

- product\_id - aus dem Token Self-Assessment des aufrufenden Clientsystems
- product\_version - aus dem Token Self-Assessment des aufrufenden Clientsystems
- professionOID - aus dem SM-B Zertifikat des aufrufenden Clientsystems

[<=]

**5.8-75.8.6 Leistungs-Anforderungen****A\_26486 -PEP HTTP Proxy - Performance**

Die Komponente PEP HTTP Proxy MUSS bei einem Deployment in eine Cloud Umgebung ermöglichen, dass eingehende Requests unter Last innerhalb von 0,1 Sekunden geprüft und bei erfolgreicher Prüfung weitergeleitet werden. [<=]

**A\_26487 -PEP HTTP Proxy -Skalierbarkeit**

Die Komponente PEP HTTP Proxy MUSS horizontal skalierbar sein, sodass mit jedem zusätzlichen Pod mindestens 75% der Leistung eines einzigen Pods verfügbar werden. [<=]

**A\_26488 -PEP HTTP Proxy - Last**

Die Komponente PEP HTTP Proxy MUSS pro Pod mehr als 300 Websocket Verbindungen und mehr als 300 Requests pro Sekunde unterstützen können. [<=]

**A\_26489 -PDP Authorization Server - Performance**

Die Komponente PDP Authorization Server MUSS bei einem Deployment in eine Cloud Umgebung ermöglichen, dass eingehende Requests unter Last innerhalb von 0,2 Sekunden bearbeitet und beantwortet werden. [<=]

**A\_26490 -PDP Authorization Server - Skalierbarkeit**

Die Komponente PDP Authorization Server MUSS horizontal skalierbar sein, sodass mit jedem zusätzlichen Pod mindestens 75% der Leistung eines einzigen Pods verfügbar werden. [<=]

**A\_26491 -PDP Authorization Server - Last**

Die Komponente PDP Authorization Server MUSS pro Pod über alle Endpunkte zusammen mehr als 300 Requests pro Sekunde unterstützen können. [<=]

*Hinweis: Anfragen an abhängige Dienste im Hintergrund, um einen Request vollständig zu bearbeiten (z.B. OCSP), werden bei den Bearbeitungszeiten mit berücksichtigt, jedoch nicht dem abfragenden Dienst zur Last gelegt.*

**5.8-85.8.7 Betriebliche Schnittstellendefinition**

Die Komponenten des ZETA Guard stellen Endpunkte zur Verfügung, um die grundlegende Funktionalität, eingebettet in einen Service, zu gewährleisten. Jeder Dienst, der die ZETA Guard-Komponenten betreibt, stellt damit folgende Endpunkte für

einen Nutzer zur Verfügung. Die Tabelle orientiert sich an den Schnittstellendefinitionen aus [gemKPT\_Betr].

### **A\_28436 -ZETA Guard, Endpunkte im Internet**

Der Anbieter eines TI 2.0 Dienstes MUSS die ZETA Guard und Kubernetes Cluster Endpunkte gemäß Tab gemSpec\_ZETA Schnittstellendefinition ZETA Guard im Internet bereitstellen. [≤]

**Tabelle 13: Tab\_gemSpec\_ZETA\_Schnittstellendefinition\_ZETA\_Guard**

Endpunkt / Anwendungsfall	Beschreibung
<a href="#">GET /.well-known/api-catalog</a>	<a href="#">Abruf des Resource Server API Catalog Well-known JSON Dokuments (<a href="https://www.rfc-editor.org/rfc/rfc9727.html">https://www.rfc-editor.org/rfc/rfc9727.html</a>)</a>
<a href="#">GET /.well-known/oauth-protected-resource/&lt;resource&gt;</a>	<a href="#">Abruf des Resource Server Well-known JSON Dokuments</a> <a href="#">Abruf des Resource Server Well-known JSON Dokuments. Default ist GET /.well-known/oauth-protected-resource/.</a> <a href="#">Wenn mehrere Ressourcen vom Resource Server bereitgestellt werden, dann werden die Well-known Dokumente über den Subpath &lt;resource&gt; bereitgestellt.</a> <a href="#">Beispiel: GET /.well-known/oauth-protected-resource/resource1 (<a href="https://www.rfc-editor.org/rfc/rfc9728.html">https://www.rfc-editor.org/rfc/rfc9728.html</a>)</a>
<a href="#">GET /.well-known/oauth-authorization-server</a>	<a href="#">Abruf des Autorisierungsserver Well-known JSON Dokuments</a>
<a href="#">GET /.well-known/openid-federation HOST &lt;ZETA Guard Authorization Server&gt;</a>	<a href="#">Abruf des Entity Statement Well-known nach OpenID Federation 1.0</a>
<a href="#">GET /.well-known/openid-configuration HOST &lt;ZETA Guard Authorization Server&gt;</a>	<a href="#">Abruf des OpenID Well-known JSON Dokuments.</a> <a href="#">Der ZETA Guard Authorization Server stellt ID Token für Workloads im ZETA Guard aus, für den Zugriff auf das SIEM und den Telemetriedaten Empfänger der gematik.</a>
<a href="#">GET /openid/v1/jwks HOST &lt;ZETA Guard Authorization Server&gt;</a>	<a href="#">JWKS des ZETA Guard Authorization Server</a> <a href="#">Enthält die Signatur-Zertifikate des ZETA Guard Authorization Server</a>
<a href="#">GET /openid/v1/jwks HOST &lt;Kubernetes Cluster IDP&gt;</a>	<a href="#">JWKS des Kubernetes Cluster IDP</a> <a href="#">Enthält die Signatur-Zertifikate des IDP</a> <a href="#">Hinweis: Die Bereitstellung erfolgt nicht durch de ZETA Guard, sondern durch den Kubernetes Cluster.</a>
GET /nonce	Nonce abrufen

Endpunkt / Anwendungsfall	Beschreibung
POST /register	<a href="#">Dynamic Client Registration</a>
<del>POST /token</del> <Subject Token> <a href="#">GET /authorize</a>	<a href="#">Für die Autorisierung ohne Refresh Token nach OAuth Authorization Code Flow</a>
POST /token <Refresh Token>	<a href="#">Autorisierung mit Refresh Token</a> Es werden verschiedene Token Requests unterstützt: <ul style="list-style-type: none"> <li>- <a href="#">Token Request mit Authorization Code</a></li> <li>- <a href="#">Token Exchange mit ID Token</a></li> <li>- <a href="#">Token Exchange mit Refresh Token</a></li> </ul>

*Hinweis: In ZETA Stufe 1 werden die Endpunkte: GET /.well-known/api-catalog und GET /.well-known/openid-federation noch nicht bereitgestellt.*

Zusätzlich werden mittels zentralen Komponenten (PIP/PAP) weitere Endpunkte zur Verfügung gestellt (PIP/PAP), welche von den ZETA Guard Policy Engines abgefragt werden, um beispielsweise aktualisierte Policy-Informationen abzuholen. Folgende Endpunkte werden nachfolgend für den Produkttyp PIP/PAP definiert.

**Tabelle 14: Tab\_gemSpec\_ZETA\_Schnittstellendefinition\_PIPPAP**

Endpunkt / Anwendungsfall	Beschreibung
GET /policies/{application}/{label}	Abruf der Policy eines Dienstes

Beim Erfassen der Daten des Funktionsaufrufs GET /policies/{application}/{label} muss der PIP/PAP-Dienst die Werte für {application} und {label} zusätzlich mit erfassen. Die Systematik zur Betriebsdatenerfassung wird im Kapitel 5.7 Telemetrie-Daten Service beschrieben.

### 5.8.8 Prozesse zur Inbetriebnahme eines ZETA Guard

Für den Betrieb eines ZETA Guard ist es erforderlich, dass ein Registrierungsprozess der gematik durchlaufen wird. In diesem Prozess werden Informationen über den Kubernetes Cluster und den Authorization Server des ZETA Guard bereitgestellt, die eine sichere Kommunikation mit Diensten der gematik (Artifact Registry, SIEM und Telemetriedaten Empfänger) und die Integration in den Federation Master der TI ermöglichen.

#### **A 28437 -ZETA Guard, Registrierung bei der gematik**

Der Anbieter des TI 2.0 Dienstes MUSS den ZETA Guard Authorization Server und den Issuer des Kubernetes Cluster IDP bei der gematik registrieren. [ <= ]

## 5.9 Anforderungen an Dienste der TI

Dienste der TI (Resource Server), die durch einen ZETA Guard geschützt sind, erhalten nur Requests von Clients oder anderen Diensten, wenn der PDP des ZETA Guard den

Zugriff gewährt und ein Access Token ausgestellt hat. Der PEP des ZETA Guards setzt durch, dass nur Requests mit gültigem Access Token zum Resource Server gelangen.

## 5.10 Anforderungen an den Test der Zero Trust-Komponenten

*Offener Punkt: Die Teststrategie der gematik für die TI 2.0 Anwendungen befindet sich aktuell in der Abstimmung mit den Gesellschaftern. Das daraus abzuleitende konkrete Testkonzept für Zero Trust und die daraus resultierenden Anforderungen an die Zero Trust Umsetzung sind dadurch noch nicht für eine Vorveröffentlichung verbindlich festgelegt. Sobald die Teststrategie der gematik abgestimmt ist, wird dieses Kapitel entsprechend angepasst.*

## 5.11 Sicherheitsleistungen des ZETA Guard für Resource Server

Der ZETA Guard bietet umfassende Sicherheitsleistungen für Resource Server in der TI 2.0, indem er das Zero Trust-Paradigma umsetzt. Seine wichtigsten Sicherheitsleistungen lassen sich wie folgt zusammenfassen:

- **Zugriffskontrolle:** Der Policy Enforcement Point (PEP) fungiert als HTTP Proxy und kontrolliert den gesamten Datenverkehr zwischen Client-Anwendungen und dem Resource Server. Er lässt nur Anfragen mit gültigem Access Token durch, das vom Policy Decision Point (PDP) ausgestellt wurde. Zusätzliche Prüfungen, gesteuert durch Attribute im Access Token, können integriert werden. Optional kann konfiguriert werden, dass im Request Header PoPP ein PoPP Token vorhanden sein muss.
- **Clientregistrierung:** Der ZETA Guard setzt eine sichere Clientregistrierung durch, bei der Clients durch verschiedene Mechanismen attestiert werden (z.B. Android Key ID Attestation, Apple DCAppAttest, ~~SM(C)-B-signiertes JWT~~, TPM signiertes JWT, Client signiertes JWT). Der Client wird an eine TI-Identität gebunden ~~(gID, SM(C)-B, eGK oder HBA)~~. Der ZETA Guard unterstützt die Nutzer bei der Registrierung und Verwaltung der Client-Registrierungen.
- **Autorisierung und Authentifizierung:** Der ZETA Guard OAuth2 Authorization Server steuert die Authentifizierung des Nutzers. Die Entscheidung zur Ausstellung des Access Token, nach erfolgreicher Client-Registrierung und Authentifizierung, wird durch die ZETA Guard Policy Engine basierend auf definierten Richtlinien (Policies) getroffen. Dies beinhaltet die Überprüfung von Nutzer- und Client-Registrierungsdaten inkl. Client-Attestierung. Unterstützte Authentifizierungsverfahren sind SM(C)-B ~~mit DPoP-signiertes ID Token~~ sowie Authentifizierung über sektorale oder zentrale IDPs mit OIDC.
- **Policy-Based-Access-Control:** Der Policy Information Point (PIP) und der Policy Administration Point (PAP) verwalten und liefern die freigegebenen Policies an die ZETA Guard Policy Engine. Die Policies sind maschinenlesbar und definieren die Zugriffsregeln, nach denen die Entscheidung zur Ausstellung von Access und Refresh Token erfolgt. Die Integrität und Authentizität der Policies wird durch Signaturen sichergestellt.



- **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.
- **Datenverschlüsselung/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. ~~Die ZETA Guard Daten (ZETA/ASL-Daten, Session-, User- und Client-Daten) werden bei der Persistenz zusätzlich verschlüsselt. Der ZETA Guard kann so konfiguriert werden, dass private Schlüssel in einem Hardware Security Module (HSM) gespeichert werden.~~
- ~~**VAU Unterstützung:** Der ZETA Guard kann optional in einer Vertrauenswürdig ausgeführungsumgebung (VAU) ausgeführt werden.~~
- **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.  
**Datenverschlüsselung:** Die ZETA Guard Daten (ZETA/ASL-Daten, Session-, User- und Client-Daten) werden bei der Persistenz zusätzlich verschlüsselt. Der ZETA Guard kann so konfiguriert werden, dass private Schlüssel in einem Hardware Security Module (HSM) gespeichert werden.
- **VAU Unterstützung:** Der ZETA Guard kann optional in einer Vertrauenswürdig ausgeführungsumgebung (VAU) ausgeführt werden.
- **Monitoring und Logging:** Der ZETA Guard sammelt Telemetrie-Daten 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 Telemetrie-Daten 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.
- **Sicherheits- und Produktgutachten:** Für die ZETA Guard Implementierung werden ein Sicherheits- und ein Produktgutachten bereitgestellt.
- **Betriebshandbuch/Produkthandbuch sowie Sicherheits- und Datenschutzkonzept:** Im Betriebshandbuch/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.

## 5.12 Weitere Leistungen des ZETA Guard für Resource Server

Der ZETA Guard übernimmt für Resource Server weitere Leistungen:

- **gematik Telemetriedaten Lieferung:** Der Telemetrie-Daten 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.

- **SIEM Daten Lieferung:** Der Telemetrie-Daten Service sammelt vom Monitoring des TI 2.0 Dienst-Anbieters SIEM Daten und leitet sie an das SIEM der gematik weiter.
- **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.
- **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.

---

## 6 Beispiele und Referenzimplementierungen

---

Die gematik stellt API-Spezifikationen und Proof-of-Concept-Implementierungen im Internet zur freien Verfügung.

Das Projekt <https://dsr.gematik.solutions> demonstriert eine Attestation mobiler Anwendungen auf gängigen mobilen Betriebssystemplattformen.

Im GitHub-Projekt [gemAPI\_ZETA] werden die Schnittstellenspezifikationen der hier spezifizierten Zero Trust-Komponenten veröffentlicht.

Die folgenden beiden Projekte <https://github.com/gematik/zero-lab> und <https://github.com/gematik/zero-lab-apple> demonstrieren die Anwendungsfälle zur Clientregistrierung auf Apple- und Android-Geräten.

---

## 7 Anhang A – Verzeichnisse

---

### 7.1 Abkürzungen

**Tabelle 15: Im Dokument verwendete Abkürzungen**

Kürzel	Erläuterung
API	Application Programming Interface
CD	Continuous Delivery
CN	Common Name
CTS	Compatibility Test Suite
DSR	Device Security Rating
eGK	elektronische Gesundheitskarte
ePA	elektronische Patientenakte
FBE	File Based Encryption
FDE	Full Disk Encryption
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
mTLS	Mutual Transport Layer Security
OCSP	Online Certificate Status Protocol

Kürzel	Erläuterung
OIDC	OpenID Connect, <a href="https://de.wikipedia.org/wiki/OpenID_Connect">https://de.wikipedia.org/wiki/OpenID_Connect</a>
OTLP	Open Telemetry Protocol
PAP	Policy Administration Point
PAR	Pushed Authorization Request
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
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
TI-ITSM	IT-Service-Management der TI
TPM	Trusted Platform Module
VAU	Vertrauenswürdige Ausführungsumgebung
ZETA	Zero Trust Access

## 7.2 Glossar

**Tabelle 16: Glossar der explizit im Dokument verwendeten Begriffe**

Begriff	Erläuterung
Authorization Server	Ein Server, der Zugriffstoken ausgibt, nachdem er die Identität eines Benutzers 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.

Begriff	Erläuterung
Policy Enforcement Point (PEP)	Ein Punkt in einem Netzwerk, an dem Sicherheitsrichtlinien durchgesetzt werden. Der PEP überwacht und kontrolliert den Zugriff auf Ressourcen 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.
Telemetrie-Daten	<p>Telemetrie-Daten 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 Telemetrie-Daten eine entscheidende Rolle bei der Überwachung und Sicherung von Netzwerken und Systemen.</p> <p>Merkmale und Arten von Telemetrie-Daten:</p> <ul style="list-style-type: none"> <li>- System- und Leistungsmetriken: Informationen über die Leistung und den Zustand von Hardware und Software, wie CPU-Auslastung, Speichernutzung, Netzwerkbandbreite und Festplattenkapazität.</li> <li>- Benutzeraktivitätsdaten: Protokolle und Aufzeichnungen über Benutzeraktionen und -verhalten, einschließlich Anmeldungen, Dateizugriffe, Anwendungsnutzung und andere Interaktionen.</li> <li>- Sicherheitsereignisse: Daten über sicherheitsrelevante Vorfälle, wie fehlgeschlagene Anmeldeversuche, erkannte Malware, unerlaubte Zugriffsversuche und andere sicherheitsbezogene Anomalien.</li> <li>- Netzwerkverkehrsdaten: Informationen über den Datenfluss im Netzwerk, einschließlich IP-Adressen, Ports, Protokolle, Datenmengen und Verbindungen zwischen verschiedenen Systemen und Diensten.</li> <li>- Konfigurationsdaten: Details zu den aktuellen Einstellungen und Konfigurationen von Systemen und Anwendungen, einschließlich Softwareversionen, installierte Patches und Sicherheitsrichtlinien.</li> </ul>
Telemetrie-Daten Service	Ein Dienst, der Telemetrie-Daten sammelt, verarbeitet und analysiert. Telemetrie-Daten 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.



Begriff	Erläuterung
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/Additional Security Layer (ZETA/ASL)	Eine auf HTTP basierende zusätzliche Verschlüsselung der Daten zwischen ZETA Client und ZETA Guard PEP. Der ZETA Client verwendet ein Self-Signed Zertifikat und der PEP verwendet eine Identität aus der Komponenten PKI, um die verschlüsselte Verbindung aufzubauen. Die verschlüsselte Verbindung wird auch ZETA/ASL-Kanal genannt.

## 7.3 Abbildungsverzeichnis

<del>Abbildung 1: NIST Zero Trust Referenzarchitektur, Quelle [NIST_SP1800-35_FIG1] .....</del>	<del>13</del>
<del>Abbildung 2: Abbildung Zero Trust Architektur der TI 2.0 .....</del>	<del>15</del>
<del>Abbildung 3: Abb_ZETA_Zugriff_auf_geschützte_Ressource .....</del>	<del>61</del>
<del>Abbildung 4: Abb_Service_Discovery .....</del>	<del>62</del>
<del>Abbildung 5: Abb_Client_Registrierung .....</del>	<del>64</del>
<del>Abbildung 6: Abb_Authentifizierung_und_Autorisierung .....</del>	<del>67</del>
<del>Abbildung 7: Beziehungen zwischen Session-, Nutzer- und Client-Daten sowie Token ...</del>	<del>71</del>
<del>Abbildung 1: NIST Zero Trust-Referenzarchitektur, Quelle [NIST SP1800-35 FIG1] .....</del>	<del>13</del>
<del>Abbildung 2: Abbildung Zero Trust-Architektur der TI 2.0 .....</del>	<del>15</del>
<del>Abbildung 3: Abb ZETA Zugriff auf geschützte Ressource .....</del>	<del>61</del>
<del>Abbildung 4: Abb Service Discovery .....</del>	<del>62</del>
<del>Abbildung 5 : Abb Client Registrierung .....</del>	<del>64</del>
<del>Abbildung 6 :Abb Authentifizierung und Autorisierung .....</del>	<del>67</del>
<del>Abbildung 7 Abb ZETA-Guard-Dienst-zu-Dienst-Kommunikation .....</del>	<del>70</del>
<del>Abbildung 8: Beziehungen zwischen Session-, Nutzer- und Client-Daten sowie Token ...</del>	<del>71</del>

## 7.4 Tabellenverzeichnis

<del>Tabelle 1: Statische Eigenschaften Clientsysteme auf Hersteller-/Anbiiterebene .....</del>	<del>20</del>
<del>Tabelle 2: Eigenschaften Clientsysteme auf Instanzebene (pro Installation) .....</del>	<del>21</del>
<del>Tabelle 3: Verwendete Device Claims für Android-Clients .....</del>	<del>22</del>
<del>Tabelle 4: Verwendete Device Claims für iOS-Clients .....</del>	<del>23</del>
<del>Tabelle 5: ZT_HTTP_Statuscodes .....</del>	<del>40</del>

Tabelle 6: PEP HTTP Proxy – Zusätzliche HTTP-Header .....	50
Tabelle 7: OPA – Konfiguration .....	53
Tabelle 8: PDP Authorization Server – Plugin-Schnittstelle Application Authorization Backend .....	56
Tabelle 9: SM(C)-B_Nutzer-Daten .....	57
Tabelle 10: id_token_Nutzer-Daten .....	58
Tabelle 11: PDP – Konfigurations-Parameter .....	73
Tabelle 12: Tab_gemSpec_ZETA_Telemetriedaten_HTTP_Statuscodes .....	77
Tabelle 13: Tab_gemSpec_ZETA_Schnittstellendefinition_ZETA_Guard .....	83
Tabelle 14: Tab_gemSpec_ZETA_Schnittstellendefinition_PIPPAP .....	84
Tabelle 15: Im Dokument verwendete Abkürzungen .....	89
Tabelle 16: Glossar der explizit im Dokument verwendeten Begriffe .....	91
Tabelle 17: Referenzierte Dokumente der gematik .....	95
Tabelle 18: Weitere Referenzen .....	97
Tabelle 1: Statische Eigenschaften Clientsysteme auf Hersteller-/Anbiiterebene .....	20
Tabelle 2: Eigenschaften Clientsysteme auf Instanzebene (pro Installation) .....	21
Tabelle 3: Verwendete Claims für Android-Clients .....	22
Tabelle 4: Verwendete Claims für iOS-Clients .....	23
Tabelle 5 Verwendete Claims für Windows und Linux Clients .....	23
Tabelle 6: ZT HTTP Statuscodes .....	40
Tabelle 7: PEP HTTP Proxy - Zusätzliche HTTP-Header .....	50
Tabelle 8: OPA - Konfiguration .....	53
Tabelle 9: PDP Authorization Server - Plugin-Schnittstelle Application Authorization Backend .....	56
Tabelle 10: SM(C)-B Nutzer-Daten .....	57
Tabelle 11: id token Nutzer-Daten .....	58
Tabelle 12: Tab_gemSpec_ZETA_Telemetriedaten_HTTP_Statuscodes .....	77
Tabelle 13: Tab_gemSpec_ZETA_Schnittstellendefinition_ZETA_Guard .....	83
Tabelle 14: Tab_gemSpec_ZETA_Schnittstellendefinition_PIPPAP .....	84
Tabelle 15: Im Dokument verwendete Abkürzungen .....	89
Tabelle 16: Glossar der explizit im Dokument verwendeten Begriffe .....	91
Tabelle 17: Referenzierte Dokumente der gematik .....	95
Tabelle 18: Weitere Referenzen .....	97

## 7.5 Referenzierte Dokumente

### 7.5.1 Dokumente der gematik

Die nachfolgende Tabelle enthält die Bezeichnung der in dem vorliegenden Dokument referenzierten Dokumente der gematik zur Telematikinfrastruktur.

**Tabelle 17: Referenzierte Dokumente der gematik**

[Quelle]	Herausgeber: Titel
[access-token.yaml]	Schema access-token.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/access-token.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/access-token.yaml</a>
[API-ZETA-Attestation-Service]	API des ZETA Attestation Service <a href="https://github.com/gematik/ZETA/blob/main/docs/api/v1/index.md#zeta-attestation-service-endpunkte">https://github.com/gematik/ZETA/blob/main/docs/api/v1/index.md#zeta-attestation-service-endpunkte</a>
[as-well-known.yaml]	Schema für das OAuth Authorization Server Well-known JSON Dokument <a href="https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/as-well-known.yaml">https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/as-well-known.yaml</a>
[client-instance.yaml]	Schema client-instance.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/client-instance.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/client-instance.yaml</a>
[gemAPI_ZETA]	gematik: OpenAPI Schnittstellen- und Schemaspezifikation ZETA <a href="https://github.com/gematik/zeta">https://github.com/gematik/zeta</a>
[gemGlossar]	gematik: Glossar der Telematikinfrastruktur
[gemF_PushNotification]	gematik:Feature: Anwendungsübergreifende Push Notification <a href="https://gemspec.gematik.de/docs/gemF/gemF_PushNotification/latest/">https://gemspec.gematik.de/docs/gemF/gemF_PushNotification/latest/</a>
[gemKPT_Betr]	gematik: Betriebskonzept Online-Produktivbetrieb <a href="https://gemspec.gematik.de/docs/gemKPT/gemKPT_Betr/latest/">https://gemspec.gematik.de/docs/gemKPT/gemKPT_Betr/latest/</a>
[gemSpec_DS_Hersteller]	gematik: Spezifikation Datenschutz- u. Sicherheitsanforderungen der TI an Hersteller <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Hersteller/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Hersteller/latest/</a>

[Quelle]	Herausgeber: Titel
[gemSpec_IDP_Dienst]	gematik: Spezifikation Identity Provider-Dienst <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_IDP_Dienst/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_IDP_Dienst/latest/</a>
[gemSpec_IDP_Sek]	gematik: Spezifikation Sektoraler Identity Provider <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_IDP_Sek/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_IDP_Sek/latest/</a>
[gemSpec_Krypt]	gematik: Übergreifende Spezifikation Verwendung kryptographischer Algorithmen in der Telematikinfrastruktur <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_Krypt/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_Krypt/latest/</a>
[gemSpec_OM]	gematik: Übergreifende Spezifikation Operations und Maintenance <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_OM/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_OM/latest/</a>
[gemSpec_Perf]	gematik: Übergreifende Spezifikation Performance und Mengengerüst TI-Plattform <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_Perf/latest/">https://gemspec.gematik.de/docs/gemSpec/gemSpec_Perf/latest/</a>
[GitHub ZETA Schemas]	Schemas für zusätzliche HTTP-Header <a href="https://github.com/gematik/zeta/tree/main/src/schemas">https://github.com/gematik/zeta/tree/main/src/schemas</a>
[ISMS]	Spezifikation Datenschutz- und Sicherheitsanforderungen der TI an Anbieter (Abschnitt 3.3) <a href="https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Anbieter/latest/#3.3">https://gemspec.gematik.de/docs/gemSpec/gemSpec_DS_Anbieter/latest/#3.3</a>
[JOSE]	JSON Object Signing and Encryption (JOSE) <a href="https://www.iana.org/assignments/jose/jose.xhtml">https://www.iana.org/assignments/jose/jose.xhtml</a>
[opr-well-known.yaml]	Schema für das OAuth Protected Resource Metadata Well-known JSON Dokument <a href="https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/opr-well-known.yaml">https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/opr-well-known.yaml</a>
[pdp-decision.yaml]	Schema pdp-decision.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/pdp-decision.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/pdp-decision.yaml</a>

[Quelle]	Herausgeber: Titel
[pip-pap-service.yaml]	gematik: OpenAPI Schnittstellenspezifikation für Policy Information Point und Policy Administration Point API <a href="https://raw.githubusercontent.com/gematik/zeta/develop/src/openapi/pip-pap-api.yaml">https://raw.githubusercontent.com/gematik/zeta/develop/src/openapi/pip-pap-api.yaml</a>
[refresh-token.yaml]	Schema refresh-token.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/refresh-token.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/refresh-token.yaml</a>
[TrustedTPM_RootCA_und_IntermediateCA]	Liste der vertrauenswürdigen TPM Stamm- und Zwischensignaturzertifikate <a href="https://learn.microsoft.com/de-at/windows-server/security/guarded-fabric-shielded-vm/guarded-fabric-install-trusted-tpm-root-certificates">https://learn.microsoft.com/de-at/windows-server/security/guarded-fabric-shielded-vm/guarded-fabric-install-trusted-tpm-root-certificates</a>
[user-info.yaml]	Schema user-info.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/user-info.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/user-info.yaml</a>
[zeta-error.yaml]	Schema zeta-error.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/zeta-error.yaml">https://raw.githubusercontent.com/gematik/zeta/main/src/schemas/zeta-error.yaml</a>

## 7.5.2 Weitere Referenzen

**Tabelle 18: Weitere Referenzen**

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[Android Platform Security Model]	The Android Platform Security Model (2023) <a href="https://research.google/pubs/the-android-platform-security-model/">https://research.google/pubs/the-android-platform-security-model/</a> (Abruf 01/2025)
[Apple Platform Security Guide]	Einführung in die Sicherheit der Apple-Plattformen <a href="https://support.apple.com/de-de/guide/security/seccd5016d31/web">https://support.apple.com/de-de/guide/security/seccd5016d31/web</a> (Abruf 01/2025)
[BSI-Grundschutz]	IT-Grundschutz - Informationssicherheit mit System <a href="https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/it-grundschutz_node.html">https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/IT-Grundschutz/it-grundschutz_node.html</a> (Abruf 01/2025)

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[CAB-Forum]	Certification Authority Browser Forum (CA/Browser Forum) <a href="https://cabforum.org/">https://cabforum.org/</a> (Abruf 01/2025)
[CAPEC OWASP]	CAPEC: OWASP Related Patterns <a href="https://www.mitre.org/cyber-attack/capec">CAPEC - CAPEC-659: OWASP Related Patterns (Version 3.9) (mitre.org)</a> (Abruf 01/2025)
[ExpBack]	Exponential Backoff <a href="https://en.wikipedia.org/wiki/Exponential_backoff">https://en.wikipedia.org/wiki/Exponential_backoff</a> (Abruf 01/2025)
[NIST_SP 1800-35_FIG1]	National Institute of Standards and Technology (NIST): NIST SP 1800-35 Publication (Figure 1 - General ZTA Reference Architecture) <a href="https://pages.nist.gov/zero-trust-architecture/VolumeB/architecture.html">https://pages.nist.gov/zero-trust-architecture/VolumeB/architecture.html</a> (Abruf 01/2025)
[OPA Bundle]	Open Policy Agent, Bundles <a href="https://www.openpolicyagent.org/docs/latest/management-bundles/">https://www.openpolicyagent.org/docs/latest/management-bundles/</a> (Abruf 01/2025)
[Open Policy Agent]	Open Policy Agent <a href="https://www.openpolicyagent.org/docs/latest/">https://www.openpolicyagent.org/docs/latest/</a> (Abruf 01/2025)
[OWASP-Top-10-Risiken]	OWASP Top 10 <a href="https://owasp.org/www-project-top-ten/">https://owasp.org/www-project-top-ten/</a> (Abruf 01/2025)
[RFC2119]	Key words for use in RFCs to Indicate Requirement Levels <a href="https://datatracker.ietf.org/doc/html/rfc2119">https://datatracker.ietf.org/doc/html/rfc2119</a> (Abruf 01/2025)
[RFC2986]	PKCS #10: Certification Request Syntax Specification <a href="https://datatracker.ietf.org/doc/html/rfc2986">https://datatracker.ietf.org/doc/html/rfc2986</a> (Abruf 01/2025)
[RFC6066]	Transport Layer Security (TLS) Extensions: Extension Definitions <a href="https://datatracker.ietf.org/doc/html/rfc6066">https://datatracker.ietf.org/doc/html/rfc6066</a> (Abruf 01/2025)
[RFC6749]	The OAuth 2.0 Authorization Framework <a href="https://datatracker.ietf.org/doc/html/rfc6749">https://datatracker.ietf.org/doc/html/rfc6749</a> (Abruf 01/2025)
[RFC7231]	Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content <a href="https://datatracker.ietf.org/doc/html/rfc7231">https://datatracker.ietf.org/doc/html/rfc7231</a> (Abruf 01/2025)
[RFC7232]	Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests <a href="https://datatracker.ietf.org/doc/html/rfc7232">https://datatracker.ietf.org/doc/html/rfc7232</a> (Abruf 01/2025)

[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[RFC7521]	Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants <a href="https://datatracker.ietf.org/doc/html/rfc7521">https://datatracker.ietf.org/doc/html/rfc7521</a> (Abruf 01/2025)
[RFC7523]	JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants <a href="https://datatracker.ietf.org/doc/html/rfc7523">https://datatracker.ietf.org/doc/html/rfc7523</a> (Abruf 01/2025)
[RFC7636]	Proof Key for Code Exchange by OAuth Public Clients <a href="https://datatracker.ietf.org/doc/html/rfc7636">https://datatracker.ietf.org/doc/html/rfc7636</a> (Abruf 01/2025)
[RFC8555]	Automatic Certificate Management Environment (ACME) <a href="https://datatracker.ietf.org/doc/html/rfc8555#section-6.5.1">https://datatracker.ietf.org/doc/html/rfc8555#section-6.5.1</a> (Abruf 01/2025)
[RFC8705]	OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens <a href="https://datatracker.ietf.org/doc/html/rfc8705">https://datatracker.ietf.org/doc/html/rfc8705</a> (Abruf 01/2025)
[RFC9449]	OAuth 2.0 Demonstrating Proof of Possession (DPoP) <a href="https://datatracker.ietf.org/doc/html/rfc9449">https://datatracker.ietf.org/doc/html/rfc9449</a> (Abruf 01/2025)
[RFC9727]	<a href="https://www.rfc-editor.org/rfc/rfc9727.html">api-catalog: A Well-Known URI and Link Relation to Help Discovery of APIs</a> <a href="https://www.rfc-editor.org/rfc/rfc9727.html">https://www.rfc-editor.org/rfc/rfc9727.html</a>
[RFC9728]	OAuth 2.0 Protected Resource Metadata <a href="https://www.rfc-editor.org/rfc/rfc9728.html">https://www.rfc-editor.org/rfc/rfc9728.html</a>
[session.yaml]	Schema session.yaml <a href="https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/session.yaml">https://raw.githubusercontent.com/gematik/zeta/refs/heads/main/src/schemas/session.yaml</a> (Abruf 06/2025)
[Shared Signals]	OpenID Shared Signals and Events Framework <a href="https://openid.net/specs/openid-sse-framework-1.0.html">https://openid.net/specs/openid-sse-framework-1.0.html</a> (Abruf 01/2025)
[SPIFFE und SPIRE]	Universal identity control plane for distributed systems <a href="https://spiffe.io/">https://spiffe.io/</a> (Abruf 01/2025)
[TR-03107-1]	BSI TR-03107 Elektronische Identitäten und Vertrauensdienste im E-Government <a href="https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03107/TR-03107-1.pdf">https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03107/TR-03107-1.pdf</a> (Abruf 01/2025)



[Quelle]	Herausgeber (Erscheinungsdatum): Titel
[TR-03161]	BSI TR-03161 Anforderungen an Anwendungen im Gesundheitswesen <a href="https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03161/tr-03161.html">https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr03161/tr-03161.html</a> (Abruf 01/2025)
[TR-03161-1]	Technische Richtlinie TR-03161: Anforderungen an Anwendungen im Gesundheitswesen Teil 1: Mobile Anwendungen; Version 3.0 <a href="https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-1.pdf?__blob=publicationFile&amp;v=13">https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03161/BSI-TR-03161-1.pdf?__blob=publicationFile&amp;v=13</a> (Abruf 01/2025)
[VerifiedBoot]	Verifizierter Start <a href="https://source.android.com/docs/security/features/verifiedboot?hl=de">https://source.android.com/docs/security/features/verifiedboot?hl=de</a> (Abruf 01/2025)