
C_12761_Anlage

Inhaltsverzeichnis

1	Änderungsbeschreibung.....	2
2	Änderung in gemSpec_Perf - Client Zertifikat.....	3
3	Änderung in gemSpec_Perf - Lieferattribute.....	5
3.1	Kapitel 2.5.4.1 Traces.....	5
3.2	Kapitel 2.5.4.2 Logs.....	14
3.3	Kapitel 2.5.4.4 Anforderungen an Telemetriedatenlieferungen.....	19
4	Änderung in gemSpec_ZETA.....	20

1 Änderungsbeschreibung

Die Einarbeitung des Änderungseintrages wurde nicht vollständig umgesetzt. Zusätzlich wurden diverse Änderungen nur für PoPP umgesetzt und nicht generell übergreifend für die Telemetriedatenlieferung von TI2.0 Diensten.

Diese Thematik wird in der Änderung C_12761 angegangen, indem die fehlenden Produkttypen noch an die Afos angehängen werden, die das Thema der Client Zertifikate regeln und die bisher nur für PoPP gelten.

2 Änderung in gemSpec_Perf - Client Zertifikat

Afo wird storniert (wie in C_12662 bereits eingebracht)

A_27719 -Telemetriedatenlieferung - Nutzung eines gültigen Client-Zertifikats

Der Anbieter MUSS für den Verbindungsaufbau zum Telemetriedaten-Service der gematik ein gültiges TLS Client-Zertifikat nutzen.

Hinweis: Ein entsprechendes TLS Client-Zertifikat muss vor Inbetriebnahme bei der gematik beantragt werden. [≤, Anb_ZT_PIP_PAP, Anb_TI-D_ZT, Anb_DiPag_FD, Anb_VSDM_2_FD, organ./betriebl. Eignung: Anbietererklärung]

Zuweisungen werden ergänzt

A_28890 -Verbindungsaufbau zum Telemetriedaten-Service

Der Anbieter MUSS für den Verbindungsaufbau zum Telemetriedaten-Service der gematik einen OAuth 2.0 Token Exchange (gem. RFC 8693) unterstützen.

Hinweis:

Die Basis für die Authentifizierung muss im Rahmen der Inbetriebnahme bei der gematik durchgeführt werden. [≤, Anb_PoPP_Service, organ./betriebl. Eignung: Anbietererklärung]

Anb_TI-D_ZT, Anb_DiPag_FD, Anb_VSDM_2_FD - Anbietererklärung

Zuweisungen werden ergänzt

A_28891 -Verbindungsaufbau zum Telemetriedaten-Service des Produkts

Das Produkt MUSS für den Verbindungsaufbau zum Telemetriedaten-Service der gematik einen OAuth 2.0 Token Exchange (gem. RFC 8693) unterstützen.

Hinweis:

Die Basis für die Authentifizierung muss im Rahmen der Inbetriebnahme bei der gematik durchgeführt werden. [≤, PoPP_Service, funkt. Eignung: Herstellererklärung]

Herst_TI-D_ZT, DiPag_FD, VSDM_2_FD - Herstellererklärung

Zuweisung VSDM2-FD wird entfernt, Afo wird nicht abgelöst (entgegen der Darstellung in C_12662)

A_26178 -Performance - Selbstauskunft - Umsetzungszeit zur Änderung des Lieferintervalls

Der Anbieter MUSS die Änderung der Konfiguration vom Lieferintervall (gemäß [A_26177*]) nach Aufforderung durch die gematik innerhalb von 5 Werktagen (ausgenommen bundeseinheitliche Feiertage) vornehmen. [≤, Anb_FD_KOM-LE, Anb_SMC-B, Anb_eRp_FD, Anb_HBA, Anb_IDP-D, Anb_X.509_TSP_eGK, Anb_TI-M, Anb_SigD, Anb_ZD, Anb_VPN_ZugD, Anb_IDP-Sek_KTR, Anb_Konn_Highspeed, Anb_Aktensystem_ePA, Anb_IDP_FedMaster, Anb_NCPeH_FD, Anb_TIM_FD, Anb_FD_VSDM, Anb_VSDM_2_FD, Anb_TI_Gateway, organ./betriebl. Eignung: Prozessprüfung]

Anb_VSDM_2_FD entfernen

3 Änderung in gemSpec_Perf - Lieferattribute

Die Trace-Attribute entsprechen teilweise nicht den Open Telemetry Vorgaben und sind, deshalb sowohl hinsichtlich der Tabelle, als auch bzgl. der Beispiele anzupassen. Darüber hinaus werden die Attribute und Beispiele sowohl aus Sicht des HTTP-Proxys des ZETA-Guard, als auch des Resource Servers, beschrieben.

3.1 Kapitel 2.5.4.1 Traces

Traces MÜSSEN dem OpenTelemetry Trace Data Model entsprechen und im Format des OpenTelemetry Trace Protobuf erzeugt werden. Die in dieser Spezifikation aufgeführten OpenTelemetry-Attribute stellen eine gezielte Teilmenge dar, die für sicherheitsrelevante Analysen und Auswertungen vorgesehen ist. Die Spezifikation definiert bewusst nicht den vollständigen Umfang möglicher OpenTelemetry-Attribute. Implementierungen DÜRFEN zusätzliche Attribute gemäß OpenTelemetry erfassen und übertragen; diese sind jedoch nicht Gegenstand der normativen Anforderungen dieser Spezifikation.

Ein gesendeter Trace enthält 1-n Spans, die grundlegende Einheit von einzelnen Operationen in einem verteilten System. Die Gesamtheit der Spans mit gleicher Trace ID ermöglicht eine Nachvollziehbarkeit der gesamten Kette eines Aufrufes.

Da es mit ZETA Guard und Resource Server unterschiedliche Komponenten gibt, die Traces erzeugen, werden die Codebeispiele und Parameter-Tabellen separat dargestellt. Bei der Übermittlung der Traces an die Telemetriedatenerfassung der gematik durch den ZETA Guard, spielt diese Trennung keine Rolle.

Hier in der Darstellung wird die OTLP/JSON Notation verwendet, die notwendigen und zugehörigen Attribute der OTLP/gRPC Notation (Proto) - welche für die technische Implementation der Anwendung relevant sind - sind der jeweils aktuell gültigen und veröffentlichten Beschreibung des OpenTelemetry Projektes zu entnehmen.

Hier ein Beispiel eines Traces Spans des HTTP Proxys, der aus zwei einzelnen Operationen zusammengesetzt wird.

```
{
  "resourceSpans": [
    {
      "resource": {
        "attributes": [
          {
            "key": "service.name",
            "value": {"stringValue": "ZETA Guard PEP HTTP proxy"}
          }
        ]
      }
    ]
  }
```

```
104     },
105     "scopeSpans": [
106         {
107             "scope": {
108                 "name": "proxy-tracing",
109                 "version": "1.0.0"
110             },
111             "spans": [
112                 {
113                     "traceId":
114                     "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa",
115                     ,
116                     "spanId": "1111111111111111",
117                     "name": "HTTP GET
118                     /vsdservice/v1/vsdbundle",
119                     "kind": 2,
120                     "startTimeUnixNano":
121                     "1714311000000000000",
122                     "endTimeUnixNano":
123                     "1714311000200000000",
124                     "attributes": [
125                         {
126                             "key": "http.method",
127                             "value": {"stringValue":
128                             "GET"}
129                         }, {
130                             "key": "http.route",
131                             "value": {"stringValue":
132                             "/vsdservice/v1/vsdbundle"}
133                         }, {
134                             "key":
135                             "http.request.header.x-
136                             forwarded-for",
137                             "value": {"stringValue":
138                             "203.0.113.10"}
139                         }, {
140                             "key": "server.address",
141                             "value": {"stringValue":
142                             "vsdm2.aok.net"}
143                         }, {
144                             "key": "product_id",
```

```

145         "value": {"stringValue":
146             "CLIENTID123456"}
147     }, {
148         "key": "product_version",
149         "value": {"stringValue":
150             "2.1.3-10"}
151     }, {
152         "key": "profession_oid",
153         "value": {"stringValue":
154             "1.2.276.0.76.4.30"}
155     }
156 ],
157 "status": {
158     "code": 1
159 }
160 }
161 ]
162 }
163 ]
164 }
165 ]
166 }

```

Die folgende Tabelle gibt Aufschluss über die möglichen Trace-Parameter des ZETA HTTP Proxy Spans. Die tabellarische Auflistung berücksichtigt nicht explizit die Hierarchie Ebene, diese kann aus dem Schema Beispiel entnommen werden.

Tabelle 1 Tab_gemSpec_Perf_Trace_HTTPSpanParameter_Telemetriedatenlieferung

Daten	Open Telemetry Attribute (OTLP/JSON)	Pflichtfeld	Begründung und Zweck
Service Name	service.name	ja	Identifikationsmerkmal, fester Wert "ZETA Guard PEP HTTP proxy"
Startzeit	startTimeUnixNano	ja	Startzeitpunkt der Operation
Endzeit	endTimeUnixNano	ja	Endzeitpunkt der Operation
HTTP Methode	http.method	ja	Erkennung von unautorisierten Aktionen oder abnormalen Anfragen. Ein unerwarteter HTTP-Methoden-Typ auf einer bestimmten Route kann auf einen

			Angriffsversuch hindeuten.
HTTP Route	http.route	ja	Überwachung von Zugriffen auf spezifische Ressourcen und Erkennung von unautorisierten Zugriffen oder Versuchen, nicht existierende Routen zu erreichen (z.B. für Brute-Force-Angriffe oder Enumeration).
HTTP FQDN	server.address	ja	Identifikation des Zielserver bei Multi-Host-Umgebungen und Erkennung von Anfragen an nicht autorisierte oder verdächtige Domänen.
HTTP Status	http.response.status_code	ja	Erkennung von Fehlern, Ausfällen oder potenziellen Angriffsversuchen (z.B. viele 401/403-Fehler bei Brute-Force-Angriffen, viele 5xx-Fehler bei Denial-of-Service-Angriffen).
Product Id	product_id	ja	Product ID aus dem Token Self-Assessment des aufrufenden Clientsystems
Product Version	product_version	ja	Product Version aus dem Token Self-Assessment des aufrufenden Clientsystems
Profession OID	profession_oid	ja - sofern möglich	Profession OID aus dem SM-B Zertifikat des aufrufenden Clientsystems. Das Attribut ist nur für die Usecases zu übermitteln, in denen ein SM-B-Zertifikat Anwendung findet.

Ergänzend zum ZETA Guard Proxy Span, hier das Beispiel eines Trace Span eines Resource Servers:

```
{
  "resourceSpans": [
    {
      "resource": {
        "attributes": [
          {
```

```
181         "key": "service.name",
182         "value": {"stringValue": "vsdm2-
183 service"}
184     }
185 ]
186 },
187 "scopeSpans": [
188     {
189         "scope": {
190             "name": "rs-tracing",
191             "version": "1.0.0"
192         },
193         "spans": [
194             {
195                 "traceId":
196                 "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
197                 ,
198                 "spanId": "2222222222222222",
199                 "parentSpanId":
200                 "1111111111111111",
201                 "name": "HTTP GET backend-
202 service",
203                 "kind": 3,
204                 "startTimeUnixNano":
205                 "1714311000205000000",
206                 "endTimeUnixNano":
207                 "1714311000900000000",
208                 "attributes": [
209                     {
210                         "key": "rs.statuscode",
211                         "value": {"stringValue":
212 "200"}
213                     }, {
214                         "key": "rs.duration",
215                         "value": {"intValue":
216 1234535345}
217                     }, {
218                         "key": "pdt.ikn",
219                         "value": {"intValue":
220 1000000000}
221                     }, {
222                         "key": "pdt.eTag",
```



```

223                                     "value": {"boolValue":
224                                     false}
225                                     }, {
226                                     "key": "pdt.eTagValue",
227                                     "value": {"boolValue":
228                                     false}
229                                     }
230                                 ],
231                                 "status": {
232                                     "code": 1
233                                 }
234                             }
235                         ]
236                     }
237                 ]
238             }
239         ]
240     }

```

242 Die folgende Tabelle gibt Aufschluss über die möglichen Trace-Parameter. Beschrieben
 243 werden die allgemein gültigen Parameter sowie die für alle Ressource Server gültigen,
 244 beginnend mit dem Suffix "rs". Im Beispiel sind zusätzlich Key / Value Paare mit dem
 245 Präfix "pdt_", diese sind produkttypspezifisch und werden in entsprechenden
 246 Unterkapiteln (siehe Kapitel 3) beschrieben. Die tabellarische Auflistung berücksichtigt
 247 nicht explizit die Hierarchie Ebene, diese kann aus dem Schema Beispiel entnommen
 248 werden.

249
 250 Tabelle 2 Tab_gemSpec_Perf_Trace_RSSpanParameter_Telemetriedatenlieferung

Daten	Open Telemetry Attribute (OTLP/JSON)	Pflichtfeld	Begründung und Zweck
Service Name	service.name	ja	Identifikationsmerkmal, fester Wert zur Erkennung des Resource Servers
Trace-Id	traceId	ja	Eindeutige ID eines Traces und Kennzeichnung zueinander gehörender Operationen (wird vom OTEL Framework gesetzt)
Parent Span-Id	parentSpanId	ja	Eindeutige ID zur Identifizierung von Hierarchien innerhalb der Operationen eines Traces (wird

			vom OTEL Framework gesetzt)
Startzeit	startTimeUnixNano	ja	Startzeitpunkt der Operation Entspricht "timestamp" der Betriebsdatenlieferung v2
Endzeit	endTimeUnixNano	ja	Endzeitpunkt der Operation
Resource Server Statuscode	rs.statuscode	ja	Ein max. 5-stelliger Statuscode des Ressource Servers gemäß [A_27722-*]. Produkttypspezifische fachliche Fehlercodes sind hier vorrangig vor "Standard-HTTP-Statuscodes" zu verwenden. Entspricht "status" der Betriebsdatenlieferung v2
Resource Server Duration	rs.duration	ja	Dies ist eine vom Ressource Server berechnete Zeitdauer (in Millisekunden), wie lange die Verarbeitung der Operation intern benötigte. Entspricht "duration_in_ms" der Betriebsdatenlieferung v2
Resource Server Instanz	rs.instance	nein	Dies ist eine zusätzliche fachliche Instanz des Resource Servers. Wenn vorhanden vom Ressource Server befüllt. Das Feld ist optional.
Produkt spezifische Attribute	pdt.*	Produkttyp spezifische Anforderungen sind zu beachten	Im Beispiel sind zusätzlich Key / Value Paare mit dem Präfix "pdt_", diese sind produkttypspezifisch und werden in entsprechenden Unterkapiteln (siehe Kapitel 3) beschrieben. Enthält die in der Betriebsdatenlieferung v2 beschriebenen Inhalte von "message".

{

"trace_id": "abcd1234efgh5678ijkl9012mnop3456",**"spans": [**

{

"span_id": "1111222233334444",

```
256 "service.name": "HTTP GET /api/user",
257 "start_time_unix_nano": 1678886400000000000,
258 "end_time_unix_nano": 1678886410000000000,
259 "attributes": [
260     {"http_fqdn": "vsdm2.aok.net"},
261     {"http_route": "/ressource/path/1"},
262     {"http.request.method": "GET"},
263     {"http.response.status_code": 200} ,
264     {"product_id": "CLIENTID1234567890AB"},
265     {"product_version": "2.1.3-10"},
266     {"prof0ID": "1.2.276.0.76.4.30"}
267 ],
268 "parent_id": null // Root-Span hat keinen Parent
269 },
270 {
271     "span_id": "5555666677778888",
272     "service.name": "rs-vsdm_2",
273     "start_time_unix_nano": 1678886430000000000,
274     "end_time_unix_nano": 1678886440000000000,
275     "attributes": [
276         {"http_fqdn": "vsdm2.aok.net"},
277         {"http_route": "/ressource/path/1"},
278         {"http.request.method": "GET"},
279         {"http.response.status_code": 200},
280         {"rs_statuscode": 200},
281         {"key": "pdt_size", "value": { "intValue":
282         456}},
283         {"key": "pdt_ikn", "value": { "intValue":
284         128}},
285         {"key": "pdt_eTag", "value": { "boolValue":
286         false}},
287         {"key": "pdt_eTagValue", "value":
288         { "boolValue": false}}
289     ],
290     "parent_id": "1111222233334444" // Parent-Span-ID
291     des Root-Spans
292 }
293 ]
294 }
295
```

Fehlt in der Kette der Operationen ein Span eines Ressource Servers kann daraus abgeleitet werden, dass der Ressource Server ausgefallen ist.

Die folgende Tabelle gibt Aufschluss über die möglichen Trace-Parameter. Beschrieben werden die allgemein gültigen Parameter sowie die für alle Ressource Server gültigen, beginnend mit dem Suffix "rs". Im Beispiel sind zusätzlich Key / Value Paare mit dem Präfix "pdt_", diese sind produkttypspezifisch und werden in entsprechenden Unterkapiteln (siehe Kapitel 3) beschrieben.

Tabelle 3: Tab_gemSpec_Perf_Trace_Parameter_Telemetriedatenlieferung

Attributname	zugehörige Operation	Beschreibung
trace_id	alle	Eindeutige ID eines Traces und Kennzeichnung zueinander gehörender Operationen. Das Feld darf nicht leer sein.
span_id	alle	Eindeutige ID einer einzelnen Operation innerhalb eines Traces. Das Feld darf nicht leer sein.
parent_id	alle	Eindeutige ID zur Identifizierung von Hierarchien innerhalb der Operationen eines Traces.
service.name	alle	Name der Operation bzw. Servicename. Das Feld darf nicht leer sein.
start_time_unix_nano	alle	Startzeitpunkt der Operation. Das Feld darf nicht leer sein.
end_time_unix_nano	alle	Endzeitpunkt der Operation. Das Feld darf nicht leer sein.
attributes:http_fqdn	alle	Vollständiger FQDN der ausgeführten Operation. Das Feld darf nicht leer sein.
attributes:http_route	alle	HTTP Route der ausgeführten Operation. Das Feld darf nicht leer sein.
attributes:http.request.method	alle	HTTP Methode der ausgeführten Operation. Das Feld darf nicht leer sein.
attributes:http.response.status_code	alle	3-stelliger HTTP Statuscode gemäß [RFC9110]. Das Feld darf nicht leer sein.

attributes:product_id	alle	Product ID aus dem Token Self-Assessment des aufrufenden Clientsystem
attributes:product_version	alle	Product Version aus demToken Self-Assessment des aufrufenden Clientsystems
attributes:profOID	alle	profession OID aus dem SM-B Zertifikat des aufrufenden Clientsystems. Das Attribut ist nur für die Usecases zu übermitteln, in denen ein SM-B-Zertifikat Anwendung findet.
attributes:rs_duration	Ressource Server des TI 2.0 Dienst	Dies ist eine vom Ressource Server berechnete Zeitdauer (in Millisekunden), wie lange die Verarbeitung der Operation intern benötigte. Das Feld ist optional.
attributes:rs_instance	Ressource Server des TI 2.0 Dienst	Dies ist eine zusätzliche fachliche Instanz des Resource Servers. Wenn vorhanden vom Ressource Server befüllt. Das Feld ist optional.
attributes:rs_statuscode	Ressource Server des TI 2.0 Dienst	Ein max. 5-stelliger Statuscode des Ressource Servers gemäß [A_27722-*]. Produktypspezifische fachliche Fehlercodes werden hier ebenfalls gemappt. Das Feld darf nicht leer sein.

...

3.2 Kapitel 2.5.4.2 Logs

Zusätzlich zur Nachvollziehbarkeit von verteilten Transaktionen in einer Anwendung mittels Traces kann OpenTelemetry Logs erfassen. Die Telemetriedatenlieferung stellt einen Custom Collector für die Selbstauskunft zur Verfügung. Der ZETA Guard greift diesen Collector auf und lässt ihn durch den Ressource Server befüllen. Die Selbstauskunft wird an den Telemetriedaten-Service über einen OTLP-Exporter als OTEL Log Signal gesendet. Der ZETA Guard stellt einen OTEL Collector für Empfang, Anreicherung und den Weiterversandt der Selbstauskunft zur Verfügung.

Logdaten MÜSSEN dem OpenTelemetry Log Data Model entsprechen und im Format des OpenTelemetry Log Protobuf erzeugt und übertragen werden. Die in dieser Spezifikation beschriebenen OpenTelemetry-Attribute stellen eine gezielte Teilmenge dar, die für sicherheitsrelevante Analysen und Auswertungen im Kontext der Selbstauskunft vorgesehen ist.

Diese Spezifikation erhebt keinen Anspruch auf Vollständigkeit hinsichtlich aller durch OpenTelemetry definierten Log-Attribute. Implementierungen DÜRFEN zusätzliche

Attribute gemäß OpenTelemetry erfassen und übertragen; diese sind jedoch nicht Gegenstand der normativen Anforderungen dieser Spezifikation, sofern sie die Auswertung der hier definierten Attribute nicht beeinträchtigen.

Hier ein Beispiel eines OTEL Log records.

```
{
  "resourceLogs": [
    {
      "resource": {
        "attributes": [
          {"key": "product_nameproduct.name", "value": {"stringValue": "VSDM2_HerstellerX"}},
          {"key": "product.code_name", "value": {"stringValue": "VSDM2X"}},
          {"key": "producttyp_nameproduct.typ_name", "value": {"stringValue": "PDT79"}},
          {"key": "lpi.ci_id", "value": {"stringValue": "CI-9876543"}},
          {"key": "lpi.instance", "value": {"stringValue": "a"}},
          {"key": "pod_name", "value": {"stringValue": "VSDM2.pod"}},
          {"key": "product.vendor_id", "value": {"stringValue": "HERST"}},
          {"key": "product.vendor_name", "value": {"stringValue": "Hersteller ABC"}}
        ]
      },
      "scopeLogs": [
        {
          "scope": {
            "name": "com.example.MyLogger",
            "version": "1.0.0"
          },
          "logRecords": [
            {
              "timeUnixNano": 1678886400000000000,
              "body": { "stringValue": "Selbstauskunft" },
              "severityText": "product_info",
              "severityNumber": 9,
```

```

364         "attributes": [
365             { "key":
366               "product_versionproduct.vers
367               ion
368             }, "value": { "stringValue":
369               "1.2.0-0" }},
370             { "key":
371               "producttype_versionproduct.
372               type_version
373             }, "value": { "stringValue":
374               "1.0.3" }},
375             { "key":
376               "configuration_versionconfig
377               uration.version
378             }, "value": {"stringValue":
379               "1.3.4-2" }}
380         ]
381     }
382 ]
383 }
384 ]
385 }
386 ]
387 }

```

Die zum OpenTelemetry Log "product_info" gehörenden Attribute werden in der folgenden Tabelle beschrieben. Hier in der Darstellung wird die OTLP/JSON Notation verwendet, die notwendigen und zugehörigen Attribute der OTLP/gRPC Notation (Proto) - welche für die technische Implementation der Anwendung relevant sind - sind der jeweils aktuell gültigen und veröffentlichten Beschreibung des OpenTelemetry Projektes zu entnehmen.

Tabelle 4: Tab_gemSpec_Perf_Telemetriedaten_product_info

Daten	OpenTelemetry Attribute (OTLP/JSON)	Pflichtfeld	Begründung und Zweck
Fester Wert zur Definition der Selbstauskunft	body	ja	"Selbstauskunft"
Fester Wert zur Definition der Selbstauskunft	severityText	ja	"product_info"
Fester Wert zur Definition der	severityNumber	ja	9

Selbstauskunft			
Produktname	product.name	Ja	Produktname, vom Hersteller vergeben Entspricht "ProductName" der Selbstauskunft v1
Produktkurzname	product.code_name	Ja	Produktkurzname, vom Hersteller vergeben Entspricht "ProductCodt" der Selbstauskunft v1
Produkttyp	product.type_name	Ja	Produkttyp mit numerischen Code, nach Vorgaben der gematik Entspricht "ProductType" der Selbstauskunft v1
CI-Identifikator	lpi.ci_id	Ja	Identifikation des Konfiguration-Elementes "logische Produktinstanz", beginnend mit "CI" Entspricht "CI-ID" der Selbstauskunft v1
CI-Instanz	lpi.instance	Nein	Identifikation der logischen Produktinstanz, sofern redundante Instanzen vorhanden sind
Pod Name	pod.name	Nein	Identifikation des laufenden Pods
Hersteller ID	product.vendor_id	Ja	Hersteller ID Entspricht "ProductVendorID" der Selbstauskunft v1
Herstellername	product.vendor_name	Ja	ausführlicher Herstellername Entspricht "ProductVendorName" der Selbstauskunft v1
Produktversion	product.version	Ja	Aktuelle Produktversion, vom Hersteller vergeben Entspricht "ProductVersion" der Selbstauskunft v1
Produkttyp Version	product.type_version	Ja	Aktuelle Produkttypversion, nach Vorgaben der gematik Entspricht "ProductTypeVersion" der

			Selbstauskunft v1
Konfigurationsversion	configuration.version	Ja	Aktuelle Konfigurationsversion, vom Hersteller vergeben

396

Parameter	Beschreibung
resource:attributes:product_name	Name des Produkts (durch Hersteller)
resource:attributes:producttyp_name	Name des Produkttyp (PDT-Nummer der gematik)
resource:attributes:pod_name	Name des Pods/der Instanz (durch Hersteller)
scopeLogs:logRecords:timeUnixNano	Zeitpunkt der letzten Änderung
scopeLogs:logRecords:body	{ "stringValue": "Selbstauskunft" }
scopeLogs:logRecords:severityText	"product_info"
scopeLogs:logRecords:severityNumber	9
scopeLogs:logRecords:attributes:product_version	Aktuelle Produktversion (durch Hersteller)
scopeLogs:logRecords:attributes:producttype_version	Aktuelle Produkttypversion (PTV des Steckbriefes)
scopeLogs:logRecords:attributes:configuration_version	Aktuelle Konfigurationsversion (durch Hersteller)

397

3.3 Kapitel 2.5.4.4 Anforderungen an Telemetriedatenlieferungen

In A_27723 muss die Referenz auf die Traceparametertabellen geändert werden. Der Fokus der Afo wird geändert und zielt nur noch auf die Lieferung zwischen RS und ZETA Guard.

...

A_27723-03 -Performance - Telemetriedatenlieferung - Lieferung RS

Der Produkttyp MUSS für jede ausgeführte Operation die Betriebsdaten (Traces) mit Traceparametern gemäß der Tabelle "Tab_gemSpec_Perf_Trace_RSspanParameter_Telemetriedatenlieferung" in einer Telemetriedatenlieferung an den Telemetriedaten-Service senden. Das Senden der

Telemetriedaten SOLL asynchron erfolgen, um die Performance der Operation nicht negativ zu beeinflussen.

Hinweis: Ist der ZETA Guard Teil des Produkttyps (z.B. VSDM2) gilt die Nachweispflicht für die Sendung an den Telemetriedaten-Service des ZETA Guards, in anderen Fällen (z.B. sektoraler IDP) an den Telemetriedaten-Service der gematik. [≤, DiPag_FD, VSDM_2_FD, PoPP_Service, Herst_TI-D_ZT, ZT_PIP_PAP, funkt. Eignung: Test Produkt/FA]

neue Afo mit Referenz auf die Traceparameter für die Lieferung zwischen ZETA Guard und Telemetriedatencloud.

A_29020 -Performance - Telemetriedatenlieferung - Lieferung ZETA Guard HTTP Proxy

Der Produkttyp MUSS für jede ausgeführte Operation die Betriebsdaten (Traces) mit Traceparametern gemäß der Tabelle

"Tab_gemSpec_Perf_Trace_HTTPSpanParameter_Telemetriedatenlieferung" in einer Telemetriedatenlieferung an den Telemetriedaten-Service senden. Das Senden der Telemetriedaten SOLL asynchron erfolgen, um die Performance der Operation nicht negativ zu beeinflussen.

[≤, ZT_Cluster, funkt. Eignung: Test Produkt/FA]

Zuweisen an ZT_Cluster - Test Produkt / FA

bei Afo A_28782 wird die Zuweisung des ZT_Clusters geändert:

alt ZT_Cluster - funkt. Eignung: Test Produkt/FA

neu ZT_Cluster - funkt. Eignung: Herstellererklärung

...

4 Änderung in gemSpec_ZETA

A_27494-01 wird angepasst und auf die neue Tabelle aus gemSpec_Perf referenziert:

A_27494-02 -Telemetriedaten-Service, Custom Collector für Selbstauskunft

Der Telemetriedaten-Service MUSS einen OTEL Collector für die Selbstauskunft des Resource Servers bereitstellen. Die Selbstauskunft des Resource Servers erfolgt für jede eigenständige Instanz als OTLP Log Record.

Der Telemetriedaten-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.

Hinweis: Der OTLP Log Record für die Selbstauskunft ist wie folgt in gemSpec_Perf (Tab_gemSpec_Perf_Telemetriedaten_product_info) 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).

【<=, ZT_Cluster, funkt. Eignung: Herstellererklärung】