AMCON
Software, die bewegt

# Dokumentation zur API

vom UIC-Sicherheitsportal

des Deutschlandtarifverbunds

ANBIETER: AMCON Software GmbH

DATUM: 26. Juni 2025

VERSION: 1.3

**1 Glossar ..... 1**

**2 Einleitung ..... 3**

**3 API-Spezifikation ..... 3**

**4 Systemgrundlagen ..... 4**

**4.1 Organisationen und RICS-Codes ..... 4**

**4.2 API-Schlüssel und Berechtigungen ..... 4**

**4.3 Ticket-Identifikation ..... 4**

**5 Authentifizierung ..... 5**

**5.1 API-Key ..... 5**

**5.2 OAuth2 ..... 5**

**6 Rate Limiting ..... 6**

**7 Datenschemata ..... 7**

**7.1 Anfrage ..... 7**

7.1.1 Ticket ..... 7

7.1.2 Tickets ..... 7

7.1.3 Kontrolle ..... 8

7.1.4 Kontrollnachweise ..... 9

7.1.5 Ausgabenachweis ..... 10

7.1.6 Ausgabenachweise ..... 12

7.1.7 Deutschlandticket ..... 13

**7.2 Antwort ..... 15**

7.2.1 Eintrag Sperrlistenübersicht ..... 15

7.2.2 Sperrlistenübersicht ..... 15

7.2.3 Sperrlisteneintrag ..... 16

7.2.4 Sperrliste ..... 16

7.2.5 Kontrollereignis ..... 17

7.2.6 Kontrollergebnis ..... 17

7.2.7 Ticket ..... 18

7.2.8 Ticket-Barcode ..... 21

**8 API-Endpunkte ..... 22**

©AMCON Software GmbH

**8.1 Authentifizierung (OAuth2) ... 22**

8.1.1 POST: /api/v1/auth/token ... 22

**8.2 Blacklist (Sperrliste) ... 24**

8.2.1 GET: /api/v1/blacklist ... 24

8.2.2 GET: /api/v1/blacklist/{id} ... 25

8.2.3 GET: /api/v1/blacklist/latest ... 26

**8.3 Ticket (-Interaktionen) ... 28**

8.3.1 POST: /api/v1/ticket/lock ... 28

8.3.2 POST: /api/v1/ticket/unlock ... 29

8.3.3 POST: /api/v1/ticket/cancel ... 30

8.3.4 POST: /api/v1/ticket/issuancerecord ... 31

8.3.5 POST: /api/v1/ticket/issuance/dticket ... 32

8.3.6 GET: /api/v1/ticket/issuance/renderbarcode ... 33

**8.4 Validation (Kontrolle) ... 35**

8.4.1 POST: /api/v1/validation/validate ... 35

8.4.2 POST: /api/v1/validation/records ... 36

**9 Fehlerbehandlung ... 38**

**10 Ergänzungen / Anmerkungen ... 40**

**10.1 Batch-Anfragen ... 40**

**10.2 Offline-Sperrliste ... 40**

**10.3 Sonstiges ... 40**

©AMCON Software GmbH

# 1 Glossar

| | |
|---|---|
| Ticket | Ein Ticket im öffentlichen Personenverkehr ist eine Berechtigung, die einem Fahrgast den Zugang zur Nutzung von Verkehrsmitteln wie Bussen, Bahnen, Zügen oder Fähren ermöglicht. Es dient als Nachweis für den Erwerb eines Beförderungsrechts und kann unterschiedliche Formen haben, darunter physische Papierfahrkarten, elektronische Tickets (e-Tickets) oder digitale Tickets, die auf mobilen Endgeräten gespeichert sind. |
| Fahrtberechtigung | Eine Fahrtberechtigung im öffentlichen Personenverkehr ist die rechtliche und organisatorische Grundlage, die einer Person das Recht einräumt, ein bestimmtes Verkehrsmittel oder eine festgelegte Strecke zu nutzen. Sie wird in der Regel durch den Besitz eines gültigen ***Tickets***, Abonnements oder einer entsprechenden Erlaubnis (z. B. Freifahrten, Dienstfahrscheine) nachgewiesen. |
| Ticketaussteller, Ausgeber | Ein Ticketaussteller (Ausgeber) ist eine natürliche oder juristische Person, die ***Tickets*** im öffentlichen Personenverkehr erstellt, verkauft oder zur Verfügung stellt. Ticketaussteller agieren als Dienstleister, um Fahrgästen eine rechtsgültige ***Fahrtberechtigung*** für die Nutzung von Verkehrsmitteln wie Bussen, Bahnen oder Fähren anzubieten.<br><br>Beispiele für Ticketaussteller sind:<br><br>- Verkehrsunternehmen<br>- Verkehrsverbünde<br>- Drittanbieter und Ticketplattformen<br>- Lokale Verkaufsstellen und Agenturen<br>- Ticketautomaten und Selbstbedienungsgeräte<br>- Internationale Verkehrsunternehmen |
| Ticketkontrolleur, Ticketprüfer, TCO | Ein Ticketkontrolleur (Ticketprüfer, TCO) ist eine natürliche oder juristische Person, die im öffentlichen Personenverkehr dafür verantwortlich ist, die Gültigkeit und Echtheit von ***Fahrtberechtigungen*** (z. B. ***Tickets***, Abonnements) der Fahrgäste zu überprüfen. Diese Tätigkeit dient der Sicherstellung, dass die Beförderungsbedingungen eingehalten werden und Fahrgäste für die Nutzung der Verkehrsmittel korrekt bezahlt haben.<br><br>Beispiele für Ticketkontrolleure sind:<br><br>- Mitarbeiter von Verkehrsunternehmen<br>- Mitarbeiter von Verkehrsverbünden<br>- Beauftragte Sicherheitsdienstleister<br>- Automatisierte Kontrollsysteme<br>- Fahrer oder Fahrzeugführer |
| Kundenvertragspartner, KVP | Ein Kundenvertragspartner (KVP) ist eine natürliche oder juristische Person, die als primäre Ansprech- und Vertragsperson für Kunden im öffentlichen Personenverkehr fungiert. Der KVP ist verantwortlich für das Aushandeln, Abschließen und Verwalten von Kundenverträgen und sorgt dafür, dass Kunden eine |

| | |
|---|---|
| | verbindliche Vereinbarung für die Nutzung von Verkehrsmitteln wie Bussen, Bahnen oder Fähren haben. KVPs bieten umfassenden Support und Dienstleistungen, um sicherzustellen, dass die Kundenbedürfnisse erfüllt und die rechtlichen Rahmenbedingungen eingehalten werden.<br><br>Beispiele für Kundenvertragspartner sind:<br><br>- Verkehrsunternehmen, die direkte Dienstleistungen und Kundenverträge anbieten.<br>- Verkehrsverbünde, die regional unterschiedliche Verkehrsangebote bündeln und Kunden vertraglich binden.<br>- Drittanbieter-Plattformen, die als Vermittler zwischen Kunden und Verkehrsanbietern agieren.<br>- Lokale Verkaufsagenturen, die spezifisch mit Kunden Verträge eingehen und als Bindeglied zu den Verkehrsdiensten fungieren.<br>- Kundendienstzentren, die als zentrale Anlaufstelle für vertragliche Anliegen der Fahrgäste agieren. |
| Security Provider | Ein Security Provider ist technisch verantwortlich für die Barcodeerzeugung und Signierung von ***Tickets***. Der Security Provider garantiert für die Echtheit von Tickets – in diesem Falle für die Echtheit der generierten UIC-Barcodes. |

# 2 Einleitung

Der Zweck des UIC Sicherheitsportals ist die Gewährleistung einer hohen Sicherheit der Kontrolle von Tickets mit UIC-Barcode. Dies wird realisiert durch die zentrale Erzeugung von Barcodes, der Möglichkeit diese zentral zu sperren und gegen die Ausgabe- und Sperrdaten online zu kontrollieren. Zusätzlich wird eine Sperrliste ausgegeben, die auch eine sicherer Kontrolle auf nicht onlinefähigen Kontrollgeräten ermöglichen soll. Kontrolleure, die nicht die Onlinekontrolle nutzen, können zudem Kontrollnachweise übertragen.

Diese Dokumentation richtet sich an Entwickler, die die API in ihre Anwendungen integrieren möchten. Sie enthält alle notwendigen Informationen und Beispiele, um die Integration erfolgreich durchzuführen.

# 3 API-Spezifikation

Die API wird durch eine OpenAPI-Spezifikation beschrieben. Diese Spezifikation ermöglicht es, für die meisten Programmiersprachen Quellcode zu generieren und beschreibt detailliert, wie die Endpunkte verwendet werden und welche Einschränkungen gelten.

Die Spezifikation kann unter folgender URL heruntergeladen werden:

**DTVG-Sicherheitsportal OpenAPI-Spezifikation** (Version 1.0.3)

Eine benutzerfreundliche UI zur Anzeige der Spezifikation ist ebenfalls verfügbar:

**Spezifikation (UI)** (Version 1.0.3)

# 4 Systemgrundlagen

## 4.1 Organisationen und RICS-Codes

Im System können von einem Administrator Organisationen angelegt werden. Eine Organisation entspricht immer genau einem RICS-Code (Railway Interchange Coding System). Für jede Organisation kann festgelegt werden, ob diese als "Ticket-Aussteller" und/oder "Ticket-Kontrolleur" fungiert.

## 4.2 API-Schlüssel und Berechtigungen

Für jede Organisation können beliebig viele API-Schlüssel angelegt werden. Mit diesen Schlüsseln können die API-Endpunkte aufgerufen werden. Jedem API-Schlüssel können spezifische Berechtigungen zugewiesen werden, die die Verwendung der einzelnen Endpunkte regeln.

Das System selbst gibt bestimmte Einschränkungen für die Berechtigungen eines API-Schlüssels vor:

- Wenn eine Organisation nicht als "Ticket-Aussteller" gekennzeichnet ist, kann sie keinen API-Schlüssel mit der Berechtigung zum Sperren oder Entsperren von Tickets erstellen.
- Jede Organisation kann nur Tickets sperren oder entsperren, die zu dieser Organisation gehören (der RICS-Code des Tickets muss mit dem der Organisation übereinstimmen).

## 4.3 Ticket-Identifikation

Im gesamten System wird ein Ticket immer durch folgende Werte eindeutig identifiziert:

- RICS (Issuer)
- TicketId (Issuer PNR)
- Gültigkeitsende des Tickets (validTo)

Es ist wichtig, dass immer genau die Werte verwendet werden, die im UIC-Ticket (FCB) gespeichert sind.

# 5 Authentifizierung

Das Sicherheitsportal bietet zwei verschiedene Authentifizierungsmechanismen. API-Keys und OAuth2. Mit OAuth2 können theoretisch alle Endpunkte authentifiziert werden, während die möglichen Endpunkte mit API-Key eingeschränkt sind.

Jeder API-Endpunkt in dieser Dokumentation hat auch definiert, was möglich ist.

## 5.1 API-Key

Die grundlegende Authentifizierungsmethode erfolgt über API-Schlüssel. Bei jeder Anfrage muss der API-Schlüssel im HTTP-Header „Authorization“ mitgesendet werden. Der API-Key wird dabei direkt als Wert gesetzt, ohne Präfix oder Suffix.

## 5.2 OAuth2

Bei OAuth2 erfolgt die Authentifizierung mithilfe von Client Credentials (bestehend aus ClientId und ClientSecret), um zunächst ein Access- und ein Refresh-Token zu erhalten (nähere Informationen dazu finden Sie beim Token-Endpunkt). Damit die Authentifizierung über Client Credentials überhaupt möglich ist, muss zuerst die Freischaltung des Logins im Sicherheitsportal erfolgen. Nachdem dieser Schritt abgeschlossen ist, kann die Authentifizierung ein einziges Mal durchgeführt werden, woraufhin der Login wieder deaktiviert wird. Um sich gegenüber einem API-Endpunkt zu authentifizieren, ist es erforderlich, das Access-Token (als JWT) im HTTP-Header unter "Authorization" mitzuführen, wobei der Präfix "Bearer" verwendet werden muss.

Nach der ersten Authentifizierung sollten Clients nicht wiederholt mit den Client Credentials authentifiziert werden. Stattdessen sollte der Refresh-Token genutzt werden, welcher ebenfalls vom Token-Endpunkt bereitgestellt wird. Dieser Refresh-Token kann gegen neue Access- und Refresh-Tokens eingetauscht werden.

Ein Access-Token hat für gewöhnlich nur eine Gültigkeit von wenigen Minuten, während der Refresh-Token über eine wesentlich längere Dauer gültig bleibt (über einen Monat). Pro Client Credentials können immer nur ein aktives Access-Token und ein Refresh-Token existieren. Sobald eine erneute Authentifizierung mit den Client Credentials erfolgt oder der Refresh-Token genutzt wird, verlieren alle anderen bestehenden Tokens ihre Gültigkeit.

# 6 Rate Limiting

Alle Anfragen gegen die API unterliegen einem Rate Limit. Das Rate Limit wird pro API-Key bzw. pro OAuth2-Credential festgelegt. Initial hinterlegt ist ein Standardwert von 300 Anfragen pro Minute, der nicht überschritten werden darf.

Technisch wird ein Token-Bucket verwendet, welcher anfangs mit einer konfigurierten Anzahl an Tokens (im Standard 300 Tokens) befüllt ist. Alle 10 Sekunden wird 1/6 des Buckets wieder aufgefüllt (im Standard 50 Tokens alle 10 Sekunden) bis zum Maximalwert (im Standard auch 300 Tokens). Jede Anfrage an die API nimmt genau einen Token aus dem Bucket heraus. Ist der Bucket leer, wird die jeweilige Anfrage mit dem Statuscode „429 – Too Many Requests“ abgelehnt.

Das Refill-Intervall von 1/6 pro 10 Sekunden bedeutet auch, dass eine Dauerbelastung der API von (1/6 * Rate Limit) / 10 s nicht überschritten werden darf. Denn bei 300 Anfragen pro Minute dürfen unter Last entsprechend nur 50 Anfragen alle 10 Sekunden durchgeführt werden.

Das aufrufende System muss sich entsprechend darauf einstellen, dass im Lastbetrieb nicht direkt in der ersten Sekunde einer Minute das gesamte Rate Limit aufgebraucht wird (im Standard 300 Anfragen alle 60 Sekunden) und ohne Pause weiter Anfragen gesendet werden – dies würde zu einer Ablehnung mit Statuscode 429 führen. Die Anfragen sollten stattdessen über die gesamte Minute verteilt werden (im Standard 50 Anfragen alle 10 Sekunden).

Der Token-Bucket hat einen automatischen Puffer. Wenn mindestens eine Minute keine Anfrage gesendet wurde und der Bucket voll ist (im Standard 300 Tokens), kann in den nächsten 60 Sekunden die doppelte Anzahl des Rate Limits an Anfragen gesendet werden (im Standard 600 Anfragen), da sich der Bucket wiederum alle 10 Sekunden um eine feste Anzahl an Tokens regeneriert (im Standard 50 Tokens alle 10 Sekunden). Der Ablauf wäre dann der folgende: Direkt zu Beginn 300 Anfragen und anschließend schrittweise 50 Anfragen alle 10 Sekunden.

Die tatsächlich erreichbare Anzahl möglicher Anfragen kann aufgrund von Race Conditions geringfügig vom eingestellten Rate Limit abweichen, jedoch nur nach oben (z.B. 305 Anfragen pro Minute bei einem Rate Limit von 300 Anfragen pro Minute).

Sollte das Standard Rate Limit von 300 Anfragen pro Minute nicht ausreichen, so kann dies auf Anfrage manuell erhöht werden.

©AMCON Software GmbH

# 7 Datenschemata

## 7.1 Anfrage

### 7.1.1 Ticket

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| rics | RICS-Code des Ticketausstellers | string | Ja | Min.: 4 Zeichen Max.: 5 Zeichen | "5143" |
| ticketId | IssuerPNR aus dem UIC-Ticket | string | Ja | | "A0815BF0" |
| validTo | Ticket gültig bis | string | Ja | ISO 8601 Date | "2025-03-01T03:00:00+01:00" |

**Beispiel – JSON**

```
{
  "rics":"5143",
  "ticketId":"A0815BF0",
  "validTo":"2025-03-01T03:00:00+01:00"
}
```

### 7.1.2 Tickets

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| tickets | Liste der Tickets, welche in der jeweiligen Anfrage verarbeitet werden sollen | Array Typ: 7.1.1 Ticket | Ja | 1 – 10.000 Elemente innerhalb des Arrays | Siehe Beispiel – JSON |

**Beispiel – JSON**

```
{
  "tickets":[
    {
      "rics":"9999",
      "ticketId":"A0815BF0",
      "validTo":"2025-03-01T03:00:00+01:00"
```

```
    },
    {
      "rics":"9999",
      "ticketId":"BC93DE99",
      "validTo":"2025-03-01T03:00:00+01:00"
    }
  ]
}
```


### 7.1.3 Kontrolle

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| rics | RICS-Code des Ticketausstellers | string | Ja | Min.: 4 Zeichen Max.: 5 Zeichen | "5143" |
| ticketId | IssuerPNR aus dem UIC-Ticket | string | Ja | | "A0815BF0" |
| validFrom | Ticket gültig von | string | Ja | ISO 8601 Date | "2025-02-01T00:00:00+01:00" |
| validTo | Ticket gültig bis | string | Ja | ISO 8601 Date | "2025-03-01T03:00:00+01:00" |
| productId | ID des Produkts | int | Ja | | 9999 |
| tariffDescription | Ticket-Name / -Beschreibung aus Tarif | string | Ja | | "Deutschlandticket" |
| issuedAt | Ticket ausgestellt am | string | Ja | ISO 8601 Date | "2025-01-25T02:00:00+01:00" |
| validatedAt | Kontrollzeitpunkt. Wenn leer, wird der aktuelle Zeitpunkt angenommen. | string | Nein | ISO 8601 Date | "2025-02-15T10:30:00+01:00" |
| keyId | KeyId, mit welcher das Ticket signiert wurde | string | Ja | 5 Zeichen | "31A33" |

| | | | | | |
|---|---|---|---|---|---|
| securityProvider-Rics | RICS-Code des Security-Providers | string | Ja | Min.: 4 Zeichen Max.: 5 Zeichen | "3634" |

**Beispiel – JSON**


```
{
  "rics":"5143",
  "ticketId":"A0815BF0",
  "validFrom":"2025-02-01T00:00:00+01:00",
  "validTo":"2025-03-01T03:00:00+01:00",
  "productId":9999,
  "tariffDescription":"Deutschlandticket",
  "issuedAt":"2025-01-25T02:00:00+01:00",
  "validatedAt":"2025-02-15T10:30:00+01:00",
  "keyId":"31A33",
  "securityProviderRics":"3634"
}
```


### 7.1.4 Kontrollnachweise

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| validationRecords | Liste der Tickets, für welche Kontrollnachweise veröffentlicht werden sollen | Array Typ: 7.1.3 Kontrolle | Ja | 1 – 10.000 Elemente innerhalb des Arrays | Siehe Beispiel – JSON |

**Beispiel – JSON**


```
{
  "validationRecords":[
    {
      "rics":"5143",
      "ticketId":"A0815BF0",
      "validFrom":"2025-02-01T00:00:00+01:00",
      "validTo":"2025-03-01T03:00:00+01:00",
      "productId":9999,
      "tariffDescription":"Deutschlandticket",
      "issuedAt":"2025-01-25T02:00:00+01:00",
      "validatedAt":"2025-02-15T10:30:00+01:00",
      "keyId":"31A33",
      "securityProviderRics":"3634"
    },
```

```
    {
      "rics":"5143",
      "ticketId":"BC93DE99",
      "validFrom":"2025-02-01T00:00:00+01:00",
      "validTo":"2025-03-01T03:00:00+01:00",
      "productId":9999,
      "tariffDescription":"Deutschlandticket",
      "issuedAt":"2025-01-25T02:00:00+01:00",
      "validatedAt":"2025-02-15T10:30:00+01:00",
      "keyId":"31A33",
      "securityProviderRics":"3634"
    }
  ]
}
```


### 7.1.5 Ausgabenachweis

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| rics | RICS-Code des Ticketausstellers | string | Ja | Min.: 4 Zeichen Max.: 5 Zeichen | "5143" |
| ticketId | IssuerPNR aus dem UIC-Ticket | string | Ja | | "A0815BF0" |
| validFrom | Ticket gültig von | string | Ja | ISO 8601 Date | "2025-02-01T00:00:00+01:00" |
| validTo | Ticket gültig bis | string | Ja | ISO 8601 Date | "2025-03-01T03:00:00+01:00" |
| productId | ID des Produkts | int | Ja | | 9999 |
| tariffDescription | Ticket-Name / -Beschreibung aus Tarif | string | Ja | | "Deutschlandticket" |
| issuedAt | Ticket ausgestellt am | string | Ja | ISO 8601 Date | "2025-01-25T02:00:00+01:00" |
| keyId | KeyId, mit welcher das Ticket signiert wurde | string | Ja | 5 Zeichen | "31A33" |

| | | | | | |
|---|---|---|---|---|---|
| securityProvider-Rics | RICS-Code des Security-Providers | string | Ja | Min.: 4 Zeichen Max.: 5 Zeichen | "3634" |
| postcode | Postleitzahl des Ticketin-habers | string | Nein | Min.: 5 Zeichen Max.: 5 Zeichen | "49661" |
| issuerAuthoriza-tionId | VDV KA Au-thorisations-ID des Ti-ckets (Be-rechtigungs-nummer) | int | Nein | | 15312312 |
| issuerKvpId | VDV KA Kun-denver-tragspartner-ID (KVP-ID) | int | Nein | | 6321 |
| uniqueOrderId | Eindeutige Bestellnum-mer als Iden-tifikator für Fremdsys-tem | string | Nein | | "6e25585b-2e7e-4ea1-99c3-a5f743693f21" |

**Beispiel – JSON**


```
{
  "rics":"5143",
  "ticketId":"A0815BF0",
  "validFrom":"2025-02-01T00:00:00+01:00",
  "validTo":"2025-03-01T03:00:00+01:00",
  "productId":9999,
  "tariffDescription":"Deutschlandticket",
  "issuedAt":"2025-01-25T02:00:00+01:00",
  "keyId":"31A33",
  "securityProviderRics":"3634",
  "postcode":"49661",
  "issuerAuthorizationId":15312312,
  "issuerKvpId":6321,
  "uniqueOrderId":"6e25585b-2e7e-4ea1-99c3-a5f743693f21"
}
```

### 7.1.6 Ausgabenachweise

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| issuanceRecords | Liste der Tickets, für welche Ausgabenachweise veröffentlicht werden sollen | Array Typ: 7.1.5 Ausgabenachweis | Ja | 1 – 10.000 Elemente innerhalb des Arrays | Siehe Beispiel – JSON |

**Beispiel – JSON**


```
{
  "issuanceRecords":[
    {
      "rics":"5143",
      "ticketId":"A0815BF0",
      "validFrom":"2025-02-01T00:00:00+01:00",
      "validTo":"2025-03-01T03:00:00+01:00",
      "productId":9999,
      "tariffDescription":"Deutschlandticket",
      "issuedAt":"2025-01-25T02:00:00+01:00",
      "keyId":"31A33",
      "securityProviderRics":"3634",
      "postcode":"49661",
      "issuerAuthorizationId":15312312,
      "issuerKvpId":6321,
      "uniqueOrderId":"6e25585b-2e7e-4ea1-99c3-a5f743693f21"
    },
    {
      "rics":"5143",
      "ticketId":"B33041F0",
      "validFrom":"2025-02-01T00:00:00+01:00",
      "validTo":"2025-03-01T03:00:00+01:00",
      "productId":9999,
      "tariffDescription":"Deutschlandticket",
      "issuedAt":"2025-01-25T02:00:00+01:00",
      "keyId":"31A33",
      "securityProviderRics":"3634",
      "postcode":"49681",
      "issuerAuthorizationId":15312313,
      "issuerKvpId":6321,
      "uniqueOrderId":"fe97b98f-b3c6-40bb-acef-d78a72297dce"
    }
  ]
}
```

### 7.1.7 Deutschlandticket

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| firstName | Vorname des Ticketinhabers | string | Ja | | "Maxima" |
| lastName | Nachname des Ticketinhabers | string | Ja | | "Musterfrau" |
| dateOfBirth | Geburtsdatum des Ticketinhabers | string | Ja | ISO 8601 Date | "1990-05-30" |
| gender | Geschlecht des Ticketinhabers | int | Ja | 0 = Unspecified<br>1 = Female<br>2 = Male<br>3 = Other | 1 |
| productId | ID des Produkts. Üblicherweise 9999 für ein Standard D-Ticket | int | Ja | Min.: 9995<br>Max.: 9999 | 9999 |
| priceInCents | Ticket-Preis in Cent | int | Ja | | 4900 |
| monthOfValidity | Gültigkeitsmonat des Tickets. Die Tagesangabe wird ignoriert. | string | Ja | ISO 8601 Date | "2025-06-01" |
| postcode | Postleitzahl des Ticketinhabers | string | Ja | Min.:<br>5 Zeichen<br>Max.:<br>5 Zeichen | "49661" |
| uniqueOrderId | Eindeutige Kennung aus dem aufrufenden (Fremd-)System. Falls zum übergebenen Wert ein Ticket | string | Ja | | "28d07a4f-02ee-4c6f-996d-89e47670157b" |

| | | | | | |
|---|---|---|---|---|---|
| | existiert, wird das bereits vorhandene Ticket storniert. | | | | |

**Beispiel – JSON**


```
{
  "firstName":"Maxima",
  "lastName":"Musterfrau",
  "dateOfBirth":"1990-05-30",
  "gender":1,
  "productId":9999,
  "priceInCents":4900,
  "monthOfValidity":"2025-06-01",
  "postcode":"49661",
  "uniqueOrderId":"28d07a4f-02ee-4c6f-996d-89e47670157b"
}
```

# 7.2 Antwort

## 7.2.1 Eintrag Sperrlistenübersicht

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| blacklistId | ID der Sperrliste | int | | 1 |
| createdAt | Erstelldatum der Sperrliste | string | ISO 8601 Date | "2025-02-12T13:00:00.065319+00:00" |
| numberOfEntries | Anzahl Sperrlisteneinträge | int | | 2 |

**Beispiel – JSON**


```
{
  "blacklistId":1,
  "createdAt":"2025-02-12T13:00:00.065319+00:00",
  "numberOfEntries":2
}
```


## 7.2.2 Sperrlistenübersicht

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| | Liste der vorhandenen Sperrlisten | Array Typ: 7.2.1 Eintrag Sperrlistenübersicht | | Siehe Beispiel – JSON |

**Beispiel – JSON**


```
[
  {
    "blacklistId":4,
    "createdAt":"2025-02-13T13:00:00.065319+00:00",
    "numberOfEntries":5
  },
  {
    "blacklistId":3,
    "createdAt":"2025-02-12T13:00:00.065319+00:00",
    "numberOfEntries":3
  }
]
```

### 7.2.3 Sperrlisteneintrag

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| rics | RICS-Code des Ticketausstellers | string | Min.: 4 Zeichen Max.: 5 Zeichen | "5143" |
| ticketId | IssuerPNR aus dem UIC-Ticket | string | | "A0815BF0" |

**Beispiel – JSON**

```
{
  "rics":"5143",
  "ticketId":"A0815BF0"
}
```

### 7.2.4 Sperrliste

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| blacklistId | ID der Sperrliste | int | | 1 |
| createdAt | Erstelldatum der Sperrliste | string | ISO 8601 Date | "2025-02-12T13:00:00.065319+00:00" |
| numberOfEntries | Anzahl Sperrlisteneinträge | int | | 2 |
| tickets | Auf der Sperrliste befindliche Tickets | Array Typ: 7.2.3 Sperrlisteneintrag | | Siehe Beispiel – JSON |

**Beispiel – JSON**

```
{
  "blacklistId":1,
  "createdAt":"2025-02-12T13:00:00.065319+00:00",
  "numberOfEntries":2,
  "tickets":[
    {
      "rics":"9999",
      "ticketId":"1337"
    },
    {
      "rics":"9999",
      "ticketId":"1338"
    }
  ]
}
```

### 7.2.5 Kontrollereignis

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| id | Gibt die Art des Kontrollereignisses an | int | | 1 |
| description | Beschreibt die Ereignisart | string | | "Gegebenenfalls die Personalien prüfen" |
| details | Weitere Details zum Kontrollereignis | string | | "Das Ticket wurde mehr als 1 Mal kontrolliert in den letzten 5 Minuten" |
| data | Gibt die rohen Daten an, die zum Ereignis geführt haben. Der Objekttyp ist abhängig von der Id. | object | | { "interval": 5, "validations": 1 } |

**Beispiel – JSON**


```
{
  "id":1,
  "description":"Gegebenenfalls die Personalien prüfen",
  "details":"Das Ticket wurde mehr als 1 Mal kontrolliert in den letzten 5 Minuten",
  "data":{
    "interval":5,
    "validations":1
  }
}
```


### 7.2.6 Kontrollergebnis

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| isValid | Gibt an, ob das abgefragte Ticket gültig ist | boolean | true/false | true |
| validityFlags | Gibt optional weitere Informationen über den Gültigkeitszustand an | Array Typ: 7.2.5 Kontrollereignis | | Siehe Beispiel – JSON |
| errorMessage | Gibt bei ungültigen Tickets einen Grund an | String | | "Ticket is locked" |

| | | | | |
|---|---|---|---|---|
| lastUpdate | Gibt den Zeitpunkt an, an dem die Informationen über das Ticket zuletzt aktualisiert wurden | string | ISO 8601 Date | "2025-02-12T13:00:00.065319+00:00" |
| lastValidation | Gibt den Zeitpunkt an, wann das Ticket zuletzt kontrolliert wurde | string | ISO 8601 Date | "2025-02-12T13:00:00.065319+00:00" |

**Beispiel – JSON**


```
{
  "isValid":true,
  "validityFlags":[
    {
      "id":1,
      "description":"Gegebenenfalls die Personalien prüfen",
      "details":"Das Ticket wurde mehr als 1 Mal kontrolliert in den letzten 5 Minuten",
      "data":{
        "interval":5,
        "validations":1
      }
    }
  ],
  "errorMessage":null,
  "lastUpdate":"2025-06-19T08:17:13.522299+00:00",
  "lastValidation":"2025-06-23T09:01:21.3458736+00:00"
}
```


### 7.2.7 Ticket

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| ticketId | Eindeutige Kennung des erzeugten Tickets | string | | "A0815BF0" |
| issuerRics | RICS-Code des Ticketausgebers | string | | "5143" |
| securityProviderRics | RICS-Code des Security-Providers | string | | "3634" |
| keyId | KeyId, mit welcher das Ticket signiert wurde | string | | "31A33" |
| ticketData | Ticket enkodiert als Hexadezimal String | string | Hexadecimal | "4F6B6579023A7DF…" |

| | | | | |
|---|---|---|---|---|
| firstName | Vorname des Ticketinhabers | string | | "Maxima" |
| lastName | Nachname des Ticketinhabers | string | | "Musterfrau" |
| dateOfBirth | Geburtsdatum des Ticketinhabers | string | ISO 8601 Date | "1990-05-30" |
| gender | Geschlecht des Ticketinhabers | int | 0 = Unspecified<br>1 = Female<br>2 = Male<br>3 = Other | 1 |
| productId | ID des Produkts | int | | 9999 |
| tariffDescription | Ticket-Name / -Beschreibung aus Tarif | string | | "Deutschlandticket" |
| priceInCents | Ticket-Preis in Cent | int | | 4900 |
| validFrom | Ticket gültig von | string | ISO 8601 Date | "2025-02-01T00:00:00+01:00" |
| validTo | Ticket gültig bis | string | ISO 8601 Date | "2025-03-01T03:00:00+01:00" |
| issuedAt | Ticket ausgestellt am | string | ISO 8601 Date | "2025-01-25T02:00:00+01:00" |
| issuerAuthorizationId | VDV KA Authorisations-ID des Tickets (Berechtigungsnummer) | int | | 15312312 |
| issuerKvpId | VDV KA Kundenvertragspartner-ID (KVP-ID) | int | | 6321 |
| uniqueOrderId | Eindeutige Kennung aus dem (Fremd-)System, bei Aufruf mitgegeben | string | | "28d07a4f-02ee-4c6f-996d-89e47670157b" |

**Beispiel – JSON**


```
{
  "ticketId":"A0815BF0",
  "issuerRics":"5143",
  "securityProviderRics":"3634",
  "keyId":"31A33",
  "ticketData":"4F6B6579023A7DF…",
  "firstName":"Maxima",
```

```
  "lastName":"Musterfrau",
  "dateOfBirth":"1990-05-30",
  "gender":1,
  "productId":9999,
  "tariffDescription":"Deutschlandticket",
  "priceInCents":4900,
  "validFrom":"2025-02-01T00:00:00+01:00",
  "validTo":"2025-03-01T03:00:00+01:00",
  "issuedAt":"2025-01-25T02:00:00+01:00",
  "issuerAuthorizationId":15312312,
  "issuerKvpId":6321,
  "uniqueOrderId":"28d07a4f-02ee-4c6f-996d-89e47670157b"
}
```

### 7.2.8 Ticket-Barcode

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| fileContents | Bild (PNG) als Base64 String | string | Base64 | "U29mdHdhcmUg…" |
| contentType | Inhaltstyp | string | | "image/png" |
| fileDownloadName | Dateiname für Download | string | | "A0815BF0.png" |
| lastModified | Zuletzt geändert am | ISO 8601 Date | | "2025-01-25T02:33:43+01:00" |
| entityTag | ETag (Entity Tag) | object | | Siehe Beispiel – JSON |
| enableRangeProcessing | Gibt an, ob Range Processing erlaubt ist | boolean | true/false | true |

**Beispiel – JSON**


```
{
  "fileContents":"U29mdHdhcmUg…",
  "contentType":"image/png",
  "fileDownloadName":"A0815BF0.png",
  "lastModified":"2025-01-25T02:33:43+01:00",
  "entityTag":{
    "tag":{
      "buffer":"dW5pcXVlVGFn",
      "offset":0,
      "length":10,
      "value":"uniqueTag",
      "hasValue":true
    },
    "isWeak":false
  },
  "enableRangeProcessing":true
}
```

# 8 API-Endpunkte

## 8.1 Authentifizierung (OAuth2)

### 8.1.1 POST: /api/v1/auth/token

**Beschreibung**

OAuth2-Endpunkt zur Anforderung eines Tokens über Client-Credentials oder mit einem Refresh-Token. Um Client-Credentials zu benutzen muss zunächst über das Portal der Login temporär freigeschaltet werden.

| | |
|---|---|
| **Authentifizierung** | via Request-Parameter |
| **Notwendige Berechtigungen** | Keine, da ausschließlich für OAuth2-Flow |

**Request Parameter**

**Type:** application/x-www-form-urlencoded

| Name | Art | Beschreibung | Datentyp | Pflichtfeld | Beispiel |
|---|---|---|---|---|---|
| client_id | Body | ID des Clients | string | Ja | "198483d7-d6f6-444e-a75e-edb7d082e24d" |
| client_secret | Body | Geheimnis des Clients | string | Ja | "ef7f4dbb-39e0-427f-879f-9041a0465323" |
| grant_type | Body | Art des Grants | string | Ja | "client_credentials" oder "refresh_token" |
| refresh_token | Body | Refresh-Token, wenn ein neues Token angefordert wird | string | Nein | "eyJhbGciOiJIUzI1... " |
| scope | Body | Umfang der Berechtigungen | string | Nein | |

**HTTP-Antworten**

**200 (OK):**

{ "access_token": "eyJhbGciOiJIUzI1...", "token_type": "bearer", "expires_in": 3600, "refresh_token": "eyJhbGciOiJIUzI1..." }

**400 (Bad Request):**

Die Anfrage konnte nicht verarbeitet werden. Es könnten bspw. falsche / invalide Parameter angegeben sein. Fehlerantworten werden im OAuth2-Standard zurückgegeben.

**401 (Unauthorized):**

Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte die falsche Client-ID und/oder das falsche Client-Secret oder ein ungültiger Refresh-Token angegeben sein. Fehlerantworten werden im OAuth2-Standard zurückgegeben.

**403 (Forbidden):**

Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.

**429 (Too Many Requests):**

Das Anfrage-Limit wurde überschritten.

**Beispiel-Anfrage**

```
curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/auth/token \ --request POST \ --header 'Content-Type: application/x-www-form-urlencoded' \ --data 'client_id=your_client_id&client_secret=your_client_secret&grant_type=client_credentials'
```

# 8.2 Blacklist (Sperrliste)

## 8.2.1 GET: /api/v1/blacklist

| **Beschreibung** | |
|---|---|
| Gibt eine Liste aller verfügbaren Sperrlisten zurück (üblicherweise die letzten zwei Wochen). | |
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Sperrliste herunterladen |
| **Request Parameter** | |
| Keine | |
| **HTTP-Antworten** | |
| **200 (OK):**<br><br>7.2.2 Sperrlistenübersicht<br><br>**400 (Bad Request):**<br><br>Die Anfrage konnte nicht verarbeitet werden.<br><br>**401 (Unauthorized):**<br><br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein falscher API-Key oder ein ungültiger Auth-/ Refresh-Token angegeben sein.<br><br>**403 (Forbidden):**<br><br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br><br>**429 (Too Many Requests):**<br><br>Das Anfrage-Limit wurde überschritten. | |
| **Beispiel-Anfrage** | |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/blacklist \ --header 'Authorization: 190c193ec2ed45d7857c854fa0378039f66d6f3ff62f4c7c84670cd6ac26be1d' | |

## 8.2.2 GET: /api/v1/blacklist/{id}

**Beschreibung**

Gibt den Inhalt einer bestimmten Sperrliste zurück. Das Format kann über den Query-Parameter „format“ angegeben werden. Mögliche Werte sind „json“ und „csv“.

| | |
|---|---|
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Sperrliste herunterladen |

**Request Parameter**

| Name | Art | Beschreibung | Datentyp | Pflichtfeld | Beispiel |
|---|---|---|---|---|---|
| id | Path | Eindeutige ID der Sperrliste | int | Ja | 1 |
| format | Query | Format der Sperrliste | string | Nein | "json" oder "csv", Default: "json" |

**HTTP-Antworten**

**200 (OK) :**

7.2.4 Sperrliste

**400 (Bad Request):**

Die Anfrage konnte nicht verarbeitet werden. Es könnten bspw. falsche / invalide Parameter angegeben sein.

**401 (Unauthorized):**

Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein falscher API-Key oder ein ungültiger Auth-/ Refresh-Token angegeben sein.

**403 (Forbidden):**

Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.

**429 (Too Many Requests):**

Das Anfrage-Limit wurde überschritten.

**Beispiel-Anfrage**

```
curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/blacklist \ --header 'Authorization: 190c193ec2ed45d7857c854fa0378039f66d6f3ff62f4c7c84670cd6ac26be1d'
```

### 8.2.3 GET: /api/v1/blacklist/latest

**Beschreibung**

Gibt den Inhalt der neuesten Sperrliste zurück. Das Format kann über den Query-Parameter „format“ angegeben werden. Mögliche Werte sind „json“ und „csv“.

Über den zusätzlichen Parameter „lastVersion“ kann die letzte bekannte Sperrliste übertragen werden. Es wird dann nur eine Sperrliste ausgeliefert, wenn sie einen neueren Stand hat.

| | |
|---|---|
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Sperrliste herunterladen |

**Request Parameter**

| Name | Art | Beschreibung | Datentyp | Pflichtfeld | Beispiel |
|---|---|---|---|---|---|
| format | Query | Format der Sperrliste | string | Nein | "json" oder "csv", Default: "json" |
| lastVersion | Query | Bereits heruntergeladene Sperrlisten-Version | int | Nein | 1 |

**HTTP-Antworten**

**200 (OK):**

7.2.4 Sperrliste

**304 (Not Modified):**

Wenn der optionale Parameter „lastVersion“ angegeben wird, liefert der Server nur dann eine neue Liste, wenn eine neuere Version verfügbar ist. Andernfalls gibt der Server den Status-Code „304 - Not Modified“ zurück.

**400 (Bad Request):**

Die Anfrage konnte nicht verarbeitet werden. Es könnten bspw. falsche / invalide Parameter angegeben sein.

**401 (Unauthorized):**

Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein falscher API-Key oder ein ungültiger Auth-/ Refresh-Token angegeben sein.

**403 (Forbidden):**

Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.

**429 (Too Many Requests):**

| Das Anfrage-Limit wurde überschritten. |
| --- |
| **Beispiel-Anfrage** |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/blacklist/latest?format=json \ --header 'Authorization: 190c193ec2ed45d7857c854fa0378039f66d6f3ff62f4c7c84670cd6ac26be1d' |

# 8.3 Ticket (-Interaktionen)

## 8.3.1 POST: /api/v1/ticket/lock

| **Beschreibung** | |
|---|---|
| Startet eine Anfrage zum Sperren einer Sammlung von Tickets. Die Tickets werden kurz darauf zur Sperrliste hinzugefügt. Um den Prozess rückgängig zu machen, verwenden Sie den Unlock Ticket-Endpunkt. | |
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Ticket sperren<br><br>Es können nur Tickets mit RICS-Codes gesperrt werden, für die der API-Schlüssel autorisiert ist. Es können maximal 10.000 Tickets pro Anfrage gesendet werden. |
| **Request Body (Content)**<br><br>**Type:** application/json | |
| 7.1.2 Tickets | |
| **HTTP-Antworten** | |
| **202 (Accepted):**<br><br>Die Anfrage und deren Inhalte wurden erfolgreich verarbeitet.<br><br>**400 (Bad Request):**<br><br>Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br><br>**401 (Unauthorized):**<br><br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein falscher API-Key oder ein ungültiger Auth-/ Refresh-Token angegeben sein.<br><br>**403 (Forbidden):**<br><br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br><br>**429 (Too Many Requests):**<br><br>Das Anfrage-Limit wurde überschritten. | |

| **Beispiel-Anfrage** |
| --- |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/lock \ --request POST \ --header 'Authorization: 190c193ec2ed45d7857c854fa0378039f66d6f3ff62f4c7c84670cd6ac26be1d' \ --header 'Content-Type: application/json' \ --data '{ "tickets": [ { "rics": "9999", "ticketId": "A0815BF0", "validTo": "2025-03-01T03:00:00+01:00" }, { "rics": "9999", "ticketId": "BC93DE99", "validTo": "2025-03-01T03:00:00+01:00" } ] }' |

## 8.3.2 POST: /api/v1/ticket/unlock

| **Beschreibung** | |
| --- | --- |
| Startet eine Anfrage zum Entsperren einer Sammlung von Tickets. Die Tickets werden kurz darauf aus der Sperrliste entfernt. | |
| **Authentifizierung** | OAuth2 |
| **Notwendige Berechtigungen** | Ticket entsperren<br><br>Es können nur Tickets mit RICS-Codes entsperrt werden, für die der OAuth2-Client autorisiert ist. Es können maximal 10.000 Tickets pro Anfrage gesendet werden. |
| **Request Body (Content)**<br><br>**Type:** application/json | |
| 7.1.2 Tickets | |
| **HTTP-Antworten** | |
| **202 (Accepted):**<br><br>Die Anfrage und deren Inhalte wurden erfolgreich verarbeitet.<br><br>**400 (Bad Request):**<br><br>Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br><br>**401 (Unauthorized):**<br><br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.<br><br>**403 (Forbidden):** | |

| |
|---|
| Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br><br>**429 (Too Many Requests):**<br><br>Das Anfrage-Limit wurde überschritten. |
| **Beispiel-Anfrage** |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/unlock \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "tickets": [ { "rics": "9999", "ticketId": "A0815BF0", "validTo": "2025-03-01T03:00:00+01:00" }, { "rics": "9999", "ticketId": "BC93DE99", "validTo": "2025-03-01T03:00:00+01:00" } ] }' |

## 8.3.3 POST: /api/v1/ticket/cancel

| | |
|---|---|
| **Beschreibung** | |
| Mit diesem Endpunkt kann eine Ticketstornierung angefordert werden. Die Stornierung kann nicht rückgängig gemacht werden. Die übergebenen Tickets werden gesperrt und auf die Sperrliste geschrieben.<br><br>Stornierte Tickets werden nicht nur gesperrt, sondern auch ihre Ausgabe zurückgenommen. | |
| **Authentifizierung** | OAuth2 |
| **Notwendige Berechtigungen** | Ticketausstellung stornieren<br><br>Es können nur Tickets mit RICS-Codes storniert werden, für die der OAuth2-Client autorisiert ist. Es können maximal 10.000 Tickets pro Anfrage gesendet werden. |
| **Request Body (Content)**<br><br>**Type:** application/json | |
| 7.1.2 Tickets | |
| **HTTP-Antworten** | |
| **202 (Accepted):**<br><br>Die Anfrage und deren Inhalte wurden erfolgreich verarbeitet.<br><br>**400 (Bad Request):** | |

| |
|---|
| Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br><br>**401 (Unauthorized):**<br><br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.<br><br>**403 (Forbidden):**<br><br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br><br>**429 (Too Many Requests):**<br><br>Das Anfrage-Limit wurde überschritten. |
| **Beispiel-Anfrage** |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/cancel \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "tickets": [ { "rics": "9999", "ticketId": "A0815BF0", "validTo": "2025-03-01T03:00:00+01:00" }, { "rics": "9999", "ticketId": "BC93DE99", "validTo": "2025-03-01T03:00:00+01:00" } ] }' |

### 8.3.4 POST: /api/v1/ticket/issuancerecord

| | |
|---|---|
| **Beschreibung** | |
| Mit Aufruf dieses Endpunkts werden ein oder mehrere Ausgabenachweise im Sicherheitsportal erfasst. | |
| **Authentifizierung** | OAuth2 |
| **Notwendige Berechtigungen** | Ausgabenachweis veröffentlichen<br><br>Ausgabenachweise können nur für Tickets mit RICS-Codes veröffentlicht werden, für die der OAuth2-Client autorisiert ist. Es können maximal 10.000 Tickets pro Anfrage gesendet werden. |
| **Request Body (Content)**<br><br>**Type:** application/json | |
| 7.1.6 Ausgabenachweise | |
| **HTTP-Antworten** | |

**202 (Accepted):**

Die Anfrage und deren Inhalte wurden erfolgreich verarbeitet.

**400 (Bad Request):**

Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.

**401 (Unauthorized):**

Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.

**403 (Forbidden):**

Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.

**429 (Too Many Requests):**

Das Anfrage-Limit wurde überschritten.

**Beispiel-Anfrage**

curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/issuancerecord \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "issuanceRecords": [ { "rics": "9999", "ticketId": "A0815BF0", "validTo": "2025-03-01T03:00:00+01:00", "productId": 1, "tariffDescription": "Deutschlandticket", "validFrom": "2025-02-01T00:00:00+01:00", "issuedAt": "2025-01-25T02:00:00+01:00", "securityProviderRics": "3333", "keyId": "31A33", "postcode": "49661", "issuerAuthorizationId": 15312312, "issuerKvpId": 6321, "uniqueOrderId": "6e25585b-2e7e-4ea1-99c3-a5f743693f21" } ] }'

## 8.3.5 POST: /api/v1/ticket/issuance/dticket

**Beschreibung**

Mit Aufruf dieses Endpunkts wird ein Deutschlandticket ausgestellt.

| | |
|---|---|
| **Authentifizierung** | OAuth2 |
| **Notwendige Berechtigungen** | Deutschlandticket ausstellen |

**Request Body (Content)**

**Type:** application/json

7.1.7 Deutschlandticket

| **HTTP-Antworten** |
|---|
| **200 (OK):**<br><br>7.2.7 Ticket<br><br>**400 (Bad Request):**<br><br>Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br><br>**401 (Unauthorized):**<br><br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.<br><br>**403 (Forbidden):**<br><br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br><br>**429 (Too Many Requests):**<br><br>Das Anfrage-Limit wurde überschritten. |
| **Beispiel-Anfrage** |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/issuance/dticket \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "firstName": "Maxima", "lastName": "Musterfrau", "dateOfBirth": "1990-05-30", "gender": 1, "productId": 9999, "priceInCents": 4900, "monthOfValidity": "2025-06-01", "postcode": "49661", "uniqueOrderId": "28d07a4f-02ee-4c6f-996d-89e47670157b" }' |

## 8.3.6 GET: /api/v1/ticket/issuance/renderbarcode

| **Beschreibung** | | | | | |
|---|---|---|---|---|---|
| Nimmt einen Hexadezimal String als Abfrageparameter entgegen, um ein PNG des bereitgestellten Tickets zu rendern. Zu Testzwecken. | | | | | |
| **Authentifizierung** | Keine | | | | |
| **Notwendige Berechtigungen** | Keine | | | | |
| **Request Parameter** | | | | | |
| **Name** | **Art** | **Beschreibung** | **Datentyp** | **Pflichtfeld** | **Beispiel** |

| | | | | | |
|---|---|---|---|---|---|
| ticketHex | Query | Ticket als Hexadezimal String | string | Ja | "4F6B6579023A7DF…" |
| width | Query | Breite des PNG | int | Nein | 320,<br>Default: 256 |
| height | Query | Höhe des PNG | int | Nein | 320,<br>Default: 256 |

**HTTP-Antworten**

**200 (OK):**

7.2.8 Ticket-Barcode

**400 (Bad Request):**

Die Anfrage konnte nicht verarbeitet werden. Es könnten bspw. falsche / invalide Parameter angegeben sein.

**429 (Too Many Requests):**

Das Anfrage-Limit wurde überschritten.

**Beispiel-Anfrage**

curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/ticket/issuance/renderbarcode?ticketHex=4F6B6579023A7DF…

# 8.4 Validation (Kontrolle)

## 8.4.1 POST: /api/v1/validation/validate

| **Beschreibung** | |
|---|---|
| Mit Aufruf dieses Endpunkts kann geprüft werden, ob das entsprechende Ticket gesperrt ist und infolgedessen auf der Sperrliste steht. | |
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Ticket kontrollieren |
| **Request Body (Content)**<br>**Type:** application/json | |
| 7.1.3 Kontrolle | |
| **Beispiel – CURL** | |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/validation/validate \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "rics": "5143", "ticketId": "A0815BF0", "validTo": "2025-03-01T03:00:00+01:00", "productId": 9999, "tariffDescription": "Deutschlandticket", "validFrom": "2025-02-01T00:00:00+01:00", "issuedAt": "2025-01-25T02:00:00+01:00", "securityProviderRics": "3634", "keyId": "31A33", "validatedAt": "2025-02-15T10:30:00+01:00" }' | |
| **HTTP-Antworten** | |
| **200 (OK):**<br>7.2.6 Kontrollergebnis<br>**400 (Bad Request):**<br>Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br>**401 (Unauthorized):**<br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.<br>**403 (Forbidden):**<br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br>**429 (Too Many Requests):** | |

| |
|---|
| Das Anfrage-Limit wurde überschritten. |

## 8.4.2 POST: /api/v1/validation/records

| **Beschreibung** | |
|---|---|
| Mit Aufruf dieses Endpunkts werden ein oder mehrere Kontrollnachweise im Sicherheitsportal erfasst. | |
| **Authentifizierung** | API-Key / OAuth2 |
| **Notwendige Berechtigungen** | Kontrollnachweis veröffentlichen |
| **Request Body (Content)**<br>**Type:** application/json | |
| 7.1.4 Kontrollnachweise | |
| **HTTP-Antworten** | |
| **202 (Accepted):**<br>Die Anfrage und deren Inhalte wurden erfolgreich verarbeitet.<br>**400 (Bad Request):**<br>Die Anfrage konnte nicht verarbeitet werden. Es könnte bspw. ein falscher / invalider Inhalt (Body) vorhanden sein.<br>**401 (Unauthorized):**<br>Die Authentifizierung ist fehlgeschlagen / nicht möglich. In diesem Fall könnte ein ungültiger Auth-/ Refresh-Token angegeben sein.<br>**403 (Forbidden):**<br>Der anfragende Client ist erfolgreich authentifiziert, verfügt jedoch nicht über die notwendigen Berechtigungen, um den Endpunkt aufrufen zu dürfen.<br>**429 (Too Many Requests):**<br>Das Anfrage-Limit wurde überschritten. | |
| **Beispiel-Anfrage** | |
| curl https://test.sicherheitsportal.deutschlandtarifverbund.de/api/v1/validation/records \ --request POST \ --header 'Authorization: Bearer eyJhbGciOiJIUzI1...' \ --header 'Content-Type: application/json' \ --data '{ "validationRecords": [ { "rics": "9999", "ticketId": "A0815BF0", "validTo": "2025-03- | |

01T03:00:00+01:00", "productId": 1, "tariffDescription": "Deutschlandticket", "validFrom": "2025-02-01T00:00:00+01:00", "issuedAt": "2025-01-25T02:00:00+01:00", "securityProviderRics": "3333", "keyId": "31A33", "validatedAt": "2025-02-15T10:30:00+01:00" } ] }'

# 9 Fehlerbehandlung

Im Fehlerfall wird einer der folgenden Fehlercodes an den aufrufenden Client zurückgemeldet.

| Fehlernummer | Fehlername | Beschreibung |
|---|---|---|
| -1 | Unknown | Unbekannter Fehler ist aufgetreten |
| 0 | Ok | Alles in Ordnung |
| 1 | Unauthenticated | Fehlerhafte Authentifizierung |
| 2 | Unauthorizied | Keine Berechtigung für die Aktion |
| 3 | InvalidRicsUsed | Es wurde ein Rics Code verwendet, für den keine Berechtigung vorhanden ist |
| 4 | SchemaValidationFailed | Ein fehlerhaftes Schema wurde verwendet |
| 5 | ObjectNotFound | Ein Objekt, auf das verwiesen wurde, existiert nicht |
| 6 | TicketIssuanceValidtiyOutOfRange | Es wurde versucht ein Ticket mit einem Gültigkeitsmonat auszugeben, welche nicht in der erlaubten Range ist. |
| 7 | TicketIssuanceNoSigningKeyConfigured | Es wurde kein Signierschlüssel zur Ausgabe des Tickets konfiguriert |
| 8 | BlacklistNotFound | Die angegeben Blacklist existiert nicht |
| 9 | RateLimitExceeded | Es wurden zu viele Anfragen gesendet |

In jedem Fehlerfall wird ein einheitliches Ergebnisobjekt zurückgeliefert. Dieses hat folgendes Schema:

| Feld | Beschreibung | Datentyp | Pflichtfeld | Wertebereich | Beispiel |
|---|---|---|---|---|---|
| errorCode | Fehlernummer. Siehe Tabelle | int | - | | 1 |
| errorCodeDescription | Fehlername | string | - | | "Unauthenticated" |
| errorMessage | Details zum aufgetretenen Fehler | string | - | | "Invalid JWT Token" |

**Beispiel – JSON**

```
{
  "errorCode": 1,
  "errorCodeDescription": "Unauthenticated",
  "errorMessage": "Invalid JWT Token"
}
```

Für den Fehlerfall 4 (SchemaValidationFailed) wird ein weiteres Feld im Ergebnis mitgeliefert. Dies gibt an welches Feld im Schema die Validierung nicht erfolgreich überstanden hat.

| Feld | Beschreibung | Datentyp | Wertebereich | Beispiel |
|---|---|---|---|---|
| validationErrors | Auflistung der fehlerhaften Felder im Schema | object | | "FirstName": "The First-Name field is required." |

**Beispiel – JSON**


```
{
   "validationErrors": {
      "FirstName": "The FirstName field is required."
   },
   "errorCode": 4,
   "errorCodeDescription": "SchemaValidationFailed",
   "errorMessage": "Error while parsing the request"
}
```

# 10 Ergänzungen / Anmerkungen

## 10.1 Batch-Anfragen

Sowohl das Sperren als auch das Entsperren und das Stornieren von Tickets kann im Batch durchgeführt werden. Es können immer bis zu 10.000 Ticket-Einträge mit einer Anfrage übermittelt werden.

Wichtig ist, dass das System nur die Syntax der Anfrage überprüft und dann den Erhalt quittiert. Die vollständige Verarbeitung der Anfrage kann einige Sekunden bis Minuten dauern. Dies bedeutet beispielsweise:

- Wenn eine Entsperrung für ein Ticket beauftragt wird, das nicht gesperrt ist, wird die Anfrage ignoriert.
- Eine Anfrage zum Sperren eines bereits gesperrten Tickets wird ebenfalls ignoriert.

Es ist daher kein Problem, wenn eine Sperranforderung erneut gesendet wird, z.B. wenn das Ausgabesystem während des Sperrvorgangs in einem ungünstigen Moment abstürzt.

## 10.2 Offline-Sperrliste

Auch für den Download der Offline-Sperrliste wird ein API-Schlüssel benötigt. Dies verhindert, dass die Sperrliste öffentlich bekannt wird und von unseriösen Anbietern genutzt wird.

Die Offline-Sperrliste wird periodisch generiert. Eine neue Liste wird nur erzeugt, wenn sich der Inhalt der Liste ändern würde. Im Produktivsystem ist das Intervall auf 60 Minuten gestellt. Im Testsystem beträgt das Intervall 5 Minuten.

Jede Sperrliste hat eine eindeutige und fortlaufende ID. Beim Anfordern der aktuellen Sperrliste kann ein System die letzte bekannte Sperrlisten-ID angeben. Die API gibt dann nur die Sperrliste zurück, wenn es eine neuere Sperrliste gibt. Andernfalls wird der Status-Code „304 - Not Modified“ zurückgegeben.

Die Sperrliste ist in den Formaten CSV und JSON verfügbar. Für die Verwendung in Vertriebs- und Kontrollsystemen wird das JSON-Format empfohlen, da es auch die Sperrlisten-Nummer enthält. Bei der CSV-Datei ist diese Nummer ausschließlich im Dateinamen enthalten.

## 10.3 Sonstiges

Alle Anfragen an das System werden von diesem protokolliert.

Um später zu Debugging-Zwecken Geräte/Systeme eindeutig identifizieren zu können, kann optional der HTTP-Header „Device-Id“ befüllt werden. Das System wertet dies nicht automatisch aus, jedoch können für einige Tage die Anfragen über die HTTP-Accesslogs nachvollzogen werden. Dieser Header ist explizit kein Pflichtfeld.