# API-Guideline 1.0

| | |
|---|---|
| Version: | 1.0 |
| Publikationsdatum: | 19.06.2024 |
| Autor: | BDEW |

## Disclaimer

Die zusätzlich veröffentlichte Word-Datei dient als informatorische Lesefassung und entspricht inhaltlich der PDF-Datei. Die PDF-Datei ist das gültige Dokument. Diese Word-Datei wird bis auf Weiteres rein informatorisch und ergänzend veröffentlicht. Der BDEW behält sich vor, in Zukunft eine kostenpflichtige Veröffentlichung der Word-Datei einzuführen.

# Inhaltsverzeichnis

# 1 Einleitung

Dieses Dokument beschreibt die Regelungen zur Nutzung und Erstellung von API-Webdiensten für regulierte Prozesse in der Energiewirtschaft. Gemäß den Festlegungen der Bundesnetzagentur zu den Universalbestellprozessen (BK6-22-128) und dem 24h Lieferantenwechsel (BK6-22-024) sind einige Prozesse über API-Webdienste zu realisieren. Die EDI@Energy API Guidelines für Web-API Schnittstellen stellen Design-Prinzipien dar, welche das primäre Ziel verfolgen, eine bestmögliche Erfahrung mit dem Umgang webbasierter APIs zu garantieren. Durch die Anwendung dieser Guidelines soll eine konsistente und intuitive API-Landschaft entstehen, die für Anbieter und Anwender von Web-API gleichermaßen einfach zu nutzen ist.

Dieses Dokument benennt nicht die ggf. existierenden rechtlichen Folgen, wenn aufgrund eines abweichenden Vorgehens kein gesicherter elektronischer Datenaustausch stattfinden kann.

# 2 Terminologie

## 2.1 Schlüsselwörter in der API-Guideline

Die Schlüsselwörter „MÜSSEN“ (Englisch „MUST“), „DÜRFEN NICHT“ (Englisch „MUST NOT“), „ERFORDERLICH“ (Englisch „REQUIRED“), „SOLL“ (Englisch „SHALL“), „SOLL NICHT“ (Englisch „SHALL NOT“), „SOLLTE“ (Englisch „SHOULD“), „SOLLTE NICHT“ (Englisch „SHOULD NOT“), „EMPFOHLEN“ (Englisch „RECOMMENDED“), „DÜRFEN“ (Englisch „MAY“), and „FREIWILLIG“ (Englisch „OPTIONAL“) in diesem Dokument sind zu interpretieren gemäß [RFC2119]. Dabei spielt die Groß- und Kleinschreibung keine Rolle.

## 2.2 Glossar

| Begriff | Beschreibung |
|---|---|
| API-Anbieter | Bietet einen Webservice an, über den die beschriebene API genutzt werden kann. |
| API-Nutzer | Nutzt als Client mittels eines angebotenen Webservice die beschrieben API. |
| Kommunikationsendpunkt | URL und Port (URI) des API-Webservice |

# 3 Fachliche Vorgaben der API-Guideline

## 3.1 Konsistenz

Im Folgenden wird beschrieben, wie eine grundlegende Konsistenz der Web-APIs durch Vereinheitlichung von URLs und unterstützte Methoden erreicht wird.

### 3.1.1 URL

URLs bilden die Grundlage von API-Webdiensten. Diese definieren den Kommunikationsendpunkt des API-Webdienstes.

MUST Eine URL darf keine Umlaute enthalten.

### 3.1.2 Struktur

SHOULD Nutzer können URLs einfach lesen und konstruieren

› Beispiel einer gut strukturierten URL ist:

*https://xyz.ztr.de/edienergy/marktlokationen/identifikation/v1*

› Beispiel einer lesbaren und nicht strukturierten URL ist:

*https://xyz.ztr.de/33/55/zdfgd/rkfnhdrfeufuiefevcuberfiu5frf54/v1*

### 3.1.3 Regeln zum Aufbau

MUST Folgende Regeln sind beim Aufbau einer URL einzuhalten:

› URLs werden ohne abschließenden Schrägstrich gebildet.

› Die Pfad Komponente einer URL besteht ausschließlich aus Buchstaben, Zahlen, Unter- und Bindestriche sowie dem Schrägstrich als Segment-Trenner.

- Ausnahme: Für die Versionsangabe darf der Punkt zwischen zwei Zahlen verwendet werden.

› Insbesondere werden keine weiteren Sonderzeichen verwendet.

› Zur besseren Lesbarkeit sollen Objekte / Ressourcen in CamelCase Schreibweise benannt werden.

› Die Namen oder Bezeichner im URL-Pfad werden im Plural angeben.

### 3.1.4 Länge

Die Länge einer URL ist im http 1.1 Nachrichtenformat nicht beschränkt (siehe RFC 7230, Section 3.1.1):

## 3.2 Versionierung

MUST Die Versionierung der Web-APIs Schnittstellen folgt der Semantik von Semantic Versioning 2.0:

*<MAJOR>.<MINOR>.<PATCH>*

- Die <MAJOR>-Version wird erhöht, wenn das API eine inkompatible Änderung beinhaltet.
- Die <Minor>-Version wird erhöht, wenn neue Funktionalitäten, die kompatibel zur bisherigen API sind, veröffentlicht werden.
- Die <Patch>-Version wird erhöht, wenn die Änderungen ausschließlich API-kompatible Bugfixes umfassen.

MUST Zur Differenzierung inkompatibler Schnittstellenversionen enthalten alle URL den MAJOR-Anteil mit einem kleiner „v“ als Präfix in der URL, wie im folgenden Beispiel gezeigt.

*Beispiel:*

*https://xyz.ztr.de/edienergy/marktlokationen/identifikation/v1*

MUST Die Antwort muss im http-Header X-BDEW-VERSION die vollqualifizierte Versionsnummer (<MAJOR>.<MINOR>.<PATCH>, bspw. 3.1.0) beinhalten.

### 3.2.1 Änderungsmanagement

Das Änderungsmanagement der EDI@Energy API-Webdienste erfolgt bis zu zweimal im Jahr, nach einem zeitlich festgelegten Ablauf (vgl. Kapitel 2.5 der Allgemeine Festlegungen). Die Veröffentlichung der zur Konsultation gestellten Dokumente erfolgt durch eine gemeinsame Mitteilung zu Datenformaten der Beschlusskammern 6 und 7 der BNetzA. In der Mitteilung wird erläutert, wie sich die Marktteilnehmer an der Konsultation beteiligen können. Diese Änderungen können zu Inkompatiblen Schnittstellenversionen und damit zu neuen Major-Version führen.

Die Veröffentlichung der Konsultationsdokumente erfolgt ebenso durch eine gemeinsame Mitteilung zu Datenformaten der Beschlusskammern 6 und 7 der BNetzA. In der jeweiligen Mitteilung wird der verbindliche Umsetzungszeitpunkt für die Änderungen genannt.

Im Rahmen von Fehlerbehebungen an den API-Webdiensten werden nur kompatible Änderungen eingeführt. Nicht zulässig ist beispielsweise das Einführen eines neuen Pflichtfelds.

### 3.2.2 Abwärtskompatibilität

Eine Änderung an einem API-Webdienst ist dann kompatibel, wenn der API-Webdienst durch den Anbieter so geändert wird, dass diese auch noch von älteren API-Nutzern verwendet werden kann. Ein Beispiel wäre das Hinzufügen eines optionalen Parameters. Um kompatibel zu bleiben, muss in diesem Fall der API-Anbieter der Schnittstelle damit zurechtkommen, dass (ältere) API-Nutzer den optionalen Parameter nicht übergeben.

MUST Bei der Behebung von Fehlern in API-Webdiensten wird immer die Abwärtskompatibilität zu der entsprechenden Major-Version sichergestellt. Im Umstellungszeitraum müssen alle API-

Nutzer auf die neue Version des API-Webdienstes umstellen. Der Umstellungszeitraum wird in der Spezifikation des API-Webdienstes beschrieben und beträgt mindestens 3 Monate ab dem Zeitpunkt der Veröffentlichung der neuen Version. Die Veröffentlichung einer neuen Version erfolgt über das Forum Datenformate.

MUST Inkompatible Änderungen an einem API-Webdienst werden im Rahmen des Änderungsmanagements angekündigt. In der Beschreibung des API-Webdienstes wird der Anwendungszeitpunkt der neuen Version angegeben. Ab dem angegeben Zeitpunkt sind alle älteren Versionen nicht mehr nutzbar und dürfen vom API-Anbieter entfernt werden.

MUST Es sind immer alle nicht abgekündigten Versionen nutzbar. Eine Abkündigung erfolgt durch die Kennzeichnung des API-Webdienstes, diese sieht wie folgt aus:

› *Deprecated ab dem dd.mm.yyyy. 00:00 Uhr*

### 3.2.3 Aufwärtskompatibilität wird nicht unterstützt

Es wird unterstützt, dass ein Aufrufer einen optionalen Parameter gemäß einer neueren Schnittstellen Spezifikation an einen Anbieter übergibt, der noch nach einer älteren Spezifikation arbeitet, die diesen Parameter noch nicht enthält.

## 3.3 Datentypen und JSON Standardisierung

MUST Grundlegend müssen sich primitive Daten gemäß [RFC 8259] in das JSON-Format serialisieren lassen.

MUST OpenAPI (basierend auf dem JSON Schema Validation Vokabular) definiert Formate ausgehend von den ISO- und IETF-Standards für Date/Time, Integers/Numbers und Binärdaten. Diese sind ausschließlich zu verwenden!

MUST Die Nutzung von Umlauten in Bezeichnern ist nicht erlaubt.

| OpenAPI Typ | OpenAPI Format | Spezifikation | Beispiel |
|---|---|---|---|
| **integer** | int32 | 4 Byte vorzeichenbehaftete Integer-Nummer zwischen $-2^{31}$ and $2^{31}-1$ | 7721071004 |
| **integer** | int64 | 8 Byte vorzeichenbehaftete Integer-Nummer zwischen $-2^{63}$ and $2^{63}-1$ | 772107100456824 |
| **integer** | bigint | Vorzeichenbehaftete Integer-Nummer unbegrenzter Länge | 77210710045682438959 |

| OpenAPI Typ | OpenAPI Format | Spezifikation | Beispiel |
|---|---|---|---|
| **number** | float | binary32 einfach präzise Dezimalnummer – IEEE754-2008/ISO 60559:2011 | 3.1415927 |
| **number** | double | binary64 dopppelt präzise Dezimalnummer – IEEE754-2008/ISO 60559:2011 | 3.141592653589793 |
| **number** | decimal | Vorzeichenbehaftete Dezimalnummer unbegrenzter Länge | 3.14159265358979323846264338327 9 |
| **string** | byte | base64url kodiertes Byte nach RFC 7493 Sektion 4.4 | “VA==” |
| **string** | binary | base64url kodierte Byte-Sequenz nach RFC 7493 Sektion 4.4 | “VGVzdA==” |
| **string** | date | RFC 3339 Internet Profil – Subset von ISO 8601 | “2019-07-30” |
| **string** | date-time | RFC 3339 Internet Profil – Subset von ISO 8601 | “2019-07-30T06:43:40.252Z” |
| **string** | time | RFC 3339 Internet Profil – Subset von ISO 8601 | “06:43:40.252Z” |
| **string** | duration | RFC 3339 Internet Profil – Subset von ISO 8601 | “P1DT30H4S” |
| **string** | period | RFC 3339 Internet Profil – Subset von ISO 8601 | “2019-07-30T06:43:40.252Z/PT3H” |
| **string** | password | | “secret” |
| **string** | email | RFC 5322 | “example@exemple.de” |
| **string** | idn-email | RFC 6531 | “hello@buecher.example” |
| **string** | hostname | RFC 1034 | “www.test.de” |
| **string** | idn-hostname | RFC 5890 | “buecher.example” |

| OpenAPI Typ | OpenAPI Format | Spezifikation | Beispiel |
|---|---|---|---|
| **string** | ipv4 | RFC 2673 | “104.75.173.179” |
| **string** | ipv6 | RFC 4291 | “2600:1401:2::8a” |
| **string** | uri | RFC 3986 | “https://www.test.de/“ |
| **string** | uri-reference | RFC 3986 | “/clothing/” |
| **string** | uri-template | RFC 6570 | “/users/{id}” |
| **string** | iri | RFC 3987 | “https://buecher.example/“ |
| **string** | iri-reference | RFC 3987 | “/buecher-sport/” |
| **string** | uuid | RFC 4122 | “e2ab873e-b295-11e9-9c02-…” |
| **string** | json-pointer | RFC 6901 | “/items/0/id” |
| **string** | relative-json-pointer | Relative JSON Pointers | “1/id” |
| **string** | regex | Reguläre Ausdrücke wie in ECMA 262 beschrieben | “^[a-z0-9]+$” |

### 3.4 Bestandteile eines jeden API-Webdienstes

MUST Jeder API-Webdienst besitzt die folgenden Schemata.

› transactionId
- OpenAPI Typ: string
- OpenAPI Format: uuid
- Zweck: ID zur eindeutigen Identifikation eines Aufrufs

› creationDateTime
- OpenAPI Typ: string
- OpenAPI Format: date-time

- Zweck: Zeitpunkt an dem der Aufruf erstellt wurde

› initialTransactionId

- OpenAPI Typ: string
- OpenAPI Format: uuid
- Zweck: ID zur eindeutigen Referenzierung der Antwort auf eine Anfrage und der Sicherstellung der Idempotenz

### 3.4.1 Nutzung der transactionId und initialTransactionId

MUST Clients müssen bei jeder Anfrage und bei jedem Retry immer eine neue „transactionId“ und die „creationTime“ vergeben und diese mitsenden. Bei einem Retry muss die „initialTransactionId“ angegeben werden mit der "transactionId“ des initialen Aufrufs, um im Falle von wiederholten Anfragen technisch die gleichen Aufrufe identifizierbar zu machen (Idempotenz). In der Antwort muss im Feld „initialTransactionId“ die „transactionId“ des ursprünglichen Aufrufs bzw. wenn vorhanden, die initialTransactionId zurückgeben werden, um eine eineindeutige Zuordnung zur Anfrage zu ermöglichen.

## 3.5 http-Status-Code

Es wird nach dem Aufruf eine direkte Antwort (Response) auf die Anfrage (Request) gesendet. Die Antwort ist ein http-Status-Code und gibt Auskunft darüber, ob der Aufruf technisch beim Empfänger empfangen werden konnte. Bei einer Antwort mit dem http-Status-Code 202 werden keine Nutzdaten (Payload) zurückgeliefert. Anschließend erfolgt eine asynchrone Rückmeldung auf den Vorgang, sofern gemäß Prozessbeschreibung eine Rückmeldung zu diesem Vorgang vorgesehen ist.

## 3.6 Status Codes (Response)

Jeder Aufruf einer Schnittstelle wird mit einem http-Statuscode (synchrone Response) beantwortet. In den von EDI@Energy erstellten API-Webdiensten kommen die folgenden Standard http-Statuscodes zur Anwendung. Die Erläuterung ist eine Übersetzung der Standardbeschreibungen aus den RFC-Definitionen.

### 3.6.1 positive Responses

| Code | Name | Erläuterung |
|---|---|---|
| **202** | accepted | Die Anfrage wurde technisch erfolgreich verarbeitet |

### 3.6.2 negative Responses

| Code | Name | Erläuterung |
|---|---|---|
| **400** | Bad request | Die Anfrage ist ungültig |
| **401** | Unauthorized | Die Anfrage ist nicht autorisiert |
| **404** | Not found | Die angeforderte Ressource konnte nicht gefunden werden |
| **405** | Method not Allowed | Die Zielressource kann nicht aufgerufen werden, obwohl diese bekannt ist |
| **415** | Unsupported Media Type | Medientyp in der Anfrage ist nicht unterstützt |
| **429** | Too Many Requests | Client hat zu viele Anfragen in einem Zeitfenster gestellt (Ratenlimitierung). Clients sollten pausieren und zu einem späteren Zeitpunkt einen Retry versuchen. |
| **500** | Internal Server Error | Interner Fehler |
| **503** | Service Unavailabe | Server kann temporär wegen Überlastung oder Wartungsarbeiten keine Anfragen bearbeiten. Clients sollten zu einem späteren Zeitpunkt einen Retry versuchen. |
| **504** | Gateway Timeout | Falls die Web-API als Proxy zu dahinterliegenden Systemen dient, kann bei deren Nichtverfügbarkeit dies angezeigt werden. Clients sollten zu einem späteren Zeitpunkt einen Retry versuchen. |

### 3.6.3 Resilienz

MUST API-Anbieter müssen grundsätzlich in der Lage sein, gleichzeitig allen berechtigten Clients (Kommunikationspartnern) einen Verbindungausbau (TCP-Connect) zu ermöglichen.

MUST Clients müssen Retries (Wiederholungen) von Anfragen vornehmen, falls die Web-API nicht erreichbar ist oder Fehler meldet. Dies ist insbesondere der Fall, wenn technische Fehler über Statuscodes gemeldet werden. Clients müssen geeignete Pausen zwischen Retries einlegen.

SHOULD Server sollen über geeignete HTTP-Statuscodes (siehe Tabelle oben) Rückmeldungen bei Überlastungen, Ausfällen und anderen technischen Problemen geben.

SHOULD Clients sollten Anfragen nach einem clientseitigen Timeout abbrechen, wenn in dieser Zeit keine Antwort von der Web-API (dem Server) erhalten wurde. In diesem Falle sollten Clients die Anfrage wiederholen (siehe oben).

## 3.7 Objekte

MUST Die im API-Aufruf genutzten Objekte werden als JSON-Objekte gemäß [RFC8259] im http-Body übermittelt. Jedes JSON-Objekt muss in UTF-8 ohne Byte Order Mark (BOM) geschrieben werden und es MUSS das Format I-JSON gemäß [RFC7493] eingehalten werden. JSON-Objekte sind **nicht** in http-Header und **nicht** in den Query-Parametern erlaubt.

# 4 Quellen

[CP-SM-PKI]: Certificate Policy der Smart Metering PKI.

[RFC2119]: Key words for use in RFCs to Indicate Requirement Levels. IETF RFC 2119. March 1997. https://www.rfc-editor.org/rfc/rfc2119

[RFC5246]: The Transport Layer Security (TLS) Protocol Version 1.2. IETF RFC 5246. August 2008. https://www.rfc-editor.org/rfc/rfc5246

[RFC6066]: Transport Layer Security (TLS) Extensions: Extension Definitions, ITEF RFC 6066. Januar 2011. https://www.rfc-editor.org/rfc/rfc6066

[RFC8446]: The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446. August 2018. https://www.rfc-editor.org/rfc/rfc8446

[RFC8449]: Record Size Limit Extension for TLS IETF RFC 8449. August 2018. https://www.rfc-editor.org/rfc/rfc8449

[RFC8259]: The JavaScript Object Notation (JSON) Data Interchange Format. IETF RFC 8259. December 2017. https://www.rfc-editor.org/rfc/rfc8259

[RFC8785]: JSON Canonicalization Scheme (JCS). IETF RFC 8785. June 2020. https://www.rfc-editor.org/rfc/rfc8785 https://datatracker.ietf.org/doc/html/rfc7493

[RFC9110]: HTTP Semantics. ITEF RFC 9110. June 2022. https://www.rfc-editor.org/rfc/rfc9110

[RFC7493]: The I-JSON Message Format. March 2015 https://www.rfc-editor.org/rfc/rfc7493.txt

[RFC9112]: HTTP/1.1. ITEF RFC 9112. June 2022. https://www.rfc-editor.org/rfc/rfc9112

[TR02102-1]: Technische Richtlinie BSI TR-02102. Kryptographische Verfahren: Empfehlungen und Schlüssellängen. [Teil 1].

[TR02102-2]: Technische Richtlinie BSI TR-02102-2. Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Teil 2: Verwendung von Transport Layer Security (TLS).

[TR03116-3]: Technische Richtlinie BSI TR-03116 Kryptographische Vorgaben für Projekte der Bundesregierung. Teil 3: Intelligente Messsysteme.