Konsultationsfassung

API-Guideline

Version:	1.0
Publikationsdatum:	02.04.2024
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 Web-Service 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/v1.0.0

› Beispiel einer lesbaren und nicht strukturierten URL ist:

https://xyz.ztr.de/33/55/zdfgd/rkfnhdrfeufuiefevcuberfiu5frf54/v1.0.0

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 und Zahlen sowie dem Schrägstrich als Segment-Trenner.

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

› Insbesondere werden keine Unter- oder Bindestriche oder sonstige 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 geschieht über ein spezifisches URL-Segment. Die Versionsnummer ist dreistellig und besitzt ein führendes kleines **v** als Präfix. Sie kommt im URL-Schema direkt nach der Ressource.

Beispiel:

https://xyz.ztr.de/edienergy/marktlokationen/identifikation/v1.0.0

MUST Jede Ressource ist ohne Angabe der Version aufrufbar und führt zur Nutzung der jeweils aktuellen Version.

Beispiel (wenn die Version 1.0.0 die aktuelle ist):

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

› ist identisch zu https:*//xyz.ztr.de/edienergy/marktlokationen/identifikation/v1.0.0*

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

Das Format ist erweitert um die Architekturversion:

<v><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 Die Antwort muss im Header die vollqualifizierte Versionsnummer (bspw. 3.1.0) beinhalten!

MUST Bei Anfrage einer nicht unterstützten Version muss die Antwort eine Liste der unterstützten Versionen mitliefern.

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.

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 Weiterentwicklungen der Fehlerbehebungen an den API-Webdienste werden diese weiterentwickelt. In einigen Fällen bringt dies eine Inkompatibilität mit sich, z.B. kann der „alte“ API-Nutzer die „neue“ Schnittstelle nicht mehr nutzen. Tritt diese Situation ein, wird eine Lösung, die die Schnittstelle verwendet, nicht mehr ordnungsgemäß funktionieren.

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 den nicht abgekündigten Versionen sichergestellt. Im Umstellungszeitraum müssen alle API-Nutzer auf die neue Version des API-Webdienstes umstellen. Der Umstellungszeitraum wird in dem API-Webdienst veröffentlicht und beträgt mindestens 3 Monate ab dem Zeitpunkt der Veröffentlichung der neuen Version.

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:

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

3.2.3 Aufwärtskompatibilität wird nicht unterstützt

Es wird nicht 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 und Nutzdaten sind 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	Willkürlich lange vorzeichenbehaftete Integer-Nummer	77210710045682438959
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	Willkürlich präzise vorzeichenbehaftete Dezimalnummer	3.14159265358979323846264338327 9
string	byte	base64url kodiertes Byte nach RFC 7493 Sektion 4.4	“VA==”

OpenAPI Typ	OpenAPI Format	Spezifikation	Beispiel
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”
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/“

OpenAPI Typ	OpenAPI Format	Spezifikation	Beispiel
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: Transaktions-Id zur eindeutigen Identifikation eines Aufrufs

› creationDateTime
  - OpenAPI Typ: string
  - OpenAPI Format: date-time
  - Zweck: Zeitpunkt an dem der Aufruf erstellt wurde

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 verarbeitet werden konnte. Erfolgte eine synchrone Antwort (Response Code 202), erfolgt anschließend eine asynchrone Rückmeldung auf den Vorgang, sofern gemäß Prozessbeschreibung eine Rückmeldung zu diesem Vorgang vorgesehen ist. Es ist zu beachten, dass die synchrone Response unverzüglich beim aufrufenden eintreffen muss, sodass die Fristen gemäß Prozessbeschreibung eingehalten und nicht verzögert werden.

3.6 Status Codes (Response)

Jeder Aufruf einer Schnittstelle wird mit einem Http-Statuscodes (synchrone Response) beantwortet. In den von EDI@Energy erstellen 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
500	Internal Server Error	Interner Fehler

3.7 Objekte

MUST Die im API-Aufruf genutzten Objekte werden als JSON-Objekte übermittelt gemäß [RFC8259]. 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.

3.8 Bekanntmachung

Die Marktpartner müssen sich vor Nutzung des API Service über die URL und die verwendeten Zertifikate verständigen. Spätestens drei Werktage (gemäß GPKE/GeLi Gas-Kalender) nach der

erstmaligen Kontaktaufnahme eines Marktpartners müssen die Daten zwischen diesen beiden Parteien ausgetauscht sein.

4 Technische Vorgaben der API-Webdienste

4.1 Netzwerk

MUST Alle Kommunikationsendpunkte werden durch DNS-Namen und nicht durch IP-Adressen bestimmt. Die beschriebenen Webservices müssen IPv4 implementieren und SOLLTEN IPv6 implementieren (Dual Stack).

Der API-Nutzer des API-Webservice MUSS in der Lage sein, die TLS-Verbindung über IPv4 herzustellen.

4.2 Transportebene

MUST Zum Aufbau der http-Verbindung MUSS das Protokoll Transport Layer Security (TLS) nach den Regeln in [TR03116-3] (Kapitel TLS-Kommunikation im WAN und in der Marktkommunikation) eingesetzt werden.

Für den Aufbau der TLS-Verbindung ist die Erweiterung Server Name Identification (SNI) gemäß [RFC6066] bzw. [RFC8449] zu unterstützen und zu verwenden.

4.3 Inhaltsdatensicherungsebene

MUST Jede API-Interaktion und der übertragene Payload wird signiert.

MUST Die dabei einzuhaltenden Verfahren, anzuwendenden Algorithmen und weitere Vorgaben sind in [TR03116-3] (Kapitel Inhaltsdatensicherung) und [CP-SM-PKI] beschrieben und, soweit hier anwendbar, einzuhalten.

Signiert wird die angesprochene Ressource (URI) inklusive aller Query-Parameter, die http-Kopfzeilen creationDateTime, transactionId und soweit vorhanden initialTransactionId, und die übermittelte Payload (JSON-Objekt).

Vor der Berechnung des Hashwerts eines JSON-Objekts muss dieses kanonisiert werden gemäß [RFC8785].

Der zu signierende Digest wird dann berechnet:

- Hash(Hash(URI) + Hash(Payload) + Hash(CreationDateTime) + Hash(transactionId) + Hash(initialTransactionId))

Digest und Signatur werden in den http-Header Feldern X-BDEW DIGEST und X-BDEW-SIGNATURE Base64-kodiert abgelegt.

Das Zertifikat mit dem zum Signieren verwendeten öffentlichen Schlüssel wird in dem HTTP-Header X-BDEW-CERT hinterlegt. Die Kodierung des Zertifikats in einem HTTP-Header Feld ist in Kapitel 2.1 in [RFC9440] für TLS Zertifikate beschrieben und ist hier in gleicher Weise zu verwenden.

Die mittels des API-Aufrufs übertragenen Parameter und Payload werden nicht verschlüsselt.

4.4 Zertifikate und Smart Metering PKI

Die Kommunikation wird durch Verwendung der Smart Metering PKI (SM-PKI) des BSI abgesichert. Die Vorgaben der Certificate Policy (CP) der SM-PKI müssen eingehalten werden. Die Vertrauensdiensteanbieter müssen eine Sub-CA-Instanz im Sinne der CP der SM-PKI sein. Die Kommunikationspartner, API-Nutzer und Anbieter, sind nach dem Rollenkonzept der SM-PKI in der Rolle passiver EMT, sofern der Kommunikationspartner nicht durch Regelungen außerhalb dieses Regelwerks zur API die Rolle eines aktiven EMT wahrnehmen muss.

5 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

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

[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.