API-Guideline

Version:	1.0b
Publikationsdatum:	01.04.2026
Autor:	BDEW

Disclaimer

Die PDF-Datei ist das allein gültige Dokument.

Die zusätzlich veröffentlichte Word-Datei dient als informatorische Lesefassung und entspricht inhaltlich der PDF-Datei. Diese Word-Datei wird bis auf Weiteres rein informatorisch und ergänzend veröffentlicht unter dem Vorbehalt, zukünftig eine kostenpflichtige Veröffentlichung der Word-Datei einzuführen.

Inhalt

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. Das EDI@Energy-Dokument „API-Guideline“ stellt Design-Prinzipien für Web-API Schnittstellen dar, welche das primäre Ziel verfolgen, eine bestmögliche Erfahrung mit dem Umgang webbasierter APIs zu garantieren. Durch die Anwendung dieser Guideline 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 unterstützter 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 Die URLs sollen so strukturiert sein, dass Nutzer die URLs einfach lesen und konstruieren können.

› 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 müssen 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 eines API-Webdienstes

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

<MAJOR>.<MINOR>.<PATCH>

- Die <MAJOR>-Version wird erhöht, wenn die 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. Diese Änderungen sind zum Beispiel redaktionelle Änderungen/Hinweise/Änderung der Referenzen.

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 den 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 den 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, die die ältere Version des API-Webdienstes nutzen, zurechtkommen. 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 BDEW-MaKo Portal (https://bdew-mako.de/).

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: true*

Zusätzlich wird in *description* der folgende Text aufgenommen: *Deprecated ab dem dd.mm.yyyy. 00:00 Uhr*

MUST Eine abgekündigte Version darf zum Abkündigungsdatum vom Anbieter nicht mehr angeboten werden.

3.2.3 Aufwärtskompatibilität wird nicht unterstützt

Es wird unterstützt, dass ein Aufrufer einen optionalen Parameter gemäß einer neueren Schnittstellenspezifikation 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

OpenAPI Typ	OpenAPI Format	Spezifikation	Beispiel
integer	int64	8 Byte vorzeichenbehaftete Integer-Nummer zwischen -2^{63} and $2^{63}-1$	772107100456824
integer	bigint	Vorzeichenbehaftete Integer-Nummer unbegrenzter Länge	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	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”

OpenAPI Typ	OpenAPI Format	Spezifikation	Beispiel
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/“
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: Sicherstellung der Idempotenz

MUST Falls eine Antwort auf eine Anfrage referenzieren muss

› referenceId

- OpenAPI Typ: string
- OpenAPI Format: uuid
- Zweck: ID zur Angabe des Inhalts der initialTransactionId aus der zuvor empfangenen Anfrage bzw. der transactionId, wenn die zuvor empfangene Anfrage keine initialTransactionId enthält (= eindeutige Referenzierung der Antwort auf eine zuvor empfangene Anfrage)

3.4.1 Nutzung der transactionId, initialTransactionId und referenceId

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 in diesen Fällen der wiederholten Anfragen technisch die gleichen Aufrufe identifizierbar zu machen (Idempotenz).

MUST Ist eine Anfrage kein Retry, dann darf die initialTransactionId nicht angegeben werden, d. h., dieser Parameter darf in der Anfrage nicht enthalten sein.

MUST In einer Nachricht, die eine asynchrone Antwort auf eine vorherige Anfrage ist, muss im Feld „referenceId“ die „initialTransactionId“ des ursprünglichen Aufrufs zurückgegeben werden, sofern die „initialTransactionId“ vorhanden ist, ansonsten muss die „transactionId“ zurückgeben werden, um eine eineindeutige Zuordnung zur ursprünglichen Anfrage zu ermöglichen.

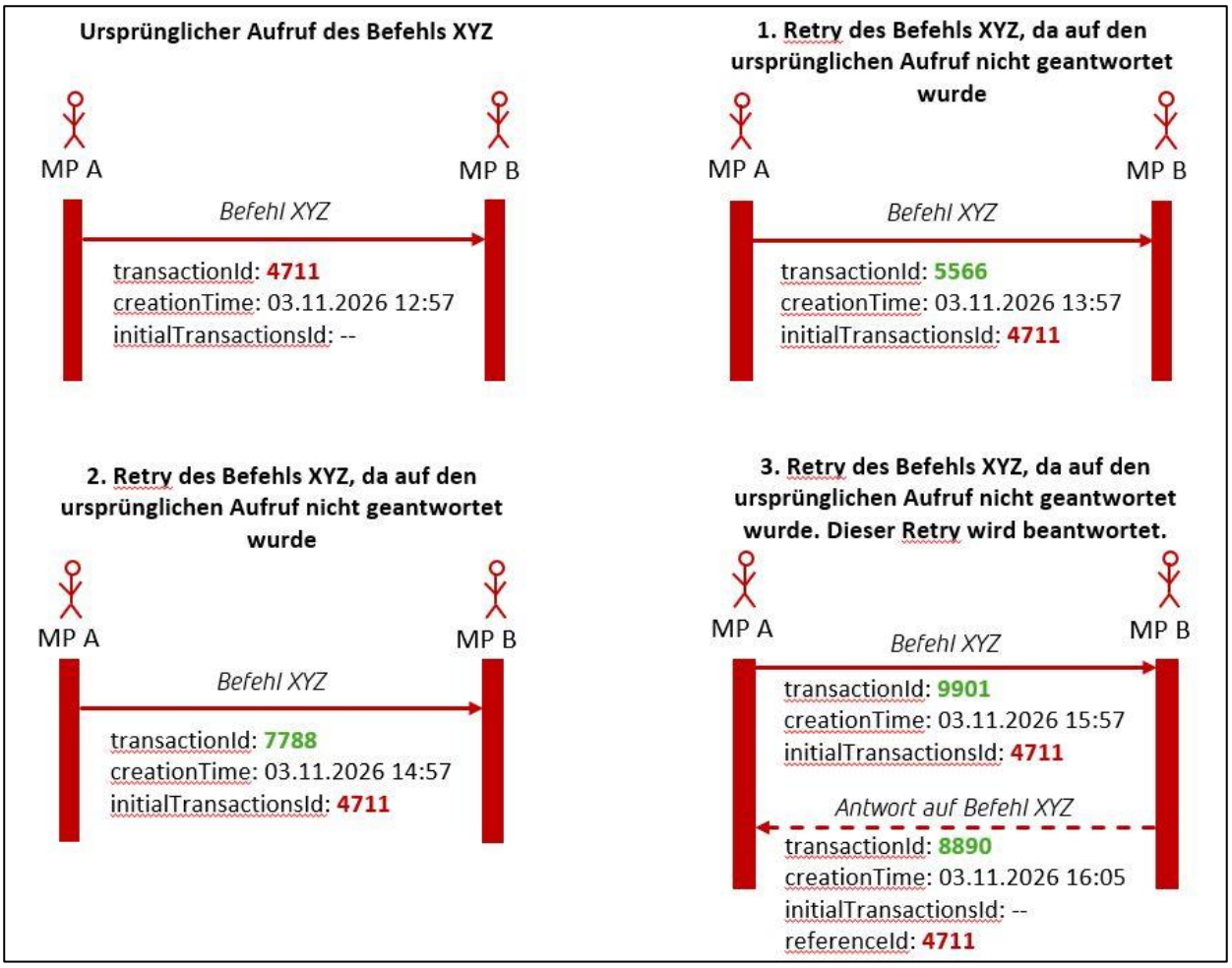

Abbildung 1: Beispielhafte Darstellung der Nutzung von transactionId, initialTransactionId und referenceId

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.

3.7.1 Identische Semantik für required und nullable Eigenschaften

MUST Die OpenAPI-Spezifikation ermöglicht Eigenschaften als erforderlich (required) und als nullwertfähig (nullable) zu kennzeichnen, um festzulegen, ob Eigenschaften fehlen oder den Wert *null* haben dürfen. Die folgende Tabelle zeigt die möglichen Kombinationen der Attribute required und nullabe an einer Eigenschaft und deren gültige Angabe im Payload.

required	**nullable**	**Wert darf fehlen?** **(Beispiel: {})**	**Wert darf „null“ sein?** **(Beispiel: {„EXAMPLE“:NULL})**
true	true	Nein	Ja

false	true	Ja	Ja
true	false	Nein	Nein
false	false	Ja	Nein

Hinweis: Ist an einer Eigenschaft das Attribut nullable nicht explizit angegeben, so ist dieser im Default *false*, das heißt, bei optionalen Eigenschaften ohne explizites nullable-Attribut oder *nullable = false* darf im Payload für diese Eigenschaft der Wert *null* nicht enthalten sein.

3.7.2 Leere Arrays und Nullwerte

MUST Für erforderliche Eigenschaften vom Typ Array, bei denen das Attribut *required = true* ist, ist die leere Liste *[]* ein gültiger Wert, sofern das Attribut *minItems* nicht angegeben ist. Ein Empfänger muss diesen Wert akzeptieren. Es ist zu beachten, dass ohne die explizite Angabe von *minItems* der Default den Wert 0 hat.

MUST In einer Liste dürfen keine Objekte mit dem Wert *null* enthalten sein.

4 Veröffentlichung der API-Webdienste auf GitHub

EDI@Energy wird alle API-Webdienste zentral in öffentlichen Verzeichnissen (im Folgenden „Repositories“) auf der GitHub-Plattform unter der Organisation EDI@Energy veröffentlichen und pflegen. Die Marktteilnehmer können somit die maschinenlesbaren Dokumentationen der Schnittstellenbeschreibungen nutzen, um Weiterentwicklungen bzw. Veränderungen an bestehenden Schnittstellen leichter nachzuvollziehen.

Folgende öffentliche Repositories sind vorhanden:

- **api-electricity**

  Dieses Repository enthält alle produktiven API-Webdienste (auch Fehlerkorrekturen) und Konsultationsversionen, die zur Umsetzung der Prozesse aus den Festlegungen der Bundesnetzagentur (z. B. GPKE, WiM-Strom, MaBiS etc.) benötigt werden.

- **api-directory-service**

  Dieses Repository enthält ausschließlich die API-Webdienste für die Verzeichnisdienste zum Austausch der Endpunktadressen.

Die EDI@Energy behält es sich vor, bei Bedarf weitere Repositories einzuführen.

4.1 Versionierung eines Repository

Jedes Repository erhält auf GitHub eine eindeutige Versionsnummer. Versionsnummern werden auf GitHub über „Releases“ abgebildet. Der Aufbau der Versionsnummer sieht wie folgt aus:

<Major>.<Minor>.<Patch>

› Die *<Major>*-Version wird erhöht,
  - bei der Eröffnung einer Konsultation (die Veröffentlichung der Ergebnisse erfolgt unter der Versionsnummer, unter der die Konsultation eröffnet wurde).

› Die *<Minor>*-Version wird erhöht,
  - wenn mindestens an einem im Repository enthaltenen API-Webdienstes die <Major>-Version erhöht wurde.

› Die *<Patch>*-Version wird erhöht,
  - wenn mindestens an einem im Repository enthaltenen API-Webdienstes die <Minor>- oder <Patch>-Version erhöht wurde.

Die Versionierung eines Repositorys erfolgt über die Veröffentlichung eines Releases am Repository selbst. Im Release erfolgt zudem die Veröffentlichung einer Änderungshistorie.

5 Ablauf einer Konsultation

Die API-Webdienste werden wie die EDIFACT-Formate bei Veränderungen oder der Einführung neuer API-Webdienste zur Konsultation gestellt und unterliegen dem bekannten Änderungsmanagement der EDI@Energy. Die Konsultationsbeiträge eines Unternehmens müssen direkt im Repository auf GitHub über Issues erfolgen. Hierzu benötigt jedes an der Konsultation teilnehmende Unternehmen oder jede teilnehmende Privatperson einen entsprechenden GitHub-Account. Das Anlegen eines GitHub-Accounts ist kostenlos möglich.

Zur Erstellung von Konsultationsbeiträgen ist pro natürliche Person, Organisation oder Unternehmen lediglich ein Account zulässig. Bei Unternehmen mit mehreren MP-IDs kann je Marktrolle ein eigener Account genutzt werden. Voraussetzung für die Berücksichtigung der Konsultationsbeiträge ist, dass der GitHub-Nutzername unter dem die Issues angelegt werden, der Beschlusskammer 6 per E-Mail (datenformate@bnetza.de) spätestens bis zum Ende des Konsultationszeitraums mitgeteilt wird. Falls es sich nicht um eine natürliche Person handelt, ist zusätzlich die Organisation bzw. das Unternehmen anzugeben.

Hinweise zu diesem Verfahren:

- Angelegte Issues sind für Dritte sofort nach der Einreichung sichtbar.
- Angelegte Issues können durch Dritte kommentiert werden.

- Das Anlegen von Issues ist nur innerhalb des Konsultationszeitraumes möglich. Zu spät angelegte Issues werden in der Konsultation nicht betrachtet.

Durch dieses Vorgehen wird die größtmögliche Transparenz in einem Konsultationsverfahren ermöglicht.

Hinweis: Grundsätzlich können Issues in GitHub verwendet werden, um Ideen, Aufgaben, Bugs oder Rückmeldungen zu einem Quellcode einzubringen, zu benennen, zu melden oder zu geben. In den EDI@Energy-Repositories ist das Anlegen von Issues nur im Rahmen einer Konsultation zulässig. Issues, die außerhalb des Konsultationszeitraums angelegt werden, werden von EDI@Energy nicht berücksichtigt. Ein angelegtes Issue kann ausschließlich von den Administratoren der Organisation gelöscht werden. Eine detaillierte Dokumentation zu Issues auf GitHub kann unter dem folgenden Link nachgelesen werden: Link

6 Abbildungsverzeichnis

7 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.

8 Änderungshistorie

Änd-ID	Ort	Änderungen		Grund der Anpassung	Status
		Bisher	Neu		
10000	Gesamtes Dokument	Version: 1.0a	Version: 1.0b	Version aktualisiert. Zusätzlich wurden im gesamten Dokument Schreibfehler, Layout, Beispiele etc. geändert, die keinen Einfluss auf die inhaltliche Aussage haben.	Genehmigt
26124	Kapitel 3.2 Versionierung	Name: Versionierung	Name: Versionierung eines API-Webdienstes	Umbenennung	Genehmigt
27075	Kapitel 3.2 Versionierung	[…] * Die <Patch>-Version wird erhöht, wenn die Änderungen ausschließlich API-kompatible Bugfixes umfassen.	[…] * Die <Patch>-Version wird erhöht, wenn die Änderungen ausschließlich API-kompatible Bugfixes umfassen. Diese Änderungen sind zum Beispiel redaktionelle Änderungen/Hinweise/Änderung der Referenzen.	Präzisierung	Genehmigt
26177	Kapitel 3.2.2 Abwärtskompatibilität	[…] 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	[…] MUST Es sind immer alle nicht abgekündigten Versionen nutzbar. Eine Abkündigung erfolgt durch die Kennzeichnung des API-Webdienstes, diese sieht wie folgt aus: › *Deprecated: true* Zusätzlich wird in *description* der folgende Text aufgenommen: *Deprecated ab dem dd.mm.yyyy. 00:00 Uhr* MUST Eine abgekündigte Version darf zum Abkündigungsdatum vom Anbieter nicht mehr angeboten werden.	Präzisierung	Genehmigt

Änd-ID	Ort	Änderungen		Grund der Anpassung	Status
		Bisher	Neu		
27068	Kapitel 3.4.1 Nutzung der transactionId, initialTransactionId und referenceId	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 in diesen Fällen der wiederholten Anfragen technisch die gleichen Aufrufe identifizierbar zu machen (Idempotenz). [...]	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 in diesen Fällen der wiederholten Anfragen technisch die gleichen Aufrufe identifizierbar zu machen (Idempotenz). MUST Ist eine Anfrage kein Retry, dann darf die initialTransctionId nicht angegeben werden, d. h., dieser Parameter darf in der Anfrage nicht enthalten sein. MUST In einer Nachricht, die eine asynchrone Antwort auf eine vorherige Anfrage ist, muss im Feld „referenceId“ die „initialTransactionId“ des ursprünglichen Aufrufs zurückgegeben werden, sofern die „initialTransactionId“ vorhanden ist, ansonsten muss die „transactionId“ zurückgeben werden, um eine eineindeutige Zuordnung zur ursprünglichen Anfrage zu ermöglichen.	Die initialTransationId darf nur bei einem Retry angegeben werden. Konkretisierung zur Nutzung der referenceId und initialTransactionId. Aufnahme einer Abbildung zur Veranschaulichung zur Nutzung der referenceId und initialTransactionId	Genehmigt
27069	Kapitel 3.7.1 Identische Semantik für required und nullable Eigenschaft	nicht vorhanden	MUST Die OpenAPI-Spezifikation ermöglicht Eigenschaften als erforderlich (required) und als nullwertfähig (nullable) zu kennzeichnen, um festzulegen, ob Eigenschaften fehlen oder den Wert null haben dürfen. Die folgende Tabelle zeigt die möglichen Kombinationen der Attribute required und nullabe an einer Eigenschaft und deren gültige Angabe im Payload.	Aufnahme von Regeln zum Umgang mit Eigenschaften die als required und nullable gekennzeichnet sind.	Genehmigt

Änd-ID	Ort	Änderungen		Grund der Anpassung	Status
		Bisher	Neu		
			[*Tabelle*] Hinweis: Ist an einer Eigenschaft das Attribut nullable nicht explizit angegeben, so ist dieser im Default false, das heißt, bei optionalen Eigenschaften ohne explizites nullable-Attribut oder nullable = false darf im Payload für diese Eigenschaft der Wert null nicht enthalten sein.		
27070	Neues Kapitel 3.7.2 Leere Arrays und Nullwerte	nicht vorhanden	MUST Für erforderliche Eigenschaften vom Typ Array, bei denen das Attribut required = true ist, ist die leere Liste [] ein gültiger Wert, sofern das Attribut minItems nicht angegeben ist. Ein Empfänger muss diesen Wert akzeptieren. Es ist zu beachten, dass ohne die explizite Angabe von minItems der Default minItems=0 ist. MUST In einer Liste dürfen keine Objekte mit dem Wert null enthalten sein.	Aufnahme von Regeln zum Umgang mit leeren Arrays und Nullwerten.	Genehmigt
27072	Neues Kapitel 4 Veröffentlichung der API-Webdienste auf GitHub	nicht vorhanden	vorhanden	Die API-Webdienste werden demnächst auf GitHub veröffentlicht.	Genehmigt
27074	Neues Kapitel 5 Ablauf einer Konsultation	nicht vorhanden	vorhanden	In diesem Kapitel wird Ablauf einer Konsultation und den Umgang mit Konsultationsbeiträgen beschrieben.	Genehmigt