

DIN EN 50090-6-3

ICS 35.240.67; 97.120

Einsprüche bis 2024-09-26

Entwurf**Elektrische Systemtechnik für Heim und Gebäude (ESHG) –
Teil 6-3: Drittanbieter ESGH IoT API;
Deutsche Fassung EN 50090-6-3:2023**

Home and Building Electronic Systems (HBES) –
Part 6-3: 3rd Party HBES IoT API;
German version EN 50090-6-3:2023

Systèmes électroniques pour les foyers domestiques et les bâtiments (HBES) –
Partie 6-3: API tierce IdO HBES;
Version allemande EN 50090-6-3:2023

Anwendungswarnvermerk

Dieser Entwurf mit Erscheinungsdatum 2024-07-26 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil das beabsichtigte Dokument von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an dke@vde.com möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, Merianstraße 28, 63069 Offenbach am Main.

Es wird gebeten, mit den Kommentaren zu diesem Entwurf jegliche relevanten Patentrechte, die bekannt sind, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 137 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE

Nationales Vorwort

Für dieses Dokument ist das nationale Arbeitsgremium DKE/K 716 „Elektrische Systemtechnik für Heim und Gebäude (ESHG)“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE (www.dke.de) zuständig.

Aktuelle Informationen zu diesem Dokument können über die Internetseiten von DKE (www.dke.de) und DIN (www.din.de) durch eine Suche nach der Dokumentennummer aufgerufen werden.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf ein Dokument ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils aktuellste Ausgabe des in Bezug genommenen Dokuments.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe des Dokuments.

Der Zusammenhang der zitierten Dokumente mit den entsprechenden deutschen Dokumenten ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation.

Das Original-Dokument enthält Bilder in Farbe, die in der Papierversion in einer Graustufen-Darstellung wiedergegeben werden. Elektronische Versionen dieses Dokuments enthalten die Bilder in der originalen Farbdarstellung.

Deutsche Fassung

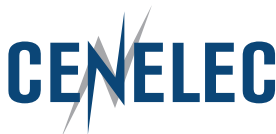
Elektrische Systemtechnik für Heim und Gebäude
(ESHG) – Teil 6-3: Drittanbieter ESGH IoT APIHome and Building Electronic Systems
(HBES) – Part 6-3: 3rd Party HBES IoT APISystèmes électroniques pour les foyers domestiques et
les bâtiments (HBES) – Partie 6-3: API tierce IdO HBES

Diese Europäische Norm wurde von CENELEC am 2022-12-22 angenommen. CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim CEN-CENELEC Management Centre oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem CEN-CENELEC Management Centre mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Kroatien, Lettland, Litauen, Luxemburg, Malta, den Niederlanden, Norwegen, Österreich, Polen, Portugal, der Republik Nordmazedonien, Rumänien, Schweden, der Schweiz, Serbien, der Slowakei, Slowenien, Spanien, der Tschechischen Republik, der Türkei, Ungarn, dem Vereinigten Königreich und Zypern.

Europäisches Komitee für Elektrotechnische Normung
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brüssel

Inhalt

	Seite
Europäisches Vorwort.....	6
1 Anwendungsbereich.....	7
2 Normative Verweisungen.....	7
3 Begriffe und Abkürzungen.....	7
3.1 Begriffe.....	7
3.2 Abkürzungen.....	14
4 ESHG-IoT-Drittanbieter-API.....	14
4.1 Einleitung.....	14
4.1.1 Abbildung der Begriffe.....	14
4.1.2 Allgemeines.....	15
4.1.3 API – Inhalt.....	17
4.1.4 API – Konformität.....	17
4.1.5 API – Version.....	18
4.1.6 API –Verfügbarkeit.....	18
4.1.7 API – Datenformat.....	18
4.1.8 API – Format der Beispiele.....	18
4.1.9 Hintergrund.....	18
4.1.10 Ressourcen.....	19
4.2 Festlegungen.....	21
4.2.1 Anforderungen an Server.....	21
4.2.2 Clientanforderungen.....	42
4.2.3 Nachrichtendatenstruktur.....	43
4.2.4 URL-Struktur.....	62
4.2.5 Ressourcenpfaddefinitionen.....	63
4.2.6 Erfassung.....	134
Bilder	
Bild 1 – Projektelemente auf dem ESHG-IoT-Drittanbieter-Server.....	16
Bild 2 – Beziehung einer ESHG-IoT-Drittanbieter-API zu Funktionen und IoT-Funktionen.....	75
Bild 3 – Aufbau der Cloud-Umgebung.....	96
Bild 4 – Aufbau des lokalen Netzwerks.....	97
Bild 5 – Verarbeitungsablauf von Subscriptions.....	110
Bild 6 – Zugriffsablauf in einer lokalen Umgebung, Anwendungsfall A.1.....	130
Bild 7 – Zugriffsablauf in einer lokalen Umgebung, Anwendungsfall A.2.....	131
Bild 8 – Zugriffsablauf in einer öffentlichen Umgebung.....	133
Tabellen	
Tabelle 1 – Abbildung der Begriffe.....	15
Tabelle 2 – Festlegung des Parameternamensfilters für den operator-Teil.....	26
Tabelle 3 – Parametername, der zwei KIM-Objekteigenschaften enthält.....	28

Tabelle 4 – Parametername, der eine KIM-Objekteigenschaft enthält.....	29
Tabelle 5 – 1xx-Antwortcodes.....	37
Tabelle 6 – 2xx-Antwortcodes.....	37
Tabelle 7 – 4xx-Antwortcodes.....	38
Tabelle 8 – 5xx-Antwortcodes.....	40
Tabelle 9 – API-Referenz-Mitglied.....	48
Tabelle 10 – Relationships-Mitglied.....	55
Tabelle 11 – URL-Pfaddefinitionen.....	63
Tabelle 12 – Ortstypen.....	73
Tabelle 13 – IoT-Funktionen.....	77
Tabelle 14 – Beispiele für IoT-Datenpunkte.....	88
Tabelle 15 – Typen von Subscriptions.....	94
Tabelle 16 – Inhalt des Transaktionsmetaobjektes.....	99
Tabelle 17 – Schritte des Sicherheitsablaufs.....	124
Tabelle 18 – Authentifizierungsdetails für die Umgebung.....	125

Europäisches Vorwort

Dieses Dokument (EN 50090-6-3:2023) wurde vom CLC/TC 205 „Elektrische Systemtechnik für Heim und Gebäude (ESHG)“ ausgearbeitet.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem dieses Dokument auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop) 2024-01-07
- spätestes Datum, zu dem nationale Normen, die diesem Dokument entgegenstehen, zurückgezogen werden müssen (dow) 2026-07-07

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. CENELEC ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren.

Rückmeldungen oder Fragen zu diesem Dokument sollten an das jeweilige nationale Komitee des Anwenders gerichtet werden. Eine vollständige Liste dieser Gremien ist auf den Internetseiten des CENELEC abrufbar.

1 Anwendungsbereich

Dieses Dokument legt eine Drittanbieter-API für das offene Kommunikationssystem für elektrische Systemtechnik für Heim und Gebäude (ESHG) fest.

2 Normative Verweisungen

Die folgenden Dokumente werden im Text in solcher Weise in Bezug genommen, dass einige Teile davon oder ihr gesamter Inhalt Anforderungen des vorliegenden Dokuments darstellen. Bei datierten Verweisungen gilt nur die in Bezug genommene Ausgabe. Bei undatierten Verweisungen gilt die letzte Ausgabe des in Bezug genommenen Dokuments (einschließlich aller Änderungen).

EN 50090-1:2011, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) Teil 1: Aufbau der Norm*

EN 50090-3-3, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 3-3: Anwendungsaspekte – ESHG-Interworking-Modell und übliche ESHG-Datenformate*

EN 50090-6-2, *Elektrische Systemtechnik für Heim und Gebäude (ESHG) – Teil 6-2: Beschreibung des IoT semantischen Ontologiemodells*

3 Begriffe und Abkürzungen

3.1 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach EN 50090-1:2011 und die folgenden Begriffe.

ISO und IEC stellen terminologische Datenbanken für die Verwendung in der Normung unter den folgenden Adressen bereit:

- IEC Electropedia: verfügbar unter <https://www.electropedia.org/>
- ISO Online browsing platform: verfügbar unter <https://www.iso.org/obp>

3.1.1

Aktor

en **actuator**

Punkt, der eine *Betätigung* vornimmt (ausgeführt von einem bestimmten *Verfahren* mit einem erwarteten *Ergebnis*), die den Zustand einer Installation zur Laufzeit ändert

Anmerkung 1 zum Begriff:

- Der Begriff „Actuator“ (Aktor) kann auf `sosa:Actuator` in der SSN-Ontologie abgebildet werden.
- Das Subjekt *Betätigung* kann auf `sosa:Actuation` in der SSN-Ontologie abgebildet werden.
- Das Subjekt *Verfahren* kann auf `sosa:Procedure` in der SSN-Ontologie abgebildet werden.
- Das Subjekt *Ergebnis* kann auf `sosa:Result` in der SSN-Ontologie abgebildet werden.

3.1.2

Anwendungsfunktion

en **application function**

Funktion, die eine Reihe von Funktionen verwendet, um das gewünschte Verhalten eines technischen Systems herbeizuführen, üblicherweise eine Kombination von Geräten, die über ihre Eingangs- und Ausgangsdatenpunkte Informationen untereinander austauschen

Anmerkung 1 zum Begriff: Eine Anwendungsfunktion kann in mehrere Funktionsblöcke mit eigenen Eingangs- und Ausgangsdatenpunkten aufgeteilt sein, die logisch miteinander verbunden sind. Die Funktionsblöcke können in einem oder mehreren Geräten untergebracht sein.

BEISPIEL Beispiele für Anwendungsfunktionen sind „direkte Elektroheizung“, „Elektroheizung mit Speichern“, „Warmwasserheizung“, „Gebläsekonvektor-Klimaanlage“ usw.

Anmerkung 2 zum Begriff: Anwendungsfunktion und Anwendung sind gleichbedeutend. Mit der Einführung des Synonyms soll eine eindeutige (nachvollziehbare) Bezugnahme von Anwendung/Anwendungsfunktion auf die entsprechende KIM-Klasse :ApplicationFunction oder die Funktion im Management-Client ermöglicht werden.

3.1.3

Aspekt

en aspect

im Allgemeinen ein bestimmter Blickwinkel auf ein System, das Dinge mit verschiedenen Eigenschaften enthält; ein Referenzierungsmechanismus für die Organisation von KIM-Elementen unter einem bestimmten Gesichtspunkt

BEISPIEL Ein Funktionspunkt ist definitionsgemäß ein Aspekt mit einem wichtigen spezifischen Blickwinkel. Es handelt sich um einen Referenzierungsmechanismus für die Organisation aller mit einem Funktionspunkt zusammenwirkenden Punkte (aller mit einer GA verknüpften GOs).

3.1.4

BIM

Bauwerksinformationsmodell

en building information model

digitaler Prozess für die Beschreibung und Dokumentation eines Bauwerks über sämtliche Phasen von dessen Lebenszyklus hinweg, von der Planung über den Bau und die Nutzung bis zum Abriss

3.1.5

Kanal

en channel

Zusammenstellung von Datenpunkten eines Gerätes, die logisch miteinander zusammenhängen, üblicherweise durch die Zuweisung zu einem Hardwaremerkmal oder einer bestimmten Funktion dieses Gerätes

Anmerkung 1 zum Begriff: Diese Datenpunkte können von einem oder mehreren definierten Funktionsblöcken abgeleitet sein, eine Erweiterung von definierten Funktionsblöcken sein oder unabhängig von einem Funktionsblock sein, wenn für die mit dem Kanal verknüpfte Funktion keiner definiert ist. Das Konzept eines Kanals ist unter den Marktteilnehmern, beispielsweise Installateuren, bekannt.

3.1.6

Datenpunkt

en datapoint

Repräsentation einer logischen Eingangsentität eines Gerätes, die als Empfänger von Zustandsdaten einer Installation agiert, während ein logischer Ausgang eines Gerätes als Quelle von Zustandsdaten einer Installation agiert

Anmerkung 1 zum Begriff: Bei einer Implementierung als Gruppenobjekt werden Zustandsdaten durch die Verwendung von Funktionspunkten übertragen.

Anmerkung 2 zum Begriff: Der Begriff „Datenpunkt“ ist ein allgemeiner Begriff; um speziell einen an einer IoT-Drittanbieter-API verfügbaren Datenpunkt zu beschreiben, wird der Begriff „IoT-Datenpunkt“ verwendet.

3.1.7

Gerät

en device

physisches Element, das Bestandteil des Netzwerks ist; es handelt sich um ein physisches, konkretes Objekt, das ein Kunde erwerben kann

3.1.8

Endpunkt

en **endpoint**

Zugangspunkt zu einem Dienst, einem Prozess oder einem Queue- oder Topic-Ziel in einer diensteorientierten Architektur

3.1.9

betrachtetes Merkmal (FOI)

en **feature of interest**

Abstraktion eines Elements aus der realen Welt (Phänomen, Ausrüstung, Person, Ereignis ...), das durch seine beobachtbaren oder betätigbaren Eigenschaften definiert wird

Anmerkung 1 zum Begriff: Einfach gesagt ist ein FOI ein Träger von Eigenschaften.

Anmerkung 2 zum Begriff: Ein Sensor bezieht sich auf ein FOI mit beobachtbaren (en: observable) Eigenschaften, ein Aktor auf ein FOI mit betätigbaren (en: actuatable) Eigenschaften.

Anmerkung 3 zum Begriff: Ein FOI stellt selbst kein „Klassifizierung/Typ“-Tag dar; „Klassifizierung/Typ“ wird mithilfe von Tags realisiert. Beispiele werden in 4.5.1.4 definiert.

3.1.10

Funktion

en **function**

Teil des vorgesehenen Verhaltens eines FB in einem Bauwerkskontext

3.1.11

Funktionsblock

en **functional block**

besteht aus einer oder mehreren Funktionen, die zusammengehören und nicht auf zwei Geräte aufgeteilt werden können, die aber umfangreich genug sind, dass ein Gerät mit nur einem solchen Funktionsblock betrieben werden könnte

Anmerkung 1 zum Begriff: Ein Funktionsblock weist ein klar definiertes Black-Box-Verhalten auf.

3.1.12

Funktionspunkt

en **function point**

Laufzeitsystem-Zustandsinformationen einer bestimmten Anwendungsfunktion

Anmerkung 1 zum Begriff: Wird von mindestens zwei Datenpunkten gemeinsam genutzt.

Anmerkung 2 zum Begriff: Verfügt über einen eindeutigen Bezeichner, der eine Gruppe gesteuerter Objekte bezeichnet. Dieser Bezeichner wird als „Gruppenadresse“ bezeichnet.

BEISPIEL <Lichtschalter> in Wohnzimmer ein-/ausgeschaltet, wobei <...> den Namen des Funktionspunktes enthält

3.1.13

Gruppenadresse

en **group address**

numerischer Bezeichner eines Funktionspunktes

3.1.14

Gruppenkommunikation

en **group communication**

Kommunikationsmodell, in dem ein Sender Informationen an einen und typischerweise mehrere Empfänger überträgt

Anmerkung 1 zum Begriff: Beim IoT kann dies durch einfache UDP-Kommunikation, durch ein Message-Broker-System oder anderweitig realisiert werden.

3.1.15

Gruppenobjekt

en **group object**

vorgesehen für Gruppenkommunikation mittels Gruppenadresse(n)

Anmerkung 1 zum Begriff: Der Zugriff kann ohne zugewiesene Gruppenadresse über Punkt-zu-Punkt-Kommunikation erfolgen. Mit zugewiesener Gruppenadresse wird es zu einem Mitglied des Funktionspunktes, den die Gruppenadresse repräsentiert.

3.1.16

ESHG-Informationsmodell

en **HBES information model**

ontologiebasiertes Modell der für ein ESHG-System relevanten Teile, einschließlich zusätzlicher semantischer (Wörterbuch-)Informationen

Anmerkung 1 zum Begriff: Die Verwaltung erfolgt durch die HBES Association, daher die Abkürzung KIM.

3.1.17

Industry Foundation Class

offener Standard zur digitalen Beschreibung von BIM-Daten

Anmerkung 1 zum Begriff: Die IFC-Daten und -Modelle werden in ISO 16739-1 festgelegt.

3.1.18

Installation

Zusammenstellung von Materialien und Komponenten (Geräten), die auf eine bestimmte Art und Weise angeordnet werden, um einen bestimmten Dienst zu erbringen

Anmerkung 1 zum Begriff: Eine Installation ist ein eingerichtetes System (z. B. ein HLK-System oder eine Brandschutzanlage) und besteht aus Ausrüstung und Funktionen, die einem bestimmten Zweck dienen.

Anmerkung 2 zum Begriff: Im Zusammenhang mit diesem Begriff beziehen sich erstellte Daten auf das in 4.2 beschriebene Installationsmodell.

[QUELLE: ISO 6707-1:2020, modifiziert – „(Geräten)“ und Anmerkungen zum Begriff hinzugefügt.]

3.1.19

IoT-Datenpunkt

en **IoT datapoint**

Repräsentation eines Endpunktes an einer IoT-Drittanbieter-API, der:

- a) einem oder mehreren Funktionspunkten entspricht, wie eine Zustandsdatendarstellung eines diskreten Zustands im Bauwerkskontext:

BEISPIEL 1 Helligkeit → der diskrete Zustand „Helligkeit“ wird durch den Wert 65 (Prozent) repräsentiert

- b) eine vollqualifizierte URL ist, die z. B. von einem IoT-Drittanbieter-Server bereitgestellt wird

BEISPIEL 2 <https://gateway.knx.local/knx/api/v1/datapoints/{Id}>

3.1.20

IoT-Funktion

en **IoT function**

Repräsentation einer Funktion an einer IoT-Drittanbieter-API, die:

- sich wie eine Zusammenstellung von IoT-Datenpunkten verhält, die ein – vom Benutzer – gewünschtes Verhalten aufweist

BEISPIEL „Wohnzimmer – Dimmen der Hintergrundbeleuchtung“, „Küche – Fußbodenheizung“

Anmerkung 1 zum Begriff: Bei einem MaC entspricht eine IoT-Funktion instanziierten Daten einer MaC-Funktion in einer Installation bzw. einem MaC-Projekt. Die MaC-Funktion selbst kann auf einer Anwendungsfunktion beruhen.

3.1.21

IoT-Drittanbieter-API

en **IoT 3rd party API**

Satz von Anforderungen und Vorschriften, in dessen Rahmen ein teilweiser Zugriff auf eine Installation ermöglicht wird, indem eine Zusammenstellung von Endpunkten bereitgestellt wird

3.1.22

IoT-Drittanbieter-Client

en **IoT 3rd party client**

Gerät oder Dienst, das/der über die IoT-Drittanbieter-API von außen mit der Installation interagiert

Anmerkung 1 zum Begriff: Der IoT-Drittanbieter-Client verbindet sich mit einem einzelnen Gerät, das die IoT-Drittanbieter-API bereitstellt, und kann über dieses Gerät vollumfänglich mit der Installation interagieren, gegebenenfalls in Abhängigkeit von einem festgelegten Autorisierungsmechanismus.

BEISPIEL Ein Mobiltelefon (das Teil des Netzwerks oder über eine Internetverbindung verbunden ist), mit dem üblicherweise jeweils nur für kurze Zeitspannen eine Verbindung hergestellt wird.

BEISPIEL Ein Wetterdienst, der die Informationen zum Wetter in einem Feed beständig über die IoT-Drittanbieter-API überträgt.

3.1.23

IoT-Drittanbieter-Server

en **IoT 3rd party server**

Gerät, das die IoT-Drittanbieter-API implementiert

Anmerkung 1 zum Begriff: Hierbei kann es sich um ein eigens dafür vorgesehenes Gerät handeln; es kann eine Funktion eines Gerätes sein, das noch weitere ESHG-IoT-Funktionalitäten und Nicht-ESHG-Funktionalitäten unterstützt; es kann sich innerhalb oder außerhalb des LAN der IoT-Installation befinden.

3.1.24

MaC-Katalogeintrag

en **MaC catalog entry**

erstellte Management-Client-Daten, die sich auf das in 4.2 beschriebene Produktmodell beziehen

3.1.25

MaC-Funktion

en **MaC function**

von dem MaC erstellte Anwendungsfunktion, die einem baulichen Element zugewiesen ist und mit der mehrere Gruppenadressen gruppiert werden

3.1.26

MaC-Projekt

en **MaC project**

von einem MaC erstelltes Projekt, das die Konfiguration einer Installation dokumentiert

3.1.27

Management-Client

Mittel für die Konfiguration und Inbetriebnahme von Geräten sowie für die Planung und Auslegung von und die Diagnosefindung für eine gesamte Installation

Anmerkung 1 zum Begriff: Der MaC dient der Konfiguration und Inbetriebnahme von Geräten sowie der Planung und Auslegung von und der Diagnosefindung für eine gesamte Installation. Im letzten Schritt schreibt der MaC spezifische Konfigurationsdaten wie die Geräteparameter auf die Geräte.

3.1.28

Ontologie

en **ontology**

konzeptuelle Beschreibung von Dingen, die in der realen Welt Gemeinsamkeiten aufweisen, zur Vermittlung von Wissen in einer bestimmten Domäne, wobei die Beschreibung zumeist mittels OWL erfolgt

Anmerkung 1 zum Begriff: Ontologien dienen der strukturierten Beschreibung der Bedeutung von Daten in Ontologieklassen und sollten nicht mit gewöhnlichen Datenmodellstrukturen verwechselt werden.

3.1.29

Objekteigenschaft

en **object property**

in OWL ein eingebautes Konzept, das Paare von Individuen **verbindet**, wobei der Ausdruck einer Objekteigenschaft die (gesamte) Beziehung zwischen den Paaren von Individuen **repräsentiert**

3.1.30

OWL

OWL 2 Web Ontology Language, informell auch „OWL 2“, festgelegt von dem World Wide Web Consortium (W3C), hauptsächlich serialisiert mit XML-Syntax für RDF (RDF/XML)

Anmerkung 1 zum Begriff: In diesem Dokument bezieht sich die Abkürzung OWL ausdrücklich stets auf OWL 2.

3.1.31

Punkt

en **point**

Repräsentation einer Schnittstelle zu Daten im System

Anmerkung 1 zum Begriff: In diesem Dokument wird der Begriff „Punkt“ als Sammelbegriff für Daten verwendet, auf die von außerhalb des Gerätes zugegriffen werden kann, beispielsweise, um mit den Punkten anderer Geräte zu interagieren. Der Begriff „Punkt“ ist dementsprechend ein allgemeiner Oberbegriff des Begriffs „Datenpunkt“ (welcher die Techniken, mit denen die „Daten“ im System strukturiert und/oder kodiert werden, genauer beschreibt).

3.1.32

Punkt-API, en: point API

en **point API**

einfache RESTful-Anwendungsprogrammierschnittstelle (CoAP oder HTTP), die für Beschränkungen unterliegende Geräte der Klasse 2 [RFC 7228] vorgesehen, aber nicht auf diese beschränkt ist und die Individualisierung von Geräten, die Verknüpfung („Linking“) von Geräten und den Zugriff auf die Gerätelaufzeitdaten (z. B. Funktionsblock- oder Kanal-Datenpunkte) unterstützt

3.1.33

Eigenschaftsart

en **quality kind**

Repräsentation einer bestimmten Kombination von beobachtbaren oder betätigbaren Eigenschaften, die als vordefinierter Teil des semantischen Wörterbuchs zur Verfügung steht oder bei der Konfiguration einzeln angelegt wird; Letzteres ist der Fall, wenn eine Eigenschaftsart mit der vorgesehenen Kombination von Eigenschaften bzw. Tags (noch) nicht im Wörterbuch enthalten ist

Anmerkung 1 zum Begriff: Eine QK stellt selbst kein „Klassifizierung/Typ“-Tag dar; „Klassifizierung/Typ“ wird mithilfe von Tags realisiert. Beispiele werden in 4.5.1.4 definiert.

3.1.34

RDF

Resource Description Framework nach <https://www.w3.org/RDF/>

Anmerkung 1 zum Begriff: RDF ist ein Framework für die Darstellung von Informationen im Web mithilfe von Tripeln. Die Informationen können serialisiert und in vielen verschiedenen Formaten wie TURTLE oder JSON(-LD) gespeichert werden. Eine allgemeine Beschreibung des Konzepts des RDF ist unter <https://www.w3.org/TR/rdf11-concepts/> zu finden.

3.1.35

Laufzeit

en **runtime**

Prozess-zu-Prozess-Übertragung von Daten zwischen Geräten, im Gegensatz zu Konfiguration

Anmerkung 1 zum Begriff: Dies betrifft in erster Linie die Übertragung von Datenpunktwerten (Steuer- und Statusinformationen).

3.1.36

semantischer Export

en **semantic export**

von dem MaC exportiertes Projekt, das eine Installation in einem Format für verknüpfte Daten widerspiegelt

Anmerkung 1 zum Begriff: Die exportierten Daten:

- werden entsprechend dem KIM strukturiert, beispielsweise mithilfe von im KIM definierten Objekteigenschaften;
- werden mit zusätzlichen semantischen Informationen aus dem semantischen Wörterbuch annotiert;
- referenzieren Konzepte von externen Ontologien.

3.1.37

semantisches Wörterbuch

en **semantic dictionary**

Satz von standardisierten Begriffen, mit denen erforderliche Teile einer Installation annotiert werden können

Anmerkung 1 zum Begriff: Für Einzelheiten siehe 4.2.8.

3.1.38

Sensor

Punkt, der eine *Beobachtung* durchführt (ausgeführt von einem bestimmten *Verfahren*, das von einem *Stimulus* ausgelöst wird) und ein *Ergebnis* als einen Installationszustand zur Laufzeit zurückgibt

Anmerkung 1 zum Begriff:

- Der Begriff „Sensor“ kann auf `sosa:Sensor` in der SSN-Ontologie abgebildet werden.
- Das Subjekt „Beobachtung“ kann auf `sosa:Observation` in der SSN-Ontologie abgebildet werden.
- Das Subjekt „Stimulus“ kann auf `ssn:Stimulus` in der SSN-Ontologie abgebildet werden.
- Das Subjekt *Verfahren* kann auf `sosa:Procedure` in der SSN-Ontologie abgebildet werden.
- Das Subjekt *Ergebnis* kann auf `sosa:Result` in der SSN-Ontologie abgebildet werden.

3.1.39

Tag

Art von Annotationsbegriff, mit dem die verfügbaren Daten um (in den meisten Fällen) bekannte standardisierte Informationen aus einem Wörterbuch ergänzt werden (im Gegensatz zu einem benutzerdefinierten, selbst gewählten Begriff)

Anmerkung 1 zum Begriff: Ein Tag ist ein konzeptloser Begriff, der nur über eine begrenzte semantische Bedeutung verfügt, wenn er nicht in ein übergeordnetes Konzept wie das Konzept eines (in einer Anwendungsfunktion verwendeten) Datenpunktes eingebettet wird.

BEISPIEL Der Begriff „Durchfluss“ hat für sich genommen eine schwache semantische Bedeutung, wenn er jedoch in einem FOI mit dem Begriff „Wasser“ in Verbindung gesetzt wird, so kommt zumindest zum Ausdruck, dass der Wasserdurchfluss beobachtet/betätigt wird.

In diesem Dokument ist ein Tag beinahe ausnahmslos ein Begriff aus dem semantischen Wörterbuch.

3.1.40

Beschreibung von Dingen

en **thing description**

Modell für semantische Metadaten zur Beschreibung von (abstrakten oder physischen) Dingen entsprechend der „Thing Description“ (TD) <https://www.w3.org/TR/wot-thing-description/> und der Thing Description Ontology <https://www.w3.org/2019/wot/td>

Anmerkung 1 zum Begriff: TD-relevante Relationen werden in dem Abschnitt zum semantischen Export beschrieben.

3.2 Abkürzungen

Für die Anwendung dieses Dokuments gelten die folgenden Abkürzungen.

GA	Gruppenadresse (en: group address)
GO	Gruppenobjekt (en: group object)
FB	Funktionsblock (en: functional block)
FP	Funktionspunkt (en: function point)
ESHG	elektrische Systemtechnik für Heim und Gebäude
IFC	Industry Foundation Class
IOO	Info ein/aus (en: info on/off)
IO	Eingang/Ausgang (en: input/output)
IoT	Internet der Dinge (en: Internet of Things)
KIM	in EN 50090-6-2 beschriebenes und von der KNX Association verwaltetes KNX-Informationsmodell
KNXA	KNX Association
M	verbindlich (en: mandatory)
MaC	Management-Client
OP	Objekteigenschaft (en: object property)
O	optional
TD	Beschreibung von Dingen (en: thing description)

4 ESHG-IoT-Drittanbieter-API

4.1 Einleitung

4.1.1 Abbildung der Begriffe

Das KNX-Informationsmodell bringt Dinge zum Ausdruck, die der in der IoT-Welt verwendeten Wortwahl und Bedeutung entsprechen bzw. diese widerspiegeln.

ANMERKUNG Die ESHG-Terminologie ist vor über 20 Jahren entstanden und wurde von der Community über die Jahre weiterentwickelt. Nun da ESHG-IoT-Installationen den Kunden über eine Schnittstelle nach dem Standard Open API zugänglich gemacht werden, besteht die Notwendigkeit, die Sprache so anzupassen, dass sie von IoT-Entwicklern intuitiv verstanden und verwendet werden kann. Leider ist es in der Kommunikation mit externen Nicht-ESHG-Entwicklern nicht möglich, die historische ESHG-Terminologie einfach unverändert zu übernehmen.

In der folgenden Tabelle 1 werden die Begriffe, die im KNX-Informationsmodell und für die ESHG-IoT-Drittanbieter-API verwendet werden, den ESHG-internen Namenskonventionen gegenübergestellt.

Tabelle 1 – Abbildung der Begriffe

Kunde <i>API-Client-Entwicklung</i>	ESHG-IoT <i>Serverentwicklung</i>	KIM	Klassische ESHG-Ter- minologie <i>Installateur</i>
Function (Funktion)	IoT Function (IoT-Funk- tion)	:ApplicationFunction	Function (Funktion)
Datapoint0F	IoT Datapoint (IoT-Da- tenpunkt)	:FunctionPoint/:Datapoint	Group Address/Group Ob- ject (Gruppenadres- se/Gruppenobjekt)
building/floor/room/site (Gebäude/Stock- werk/Raum/Standort)	building/floor/room/site (Gebäude/Stock- werk/Raum/Standort)	loc:Building/loc:Floor/ loc:Room/loc:Site	Building/Floor/ Room/- (Gebäu- de/Stock- werk/Raum/-)
k. A.	k. A.	:Channel	Channel (Kanal)
Device (Gerät)	Device (Gerät)	:Device	Device (Gerät)

- Die Terminologie des *Kunden* ist relevant für einen ESHG-IoT-Drittanbieter-Client bzw. für einen Entwickler, der einen solchen Client auf Grundlage der ESHG-IoT-Drittanbieter-API einrichtet. Dieser Entwickler kennt sich mit ESHG üblicherweise nicht aus.
- Die ESHG-IoT-Terminologie und die KIM-Terminologie richten sich an den ESHG-Hersteller, der einen ESHG-IoT-Drittanbieter-Server einrichtet, bzw. sind auch für die KNX Association selbst von Belang, die die Ontologie festlegt.
- Mit der *klassischen* ESHG-Terminologie sind Installateure von ESHG-S-Mode-Installationen bestens vertraut.

Für die Serverentwicklung ist es wichtig, zu verstehen, wie sich die Begriffe in obiger Tabelle in technischer Hinsicht unterscheiden; siehe dazu die folgenden Absätze.

Relationen

In der klassischen ESHG-Terminologie finden die Konzepte einer Gruppenadresse und eines Gruppenobjektes Anwendung: eine **Gruppenadresse** verknüpft mehrere **Gruppenobjekte** miteinander (eine **Funktion** fasst mindestens eine **Gruppenadresse** zusammen, einem **Gerät** ist mindestens ein **Gruppenobjekt** zugewiesen).

Sowohl eine GA als auch ein GO werden jedoch als **Datenpunkt/IoT-Datenpunkt** wiedergegeben.

Möglichkeit der Differenzierung zwischen GA und GO auf Ebene eines **Datenpunktes/IoT-Datenpunktes**:

a) Auf Ebene der ESHG-IoT-Server-Entwicklung

Verwendung der entsprechenden KIM-Konzepte für eine GA bzw. ein GO aus der Tabelle oben. Zudem enthält ein Funktionspunkt eine Eigenschaft für den GA-Wert.

b) Auf Ebene der API-Client-Entwicklung

Wenn der Server in seine Antwort die Typinformationen der obigen KIM-Konzepte und/oder die optionale Eigenschaft für die GA aufnimmt, kann anhand dessen unterschieden werden, ob es sich bei dem **Datenpunkt/IoT-Datenpunkt** um eine GA oder einen GO in einer klassischen KNX-Installation handelt.

Für Einzelheiten zum semantischen Ausdruck dieser Relationen wird auf EN 50090-6-2 verwiesen.

4.1.2 Allgemeines

Die ESHG-IoT-Drittanbieter-API muss die folgenden Merkmale unterstützen:

- Erfassung des ESHG-IoT-Drittanbieter-Servers
- Lesen und Schreiben von Endpunkten
- Einrichten von Benachrichtigungen für Endpunkte
- Kontrolle der Zugriffsberechtigungen für Endpunkte

Informationen zur Zuverlässigkeit und Aktualität der Zustandsdaten einer Installation werden in dieser Version der ESHG-IoT-Drittanbieter-API nicht behandelt.

Die ESHG-IoT-Drittanbieter-API hostet auf dem ESHG-IoT-Drittanbieter-Server das Hauptserverprojekt und die Installationskonzepte, wie in Bild 1 dargestellt. In diesem Bild werden die folgenden Aspekte beschrieben:

- mit roten gestrichelten Linien: die verfügbaren Relationen auf der API (dies sind weniger als im System nach EN 50090 oder im KIM beschrieben, siehe Erläuterung zu D unten)
- die Kardinalität der Einheit am Ende jeder Relation (in UML-Notation)

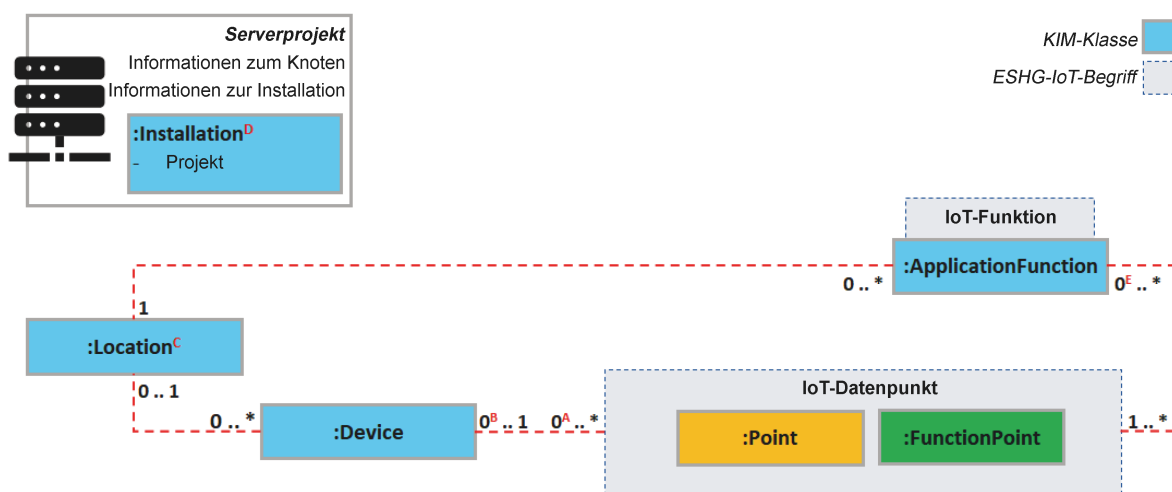


Bild 1 – Projektelemente auf dem ESHG-IoT-Drittanbieter-Server

Die Buchstaben im obigen Bild stehen für Folgendes:

- wenn ein Gerät einen Punkt nicht unterstützt (bloße Hardware wie Stromversorgung, ohne jede Anwendung) oder wenn der IoT-Datenpunkt einen Funktionspunkt repräsentiert
- wenn ein IoT-Datenpunkt einen Funktionspunkt repräsentiert
- umfasst Stockwerke/Räume/... und Weiteres
- Eine Beziehung einer Installation zu einem Ort ist keine explizite Relation in der API; es wird erwartet, dass die Ressourcenpfaddefinitionen der Kategorie „location“ (oder Ähnliches) stets der bekannte und feststehende Zugangspunkt für eine Installation (oder mehrere Installationen) sind. Im Falle mehrerer Installationen wird der gesamte Inhalt zusammengeführt, die API-Endpunkte bieten eine ganzheitliche Perspektive, siehe auch 4.2.5.1.3.
- wenn ein IoT-Datenpunkt einen Punkt repräsentiert

Serverprojekt – Informationen zum Knoten

Enthält Metainformationen zum Knoten, die dem Server zugewiesen sind: geografisch zugewiesene Adresse des Ortes, (Firmware-)Version des Knotens und vieles mehr.

Serverprojekt – Informationen zur Installation

Das dem Server zugewiesene MaC-Projekt in Form des semantischen Exports.

- Funktionen, Datenpunkte, Geräte oder Ortselemente wie Gebäude, Stockwerk oder Raum;
- die Version des MaC-Projekts zur Nachverfolgung von Änderungen der Serverprojektkonfiguration. Zur Identifizierung einer Änderung kann die OP `:lastModified` der `:Installation` verwendet werden, siehe „Installation“ in EN 50090-6-2;
- die Version des ESHG-Informationsmodells (siehe „Einleitung“ in EN 50090-6-2) als Teil des semantischen Exports.

ANMERKUNG Daten und Benutzer werden dem tatsächlichen Serverprojekt auf dem ESHG-IoT-Drittanbieter-Server nach dem Plug-and-play-Prinzip hinzugefügt. Dementsprechend ist das Serverprojekt eine Obermenge/Erweiterung des ursprünglichen MaC-Projekts. Das bedeutet, dass sämtliche zusätzlichen Daten des Serverprojekts von dem ESHG-IoT-Drittanbieter-Server und nicht vom MaC verwaltet werden müssen.

4.1.3 API – Inhalt

Die ESHG-IoT-Drittanbieter-API setzt sich aus Folgendem zusammen:

- Methoden für den Zugriff auf Ressourcen (zum Einstellen oder Abrufen von Zustandsdaten der Installation);
- Endpunkte, auf denen sich Konzepte befinden, die die Laufzeitkommunikation umfassen oder die Konfiguration einer (klassischen ESHG-)Installation repräsentieren, darüber hinaus Endpunkte, die das Einrichten von Benachrichtigungen über Änderungen der Installationszustandsdaten ermöglichen, die an Subscriber gesendet werden, die Clients der Installation sind;
- Methoden zum Filtern bestimmter Ressourceneinheiten aus einer Zusammenstellung von Ressourceneinheiten;
- Methoden zur Autorisierung durch einen ESHG-IoT-Drittanbieter-Client einschließlich einer sicheren Client/Server-Kommunikation;
- Geltungsbereich von Zugriffsberechtigungen: aus Sicherheitsgründen wird der tatsächliche Zugriff auf bestimmte Endpunkte (z. B. das Schreiben eines Datenpunktes) vom ESHG-IoT-Drittanbieter-Server beschränkt; der Zugriff wird im Rahmen der Autorisierung gewährt.

Für keinen der Endpunkte werden die erwarteten Formate von Anforderungs-/Antwortdokumenten und die zugehörigen Inhalte in diesem Dokument vollumfänglich beschrieben. Ob ein Element obligatorisch oder optional ist, kann dem elektronischen Dokument entnommen werden (siehe 4.1.6). Bezüglich obligatorischer und optionaler Endpunkte siehe 4.2.5.

ANMERKUNG Der Standard Open API erlaubt es nicht, mittels eines Statements festzulegen, dass ein Endpunkt optional ist, außer in Form eines schriftlichen Text-Statements.

4.1.4 API – Konformität

Die ESHG-IoT-Drittanbieter-API umfasst den oben beschriebenen Inhalt und setzt sich aus Folgendem zusammen:

- Der Inhalt des elektronischen API-Dokuments wird mit Open API (OAS3) beschrieben.
- Die API-Konzepte werden entsprechend dem JSON-API-Standard beschrieben, bei Abweichungen von JSON API wird in diesem Dokument darauf hingewiesen. Der ESHG-IoT-Drittanbieter-Server muss:
 - jedweden gültigen JSON-API-Inhalt ignorieren, der vom Server nicht unterstützt oder nicht erwartet wird;
 - den JSON-API-Standard einhalten, wenn andere Mitglieder (en: member) unterstützt werden, auch wenn sie in diesem Dokument nicht beschrieben werden.

BEISPIEL JSON-API-Mitglieder der oberen Ebene (en: top level), die in diesem Dokument nicht vorgeschrieben oder beschrieben werden, wie ein `links`- oder ein `included`-Mitglied der oberen Ebene.

BEISPIEL ESHG-Herstellerattribute, die in diesem Dokument nicht definiert werden, als Teil des `attributes`-Mitglieds eines bestimmten Ressourcenobjekts.

BEISPIEL Bereitstellung eines einzelnen JSON-Objekts oder eines Arrays von JSON-Objekten, wenn keine Ressourceneinheiten verfügbar sind.

4.1.5 API – Version

Die ESHG-IoT-Drittanbieter-API muss versioniert werden, um Kompatibilität zwischen Kommunikationspartnern zu ermöglichen; versioniert werden darf jedoch nur die vollständige API und nicht ein Teil von ihr, auch wenn manche Elemente der API von einer Version zur anderen nicht geändert werden.

Die Version dieses Dokuments (siehe Abschnitt zur Dokumentaktualisierung) bezieht sich auf Version 2.0.0 des elektronischen Dokuments (siehe 4.1.6). Für Einzelheiten zur API-Versionierung siehe 4.2.5.1.1.1.

4.1.6 API –Verfügbarkeit

Unter der URL <https://schema.knx.org> kann das jeweils aktuelle elektronische Dokument der ESHG-IoT-Drittanbieter-API abgerufen werden. Ältere Versionen stehen ebenfalls zur Verfügung.

Die vorgenannte URL hat nichts mit der allgemeinen ESHG-IoT-Drittanbieter-API-URL-Struktur zu tun, siehe dazu 4.2.4.

4.1.7 API – Datenformat

Das elektronische Dokument der ESHG-IoT-Drittanbieter-API steht als YAML-Datei zur Verfügung.

4.1.8 API – Format der Beispiele

Zwecks Lesbarkeit werden alle Codebeispiele (Snippets) für die ESHG-IoT-Drittanbieter-API in diesem Dokument im YAML-Format angegeben, unabhängig von ihrem tatsächlichen Format zur Laufzeit.

4.1.9 Hintergrund

Ziel dieses Dokuments ist es, sicherzustellen, dass die ESHG-IoT-Drittanbieter-API:

- von Clients mit grundlegender HTTP-Unterstützung einfach und konsistent „konsumiert“ werden kann;
- einfach und intuitiv nutzbar ist und so einen angenehmen Entwicklungsprozess ermöglicht.

Dienste stellen üblicherweise sprachspezifische Frameworks bereit, mit denen sie ihre APIs umschließen („wrappen“). Die meisten Vorgänge laufen letztlich auf Anforderungen/Antworten mit JSON-Nutzdaten hinaus.

Die meisten modernen APIs verwenden HTTP-Protokollkonzepte, kombiniert mit einigen Konzepten anderer Rechentechnologien.

ANMERKUNG Viele Web-APIs werden über Endpunkte mit Parametern definiert. Endpunkte und Parameter sind keine Begriffe oder Konzepte, die aus HTTP oder REST hervorgegangen sind, sondern wurden aus der Technik „Remote Procedure Call (RPC)“ und verwandten Techniken übernommen.

Eine API sollte den Fokus auf die zugrunde liegenden Entitäten der betrachteten Domäne (en: domain of interest) legen, die sie bereitstellt, anstatt auf eine große Menge von Funktionen, die mit diesen Entitäten arbeiten. So müssen letztlich weniger einzelne Merkmale der API verwaltet werden. In diesem Ansatz kommt der grundlegende Anspruch zum Ausdruck, nur diejenigen Konzepte zu definieren, die tatsächlich notwendig sind, um das Problem zu lösen.

Eine geringere Anzahl von Konzepten erhöht die Konsistenz und ermöglicht es Teams, Code, Muster und Dokumentation gemeinsam zu nutzen und gemeinsame Design-Entscheidungen zu treffen. Mit diesen Leitlinien soll Folgendes erreicht werden:

- Dritte erhalten die Möglichkeit, Dienste in die Produkte von ESHG-Herstellern zu integrieren, die die ESHG-IoT-Drittanbieter-API implementieren.

- Anwendungsentwickler erhalten über REST-Schnittstellen einen einfachen Zugang zu den ESHG-IoT-Diensten.
- Geräteentwicklern wird die Integration der ESHG-IoT-Drittanbieter-API erleichtert.

4.1.10 Ressourcen

4.1.10.1 Allgemeines

Eine Ressource ist das Herzstück eines jeden REST-orientierten API-Designs, sie modelliert und repräsentiert die Domäne der vorgesehenen Anwendung. Durch einen URI wird eine Ressource eindeutig definiert, eine URL definiert eine Methode für den Zugriff auf die Ressource.

Ressourcen müssen im Allgemeinen mit einem URI beschrieben werden.

4.1.10.2 Einheiten, Zusammenstellungen und Relationen

Beim Designen einer API wird üblicherweise zwischen den Ressourcentypen Dokument (en: document), Zusammenstellung (en: collection), Speicher (en: store) und Controller unterschieden. In diesem Dokument liegt der Schwerpunkt auf Dokumenten und Zusammenstellungen.

- Ein Dokument (im Folgenden als „Einheit“ bezeichnet) ist ein einzelnes betrachtetes Objekt (en: object of interest).
- Eine Zusammenstellung ist ein Verzeichnis einzelner Einheiten, die auf einem Server gehostet werden.

Einheiten und Zusammenstellungen (von Einheiten) bilden das erste grundlegende Konzept.

Neben einzelnen Einheiten umfassen die meisten betrachteten Domänen auch konzeptuelle Relationen. Relationen werden in HTTP üblicherweise über Verknüpfungen (en: link) ausgedrückt. Verknüpfungen zeigen an, was vom aktuellen Standort aus erreichbar ist. Sie sind wie Straßenschilder für die Straßen, die das System durchziehen.

Relationen (verknüpfte Daten) bilden das zweite grundlegende Konzept.

Weiter unten sind Beispiele aufgeführt.

4.1.10.3 Ressourcenpfad

Pfade zu Ressourcen sollten die folgende Struktur haben, wobei der eine Pfad dem Abrufen einer einzelnen Einheit einer bestimmten Ressource mittels einer Ressourcen-`{id}` und der andere Pfad dem Abrufen einer Zusammenstellung solcher Einheiten dient.

- Pfad zu einer.../item/{id}
Einheit
- Pfad zu ei-.../item (dem gleichen Prinzip für Ressourcenpfade folgt auch .../item/{id}/
ner Zusammen-item)
stellung

Die einzelnen Segmente eines Pfades müssen mit Bindestrichen voneinander getrennt werden, z. B. .../segment1-segment2/.

4.1.10.4 Ressourcennamen

Eine domänenspezifische Nomenklatur für Ressourcennamen macht es Entwicklern leichter, die Funktionalität und die grundlegende Semantik der Ressourcen zu verstehen. Auch der Dokumentationsaufwand für die API wird durch eine solche Nomenklatur geringer.

Eine Zusammenstellung beinhaltet üblicherweise mehrere Einheiten; die entsprechenden Ressourcennamen sollten im Plural stehen.

`/buildings`

`/buildings/{id}`

Bei der Verarbeitung von URLs wird häufig die Groß-/Kleinschreibung berücksichtigt, weswegen die Kleinschreibung gewählt werden muss.

4.1.10.5 Ressourcen-IDs

Der JSON-API-Standard fordert für die `{id}` ein Zeichenkettenformat. Es wird empfohlen, für die `{id}` das UUID-Format zu verwenden.

Um die Kodierung einer Ressourcen-`id` in URLs zu erleichtern, dürfen für die `{id}` keine reservierten Zeichen nach [RFC 3986], 2.2, verwendet werden.

Wenn die Ressourcen-`{id}` durch einen UUID repräsentiert wird, ist die vorgenannte Regel automatisch erfüllt, da die genannten reservierten Zeichen in einem UUID nicht verwendet werden.

4.1.10.6 Beispiele

Die nachstehenden Beispiele für Ressourcenpfade zeigen jeweils einen Pfad zu einer einzelnen Orts- und einer einzelnen Funktionseinheit sowie einen Pfad zu einer Zusammenstellung von Orten. Die IDs sind dabei als UUID-Ressourcen-`{id}` formatiert.

`/locations/{id}` A specific location item as part of the locations collection in an Installation,

`https://.../locations/3a19fee4-ba8a-4fd2-ba10-42b7f40f5acf`

`/functions/{id}` A specific function item as part of the functions collection in an Installation,

`https://.../functions/7e1418b3-58cb-43dc-a743-1280bc2c6bd7`

`/locations` A collection of all locations in an Installation,

`https://.../locations`

Das nachstehende Ressourcenpfadbeispiel zeigt eine Relation eines Ortes zu den dort enthaltenen Funktionen.

`/locations/{id}/functions` A collection of all function items in a specific location, `https://.../locations/3a19fee4-ba8a-4fd2-ba10-42b7f40f5acf/functions`

4.2 Festlegungen

4.2.1 Anforderungen an Server

Es ist zulässig, übliche Web-Techniken zu verwenden (z. B. zusätzliche HTTP-Header-Tags, TLS-Verschlüsselungssuiten usw.), die über die in diesem Abschnitt sowie in 4.2.5.8 festgelegten Techniken hinausgehen, sofern sie keine der expliziten Protokoll- und Sicherheitsanforderungen verletzen.

4.2.1.1 Protokolle

4.2.1.1.1 IP

Die Kommunikation über die ESHG-IoT-Drittanbieter-API muss sowohl mit IPv4 als auch mit IPv6 möglich sein. Der Implementierer darf die Version nach eigenem Ermessen wählen und auch Dual-Stack-Implementierungen sind möglich.

4.2.1.1.2 HTTP

Ein ESHG-IoT-Drittanbieter-Server muss HTTP/1.1 entsprechend der Dokumentation in den folgenden RFCs unterstützen:

- [RFC 7230] – *Message Syntax and Routing* → allgemeine Zwecke
- [RFC 7231] – *Semantic and Content* → Content-Type-Header

Ein ESHG-IoT-Drittanbieter-Server darf HTTP/1.1 entsprechend der Dokumentation in den folgenden RFCs unterstützen:

- [RFC 7232] – *Conditional Requests* → ETag-Header

HTTP/2 mit [RFC 7540] wird empfohlen, ist aber optional.

Nachstehend werden einige der erforderlichen Elemente aus dem vorgenannten Dokument *Semantic and Content* aufgeführt:

- Verarbeiten verschiedener HTTP-Zugangsmethoden für Ressourcen;
- Verarbeiten verschiedener (URL-)Ressourcenpfade;
- Verarbeiten möglicher Abfrageparameter für (URL-)Ressourcenpfade;
- Verarbeiten möglicher Anforderungs-/Antwort-Header;
- Verarbeiten einer HTTP-Versionsnummer.

Auch wenn in den in diesem Dokument festgelegten Anforderungen auf HTTP/1.1 verwiesen wird, wird ausdrücklich die Annahme zugrunde gelegt, dass diese Anforderungen sowohl für HTTP/2 als auch für HTTP/1.1 gelten.

ANMERKUNG Die weiteren Beispiele in diesem Dokument verwenden HTTP/1.1.

4.2.1.2 Zugangsmethode

Der Zugriff auf Methoden kann mit den standardmäßigen HTTP-Zugangsmethoden für RESTful-Dienste erfolgen.

Zum Abrufen von lesbaren Ressourcen kann eine GET-Anforderung an den entsprechenden Endpunkt gesendet werden. Zum Aktualisieren von schreibbaren Ressourcen kann eine PUT-Anforderung an den entsprechenden Endpunkt gesendet werden. In bestimmten Fällen kann auf eine schreibbare Ressource auch mittels POST-Anforderungen (erstellen), PATCH-Anforderungen (aktualisieren) und DELETE-Anforderungen zugegriffen werden.

Bezüglich der Frage, welche HTTP-Methoden in welchem Anwendungsfall zum Einsatz kommen, wird auf den ESHG-IoT-Drittanbieter-API-Standard in 4.2.5 verwiesen.

Wenn ein Client eine Zugangsmethode verwendet, die für einen bestimmten Endpunkt nicht definiert ist, muss der ESHG-IoT-Drittanbieter-Server mit dem Fehlercode 405 antworten.

4.2.1.3 Header

Der ESHG-IoT-Drittanbieter-Server **muss** die Verarbeitung der folgenden HTTP-Header unterstützen.

- `Accept`
Client-Anforderungs-Header, der einen bestimmten angeforderten `Content-Type` bzw. `media-type` enthält
- `Accept-Charset`
Client-Anforderungs-Header, der einen bestimmten angeforderten `Accept-Charset` enthält

ANMERKUNG Bekannte Browser-Clients lassen diesen Header üblicherweise weg.

Das vorstehend beschriebene Header-Verhalten hängt eng mit der Verarbeitung der Inhaltsaushandlung zusammen, siehe 4.2.1.7.

- `Authorization`
Client-Anforderungs-Header, der Autorisierungsdaten eines bestimmten Benutzers enthält
- `Host`, `Content-Length`, `Date`
Werden für Subscription-POST-Benachrichtigungen verwendet, siehe 4.2.5.6.4.6.

Der ESHG-IoT-Drittanbieter-Server darf die Verarbeitung des folgenden HTTP-Headers unterstützen.

- `ETag`
Server-Antwort-Header entsprechend der Beschreibung in [RFC 7232], 2.3.

ANMERKUNG `ETag` ist ein Zwischenspeichermechanismus, der der Zuverlässigkeit und Skalierbarkeit dient, und ein Indikator für die Aktualität/Gültigkeit von Domänen- bzw. Anwendungsdaten. Er betrifft die Serverantwort, die einen bestimmten Ressourcenbezeichner enthält (Version oder Fingerabdruck).

An bestimmte Endpunkte der ESHG-IoT-Drittanbieter-API gerichtete Clientanforderungen mit anderen Anforderungs-Headern als den vorstehend festgelegten (z. B. `If-None-Match`) liegen außerhalb des Anwendungsbereichs dieses Dokuments.

4.2.1.4 Abfrageparameter

In diesem Dokument kommen die folgenden Parametertypen zum Einsatz:

- Abfrage (Parameter ist Teil der URL, siehe nachstehende Abschnitte)
- Pfad (Parameter ist Teil der URL und wird als Teil des Ressourcenpfades beschrieben, siehe 4.2.5)
- Header (Parameter ist Teil eines HTTP-Headers)

Dieses Dokument beschreibt optionale und obligatorische Abfrageparameter. Die verwendeten Namen der Abfrageparameter entsprechen dem JSON-API-Standard. Andere (anbieterspezifische) Abfrageparameter sind optional.

Wenn eckige Klammern „[]“ und der Doppelpunkt “:” in den Abfrageparameternamen verwendet werden, müssen sie %-kodiert werden. Zur allgemeinen Definition von Namen für Abfrageparameter siehe JSON-API-Standard, Abschnitt „Query Parameters“.

Der ESHG-IoT-Drittanbieter-API-Server muss eine Abfrageparameter-Zeichenkette nach [RFC 3986], 3.4, unterstützen. Die folgenden Abschnitte enthalten Beispiele.

- Bei der Verkettung mehrerer Abfrageparameter muss das Et-Zeichen „&“ als Trennzeichen verwendet werden. Der Server muss alle Abfrageparameter verarbeiten. Die Verkettung darf nicht mit einer logischen Operation für die Parameter verarbeitet werden, siehe dazu 4.2.1.4.1 und 4.2.1.4.2 unten.
- Mehrere Abfrageparameterwerte für denselben Abfrageparameter müssen in Form einer Liste mit dem Komma als Trennzeichen unterstützt werden. Die Liste darf nicht mit einer logischen Operation für die Werte verarbeitet werden, siehe dazu 4.2.1.4.2 unten.

Der Abfrageparameter setzt sich aus einem Parameter**namen** und einem Parameter**wert** zusammen, die durch das Gleichzeichen „=“ getrennt werden. Aus technischer Sicht handelt es sich um ein Schlüssel/Wert-Paar, mit dem nach Ressourcen gesucht wird, die mit dem Schlüssel zusammenhängen und auch mit dem angeforderten (Parameter-)Wert übereinstimmen.

ANMERKUNG Die Servertechnologie zur Definition dieser Relationen liegt außerhalb des Anwendungsbereichs (SQL-Datenbank, Triplestore usw.).

1. Der Abfrageparameter**name** (Schlüssel) ist als Zeichenkette definiert, die eine Menge von Begriffen abdeckt, siehe nachfolgende Abschnitte.
2. Der Abfrageparameter**wert** ist als Zeichenkette definiert, die beliebige Begriffe mit Zeitstempeln und Zahlen abdeckt. Je nach dem dazugehörigen Ressourceneinheitsmitglied muss der Wert unterschiedlich interpretiert werden. Der Parameterwert darf in Anführungszeichen gesetzt werden, die von dem ESHG-IoT-Drittanbieter-Server ignoriert werden müssen.

ANMERKUNG Das `attributes`-Mitglied `value` des Datenpunktes ist ein möglicher Abfrageparameter, siehe auch Beispiele in nachstehenden Abschnitten. Es ist als JSON-Zeichenkette definiert, siehe 4.2.5.4. Ein Abfrageparameterwert von 4.10 wird bei einem zeichenkettenbasierten DPT als Zeichenkette 4.10 und bei einem zahlenbasierten DPT als Zahl 4,1 interpretiert.

Zeichenkette

Zeichenkettenwerte werden mittels %-Dekodierung des gegebenen Wertes interpretiert.

Zahl

Zahlen werden nach JSON, Abschnitt 6, interpretiert.

Zeit

Zeiten werden entsprechend dem API-Referenz-Mitglied „timestamp“ interpretiert.

IRI

IRIs werden ebenso interpretiert, wie oben für **Zeichenkette** beschrieben. Sie decken KIM-Eigenschaften und optional Eigenschaften externer Ontologien ab. Einen vollständigen IRI als Wert zu verwenden, wird nicht empfohlen, dafür sollte das RDF-Namensraum-IRI-Konzept verwendet werden (im Folgenden „Namensraumpräfix“).

Das entsprechende Namensraumpräfix kann in der Clientanforderung weggelassen werden. Der ESHG-IoT-Drittanbieter-API-Server muss stets die KIM-Eigenschaften priorisieren (siehe Beispiele in folgenden Abschnitten).

ANMERKUNG Die Möglichkeit, ein Namensraumpräfix wie `dic-tag:`, `dic:`, `loc:` oder `unit:` wegzulassen, bedeutet, dass der ESHG-IoT-Drittanbieter-API-Server unterschiedliche Prioritäten für die Suche nach entsprechendem Inhalt implementieren muss.

1. in KIM enthaltene Eigenschaften mit oder ohne Namensraumpräfix
2. anbieterspezifische Eigenschaften

Wenn eine anbieterspezifische Eigenschaft oder eine (Nicht-KIM-)Eigenschaft einer externen Ontologie den gleichen Wert hat wie im KIM, muss ein Client dies mit dem entsprechenden Namensraumpräfix angeben.

Andernfalls wird sie von dem Server als die KIM-Eigenschaft behandelt (in diesem Fall erkennt der Server nicht, dass eigentlich ein anderer Abfrageparameter beabsichtigt war).

Für das Serververhalten im Falle zweier KIM-Eigenschaften, deren Namen identisch sind, wenn die beiden Präfixe weggelassen werden, gibt es keine Festlegungen.

Eine für den Benutzer sichtbare Übersetzung des Abfrageparameter**namens** und **-wertes** (einschließlich spezifischer Teile des Namens wie [operator]) muss vom Client gehandhabt werden.

4.2.1.4.1 Paginierung

Einschränkungen unterliegende Server/Clients haben möglicherweise Probleme mit der Verarbeitung großer Mengen von Zusammenstellungseinheiten in einer Antwort. Aus diesem Grund unterstützt die ESHG-IoT-Drittanbieter-API eine seitenbasierte Paginierungsstrategie.

- Ein ESHG-IoT-Drittanbieter-Client darf die erwartete Anzahl von Antworteinheiten aus einer Zusammenstellung mittels eines bestimmten „Zusammenstellungs-Offsets“ oder einer bestimmten Antwortgröße begrenzen, um letztlich alle Einheiten einer Ressourcenzusammenstellung Einheit für Einheit auszulesen.
- Ein ESHG-IoT-Drittanbieter-Server darf die maximale Anzahl von Einheiten in einer Antwort begrenzen. Die Begrenzung darf von den Server(ressourcen)kapazitäten und der erwarteten Anzahl von Einheiten in einer Zusammenstellung (die eng mit dem Umfang eines MaC-Projekts zusammenhängt) abhängig gemacht werden.

Der ESHG-IoT-Drittanbieter-Server muss die folgenden **Abfrageparameter** unterstützen.

- `page [...]` Paginieren der Antwortdaten nach einem bestimmten Kriterium, für Werte von „...“ siehe unten

Wenn in einer einzelnen Anforderung verschiedene (Paginierungs-)Abfrageparameter aneinandergereiht werden, müssen diese alle einzeln erfüllt werden, die Parameter werden nicht über ein logisches Und verbunden.

4.2.1.4.1.1 `page[number]`

Der Parameter überspringt mehrere Seiten, bevor er mit dem Zusammenstellen des Ergebnis-Einheitensatzes beginnt. Dies darf nur einmal je Anforderung auftreten, andernfalls muss der ESHG-IoT-Drittanbieter-API-Server einen Fehler ausgeben (siehe 4.2.1.4.4).

Dieser Filter wird auf zurückgegebene Ressourcenzusammenstellungen der Ressourcenpfadkategorie **function**, **location**, **datapoint**, **device**, **information**, **subscription** oder **timeseries** angewendet; siehe dazu das elektronische Dokument (siehe 4.1.6).

- Der Parameter**name** wird mit `page[number]` festgelegt.
- Der Parameter**wert** ist eine Zahl.

Der Standard-Parameterwert für den ESHG-IoT-Drittanbieter-Server muss null sein.

BEISPIEL Eine Clientanforderung ohne einen Abfrageparameter **page[number]** wäre nicht in der Lage, alle Einheiten zu lesen, die gegebenenfalls vor dem Standardwert stehen, wenn dieser nicht auf null gesetzt ist.

Wenn ein Abfrageparameter in der Clientanforderung, mit der auf eine Zusammenstellung zugegriffen wird, nicht gesetzt wurde, muss der ESHG-IoT-Drittanbieter-Server mit dem Standard-Parameterwert (entsprechend obiger Festlegung) als Teil des `collection`-Mitglieds (siehe 4.2.3.2.1) antworten.

Der Serverwert für den Umfang einer Seite als solcher ist anbieterspezifisch.

ANMERKUNG Ansprechen einer Zusammenstellungseinheit (als nullbasierter Offset) → Zahl = Offset/Seitenumfang

4.2.1.4.1.2 page[size]

Der Parameter definiert die maximale Anzahl von Einheiten, die in der Antwort zurückgegeben sind. Dies darf nur einmal je Anforderung auftreten, andernfalls muss der ESHG-IoT-Drittanbieter-API-Server einen Fehler ausgeben (siehe 4.2.1.4.4).

Dieser Filter wird auf zurückgegebene Ressourcenzusammenstellungen der Ressourcenpfadkategorie **function**, **location**, **datapoint**, **device**, **information**, **subscription** oder **timeseries** angewendet; siehe dazu das elektronische Dokument (siehe 4.1.6).

- Der Parametername wird mit *page[size]* festgelegt.
- Der Parameterwert ist eine Zahl.

Der Standard-Parameterwert für den ESHG-IoT-Drittanbieter-Server muss auf einen Wert größer null eingestellt werden, der tatsächliche Wert ist anbieterspezifisch.

ANMERKUNG Bei der Einstellung auf null würde eine Clientanforderung ohne den Abfrageparameter *page[size]* eine leere Liste zurückgeben.

Wenn ein Abfrageparameter in der Clientanforderung, mit der auf eine Zusammenstellung zugegriffen wird, nicht gesetzt wurde, muss der ESHG-IoT-Drittanbieter-Server mit dem tatsächlichen Parameterwert (unter Berücksichtigung des Standard-Parameterwertes) als Teil des *collection*-Mitglieds (siehe 4.2.3.2.1) antworten.

ANMERKUNG Der tatsächliche (Umfangs-)Parameterwert in einer Antwort darf kleiner sein als der Standard-(Umfangs-)Parameterwert, wenn die letzte Seite nicht genügend Einheiten enthält. Wenn der Standard-(Umfangs-)Parameterwert größer ist als die Anzahl von Einheiten in einer Zusammenstellung, enthält die Antwort die gesamte Zusammenstellung.

BEISPIEL Eine (%-kodierte) GET-Anforderung ruft alle Ortseinheiten aus der */locations*-Zusammenstellung ab, beginnend mit *page[number]* „20“, und gibt die nachstehenden Zusammenstellungsinformationen zurück.

Anforderung

```
GET /locations?page%5Bnumber%5D%3D20 HTTP/1.1
Accept: application/vnd.api+json
```

Antwort

```
meta:
  collection: // a response with collection information, see 4.2.3.2.1
  number: 20 // client requested to skip 20 pages, return from page #21
  size: 10 // no size limitation from client, use server default value (10)
  total: 400 // a client may derive from the value above a server page size
10
  data: [{...}]
```

4.2.1.4.2 Filter

Ein Client darf nach bestimmten Einheiteneigenschaften in Zusammenstellungen filtern. Zu diesem Zweck unterstützt der ESHG-IoT-Drittanbieter-API-Server eine Filterstrategie.

- Ein ESHG-IoT-Drittanbieter-Client möchte die Ressourceneinheiten in einer Antwort derart begrenzen, dass nur Einheiten angezeigt werden, die einem bestimmten Suchkriterium des Benutzers entsprechen; alle anderen Einheiten der Zusammenstellung sind für den Client nicht von Belang.

Der ESHG-IoT-Drittanbieter-Server darf die folgenden **Abfrageparameter** unterstützen.

- *filter[...][operator]* Filtern von Ressourcen entsprechend einem gegebenen Filterkriterium und Operator; hinsichtlich der Optionen für „...“ und *operator* siehe nachfolgende Abschnitte

Der Filter wird auf zurückgegebene Ressourcenzusammenstellungen der Ressourcenpfadkategorie **function**, **location**, **datapoint**, **timeseries** oder **device** angewendet; siehe dazu das elektronische Dokument (siehe 4.1.6).

Wenn in einer einzelnen Anforderung verschiedene (Filter-)Abfrageparameter aneinandergereiht werden, müssen diese alle einzeln erfüllt werden, die Parameter werden über ein logisches Und verbunden.

ANMERKUNG Ein Filterung mit einem logischen Oder kann mittels mehrerer Anforderungen erreicht werden.

4.2.1.4.2.1 filter[...][operator]

Der *operator*-Teil des Parameternamens filtert nach bestimmten Ressourceneinheiten, die Teil einer Zusammenstellung sind und mit einer logischen/Vergleichsoperation zusammenhängen, z. B. alle *datapoints* mit einem Temperatur(attribut)wert unter 30 °C.

— Der Parametername *filter[...][operator]* wird für den *operator*-Teil wie folgt festgelegt:

Tabelle 2 – Festlegung des Parameternamensfilters für den operator-Teil

Op.	Definition	Wahrheitsregel	Abfrageparameter-Werteliste
eq	gleich	Ressourceneinheit-Mitgliedswert = Abfrageparameterwert	nicht zulässig, siehe Fehlerbehandlung
le	kleiner oder gleich	Ressourceneinheit-Mitgliedswert = < Abfrageparameterwert	nicht zulässig, siehe Fehlerbehandlung
ge	größer oder gleich	Ressourceneinheit-Mitgliedswert = > Abfrageparameterwert	nicht zulässig, siehe Fehlerbehandlung
lt	kleiner als	Ressourceneinheit-Mitgliedswert < Abfrageparameterwert	nicht zulässig, siehe Fehlerbehandlung
gt	größer als	Ressourceneinheit-Mitgliedswert > Abfrageparameterwert	nicht zulässig, siehe Fehlerbehandlung
and	logisches Und	alle angeforderten Abfrageparameterwerte sind in einer Liste eines Ressourceneinheitsmitglieds enthalten ^a	zulässig
or	logisches Oder	mindestens ein angeforderter Abfrageparameterwert ist in einer Liste eines Ressourceneinheitsmitglieds enthalten/stimmt mit dem Wert eines Ressourceneinheitsmitglieds überein	zulässig
^a Anmerkung für Implementierer: Dies ist ebenfalls falsch, wenn ein Ressourceneinheitsmitglied mit einem (einzelnen) Wert oder einer Liste, die nicht ausreichend vergleichbare Elemente enthält, angesprochen wird.			

Die Reihenfolge für den Vergleichsoperator **le**, **ge**, **lt**, **gt**, **eq** ist wie folgt definiert:

1. chronologisch, wenn beide Werte als Zeit formatiert sind (siehe **Zeit** in 4.2.1.4);
2. numerisch, wenn beide Werte als Zahl formatiert sind (siehe **Zahl** in 4.2.1.4);
3. andernfalls lexikografisch.

Zu den Punkten 1 und 2 oben: Mit Ausnahme des Datenpunktwertes des attribute-Mitglieds „value“ muss hier stattdessen der entsprechende DPT verwendet werden (z. B. die Zeichenkette „4.10“ bei einem Zeichenketten-DPT oder die Gleitkommazahl „1.3E-3“ bei einem Gleitkomma-DPT).

Der Parameternamensteil *operator* ist optional; wenn er nicht vorhanden ist, wird dies von dem Server als der Standardoperator „eq“ interpretiert.

4. Der Parameterwert wird in den nachfolgenden Abschnitten für jeden [...] -Teil festgelegt.

BEISPIEL Logische Filterausdrücke in verschiedenen Varianten.

```
1. filter[value][le]=30
   // datapoint (runtime) attributes member value shall be lower/equal than 30
2. filter[timestamp][gt]=2021-02-17T17:17:13Z
   // datapoint shall be last updated strictly after 2021-02-17, 5:17:13PM
3. filter[state][or]=Tested,Accepted
   // location attributes member state shall be "Tested" or "Accepted"
4. filter[enum][and]=on,off
   // datapoint attributes member enum shall contain "on" and "off"
   // → this is a valid combination for the DPT 1.001
5. filter[unit][and]=DEG_C,DEG_F
   // datapoint attributes member unit shall contain "DEG_C" and "DEG_F"
   // → such a DPT does not exist
6. filter[unit]=DEG_C&filter[value][lt]=30
   // datapoint attributes member unit shall be "DEG_C" and the value shall be
   lower than 30
   // several and'ed parameters, each of them shall be handled and met
   individually
```

4.2.1.4.2.2 filter[meta.@type][operator]

Der Parameter filtert nach bestimmten Ressourceneinheiten, die Teil einer Zusammenstellung sind und mit einem bestimmten Ressourcentyp zusammenhängen, z. B. alle *datapoints* mit dem zusätzlichen Typ loc:Floor.

ANMERKUNG Die zusätzliche Ressourcentypdefinition wird in einer Antwort als Teil des meta-Mitglieds angegeben, so dass kein Konflikt mit anderen Mitgliedern wie der allgemeinen Typdefinition innerhalb des data-Mitglieds entstehen kann.

- Der Parameter **name** wird mit *filter[meta.@type]* festgelegt, der *operator*-Teil nach 4.2.1.4.2.1.
- Der Parameter **wert** wird mit einem KIM-Klassennamen oder (sofern verfügbar) mit dem dieser Klasse zugewiesenen URN festgelegt.

Die KIM-Klassen für Datenpunkt-, Geräte- und Funktionsressourcen, deren entsprechende Namensraumpräfixe und die tatsächlichen Klassennamen können EN 50090-6-2 bzw. der elektronischen Datei entnommen werden.

Die KIM-Klassen für Ortsressourcen, deren entsprechende Namensraumpräfixe und die tatsächlichen Klassennamen können Tabelle 12 entnommen werden.

ANMERKUNG Der KIM-Klassename ist das, was von dem IRI übrig bleibt, wenn das entsprechende Namensraumpräfix gestrichen wird, z. B. loc: von <http://schema.knx.org/2020/ontology/loc#Floor> → Floor.

ANMERKUNG „Vererbungen“ (zwischen über- und untergeordneten Ressourcen) werden beim Filtern der in Tabelle 12 definierten Ortsressourcen nicht berücksichtigt. Beispielsweise werden beim Filtern nach allen Gebäuden keine Räume zurückgegeben.

BEISPIEL Typfilterausdrücke in verschiedenen Varianten.

```
— filter[meta.@type]=Floor           // KIM location type without
                                     namespace
— filter[meta.@type]=loc:Floor        // KIM location type with namespace
— filter[meta.@type][or]=Floor,Space  // locations either a Floor or Space
— filter[meta.@type]=Floor,Space      // two parameter values, may cause a
                                     422
```

```

— filter[meta.@type][lt]=Floor // types lexicographically lower than

                                // string Floor (may not be very
                                // useful)

— filter[meta.@type]=dpa.321.51 // KIM datapoint type without
                                // namespace

— filter[meta.@type]=dic:dpa.321.51 // KIM datapoint type with namespace

— filter[meta.@type]=urn:knx:dpa.321.51 // KIM datapoint type as URN

— filter[meta.@type]=td:Thing // external element with namespace

— >filter[meta.@type]=oms:HeatMeter // external element with namespace

— filter[meta.@type]=switchinCtrl // KIM app. function without
                                // namespace

— filter[meta.@type]=urn:knx:fct.switchingCtrl // KIM app. function as URN

```

4.2.1.4.2.3 filter[tagKey][operator]

Der Parameter filtert nach bestimmten Ressourceneinheiten, die Teil einer Zusammenstellung sind und mit bestimmten Tags zusammenhängen, z. B. ein Datenpunkt, der mit dem (Phänomen-)Tag *temperature* zusammenhängt, ein Gebäude, das mit dem (Ortsverwendungs-)Tag *airport* zusammenhängt, oder eine Anwendungsfunktion, die mit dem (Sparten-)Tag *HVAC* zusammenhängt.

- Der Parametername *filter[tagKey]* wird für den *operator*-Teil nach 4.2.1.4.2.1 und für den *tagKey*-Teil mit einer Struktur der KIM-Objekteigenschafts-Relationsprädikate wie folgt festgelegt:

1. *hasTagSet.hasTag*

Die Struktur drückt eine Kombination zweier KIM-OPs aus, die Tags ansprechen, die **indirekt** einer Quelle zugeordnet sind (z. B. ein Ort oder ein Datenpunkt). Die Struktur muss mit den folgenden OPs verwendet werden:

Tabelle 3 – Parametername, der zwei KIM-Objekteigenschaften enthält

Übergeordnete OP	hasTagSet	hasTag	Bemerkung
untergeordnete OP	<ul style="list-style-type: none"> — hasQualityKind — actsOnQualityKind — observesQualityKind — adjustsQualityKind — measuresQualityKind 	<ul style="list-style-type: none"> — hasEquipmentType — hasPhenomenonType — hasLocality — hasQuantityKind 	spricht eine Eigenschaftsart (QK, en: quality kind) an
untergeordnete OP	hasProcessKind	<ul style="list-style-type: none"> — hasEquipmentType — hasPhenomenonType — hasVirtualEntityType — hasLocality — hasStateKind 	spricht eine Prozessart (PK, en: process kind) an
untergeordnete OP	keine untergeordneten Eigenschaften definiert	<ul style="list-style-type: none"> — hasOrigin — hasPointOperation — hasPointFunctionType 	spricht eine Betriebsart (en: operation kind) an

ANMERKUNG 1 Eine QK/PK oder Betriebsart kann einer Quelle mehrmals zugewiesen worden sein, jede der obigen Strukturen muss jedoch einzeln betrachtet werden, siehe dazu die nachstehenden Beispiele.

Die Struktur darf nicht in einzelne Teile aufgeteilt werden, um beispielsweise QKs/PKs direkt mit der ersten OP anzusprechen.

ANMERKUNG 2 Um QKs/PKs direkt anzusprechen, müsste eine strukturierte Benennungskonvention für sie eingeführt werden. Ein Parameterwert mit einem solchen Namen würde keine(n) aussagekräftige(n) Bereich/Struktur zum Filtern einer bestimmten QK/PK angeben; von Interesse sind die Tags als solche und nicht die Namen der Container, die die Tags enthalten.

2. hasTag

Die Struktur drückt eine einzelne KIM-OP aus, die Tags anspricht, die direkt einer Quelle zugeordnet sind (z. B. ein Ort oder ein Datenpunkt). Die Struktur muss mit den folgenden OPs verwendet werden:

Tabelle 4 – Parametername, der eine KIM-Objekteigenschaft enthält

Übergeordnete OP	hasTag
untergeordnete OP	hasPointInterface operatesForTrade operatesOnProtocol hasLocationUsage

ANMERKUNG 3 Wie vorstehend beschrieben, dürfen manche Tags einer Quelle nur indirekt zugeordnet werden, das Filterergebnis entspricht also möglicherweise nicht den Erwartungen, wenn untergeordnete Eigenschaften von :hasTag verwendet werden, die in Tabelle 4 nicht aufgeführt sind.

Der Bereich der beschriebenen übergeordneten OPs erstreckt sich auf alle möglichen Tag-Individuen der KIM-Klasse :Tag. Wenn sie verwendet werden, werden alle Tags gefiltert und in das Ergebnis aufgenommen. Um die Ergebnisse zu begrenzen, müssen die zugehörigen untergeordneten OPs verwendet werden, vorrangig von hasTag, aber auch von hasTagSet (oder einer Kombination).

ANMERKUNG 4 Der obigen Aussage liegt die Annahme zugrunde, dass die Serverdaten untergeordnete Eigenschaften enthalten müssen, andernfalls ist das zurückgegebene Ergebnis stets leer.

Wenn eine gewählte Kombination aus Parametername und -wert inhaltlich oder logisch nicht zusammenpasst, ist der Server möglicherweise nicht in der Lage, ein passendes Ergebnis abzurufen/zu berechnen; siehe dazu den Abschnitt zur Fehlerbehandlung unten.

— Der Parameterwert wird mit dem Namen eines KIM-Individuums der Klasse :Tag festgelegt.

Die KIM-Individuen für die Tag-Ressourcen, deren entsprechende Namensraumpräfixe und die tatsächlichen Individuennamen können EN 50090-6-2 bzw. der elektronischen Datei entnommen werden.

Einem Client müssen die möglichen Filterwerte bekannt sein. Üblicherweise erfährt ein Client diese in den Antworten. Deshalb wird dringend empfohlen, dass eine Antwort die einzelnen Tags und/oder die QKs/PKs als vollständige Ressourcenobjekte entsprechend der Beschreibung in 4.2.3.3.5 enthält. Andere Optionen – beispielsweise sie im Clientcode vorzuprogrammieren oder sie extern (z. B. aus der Ontologie) auszulesen – bieten weniger Transparenz.

ANMERKUNG 5 Der KIM-Individuenname ist das, was von dem IRI übrig bleibt, wenn das entsprechende Namensraumpräfix gestrichen wird, z. B. dic-tag: von <http://schema.knx.org/2020/ontology/dictionaries/dic-tag#actuator> → actuator.

BEISPIEL Filterausdrücke in verschiedenen Varianten.

```

— filter[actsOnQualityKind.hasEquipmentType]=actuator
  // Source: Equipment, Datapoint (Device Catalog/Instance Data), Location,
  FP
  // parameter name addresses a source with indirectly assigned tags as
  part of a QK
  // query parameter with one value; filter all rooms with a QK with
  equipment type actuator

— filter[hasLocationUsage]=airport
  // Source: Location (Instance Data)
  // parameter name addresses a Location with directly assigned tags (with
  a parent OP)

— filter[hasLocationUsage][or]=pool,wellness
  // Source: Location (Instance Data)
  // parameter name addresses a Location with directly assigned tags (with
  a child property)
  // query parameter with two values; filter all rooms with location usage
  pool or wellness

— filter[hasTag]=switch
  // Source: Datapoint (Device Catalog/Instance Data), Location, FP,
  Application Function
  // parameter name addresses a source with directly assigned tags (with a
  parent OP)

— filter[hasPointInterface][and]=if.p,if.i

  // Source: Datapoint (Device Catalog/Instance Data)

  // parameter name addresses a Datapoint with directly assigned tags (with a
  child OP)

```

BEISPIEL Filterausdrücke mit verschiedenen Abfrageparameterstrukturen, die QKs enthalten. Um die Filterergebnisse zu verstehen, wird die folgende Beispieleinrichtung angenommen.

Datenpunkt A	Datenpunkt B	Datenpunkt C
QK1: Ort: <i>wire</i>	QK1: Ort: <i>wire</i>	QK1: Ort: <i>wire, site</i>
QK2: Ort: <i>site</i>		

```

— /datapoints?filter[hasQualityKind.hasLocality][or]=wire,site
  // parameter name addresses a Datapoint with indirectly assigned tags as
  part of a QK
  // succeeds for Datapoint A,B,C → at least one assigned QK has one tag
  included

— /datapoints?filter[hasQualityKind.hasLocality][and]=wire,site
  // parameter name addresses a Datapoint with indirectly assigned tags as
  part of a QK
  // succeeds for Datapoint C → at least one assigned QK has both tags
  included

— /datapoints?filter[hasQualityKind.hasLocality]=wire&filter[hasQualityKind.
  hasLocality]=site
  // each parameter name addresses a Datapoint with indirectly assigned
  tags as part of a QK
  // succeeds for Datapoint A,C → each given query parameter definition
  needs to be
  // resolved individually to generate the overall result

```

4.2.1.4.2.4 filter[attributeKey][operator]

Der Parameter filtert nach bestimmten Ressourceneinheiten, die Teil einer Zusammenstellung sind und mit bestimmten Ressourcenattributen zusammenhängen, z. B. alle Geräte mit der Bestellnummer (dem Attribut) *5WG100-200-AB*.

- Der Parametername *filter[attributeKey]* wird für den *operator*-Teil nach 4.2.1.4.2.1 und für den *attributeKey*-Teil mit den *attributes*-Mitgliedern entsprechend der Definition in 4.2.3.3.1.1 festgelegt. Anbieterspezifische Attribute sind ebenfalls möglich, siehe Beispiel unten.

Mitglieder von *attributes*, die als eine Kombination zusammen mit (Unter-)Eigenschaften definiert sind, müssen eine Verkettung aller einzelnen Individuen, getrennt durch einen Punkt, sein. Diese Methode ist die gleiche wie im vorangehenden Abschnitt für QKs/PKs definiert. Für ein weiteres Beispiel siehe unten.

- Der Parameterwert wird bezogen auf die verwendeten *attributes*-Mitglieder wie folgt festgelegt:
 1. MaC-Projekt(-Zeichenketten)-Begriffe;
 2. Zeitstempel(-Zeit)-Werte;
 3. Datenpunkt(-Zeichenketten-/Zahlen)-Werte;
 4. KIM-Individuen(-IRI)-Namen.

Zur Interpretation des Wertes siehe 4.2.1.4.

Die KIM-Individuen für die entsprechenden Ressourcen, deren entsprechende Namensraumpräfixe und die tatsächlichen Individuennamen können EN 50090-6-2 bzw. der elektronischen Datei entnommen werden.

ANMERKUNG Der KIM-Individuennamen ist das, was von dem IRI übrig bleibt, wenn das entsprechende Namensraumpräfix gestrichen wird, z. B. unit: von https://qudt.org/vocab/unit/DEG_C → DEG_C.

BEISPIEL Filter mit einigen *attributes*-Mitgliedern.

1. `filter[orderNumber]=5WG100-200-AB`
// device order number (*attributes* member orderNumber) is queried with *5WG100-200-AB*
2. `filter[gira:deviceUpdateCycle]=immediately`
// device (KNX manufacturer member deviceUpdateCycle) is queried with *immediately*// the namespace prefix separator "": shall also be % encoded
3. `filter[unit]=DEG_C`
// datapoint unit (attributes member unit) is queried with *DEG_C*// the namespace prefix unit: is not needed (property is part of the KIM)
4. `filter[version.hardware]=2.0`
// device hardware version (attributes member version) is queried with (string) *2.0*

BEISPIEL Filter mit *attributes*-Mitglied **timestamp** nach Aktualisierungen eines einzelnen Datenpunktes (.../timeseries).

1. `filter[timestamp][le]=2021-02-17T17:17:17Z`
// filter all updates, occurred before...
2. `filter[timestamp][ge]=2021-02-17T17:17:13Z`
// filter all updates, occurred after...
3. `filter[timestamp][ge]=2021-02-17T17:17:13Z&filter[timestamp][le]=2021-02-17T17:17:17Z`
// filter all updates, occurred between...

BEISPIEL Filter mit *attributes*-Mitglied **timestamp** nach Aktualisierungen aller Datenpunkte (.../datapoints).

E DIN EN 50090-6-3:2024-08
EN 50090-6-3:2023

1. `filter[timestamp][le]=2021-02-17T17:17:17Z`
// filter where the last update, occurred before...
2. `filter[timestamp][ge]=2021-02-17T17:17:13Z`
// filter where the last update, occurred after...
3. `filter[timestamp][ge]=2021-02-17T17:17:13Z&filter[timestamp][le]=2021-02-17T17:17:17Z`
// filter where the last update, occurred between...

4.2.1.4.3 Informationsabfragen

Dieser Abschnitt enthält einige (vereinfachte) HTTP-Abfragen (Fragen), die (unter Verwendung von Filtern) an eine Installation gerichtet werden. In allen nachstehenden Beispielen wird von der Möglichkeit Gebrauch gemacht, KIM-spezifische Namensräume für den Parameternamen und -wert wegzulassen.

1. Welche Orte (Räume) im Gebäude werden als Küche oder Bad verwendet?

```
GET      /locations?filter[hasLocationUsage][or]=kitchen, bath
RESPONSE: Zusammenstellung von Orten
```

2. Wie lauten die Sollwerte und die aktuelle Temperatur eines bestimmten Ortes?

Schritt 1

```
GET      /locations/{locationId}/functions?
        filter[meta.@type]=temperatureAir
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionId}/datapoints?filter[hasTagSet.
        hasPointFunctionType][or]=sp,processValue
RESPONSE: Zusammenstellung von Datenpunkten
```

3. Wie lautet die aktuelle Betriebsart des Primärsystems?

Schritt 1

```
GET      /functions?
         filter[meta.@type]=generationCtrl&filter[operatesForTrade]=heating
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionsId}/datapoints?filter[hasTagSet.
         hasStateKind]=controlMode
RESPONSE: Zusammenstellung von Datenpunkten
```

4. Wie lautet die aktuelle HLK-Betriebsart eines bestimmten Ortes?

Schritt 1

```
GET      /locations/{locationId}/functions?filter[meta.@type]=airCtrl&
         filter[operatesForTrade][or]=heating,ventilation,cooling
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionsId}/datapoints?filter[hasQualityKind.
         hasPointOperation]=enumeration
RESPONSE: Zusammenstellung von Datenpunkten
```

5. Welche Heizzone verfügt über Luftbehandlungsgeräte?

Schritt 1

```
GET      /functions?
        filter[meta.@type]=emissionCtrl&filter[operatesForTrade]=heating
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionsId}/datapoints&filter[hasQualityKind.
        hasEquipmentType]=ahu
RESPONSE: Zusammenstellung von Datenpunkten
```

6. In einem Raum ist es kalt. Wo kann ich den Sollwert ändern?

Schritt 1

```
GET      /locations/{locationId}/functions?filter[meta.@type]=emissionCtrl
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionsId}/datapoints?
        filter[hasQualityKind.hasPhenomenType]=air
        &filter[hasQualityKind.hasQuantityKind]=Temperature&filter[hasTagSet.
        hasPointFunctionType]=sp
RESPONSE: Zusammenstellung von Datenpunkten
```

7. Alle historischen Datenpunktwerte vor einem gegebenen Zeitpunkt auslesen.

```
GET      /datapoints/{datapointsId}/timeseries?filter[timestamp]
        [le]=2019-04-22T14:07:17Z
RESPONSE: Zeitreiheneinheiten-Zusammenstellung für diesen Datenpunkt
```

8. Alle historischen Datenpunktwerte ab einem gegebenen Zeitpunkt auslesen.

```
GET      /datapoints/{datapointsId}/timeseries?filter[timestamp]
        [ge]=2019-04-22T14:07:17Z
RESPONSE: #x0009;#x0009;Zeitreiheneinheiten-Zusammenstellung für diesen
        Datenpunkt
```

9. Alle historischen Datenpunktwerte für eine gegebene Zeitspanne auslesen.

```
GET      /datapoints/{datapointsId}/timeseries?filter[timestamp]
        [ge]=2019-04-22T14:07:17Z&
        filter[timestamp][le]=2021-04-22T14:07:17Z
RESPONSE: Zeitreiheneinheiten-Zusammenstellung für diesen Datenpunkt
```

10. Alle Datenpunktwerte eines bestimmten Wärmeverteilungssystems (Heizkreises) auslesen, das der Energiezählung (zur Zusammenfassung des Energieverbrauchs) dient.

Schritt 1

```
GET      - id: 8951704b-269b-34f9-9a8f-4e415c5723e2
RESPONSE: Zusammenstellung von Anwendungsfunktionen
```

Schritt 2

```
GET      /functions/{functionsId}/datapoints?
        filter[operatesForTrade]=metering
RESPONSE: datapoint collection data:
        - id: 8951704b-269b-34f9-9a8f-4e415c5723e1 type: datapoint
          meta:
            "@type": [dic:dpa.1103.51, dic:meteringValue, oms:VM1]
          attributes:
            title: "[M-BUS INTEGRATED] Total volume"
            readable: true
            writable: false
            unit: unit:M3
            valueType: string
            value: "10.055"
            timestamp: 2021-09-08T11:01:44+02:00
            KNX:operatesForTrade: dic-tag:metering
            KNX:operatesOnProtocol: dic-tag:mbus
          relationships:
            datapointDevice:
              links:
                related: /
              datapoints/8951704b-269b-34f9-9a8f-4e415c5723e1/device
            - id: 8951704b-269b-34f9-9a8f-4e415c5723e2
            ...
```

4.2.1.4.4 Fehlerbehandlung

Die vorstehend beschriebenen Abfrageparameter können in einer Anforderung unabhängig voneinander verwendet werden. Zudem darf in einer einzelnen Anforderung auch mehr als ein Abfrageparameter verwendet werden.

Es ist möglich, dass Abfrageparameter für einen bestimmten Serverendpunkt nicht anwendbar sind oder überhaupt nicht für einen Endpunkt festgelegt sind (für Einzelheiten zu der Frage, auf welchen Ressourcenpfad welche Abfrageparameter angewendet werden können, wird auf das elektronische Dokument in 4.1.6 verwiesen). Würde ein Server unerwartete Abfrageparameter oder deren Werte in solchen Situationen schlicht

ignorieren (und mit etwaigen nachfolgenden gültigen Parametern fortfahren), so wären die Antwortdaten für einen Client nicht vorhersehbar.

Der ESHG-IoT-Drittanbieter-Server:

- muss den Antwortcode 400 (sowie optional ein Fehlerobjekt) zurückgeben, wenn die Benennungsregel für Abfrageparameter verletzt wurde (siehe JSON-API-Standard, Abschnitt „Query Parameters“).
- muss das Abfrageergebnis als falsch behandeln, wenn ein ESHG-IoT-Drittanbieter-Client in seiner Anforderung einen (syntaktisch korrekten) Abfrageparameter angibt, dessen Parameterwert für den Server nicht festgelegt oder nicht anwendbar ist.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `filter[meta.@type]=Kitchen`, die an einen Endpunkt gerichtet ist, der eine Zusammenstellung von Orten zurückgibt; der Wert „Kitchen“ ist kein im KIM definierter Ortstyp.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `filter[meta.@type]=Datapoint`, die an einen Endpunkt gerichtet ist, der eine Zusammenstellung von Orten zurückgibt; der Wert „Datapoint“ als Ressourcentyp ist für eine Ortsressource nicht anwendbar.

Im Ergebnis sind die Nutzdaten der Antwort für die Ressource leer. Das Ergebnisverhalten ist das gleiche wie in dem Standardfall einer Abfrage mit einem festgelegten/anwendbaren Abfrageparameterwert, die sich an eine Zusammenstellung richtet, die derzeit keine solche „übereinstimmenden“ Ressourceneinheiten enthält.

- muss den Antwortcode 422 (und optional ein Fehlerobjekt) zurückgeben, wenn der ESHG-IoT-Drittanbieter-Client in seiner Anforderung an einen Ressourcenpfad einen (dem RFC nach syntaktisch korrekten) Abfrageparameter angibt, der ein Individuum mit reservierten Abfrageparametern `page[...]` oder `filter[...]` anspricht.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `filter[hasPointInterface]=if.i`, die an einen Endpunkt gerichtet ist, der ein Datenpunktindividuum zurückgibt; der Abfrageparameter „filter[...]“ ist für ein Individuum nicht zulässig.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `page[number]=2&page[size]=3`, die an einen Endpunkt gerichtet ist, der ein Ortsindividuum zurückgibt; der Abfrageparameter „page[...]“ ist für ein Individuum nicht zulässig.

- darf den Antwortcode 422 (und optional ein Fehlerobjekt) zurückgeben, wenn ein ESHG-IoT-Drittanbieter-Client in seiner Anforderung einen (dem RFC nach syntaktisch korrekten) Abfrageparameter mit einer für den Server unbekannten Abfrageparameterdefinition angibt.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `filter[value][and]=10,12`, die an einen Endpunkt gerichtet ist, der eine Zusammenstellung von Datenpunktwerten zurückgibt; dies ist ein gültiger Abfrageparameterausdruck für eine Zusammenstellung, die aus **Anwendungssicht** jedoch keinen Sinn ergibt (der tatsächliche Wert eines Datenpunktes kann zu einem gegebenen Zeitpunkt stets nur einen Wert haben).

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `filter[value][or]=10` oder `filter[value][and]=12` oder `filter[value][eq]=10,12` oder `filter[value]=10,12`, die an einen Endpunkt gerichtet ist, der eine Zusammenstellung zurückgibt; dies ist ein **ungültiger** Abfrageparameterausdruck für eine Zusammenstellung.

BEISPIEL Eine Clientanforderung mit dem Abfrageparameter `foo=bar`; dies ist ein unbekannter Abfrageparameterausdruck.

4.2.1.5 Antwortobjekt

Eine Anforderung an einen Ressourcenpfad, die eine Einheit oder eine Zusammenstellung auf der ESHG-IoT-Drittanbieter-API anspricht, führt zur Erstellung eines Antwortdokuments.

Ein solches Antwortdokument enthält (Nachrichtenhauptteil-)Daten; mittels einer Struktur, die Metainformationen oder Relationen zu anderen Ressourcen enthält, kann es weiter verfeinert werden.

Die ESHG-IoT-Drittanbieter-API-Struktur der Nachrichtenhauptteildaten ist darauf ausgelegt, für Menschen lesbaren Inhalt bereitzustellen (bei sicherheitsdekodierten Inhalten). Für die Definition von Nachrichtenhauptteildaten siehe 4.2.3.

4.2.1.6 Antwortstatuscodes

Der ESHG-IoT-Drittanbieter-Server muss mit definierten HTTP-Antwortstatuscodes aus [RFC 7231] antworten, unter Einhaltung von deren vorgesehener Semantik. Andere Antwortstatuscodes als jene, die in den nachstehenden Abschnitten beschrieben werden, sind anbieterspezifisch (einschließlich jener, die der ordnungsgemäßen Unterstützung des ETag-Mechanismus dienen).

Die Spalte „Einzelheiten“ gibt an, ob für einen Antwortstatuscode ein bestimmtes Verhalten der ESHG-IoT-Drittanbieter-API erforderlich ist.

4.2.1.6.1 1xx-Statuscodes

1xx-Antwortstatuscodes zeigen eine Zwischenantwort für den Kommunikationsverbindungsstatus an.

Tabelle 5 – 1xx-Antwortcodes

Status	Bezeichnung	Unterstützung	Einzelheiten
101	Switching Protocols (Protokollwechsel)	M	–

4.2.1.6.2 2xx-Statuscodes

2xx-Antwortstatuscodes zeigen an, dass die Clientanforderung erfolgreich empfangen und akzeptiert wurde.

Tabelle 6 – 2xx-Antwortcodes

Status	Bezeichnung	Unterstützung	Einzelheiten
200	OK	M	<p>Anforderung akzeptiert.</p> <p>ANMERKUNG Nach dem JSON-API-Standard wird der Antwortcode bei einer erfolgreichen an eine einzelne Ressource oder eine Ressourcenzusammenstellung gerichteten Anforderung verwendet.</p> <p>BEISPIEL Der Endpunkt, auf den über den URL-Ressourcenpfad <i>/functions</i> zugegriffen wird, enthält keine Einheiten, da der Installation die Daten nicht zur Verfügung standen (das MaC-Projekt verfügt über keine MaC-Funktionen). Die „erfolgreich“-Antwort (200) enthält JSON-API-konforme Nutzdaten, die mittels des data-Mitglieds und optional des collection-Mitglieds (siehe 4.2.3.3) leere Daten anzeigen.</p> <p>BEISPIEL Der Endpunkt, auf den über den URL-Ressourcenpfad <i>/functions/{id}</i> zugegriffen wird, enthält die Ressourceneinheit mit {id} nicht, weswegen eine „nicht erfolgreich“-Antwort (404) verwendet werden muss.</p>
201	Created (erstellt)	M	–

Status	Bezeichnung	Unterstützung	Einzelheiten
202	Accepted (akzeptiert)	M	<p>Anforderung akzeptiert; weitere Verarbeitung nicht abgeschlossen.</p> <p>ANMERKUNG Ein ESHG-IoT-Drittanbieter-Server zeigt an, dass er nicht garantieren kann, dass er die Anforderung erfolgreich ausführen kann. Üblicherweise wird der Client in einem solchen Fall umgehend von der Aufgabe entbunden, die Anforderung weiter zu überwachen, obwohl die Anforderung von dem Server noch nicht verarbeitet wurde.</p> <p>BEISPIEL Wenn Datenpunktwerte geschrieben werden, muss der Server gegebenenfalls auf eine (positive/negative) Bestätigung durch ein Subsystem warten. Ohne diese Bestätigung enthält der Antwortcode keinen gesicherten Anforderungsstatus. Dies wird durch die Verwendung dieses Antwortcodes angezeigt.</p>
204	No Content (kein Inhalt)	M	<p>Erfolgreiche Anforderung, es werden keine weiteren Antwortdaten gesendet werden.</p> <p>ANMERKUNG Ein ESHG-IoT-Drittanbieter-Server zeigt an, dass die angeforderten Änderungen wie vorgesehen vorgenommen wurden. Der übliche Anwendungsfall besteht darin, den Client lediglich über die Ausführung einer Anforderung zu informieren; zusätzlich darf auch ein ETag-Flag verwendet werden.</p> <p>BEISPIEL Das Aufheben einer Server-Subscription erfordert üblicherweise keine Bestätigung durch ein Subsystem, der Antwortcode zeigt den sofortigen Erfolg an.</p> <p>BEISPIEL Wenn Datenpunktwerte geschrieben werden, erhält der Server möglicherweise sofort eine Bestätigung, z. B. bei Subsystemen mit sehr geringer Latenz oder wenn der Datenpunkt nur auf dem Server gehostet wird. Der Antwortcode zeigt den sofortigen Erfolg an.</p>

4.2.1.6.3 4xx-Statuscodes

4xx-Antworten zeigen an, dass der Client augenscheinlich fehlerhafte Anforderungen gesendet hat.

Tabelle 7 – 4xx-Antwortcodes

Status	Bezeichnung	Unterstützung	Einzelheiten
400	Bad Request (fehlerhafte Anforderung)	M	–
401	Unauthorized (fehlende Autorisierung)	M	–
403	Forbidden (verboten)	M	–

Status	Bezeichnung	Unterstützung	Einzelheiten
404	Not Found (nicht gefunden)	M	<p>Der Server kann die angeforderte Ressource bzw. den angeforderten Ressourcenpfad nicht finden.</p> <p>ANMERKUNG Nach dem JSON-API-Standard wird der Antwortcode bei einer an eine nicht vorhandene Ressource gerichteten Anforderung verwendet.</p> <p>BEISPIEL Der Endpunkt, auf den über den URL-Ressourcenpfad <i>/functions/{id}</i> zugegriffen wird, enthält die Ressourceneinheit mit {id} nicht.</p> <p>BEISPIEL Der Endpunkt, auf den über den URL-Ressourcenpfad <i>api/v200/functions</i> zugegriffen wird, existiert nicht. Üblicherweise fordert ein Client im Vorhinein alle von einem Server unterstützten API-Versionen mit ihren jeweiligen Basispfaddefinitionen (<i>api/v200</i>) an, indem er sie von dem entsprechenden Endpunkt „/.well-known/knx“ ausliest.</p>
405	Method Not Allowed (Methode unzulässig)	M	–
406	Not Acceptable (nicht akzeptabel)	M	–
408	Request Timeout (Zeitüberschreitung der Anforderung)	O	–
409	Conflict (Konflikt)	M	–
412	Precondition Failed (Vorbedingung nicht erfüllt)	O	–
413	Request Entity Too Large (Anforderungsentität zu groß)	O	–
414	URI Too Long (URI zu lang)	M	–
415	Unsupported Content-Format (Inhaltsformat nicht unterstützt)	M	–
422	Unprocessable Entity (Entität kann nicht verarbeitet werden)	M	–
429	Too Many Requests (zu viele Anforderungen)	M	–

4.2.1.6.4 5xx-Statuscodes

5xx-Antwortstatuscodes zeigen an, dass der Server die angeforderte Methode nicht ausführen kann.

Tabelle 8 – 5xx-Antwortcodes

Sta- tus	Bezeichnung	Unter- stüt- zung	Einzelheiten
500	Internal Server Error (interner Serverfehler)	M	–
501	Not Implemented (nicht implementiert)	M	–
503	Service Unavailable (Dienst nicht verfügbar)	M	–
504	Gateway Timeout (Gateway-Zeitüberschreitung)	M	–
505	HTTP Version Not Supported (HTTP-Version nicht unterstützt)	M	–

4.2.1.6.5 Gesammelte Fehlerbehandlung

Wenn ein Server auf mehrere Probleme trifft, darf er die Verarbeitung fortsetzen. Beispielsweise darf ein Server mehrere Attribute verarbeiten und dann mehrere Validierungsprobleme in einer einzigen Antwort zurückgeben.

Wenn ein ESHG-IoT-Drittanbieter-Server bei ein und derselben Anforderung auf mehrere Probleme trifft, die dem **gleichen** Fehlerbereich angehören (z. B. 4xx), muss in der Antwort der am häufigsten anwendbare HTTP-Fehlercode verwendet werden.

Wenn ein ESHG-IoT-Drittanbieter-Server bei ein und derselben Anforderung auf mehrere Probleme trifft, die **unterschiedlichen** Fehlerbereichen angehören (z. B. 4xx und 5xx), muss in der Antwort der Fehlercode 500 mit zusätzlichen Fehlerdetails in einem Fehlerobjekt (siehe 4.2.1.6.6) verwendet werden.

ANMERKUNG Für mehrfache 4xx-Fehler könnte beispielsweise der Antwortstatuscode 400 `Bad Request` und für mehrere 5xx-Fehler der Antwortstatuscode 500 `Internal Server Error` anwendbar sein.

Die vorstehend aufgeführten Punkte wurden vom JSON-API-Standard abgeleitet, siehe Abschnitt „Errors“.

4.2.1.6.6 Fehlerobjekte

Wenn während des Serverbetriebs bei der Datenverarbeitung Fehler auftreten, sollte der Server stets Fehlerobjekte zurückgeben (zusammen mit einem geeigneten HTTP-Antwortstatuscode). Dadurch können mehr Einzelheiten übermittelt werden als durch den bloßen Statuscode, und zwar unabhängig davon, ob der Fehler durch den Client oder den Server verursacht wurde (d. h. sowohl für [4xx]- als auch für [5xx]-Fehlercodes).

Ein ESHG-IoT-Drittanbieter-Server muss Folgendes unterstützen.

1. Ein ESHG-IoT-Drittanbieter-Server muss Fehler mit dem `errors`-Mitglied der oberen Ebene melden, einschließlich eines **nicht leeren** Arrays einzelner Fehlerobjekte entsprechend der Beschreibung in JSON API.
2. Ein ESHG-IoT-Drittanbieter-Server darf Fehlerdetails entsprechend der Definition in [RFC 7807] oder im JSON-API-Standard, Abschnitt „Error Objects“, in seine Antwort aufnehmen.

Das Fehlerobjekt muss mindestens ein `title`-Mitglied (als Bezeichner des Fehlerobjekts) und ein `status`-Mitglied enthalten und sollte ein `detail`-Mitglied mit zusätzlichen Fehlerdetails für die jeweilige Anforderung enthalten. Darüber hinaus darf ein Fehlerobjekt zusätzliche benutzerdefinierte Fehlermitglieder enthalten.

BEISPIEL Ein JSON-API-`errors`-Mitglied, das zwei (künstliche) Fehlerobjekte mit obligatorischen/optionalen Mitgliedern und einem benutzerdefinierten Fehlermitglied `authentication_uri` enthält.

```
errors:
  - title: Forbidden // mandatory
status: 403 // mandatory
detail: Service requires other access rights // optional, JSON API
links: https://schema.knx.org/2020/api // optional, JSON API
authentication_uri: /oauth // optional, custom
title: Not Implemented
  status: 501
  detail: Not implemented for this version of the API
```

Stacktraces dürfen nicht Bestandteil eines Fehlerobjekts sein. Durch Stacktraces können sensible Informationen an Partner und Dritte gelangen, die nicht für diese bestimmt sind, und Interna über Schwachstellen gegenüber Angreifern offengelegt werden.

4.2.1.7 Inhaltsaushandlung

Die Inhaltsaushandlung ermöglicht es einer künftigen ESHG-IoT-Drittanbieter-API, den API-Inhalt zu erweitern oder unterschiedliche Inhaltstypformate auf demselben API-Ressourcenpfad und/oder derselben Namensstruktur zu unterstützen.

Der ESHG-IoT-Drittanbieter-Server muss einen definierten API-Inhaltstyp und eine definierte Zeichenkodierung auf der ESHG-IoT-Drittanbieter-API unterstützen. Hierzu werden das HTTP-Element `Content-Type` und der/ das darin enthaltene `Internet-Media-Type` und `charset` verwendet.

ANMERKUNG Der HTTP-Content-Type definiert ein Header-Feld [RFC 7231], in dem der verwendete HTTP-Nutzdaten-(Internet-)Media-Type [RFC 2046] angegeben wird, der früher als „MIME-Type“ bezeichnet wurde.

4.2.1.7.1 Inhaltstyp

1. Ein ESHG-IoT-Drittanbieter-Server muss die folgenden `Media-Type`-Varianten unterstützen. Die Nachrichtenhauptteildaten müssen also dem im Hauptteil angegebenen `Media-Type` entsprechen.

<code>application/vnd.api+json</code>	(normaler Anforderungs-/Antwort-Endpunkt und Fehlerbehandlung)
<code>application/json</code>	(Erkennungsantworten und Autorisierungsantworten)
<code>application/x-www-form-urlencoded</code>	(Autorisierungsanforderungen)
<code>application/pkcs7-mime</code>	(Zertifikatantworten)

ANMERKUNG Alle `media-types` sind bereits bei der IANA registriert.

2. Ein ESHG-IoT-Drittanbieter-Server muss das folgende Standard-charset unterstützen.

utf-8

(für die allgemeine Inhaltskodierung)

Der vorstehend beschriebene Media-Type bzw. das entsprechende JSON-Format aus dem JSON-API-Standard erfordert die Verwendung des oben definierten charset; siehe dazu JSON-Standard, 8.1.

4.2.1.7.2 Umgang mit dem Inhaltstyp

Für den üblichen Umgang mit Media-Type und Content-Type hinsichtlich Clientanforderungen und der Erzeugung von Server-Antwortnutzdaten siehe JSON-API-Standard, 4.2.

ANMERKUNG Der JSON-API-Standard beschreibt den Umgang mit dem Media-Type als solchem sowie mit den hier nicht zugelassenen Parametern, einschließlich der entsprechenden Fehlerbehandlung mit dem Fehlercode 406/415.

1. Ein ESHG-IoT-Drittanbieter-Server muss mit dem Fehlercode 400 antworten, wenn ein Client den folgenden ungültigen Anforderungsrahmen verwendet:
 - kein Nutzdateninhalt enthalten, obwohl der Endpunkt einen solchen erwartet (z. B. bei einem POST-Dienst)
 - Nutzdateninhalt enthalten, obwohl der Endpunkt keinen solchen erwartet (z. B. bei einem GET-Dienst)
2. Ein ESHG-IoT-Drittanbieter-Server darf sämtliche Daten in seinen Antwortdokumenten ausschließlich entsprechend seinem Nutzdaten-Media-Type senden, der auch in seinem Content-Type-Antwort-Header festgelegt werden muss.
3. Zusätzlich zu den Anforderungen im JSON-API-Standard gilt, dass ein ESHG-IoT-Drittanbieter-Server mit dem Fehlercode 406 antworten muss, wenn ein Client die folgenden **Accept-Header-Elemente** als Teil seiner Anforderung definiert:
 - kein Media-Type entspricht dem festgelegten (dies deckt auch eine Liste innerhalb der Anforderung ab)
4. Zusätzlich zu den Anforderungen im JSON-API-Standard gilt, dass ein ESHG-IoT-Drittanbieter-Server mit dem Fehlercode 415 antworten muss, wenn ein Client die folgenden **Content-Type-Header-Elemente** als Teil seiner Anforderung definiert:
 - ein anderer Media-Type als der für den Endpunkt festgelegte
 - ein leerer oder nicht vorhandener Media-Type

Ein ESHG-IoT-Drittanbieter-Server muss mit dem Media-Type application/vnd.api+json antworten, wenn eine Clientanforderung den Content-Type mit dem Media-Type application/json anstelle eines angenommenen/festgelegten Media-Type application/vnd.api+json festlegt.

ANMERKUNG Die Anforderungen unter diesem Punkt beschreiben die Fehlerbehandlung/Fallback-Mechanismen im Falle von **entity-Headern**. Dass ein Content-Type in einer Anforderung vorliegt, ist zumeist bei Zugangsmethoden mit zusätzlichen Daten in der Anforderung der Fall, z. B. bei POST-/PUT-/PATCH-Methoden. Die vorstehende Definition im zweiten Aufzählungspunkt wird als Fallback-Mechanismus verwendet, wenn ein Client ein ähnliches Format anfordert, das der Server verstehen kann. Die Anzeige eines Fehlers ist in einem solchen Fall nicht vorgesehen.

1. Ein ESHG-IoT-Drittanbieter-Server muss mit dem Standard-charset antworten:
 - für alle in seinen Antwortdokumenten enthaltenen Daten
 - als Teil des Content-Type, wenn es von einem Client als Teil des Accept-Charset angefordert wurde; oder wenn das vom Client angeforderte Accept-Charset nicht bereitgestellt werden kann (dieses Vorgehen wird dem Anhalten der Kommunikation mit einem expliziten Fehler vorgezogen).
2. Ein ESHG-IoT-Drittanbieter-Server darf ohne Zeichensatz antworten, wenn in der Clientanforderung kein Accept-Charset definiert wurde.

4.2.2 Clientanforderungen

Clients im Allgemeinen und ESHG-IoT-Drittanbieter-Clients im Speziellen liegen außerhalb des Anwendungsbereichs dieses Dokuments, so dass für sie keine vollständigen Anforderungen formuliert werden