DALI-2 IoT

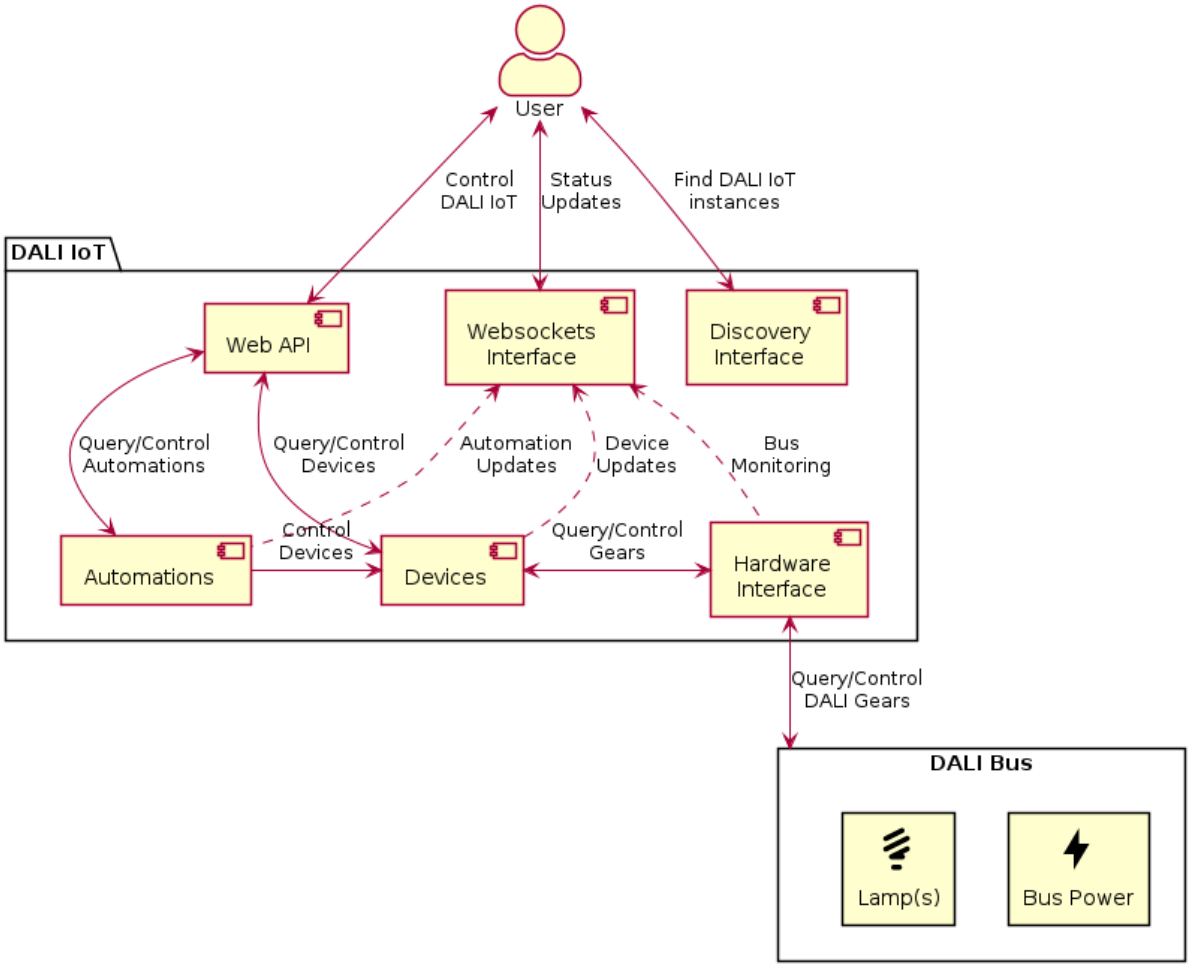
Anleitung

API Dokumentation

Programmierschnittstellenbeschreibung
des DALI-2 IoT Moduls

Art.Nr. 89453886
GTIN: 9010342013607

© 2022-04-06, Lunatone Industrielle Elektronik GmbH


User
Control DALI IoT
Status Updates
Find DALI IoT instances
DALI IoT
Web API
Websockets Interface
Discovery Interface
Query/Control Automations
Query/Control Devices
Automation Updates
Device Updates
Bus Monitoring
Automations
Control Devices
Devices
Query/Control Gears
Hardware Interface
Query/Control DALI Gears
DALI Bus
Lamp(s)
Bus Power

Inhaltsverzeichnis

1 Bestimmung der IP Adresse

Für die Nutzung der API Dokumentation ist es nötig die IP Adresse des DALI-2 IoT Geräts zu kennen. Die Bestimmung der IP Adresse hängt davon ab, ob eine direkte Verbindung zu einem Computer genutzt wird (Abschnitt 1.1), oder das DALI-2 IoT in einem Netzwerk betrieben wird (Abschnitt 1.2).

1.1 Direkte Verbindung zu einem Computer

Im Auslieferzustand ist das DALI-2 IoT so konfiguriert, dass es eine automatische IP Adresse anhand des DHCP Protokolls beantragt. Falls das DALI-2 IoT keinen DHCP-Server erreichen kann, wird stattdessen auf die fixe IP Adresse `169.254.0.1`, mit der Subnetzmaske `255.255.0.0` zurückgegriffen. Diese kann z.B. verwendet werden, um das DALI-2 IoT über eine direkte Ethernet-Verbindung mit einem Computer zu erreichen.

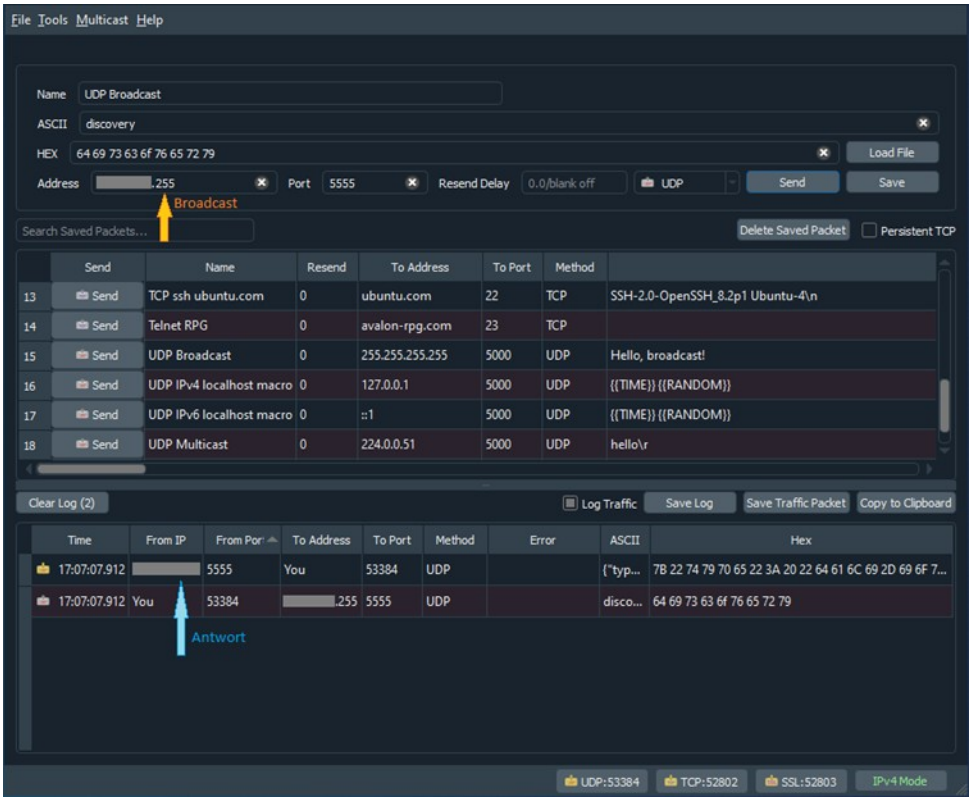
1.2 Identifizieren der DALI-2 IoT Geräte in einem Netzwerk

Die IP Adresse kann auch anhand eines „Discovery“ Protokolls, welches vom DALI-2 IoT zur Verfügung gestellt wird, bestimmt werden. Das DALI-2 IoT führt einen Service aus, der am Port `5555` auf UDP-Pakete mit dem Datenpaket `discovery` reagiert, indem es ein Datenpaket mit `{"type": "dali-2-iot"}` zurück sendet. Auf diese Weise lassen sich laufenden DALI-2 IoT Instanzen in einem Netzwerk identifizieren, indem entweder ein UDP Broadcast-Paket oder einzelne UDP-Pakete an die jeweiligen Adressen im Netzwerk verschickt werden.

Seit Versionen `1.2` enthält die Antwort einen zusätzlichen konfigurierbaren Namen (Abschnitt 5.1).

```
{
  "type": "dali-2-iot",
  "name": "user defined name"
}
```

Ein simples Programm zum Versenden von Broadcasts ist „Packet Sender“.

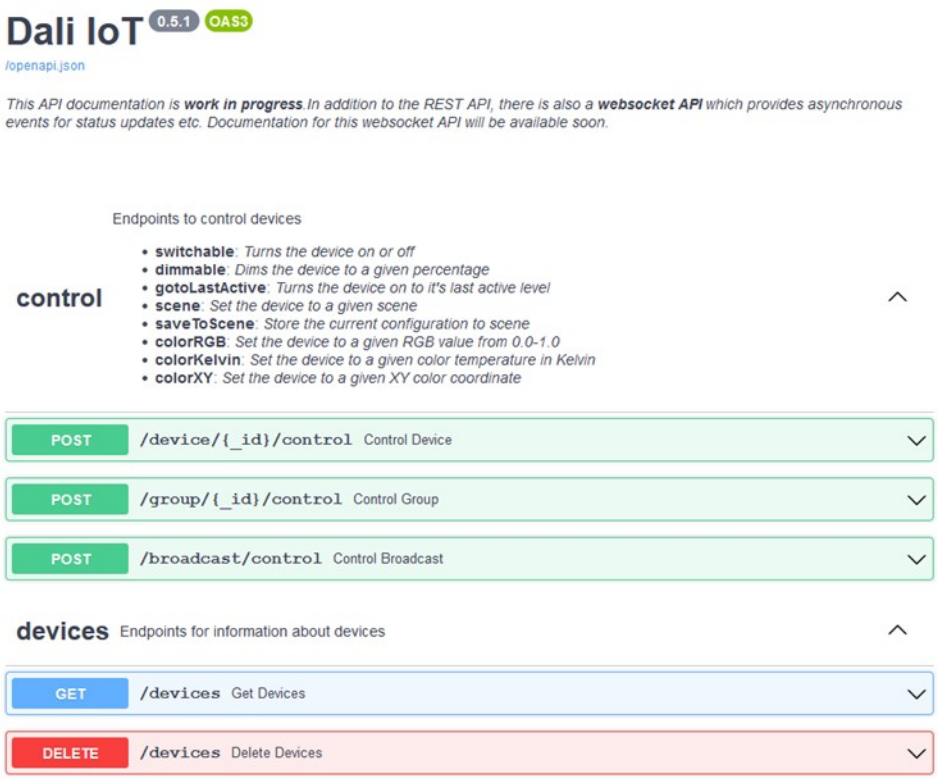


Um mit Packet Sender die DALI-2 IoT Instanzen in einem Netzwerk zu finden, muss das „ASCII“-Eingabefeld auf `discovery` gesetzt werden, das „Address“-Feld auf die Broadcast-Adresse des Netzwerks (endet auf `.255`) bzw. eine einzelne Adresse die abgefragt werden soll, und das „Port“-Feld auf `5555`.

Klickt man nach diesen Einstellungen auf „Send“, dann wird ein UDP-Paket im Netzwerk versendet. Antworten auf dieses Paket scheinen im „Log“ unterhalb der Vorlagen für Pakete auf. Dort können die IP Adressen von DALI-2 IoT Instanzen abgelesen werden.

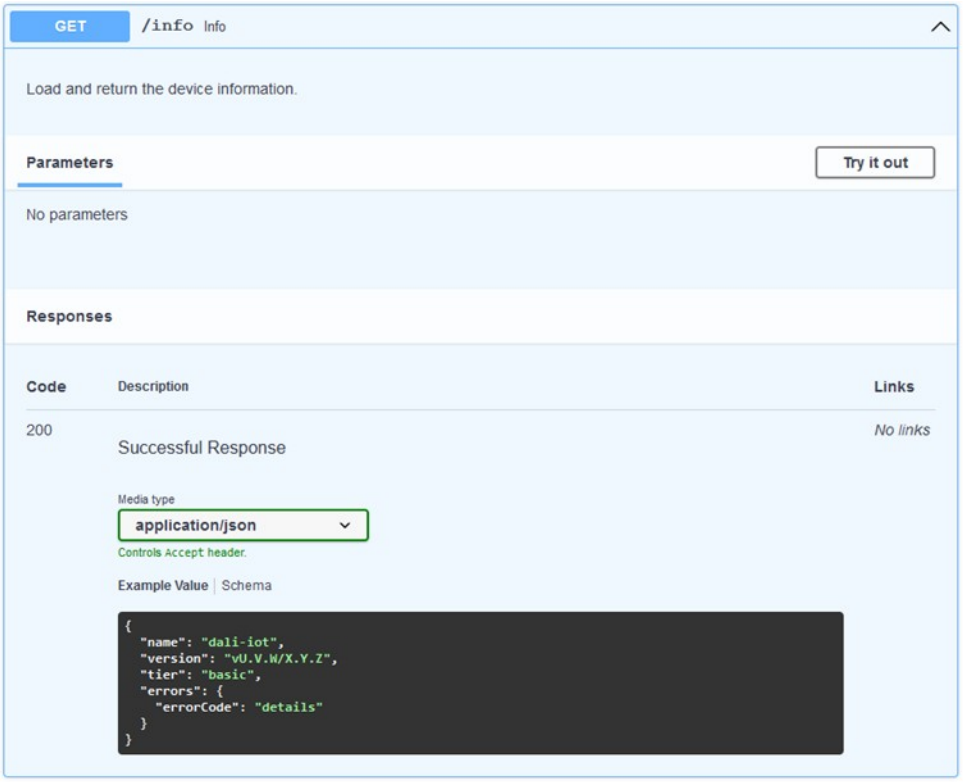
2 Aufruf der DALI-2 IoT API Dokumentation

Die API Dokumentation („docs“-Seite) des DALI-2 IoT kann über `http://<IP_ADDRESS>/docs` mit einem Webbrowser aufgerufen werden, wobei `<IP_ADDRESS>` durch die IP Adresse des Geräts ersetzt werden muss. Auf dieser Seite sind zunächst die API Endpunkte, geordnet nach Kategorien (`control`, `devices`, `dali`, etc.) aufgelistet. In den einzelnen Kategorien befinden sich sogenannte Endpunkte mit denen mit dem DALI-2 IoT System interagiert werden kann. Endpunkte sind eine Kombination aus Anforderungsmethode (GET, POST, PUT oder DELETE) und dem Endpunktnamen (z. B. `/broadcast/control`). Der gleiche Endpunktname kann für mehrere Endpunkte mit unterschiedlichen Anforderungsmethoden und Funktionalitäten genutzt werden.

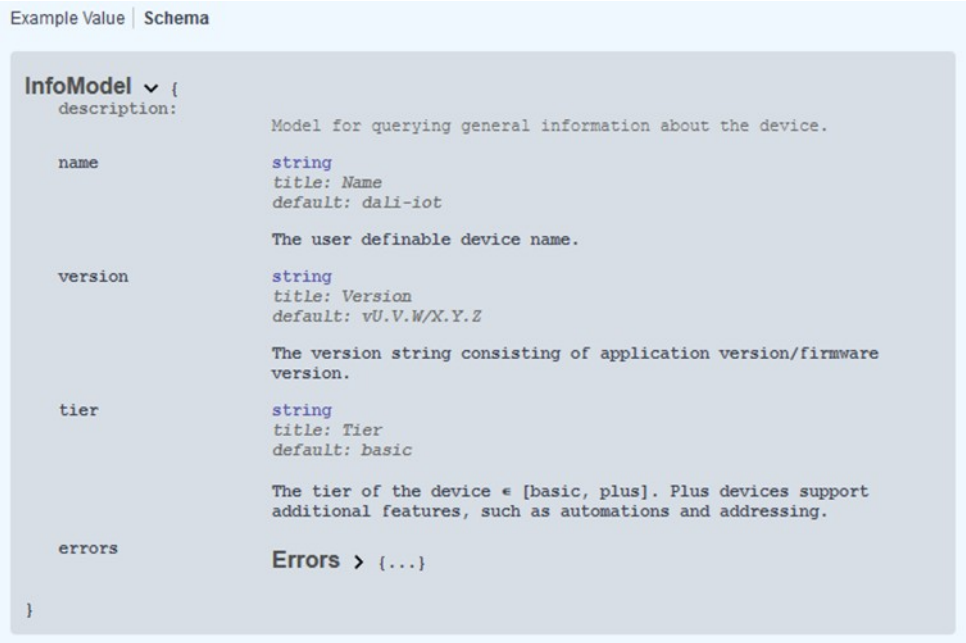


2.1 Kategorien und Endpunkte

Beim Aufruf der API Dokumentation sind die Endpunktbeschreibungen zunächst zusammengeklappt; diese können mit einem Klick auf den Endpunkt aufgeklappt werden um weitere Details zu erhalten. Dabei werden eine kurze Beschreibung des Endpunkts, die zu übertragenden Parameter und die Rückmeldungen der API genauer ausgeführt. Als Beispiel ist `GET /info` aufgeführt.

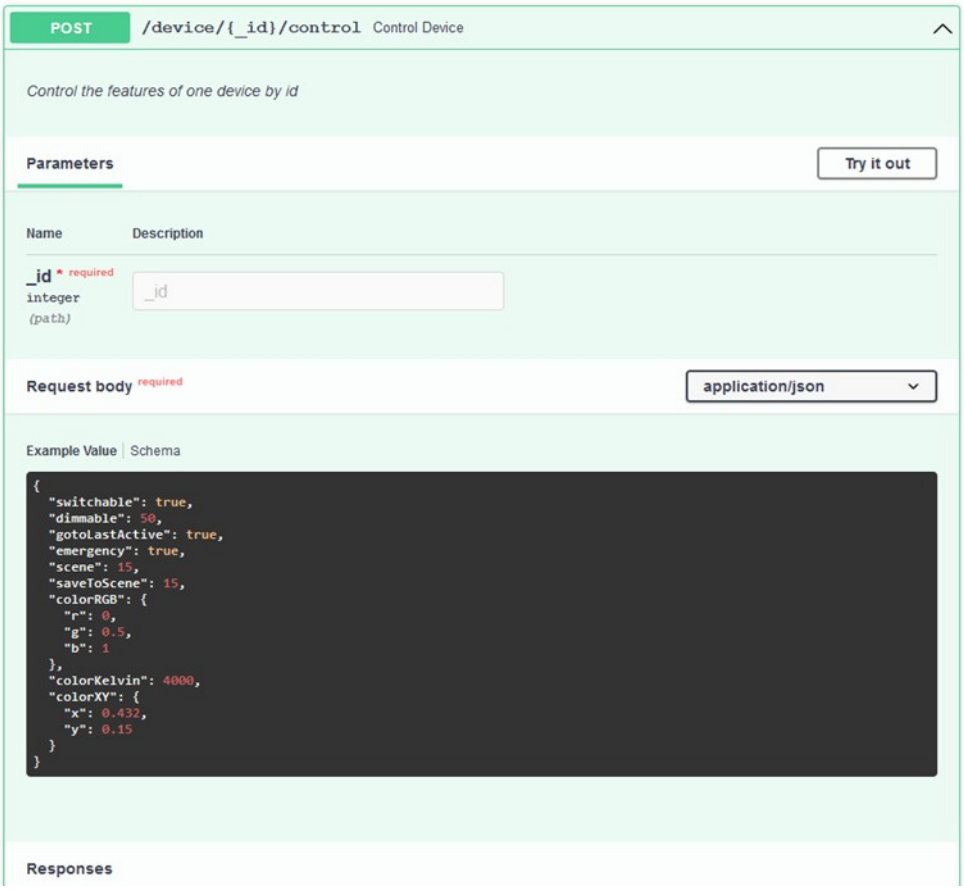


Eine Anforderung dieses Endpunkts benötigt keine Eingaben und überträgt als Rückgabewert von der API ein `InfoModel` Objekt. Ein Beispiel für die Struktur dieses Modells wird in der Dokumentation des Endpunkts gezeigt. Bei `GET /info` werden unter anderem der Name des Geräts und dessen Software- und Firmware-Versionen übertragen. Um ein genaueres Bild über die Zusammensetzung des Rückgabewerts zu erhalten lässt sich das Schema des Modells darstellen, indem man auf die Schaltfläche „Schema“ oberhalb des Beispiels klickt. Dabei werden die Datenfelder des Modells, deren Datentypen und eine kurze Erklärung der Daten dokumentiert.

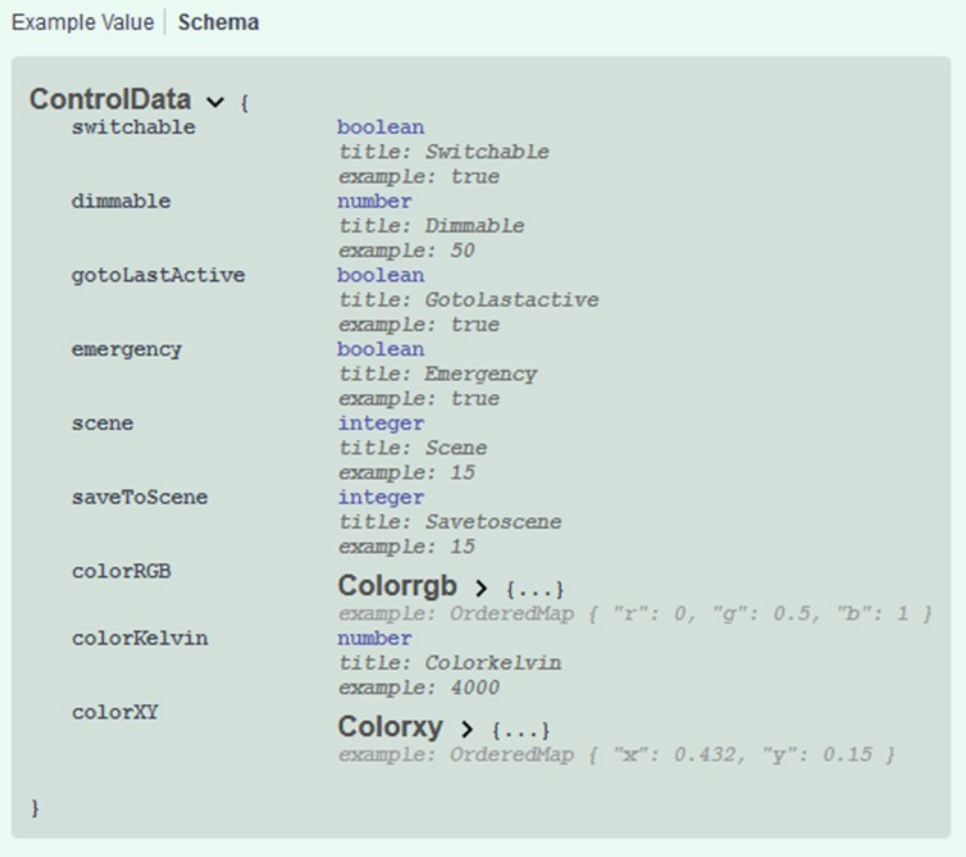


Manche Endpunkte benötigen zusätzliche Angaben um eine Anforderung verarbeiten zu können. Zum Beispiel muss für den Endpunkt `POST /device/{_id}/control`, mit dem einzelne Geräte angesteuert werden können, eine Identifikationsnummer und ein `ControlData` Objekt gesendet werden. Die Identifikationsnummer bestimmt welches Gerät angesteuert wird, während `ControlData` den gewünschten Zustand für das Gerät angibt. Ein Beispielwert für `ControlData` Objekte befindet sich in der Dokumentation des Endpunkts (Example Value).

Der Endpunkt `POST /device/{_id}/control`, enthält die Identifikationsnummer als Teil der Adresse des Endpunkts, während das `ControlData` Objekt in den Abfragedaten übermittelt wird. In der API Dokumentation wird die Identifikationsnummer durch ein separates Eingabefeld angegeben. Diese Nummer verändert den Endpunktnamen der Abfrage, was durch geschwungene Klammern im Endpunktnamen angezeigt wird. Um zum Beispiel ein Gerät mit Identifikationsnummer `"id": 1` anzusteuern werden Abfragen an den Endpunkt `POST /device/1/control` gesendet.



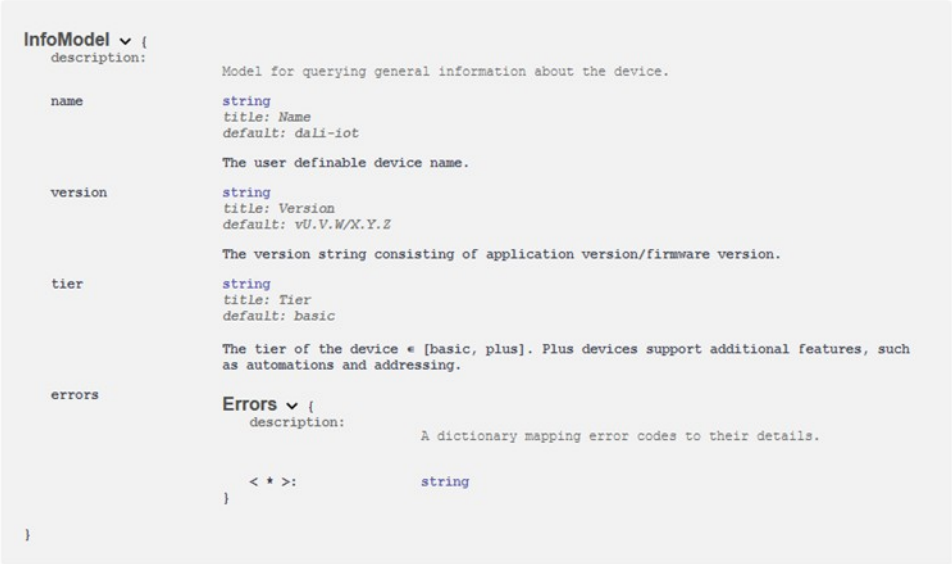
Das Schema dieses Datentyps lässt sich, so wie bei den Rückgabewerten, anzeigen, indem man auf „Schema“ oberhalb des Beispielwerts klickt, neben dem Button „Example Value“.



2.2 Liste der Schemata

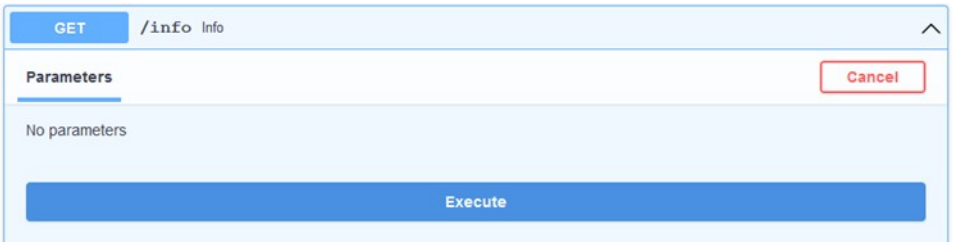
Anschließend an die Kategorien und Endpunkte befindet sich eine Liste von Schemata, die von den Endpunkten übertragen werden.

Diese können durch einen Klick auf das Schema ausgeklappt werden um deren Details anzuzeigen.

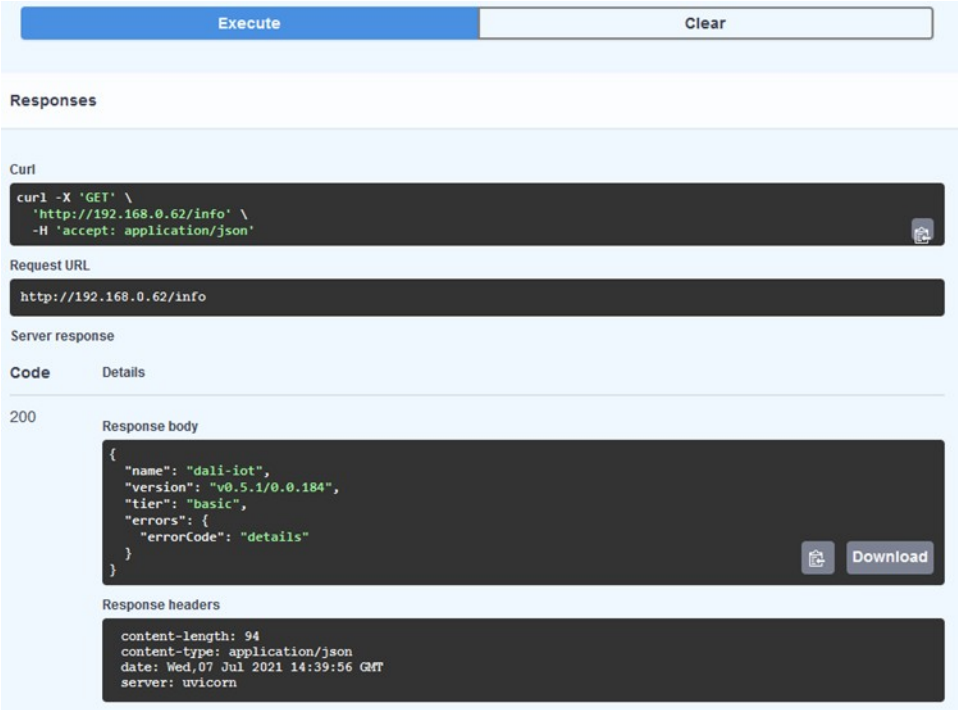


2.3 Direktes Testen der Endpunkte über die Dokumentation

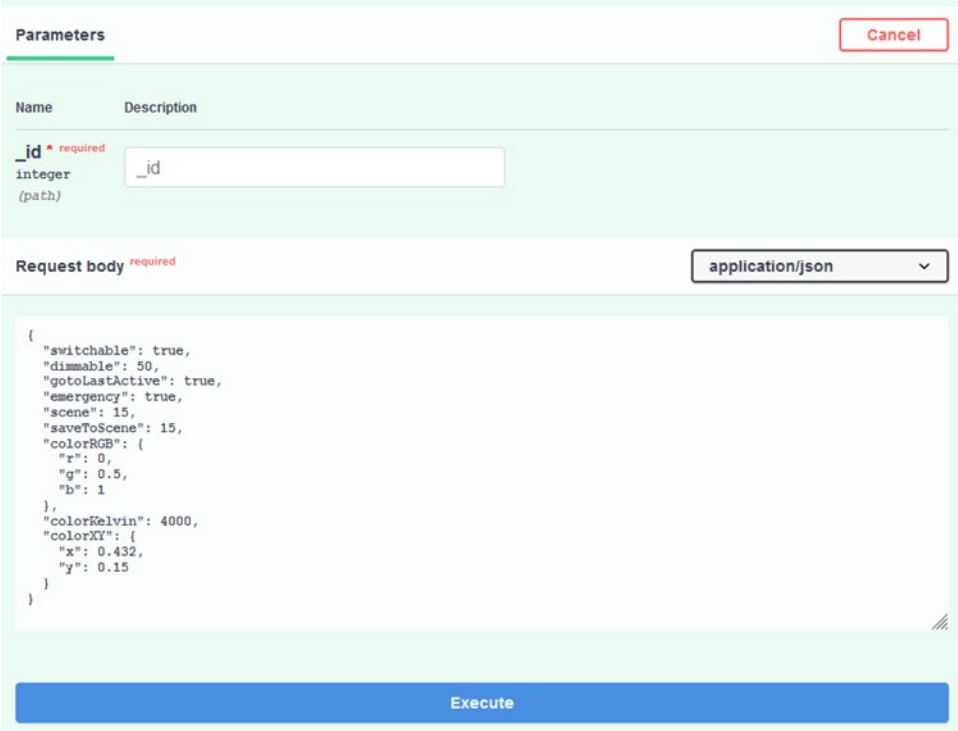
Endpunkte können über den „Try it out“ Button direkt in der Dokumentation getestet werden. Durch eine Betätigung des Buttons öffnen sich zusätzliche Eingabefelder (die von der Beschaffenheit eines Endpunkts abhängen), sowie der Button „Execute“. Da beim Endpunkt `GET /info` keine Parameter vorhanden sind, ist in diesem Fall keine weitere Eingabe nötig.



Wird dieser Button betätigt, dann sendet der Browser eine Anforderung an den Endpunkt. Nach dem Absenden werden: ein Beispiel, wie eine äquivalenter Anforderung mit dem Programm „curl“ gesendet werden kann, die Adresse an welche die Anforderung geschickt wird, und die Antwort auf die Anforderung angezeigt. Darüber hinaus wird ein Button „Clear“ zum Entfernen der Anforderungen hinzugefügt.



Da manche Endpunkte zusätzliche Angaben für deren Verarbeitung benötigen, können weitere Eingabefelder aufgeklappt werden. Zum Beispiel benötigt der Endpunkt `POST /device/{_id}/control` eine Identifikationsnummer um ein Gerät zu identifizieren, und ein `ControlData` Objekt, um dieses zu steuern. Pflichtfelder werden neben den Eingabefeldern mit „required“ gekennzeichnet. Felder, die keine Pflichtfelder sind, können leer belassen werden.



3 Ansteuern von DALI Geräten

3.1 Das ControlData-Objekt

Geräte werden über ihre Features, wie z. B. „`dimmable`“ für dimmbare Geräte, angesteuert. Um Geräte einzustellen, müssen deren Features und zugehörige Parameter durch ein `ControlData` Objekt angegeben werden. Das `ControlData` Schema ist eine Abbildung, die die gewünschten Werte den Namen der Features zuordnet. Einzelne Features sind optional; daher ist es ausreichend nur die gewünschten in einer Anforderung anzugeben. Es können mehrere, durch Beistriche getrennte Features gleichzeitig übertragen werden.

3.1.1 Ein- oder Ausschalten der Geräte

Um Geräte ein- oder auszuschalten, wird das „`switchable`“-Feature genutzt, welches einen Boolean Wert benötigt, um das Gerät entweder ein (`true`) oder aus (`false`) zu schalten. Der „Request Body“ zum Einschalten aller Geräte setzt sich aus dem Namen des Features und dem Wert zusammen.

```
{
  "switchable": true
}
```

3.1.2 Dimmen der Geräte

Um Geräte zu dimmen, wird das „`dimmable`“-Feature genutzt. Dieses akzeptiert eine Prozentzahl von 0 bis 100. Wird ein „`dimmable`“-Wert von 0 angegeben, dann werden die Geräte ausgeschaltet. Bei allen anderen Werten werden die Geräte auf das angegebene Level eingestellt.

```
{
  "dimmable": 50
}
```

3.1.3 Letztes aktives Level abrufen

Um das letzte aktive Level wieder aufzurufen kann das „`gotoLastActive`“ Feature genutzt werden. Dabei wird ein `true`-Wert angegeben.

```
{
  "gotoLastActive": true
}
```

3.1.4 Aufrufen einer Szene

Szenen können mit dem „`scene`“ Feature aufgerufen werden. Dabei wird die Nummer der Szene (0 bis 15) angegeben, welche aufgerufen werden soll.

Szenen werden in den jeweiligen DALI Geräten gespeichert. Viele DALI Geräte haben im Auslieferzustand keine Szenenwerte vorgegeben. Zum Speichern von Szenenwerten, siehe Abschnitt 3.1.5.

```
{
  "scene": 15
}
```

3.1.5 Eine Szene abspeichern

Um das aktuelle Level für eine Szene zu speichern kann das „saveToScene“ Feature genutzt werden. Dabei wird die Nummer der Szene (0 bis 15) angegeben, in die gespeichert werden soll.

```
{
  "saveToScene": 15
}
```

3.1.6 Konfigurieren von Farben mittels Rot-, Grün- und Blau (RGB) Werten

Um die Farbe einer Rot- Grün- und Blau- (RGB / RGBW) Leuchte einzustellen kann das „colorRGB“ Feature genutzt werden. Dabei wird eine Abbildung übertragen, die den Schlüsseln "r", "g" und "b" relative Farbwerte von 0 bis 1 zuordnet.

```
{
  "colorRGB": {
    "r": 0,
    "g": 0.5,
    "b": 1
  }
}
```

3.1.7 Konfigurieren von Farben mittels Weiß-, Bernsteinfarben- und frei gewählten Farb (WAF) Werten

Um die Farbe einer Weiß- Bernsteinfarben- und frei gewählter Farbe- (RGBWAF) Leuchte einzustellen kann das „colorWAF“ Feature genutzt werden. Dabei wird eine Abbildung übertragen, die den Schlüsseln "w", "a" und "f" relative Farbwerte von 0 bis 1 zuordnet.

```
{
  "colorWAF": {
    "w": 0,
    "a": 0.5,
    "f": 1
  }
}
```

3.1.8 Konfigurieren von Farbtemperatur

Um die Farbe einer Farbtemperatur-Leuchte (CW-WW) einzustellen kann das „colorKelvin“ Feature genutzt werden. Dabei wird die Farbtemperatur in Kelvin angegeben (typischer Wertebereich von Lunatone CW-WW Dimmern: 100 bis 20000 K).

```
{
  "colorKelvin": 4000
}
```

3.1.9 Konfigurieren von Farben mittels Farbkoordinaten (XY)

Um die Farbe einer Leuchte mit Farbraumkoordinaten einzustellen kann das „colorXY“ Feature genutzt werden. Dabei wird eine Abbildung übertragen, die den Schlüsseln "x" und "y" die Farbkoordinaten von 0 bis 1 zuordnet.

```
{
  "colorXY": {
    "x": 0.432,
```

```
    "y": 0.15
  }
}
```

3.2 Wirkbereich: das DeviceModel Objekt

Zonen (Abschnitt 3.6) und Automatisierungen (Abschnitt 4) haben gemeinsam, dass sie sich auf einen Wirkbereich beziehen und die Features aller Geräte innerhalb des Bereichs gemeinsam kontrollieren. Ein Wirkbereich ist eine Liste von Geräten, Gruppen, Zonen und Broadcasts. Jeder Eintrag des Wirkbereichs ist ein `DeviceModel` Objekt, welches sich aus einem Typ (`type`) und einer optionalen Identifikationsnummer (`id`) zusammen setzt. Für den Typ kommen Broadcasts (`broadcast`), Gruppen (`group`), Geräte (`device`) und Zonen (`zone`) in Frage, wobei für Gruppen, Geräte und Zonen die Angabe der Identifikationsnummer nötig ist, um diese anzusteuern.

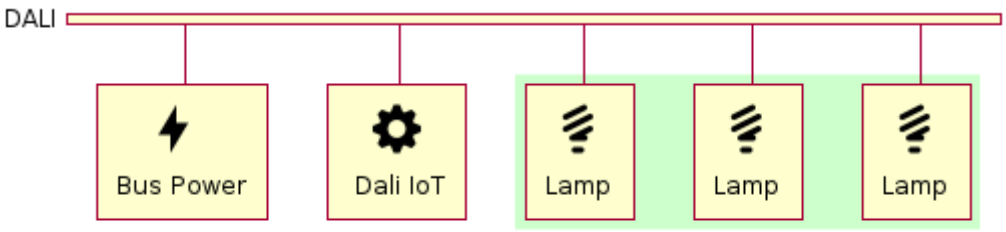
```
{
  "type": "broadcast"
}
```

```
{
  "type": "device",
  "id": 1
}
```

```
{
  "type": "group",
  "id": 1
}
```

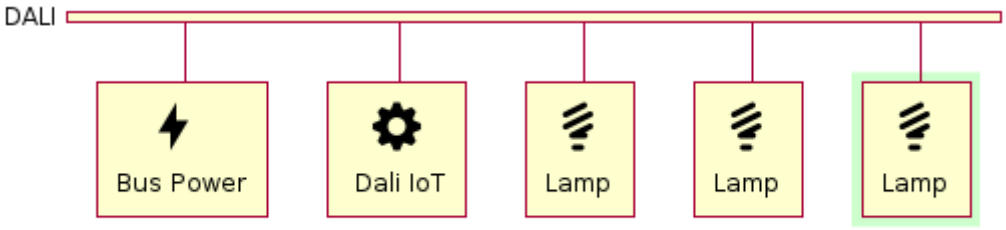
```
{
  "type": "zone",
  "id": 1
}
```

3.3 Ansteuern aller DALI Geräte auf dem DALI Bus

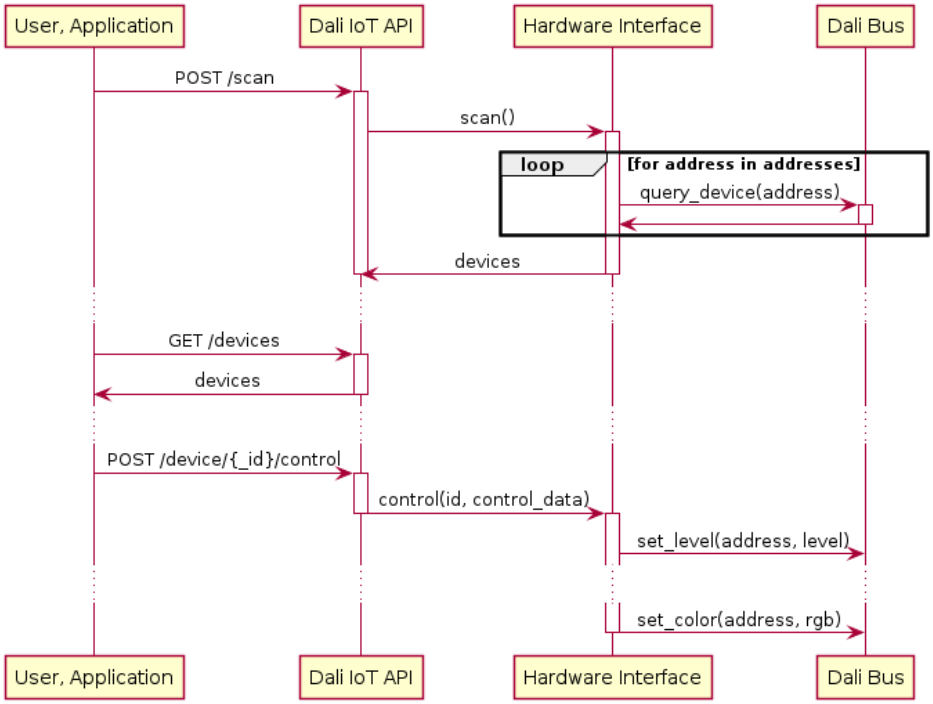


Um alle DALI-Geräte auf dem DALI Bus zu steuern ist keine zusätzliche Einrichtung notwendig; die Geräte können mit dem Endpunkt `POST /broadcast/control` direkt angesteuert werden. Dabei muss ein `ControlData` Objekt übertragen werden.

3.4 Ansteuern einzelner Geräte



Um mit dem DALI-2 IoT einzelne Geräte auf dem DALI Bus zu steuern, müssen diese zunächst einmalig erfasst und gespeichert werden. Jedes gespeicherte Gerät erhält dabei eine Identifikationsnummer, um es bei Anforderungen oder Befehlen von anderen Geräten zu unterscheiden. Nach dem Abschluss eines Geräte-Scans können die gespeicherten Geräte abgefragt werden, um deren Zustand und Identifikationsnummern zu erhalten. Diese Nummern werden intern im DALI-2 IoT genutzt um einzelne Geräte anzusteuern und unterscheiden sich von den DALI-Adressen der Geräte.



3.4.1 Geräte Scan

Ein sogenannter „Geräte-Scan“ dient dazu die Geräte auf dem DALI Bus zu adressieren, erfassen und speichern. Ein neuer Scan wird durch den `POST /dali/scan` Endpunkt gestartet, und benötigt zwei Angaben: `newInstallation` und `noAddressing`.

```
{
  "newInstallation": false,
  "noAddressing": false
}
```

`"newInstallation": true` wird bei einer Neuinstallation genutzt und bewirkt, dass die bekannten Geräte vor dem Scan gelöscht werden. Bei `"new Installation": false` werden die bestehenden Adressen beibehalten und neu gefundene Geräte zu den bekannten hinzugefügt.

Außerdem kann vor der Erfassung der Geräte eine Adressierung gestartet werden, bei der den DALI-Geräten eine eindeutige Bus-Adresse zugewiesen wird. Falls dieser Schritt bereits anderweitig durchgeführt wurde (z. B. über eine zweite DALI-2 IoT Instanz) müssen die DALI Geräte nur noch vom DALI-2 IoT erfasst werden. In diesem Fall lässt sich die DALI Adressierung deaktivieren, indem der Wert `noAddressing` auf `true` gesetzt wird.

Da der Geräte-Scan alle Adressen des DALI Bus abfragt kann dieser bis etwa eine Minute dauern. Währenddessen kann der Fortschritt des Scans über den Endpunkt `GET /dali/scan` abgefragt werden. Dieser Endpunkt liefert als Antwort ein `ScanModel`, mit der Identifikationsnummer des Scans, dessen Fortschritt (in Prozent), der Anzahl gefundener Geräte und dem Status des Scans.

```
{
  "id": "e9160f03-0982-4cd7-88ab-00a4755bd17d",
  "progress": 38.671875,
  "found": 1,
  "status": "in progress"
}
```

Während ein Scan durchgeführt wird, kann kein neuer Scan gestartet werden. Allerdings können Scans abgebrochen werden, indem eine Abfrage an den Endpunkt `POST /dali/scan/cancel` geschickt wird.

3.4.2 Abfrage der Geräte

Die Geräte, die vom DALI-2 IoT erfasst wurden, können mit dem `GET /devices` Endpunkt abgefragt werden. Dieser benötigt keine zusätzlichen Angaben und antwortet mit einer Liste der bekannten Geräte, deren Identifikationsnummern (`id`), Namen, Typen, den Zuständen ihrer Features, sowie deren Szenen- und Gruppenzugehörigkeit. Außerdem enthält die Antwort eine Signatur bestehend aus einem Zeitstempel und einem Zähllstand.


```
{
  "devices": [
    {
      "id": 1,
      "name": "DALI #0",
      "address": 0,
      "line": 0,
      "type": "default",
      "features": {
        "switchable": {
          "status": false
        },
        "dimmable": {
          "status": 0
        },
        "scene": true,
        "colorRGB": {
          "status": {
            "r": 1,
            "g": 1,
            "b": 1
          }
        },
                                    →
```

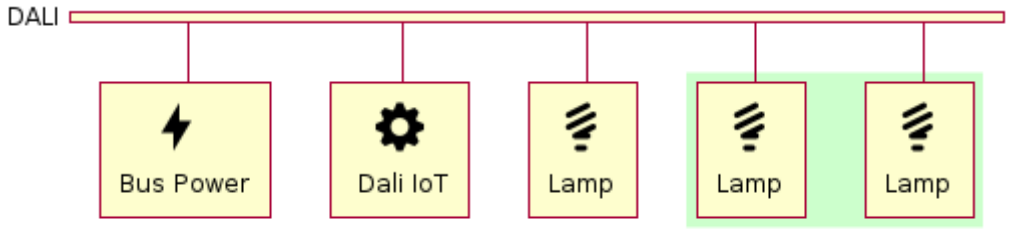
```
←
        "colorKelvin": {
          "status": 2700
        },
        "colorXY": {
          "status": {
            "x": 0,
            "y": 0
          }
        },
        "saveToScene": true,
        "gotoLastActive": {}
      },
      "scenes": [],
      "groups": [],
      "daliTypes": [8]
    },
    { "id": 2, … }, …
  ],
  "timeSignature": {
    "timestamp": 1625747234.3620,
    "counter": 4
  }
}
```


Geräten hatten keine `address`- und `line`-Eigenschaften in Versionen vor `1.2`, in welchen stattdessen ein `info`-Attribut mit einem String-Wert genutzt wurde. Die `daliTypes`-Eigenschaft wurde in Version `1.4` hinzugefügt.

3.4.3 Befehle an Einzelgeräte senden

Geräte können über den `POST /device/{_id}/control` Endpunkt einzeln angesteuert werden. Dabei wird, genauso wie beim Ansteuern aller Geräte, ein `ControlData` Objekt übermittelt. Zusätzlich zum `ControlData` Objekt in der „Request Body“ Eingabe muss allerdings die Identifikationsnummer des Geräts angegeben werden.

3.5 Konfigurieren und Ansteuern von Gruppen



Bevor Gruppen angesteuert werden können, müssen diese zunächst den DALI Geräten zugewiesen werden. Dieser Schritt kann übersprungen werden, falls die Geräte bereits Gruppen hinzugefügt wurden. Sind den Geräten Gruppen zugewiesen, dann können diese gemeinsam mit nur einer Anforderung angesteuert werden.

3.5.1 Zuweisen von Gruppen

Um Gruppen zuzuweisen kann der Endpunkt `PUT /device/{_id}` genutzt werden. Dieser Endpunkt benötigt die Identifikationsnummer eines Geräts, sowie ein `DeviceUpdateModel`, bestehend aus einem (optionalen) Namen für das Gerät und einer (ebenso optionalen) Liste von Gruppennummern (0 bis 15). Um zum Beispiel die Gruppe 0 für ein Gerät einzustellen, muss ein „Request body“ mit dem Schlüssel `groups`, dem eine Liste mit der Nummer 0 zugewiesen ist, an den Endpunkt gesendet werden.

```
{
  "groups": [
      0
  ]
}
```

Der Rückgabewert der API enthält den Zustand des Geräts mit der bereits veränderten Gruppenzugehörigkeit. Diese Antwort ist ähnlich zu Antworten an Anforderungen des `GET /device/{_id}` Endpunkts – allerdings eingeschränkt auf das geänderte Gerät – und beinhaltet deshalb auch die Zustände der Features und die gespeicherten Szenen.

```
{
  "id": 1,
  "name": "DALI #0",
  "address": 0,
  "line": 0,
  "type": "default",
  "features": {
    "switchable": {
      "status": true
    },
    "dimmable": {
      "status": 100
    },
    "scene": true,
    "colorRGB": {
      "status": {
        "r": 1,
        "g": 1,
        "b": 1
      }
    },                                    →
```

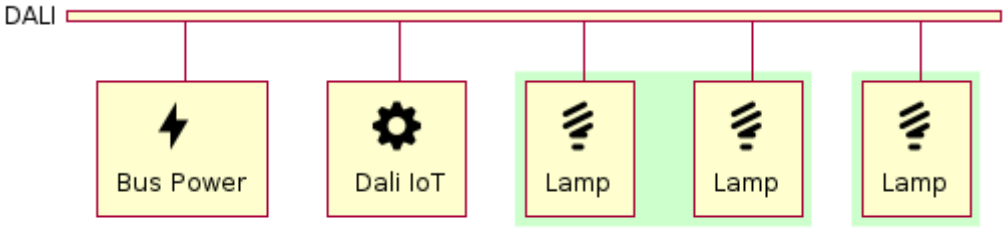
```
←
    "colorKelvin": {
      "status": 2700
    },
    "colorXY": {
      "status": {
        "x": 0,
        "y": 0
      }
    },
    "saveToScene": true,
    "gotoLastActive": {}
  },
  "scenes": [],
  "groups": [0],
  "daliTypes": [8],
  "timeSignature": {
    "timestamp": 1625754320.503,
    "counter": 159
  }
}
```

Geräten hatten keine `address`- und `line`-Eigenschaften in Versionen vor `1.2`, in welchen stattdessen ein `info`-Attribut mit einem String-Wert genutzt wurde. Die `daliTypes`-Eigenschaft wurde in Version `1.4` hinzugefügt.

3.5.2 Ansteuern von Gruppen

Gruppen können über den `POST /group/{_id}/control` Endpunkt angesteuert werden. Dabei wird, genauso wie beim Ansteuern aller Geräte, ein `ControlData` Objekt übermittelt. Zusätzlich zum `ControlData` in der „Request Body“ Eingabe muss allerdings die Gruppennummer der Gruppe angegeben werden.

3.6 Konfigurieren und Ansteuern von Zonen



Zonen sind logische Sammlungen von Gruppen, Geräten und anderen Zonen innerhalb des DALI-2 IoT, die gemeinsam angesteuert werden können. Zonen müssen konfiguriert werden, bevor sie angesteuert werden können, und sind durch die folgenden Eigenschaften charakterisiert:

- eine Identifikationsnummer (`id`),
- ein optionaler Namen (`name`), und
- eine Liste von Zielobjekten, die gemeinsam angesteuert werden (`targets`).

Zielobjekte werden durch das `DeviceModel` (Abschnitt 3.2) angegeben.

3.6.1 Hinzufügen von Zonen

Um eine neue Zone hinzuzufügen, kann der `POST /zone` Endpunkt benutzt werden. Hierbei müssen nur die Zielobjekte der Zone übertragen werden. Als Beispiel wird eine Zone, die sich aus Gruppe 0 und Gerät 1 zusammensetzt, angegeben.

```
{
  "targets": [
    {
      "type": "group",
      "id": 0
    },
    {
      "type": "device",
      "id": 1
    }
  ]
}
```

3.6.2 Abfragen von Zonen

Die Zonen, die im DALI-2 IoT erzeugt wurden, können mit dem `GET /zones` Endpunkt abgefragt werden. Dieser benötigt keine zusätzlichen Angaben und antwortet mit einer Liste der bekannten Zonen, deren Identifikationsnummern (`id`), Namen (`name`) und Zielobjekten (`targets`). Außerdem enthält die Antwort eine Signatur bestehend aus einem Zeitstempel und einem Zählstand.

```
{
  "zones": [,
    {
      "id": 1,
      "name": null,
      "targets": [
```

```
        {
          "type": "group",
          "id": 0
        },
        {
          "type": "device",
          "id": 1
        }
      ]
    }
  ],
  "timeSignature": {
    "timestamp": 1644507239.4869525,
    "counter": 342
  }
}
```


3.6.3 Ändern von Zonen

Um Zonen zu verändern kann der Endpunkt `PUT /zone/{_id}` genutzt werden. Dieser Endpunkt benötigt die Identifikationsnummer einer Zone, sowie die Werte, die geändert werden sollen. Es ändern sich nur jene Werte, die in der Anfrage übergeben werden. Um zum Beispiel den Namen einer Zone zu ändern muss ein „Request body“ mit dem Schlüssel `name`, dem der neue Name zugewiesen ist, an den Endpunkt gesendet werden.

```
{
  "name": "Group 0 and device 1"
}
```

Der Rückgabewert der API enthält den Zustand der Zone mit den geänderten Werten. Diese Antwort ist äquivalent zu Antworten an Anforderungen des `GET /zone/{_id}` Endpunkts.


```
{
  "id": 1,
  "name": "Group 0 and device 1",
  "targets": [
    {
      "type": "group",
      "id": 0
    },
    {
      "type": "device",
      "id": 1
    }
  ],
  "timeSignature": {
    "timestamp": 1644507239.4869525,
    "counter": 342
  }
}
```


3.6.4 Ansteuern von Zonen

Zonen können über den `POST /zone/{_id}/control` Endpunkt angesteuert werden. Dabei wird, genauso wie beim Ansteuern aller Geräte, ein `ControlData` Objekt übermittelt. Zusätzlich zum `ControlData` in der „Request Body“ Eingabe muss allerdings die Identifikationsnummer der Zone angegeben werden.

4 Automatisierungen

Automatisierungen können genutzt werden, um komplexe Abfolgen von Befehlen zu editieren, speichern und durchzuführen. Mehrere verschiedene Automatisierungen stehen dazu zur Verfügung: Sequenzen, Zeit und Wochentag gesteuerte Befehle (Schedules), und Circadiane Tageslichtverläufe.

4.1 Sequenzen

Sequenzen sind automatische Wiedergaben von Befehlsfolgen und sind durch die folgenden Eigenschaften charakterisiert:

- eine Identifikationsnummer (`id`),
- eine Reihe von "Schritten" (`steps`),
- einem Parameter um die Sequenz zuzulassen (`enabled`),
- einem optionalen Namen (`name`),
- einem optionalen Parameter für endlose Sequenzen (`loop`) und
- einem optional Parameter für eine endliche Anzahl an Wiedergaben (`repeat`) charakterisiert.

Jeder Schritt in einer Sequenz besteht außerdem aus einem Parameter (`enabled`) um den Schritt zuzulassen, einem Typ (`type`), einer Aktion (`data`) und einer optionalen Zeitverzögerung in Sekunden (`delay`), bevor der Schritt durchgeführt wird. Für den Typ eines Schritts ist derzeit nur `features` verfügbar. Die Aktion eines `features`-Schritts setzt sich aus einer Liste von `DeviceModel` (`targets`), die kontrolliert werden sollen, und deren Features (`features`) zusammen. Da der Parameter `enabled` standardmäßig auf `true` gesetzt wird kann dieser ausgelassen werden.

Als Beispiel wird ein einzelner Schritt, der nach zwei Sekunden alle Geräte ausschaltet, angegeben.

```
{
  "type": "features",
  "data": {
    "targets": [
     {
        "type": "broadcast"
      }
    ],
    "features": {
      "switchable": false
    }
  },
  "delay": 2
}
```

4.1.1 Hinzufügen von Sequenzen

Um eine neue Sequenz hinzuzufügen, kann der `POST /automations/sequence` Endpunkt benutzt werden. Hierbei muss die komplette Sequenz mit allen Schritten übertragen werden, wobei „`enabled`“-Felder ausgelassen werden können (wird standardmäßig auf `true` gesetzt). Als Beispiel wird eine endlose Sequenz aus Ein- und Ausschalt-Befehlen, die sich an alle Geräte richten, gezeigt. Zwischen den Befehlen ist eine Verzögerung von 2 s.

```
{
  "name": "blink_slowly",
  "loop": true,
  "steps": [
    {
      "type": "features",
      "data": {
        "targets": [
          {
            "type": "broadcast"
          }
        ],
        "features": {
          "switchable": true
        }
      },
      "delay": 2
                                                    →
```

```
←
    },
    {
      "type": "features",
      "data": {
        "targets": [
          {
            "type": "broadcast"
          }
        ],
        "features": {
          "switchable": false
        }
      },
      "delay": 2
    }
  ],
}
```


4.1.2 Abfragen von Sequenzen

Alle bereits gespeicherten Sequenzen können mit dem `GET /automations/sequences` Endpunkt abgerufen werden. Hierbei wird eine zusätzliche Angabe `active` übertragen, die anzeigt ob die Sequenz derzeit ausgeführt wird (`true`) oder nicht (`false`).


```
{
  "sequences": [
    {
      "name": "blink_slowly",
      "enabled": true,
      "loop": true,
      "repeat": 0,
      "steps": [
        {
          "enabled": true,
          "type": "features",
          "data": {
            "targets": [
              {
                "type": "broadcast"
              }
            ],
            "features": {
              "switchable": true
            }
          },
          "delay": 2
                                                    →
```

```
←
        },
        {
          "enabled": true,
          "type": "features",
          "data": {
            "targets": [
              {
                "type": "broadcast"
              }
            ],
            "features": {
              "switchable": false
            }
          },
          "delay": 2
        }
      ],
      "id": 1,
      "active": false
    }
  ]
}
```


Einzelne Sequenzen können mit dem `GET /automations/sequence/{_id}` Endpunkt abgefragt werden. Hierbei muss die Identifikationsnummer der Sequenz eingesetzt werden.

4.1.3 Starten und Stoppen von Sequenzen

Sequenzen müssen von Hand gestartet und gestoppt werden. Um eine Sequenz zu starten, kann der Endpunkt `POST /automations/sequence/{_id}/start` genutzt werden, wobei die Identifikationsnummer der Sequenz eingesetzt werden muss. Es werden keine Daten als „Request Body“ übertragen. Eine laufende Sequenz kann mit einer Anforderung an den `POST /automations/sequence/{_id}/stop` angehalten werden.

4.1.4 Ändern von Sequenzen

Um Sequenzen zu ändern kann der `PUT /automations/sequence/{_id}` Endpunkt genutzt werden, wobei die Identifikationsnummer der Sequenz eingesetzt werden muss. Die Struktur des „Request Body“ ist gleich wie die Struktur zum Hinzufügen neuer Sequenzen (Abschnitt 4.1.1), allerdings sind bei Änderungen alle Parameter optional: nur diejenigen Parameter, die geändert werden sollen, müssen auch angegeben werden.

Soll zum Beispiel eine endlose Sequenz nur noch 5 mal abgespielt werden, dann reicht es aus, die `loop` und `repeat` Eigenschaften der Sequenz anzugeben.

```
{
  "loop": false,
  "repeat": 5
}
```

Wichtig zu beachten: die Schritte einer Sequenz gelten als ein einziger Parameter. Es müssen daher auch dann alle Schritte vollständig übertragen werden, wenn nur ein Schritt geändert wird.

4.2 Zeit und Wochentag gesteuerte Befehle (*Schedules*)

Schedules sind durch Zeit und Wochentag gesteuerte Befehle und sind durch die folgenden Eigenschaften charakterisiert:

- eine Identifikationsnummer (`id`),
- einer auszuführenden Funktion (`action`),
- einer Liste von `DeviceModels` auf die die Funktion angewandt wird (`targets`),
- einer Uhrzeit zu der die Funktion geplant ist (`recallTime`),
- einem Aufrufmodus um die Zeitplanung zu kontrollieren (`recallMode`),
- einem Parameter um das Schedule zuzulassen (`enabled`),
- einem optionalen Namen (`name`), und
- optionalen aktiven Perioden (`activePeriod`), Monaten (`activeMonths`), Wochentagen (`activeWeekdays`) oder Tagen im Monat (`activeDays`) charakterisiert.

Gemeinsam bestimmen die Aufrufzeit (recallTime) und der Aufrufmodus (recallMode) eines Schedules, zu welcher Uhrzeit die zeitgesteuerten Befehle angewandt werden sollen. Die Aufrufzeit ist eine Zeitangabe in Stunden, Minuten und Sekunden; und wird je nach Aufrufmodus anders ausgewertet.

```
"recallTime": {
  "hour": 6,
  "minute": 30,
  "second": 0
}
```

Unterstützt werden die folgenden Aufrufmodi:

- `timeOfDay`: Die Befehle werden zur Aufrufzeit angewandt.
- `beforeSunrise`: Die Befehle werden um die Aufrufzeit versetzt vor dem Sonnenaufgang angewandt.
- `afterSunrise`: Die Befehle werden um die Aufrufzeit versetzt nach dem Sonnenaufgang angewandt.

- `beforeSunset`: Die Befehle werden um die Aufrufzeit versetzt vor dem Sonnenuntergang angewandt.
- `afterSunset`: Die Befehle werden um die Aufrufzeit versetzt vor nach Sonnenuntergang angewandt.

Wichtig: Sonnenauf- und -untergang werden anhand des konfigurierten Orts berechnet. Außerdem hängen alle Aufrufmodi von der Uhrzeit des DALI-2 IoT ab. Für einen korrekten Betrieb sollten Zeit und Ort des Geräts eingestellt werden (Abschnitte 5.2 und 5.3).

Die optionalen Parameter für aktive Perioden, Monate, Wochentage und Tage im Monat können einschränken an welchen Tagen die zeitgesteuerten Befehle angewandt werden sollen. Eine aktive Periode setzt sich aus einem Start- und End-Monat, sowie -Tag zusammen und umfasst alle Tage zwischen Start und Ende. An Tagen außerhalb der aktiven Periode wird das Schedule nicht ausgeführt. Für aktive Monate (und Wochentage) werden den Monatsnamen (bzw. Namen der Wochentage) Boolean-Werte zugewiesen, die angeben ob ein Monat (bzw. Wochentag) aktiv (`true`) oder inaktiv (`false`) ist. Aktive Tage innerhalb eines Monats können über die Nummer des Tags (von 1 bis 31) angegeben werden. Hierbei wird an jenen Tagen, die nicht als aktive Tage angegeben werden, das Schedule nicht ausgeführt.

```
"activePeriod": {
  "startDay": 24,
  "startMonth": 5,
  "endDay": 24,
  "endMonth": 9
}
```

```
"activeMonths": {
  "january": false,
  "february": false,
  "march": true,
  …
}
```

```
"activeWeekDays": {
  "monday": false,
  "tuesday": false,
  "wednesday": true,
  …
}
```

```
"activeDays": {
  "days": [
    1,
    2, …
  ]
}
```

4.2.1 Hinzufügen von Schedules

Um einen neuen Schedule hinzuzufügen, kann der `POST /automations/scheduler` Endpunkt benutzt werden. Hierbei muss der komplette Schedule mit dessen Uhrzeit (`recallTime`), den `DeviceModels` (`targets`), Aufrufmodus (`recallMode`) und die Aktion (`action`) angegeben werden, wobei das `enabled`-Feld ausgelassen werden kann (wird standardmäßig auf `true` gesetzt).

Als Beispiel wird ein Schedule gezeigt, der an allen Wochentagen die Gruppe 0 einschaltet. Da die Parameter `activePeriod`, `activeMonths`, `activeWeekdays` und `activeDays` optional sind, müssen diese nicht angegeben werden. Im Beispiel werden Wochenendtage deaktiviert; alle anderen Tage sind standardmäßig aktiv und müssen daher auch nicht angegeben werden.

```
{
  "name": "lights-on",
  "targets": [
    {
      "type": "group",
      "id": 0
    }
  ],
```

```
←
  "recallTime": {
    "hour": 6
    "minute": 30
  },
  "action": {
    "type": "features",
    "data": {
```

```
  "activeWeekdays": {
    "saturday": false,
    "sunday": false
  },
  "recallMode": "timeOfDay",
                                    →
      "features": {
        "switchable": true
      }
    }
  }
}
```

4.2.2 Abfragen von Schedules

Alle bereits gespeicherten Schedules können mit dem `GET /automations/schedules` Endpunkt abgerufen werden.

```
{
  "schedulers": [
    {
      "name": "lights-on",
      "targets": [
        {
          "type": "group",
          "id": 0
        }
      ],
      "enabled": true,
      "activePeriod": {},
      "activeMonths": {},
      "activeWeekdays": {
        "monday": true,
        "tuesday": true,
        "wednesday": true,
        "thursday": true,
        "friday": true,
        "saturday": false,
                                    →
←       "sunday": false
      },
      "activeDays": {},
      "recallMode": "timeOfDay",
      "recallTime": {
        "hour": 6,
        "minute": 30,
        "second": null
      },
      "action": {
        "type": "features",
        "data": {
          "features": {
            "switchable": true
          }
        }
      },
      "id": 1
    }
  ]
}
```

Einzelne Schedules können mit dem `GET /automations/scheduler/{_id}` Endpunkt abgefragt werden. Hierbei muss die Identifikationsnummer des Schedules eingesetzt werden.

4.2.3 Ändern von Schedules

Um Schedules zu ändern kann der `PUT /automations/scheduler/{_id}` Endpunkt genutzt werden, wobei die Identifikationsnummer des Schedules eingesetzt werden muss. Die Struktur des „Request Body“ ist gleich wie die Struktur zum Hinzufügen neuer Schedules (Abschnitt 4.2.1), allerdings bei Änderungen alle Parameter optional: nur diejenigen Parameter, die geändert werden sollen, müssen auch angegeben werden.

Soll zum Beispiel ein Schedule nur noch Samstags aktiv sein, dann reicht es aus die „`activeWeekdays`“ Eigenschaft zu ändern.

```
{
  "activeWeekdays": {
    "saturday": true
  }
}
```

4.3 Circadiane Tageslichtverläufe

Circadiane Tageslichtverläufe simulieren den Verlauf natürlichen Tageslichts durch kontinuierliche Anpassung von Farbtemperatur und Helligkeit und sind durch die folgenden Eigenschaften charakterisiert:

- eine Identifikationsnummer (`id`),
- zwei Tageslichtkurven für den längsten (`longest`) und kürzesten (`shortest`) Tag des Jahres,
- eine Liste von `DeviceModels` auf die der Verlauf angewandt wird (`targets`),
- einem Parameter um den Verlauf zuzulassen (`enabled`) und
- einem optionalen Namen (`name`) charakterisiert.

Die Objekte für die beiden Tageslichtkurven bestehen aus einem Tag (`day`), Monat (`month`) und einer Liste an Schritten, die die Kurve beschreiben. Jeder Schritt enthält dabei eine Stunde (`hour`), einen optionalen Helligkeitswert (`dimmable`) und eine ebenso optionale Farbtemperatur (`colorKelvin`). Helligkeits- und Farbwerte werden minütlich über die angegeben Zeitpunkte in den Kurven, sowie über das Jahr hinweg zwischen dem kürzesten und längsten Verlauf, interpoliert.

```
{
  "hour": 12,
  "colorKelvin": 5800,
  "dimmable": 20
}
```

Wichtig: Für einen korrekten Betrieb muss die Uhrzeit des DALI-2 IoT eingestellt sein (Abschnitt 5.2).

4.3.1 Hinzufügen von Circadianen Tageslichtverläufen

Um einen neuen Circadianen Tageslichtverlauf hinzuzufügen, kann der `POST /automations/circadian` Endpunkt benutzt werden. Hierbei muss der komplette Verlauf, mit dessen `DeviceModels` (`targets`), sowie dem längsten (`longest`) und kürzesten (`shortest`) Tag im Jahr angegeben werden, wobei „`enabled`“-Felder ausgelassen werden können (werden standardmäßig auf `true` gesetzt).

```
{
  "targets": [
    {
      "type": "device",
      "id": 2
    }
  ],
  "longest": {
    "day": 21,
    "month": 6,
    "steps": [
      {
        "hour": 0,
        "colorKelvin": 2700
      },
      …
      {
        "hour": 23,
```

```
←
  },
  "shortest": {
    "day": 21,
    "month": 12,
    "steps": [
      {
        "hour": 0,
        "colorKelvin": 2700
      },
      {
        "hour": 1,
        "colorKelvin": 2700
      },
      …
      {
        "hour": 23,
        "colorKelvin": 2700
```

```
        "colorKelvin": 2700
      }
    ]
                                    →
```

```
      }
    ]
  }
}
```

Die Werte in den Code-Beispielen zu circadianen Tageslichtverläufen wurden zur besseren Lesbarkeit abgekürzt. Ein kompletter Beispielverlauf für die Steuerung von Farbtemperaturen ist in Tabelle 1 angegeben.

Tabelle 1: Eine Beispielkurve für Farbtemperaturverläufe.

Stunde	Temperatur [K]	Stunde	Temperatur [K]	Stunde	Temperatur [K]
0	2700	8	4301	16	4101
1	2700	9	4767	17	3412
2	2700	10	5318	18	2700
3	2700	11	5685	19	2700
4	2700	12	5800	20	2700
5	2700	13	5685	21	2700
6	2700	14	5218	22	2700
7	3412	15	4767	23	2700

4.3.2 Abfragen von Circadianen Tageslichtverläufen

Alle bereits gespeicherten Verläufe können mit dem `GET /automations/circadians` Endpunkt abgerufen werden.

```
{
  "circadians": [
    {
      "name": "New",
      "targets": [
        {
          "type": "device",
          "id": 2
        }
      ],
      "longest": {
        "day": 21,
        "month": 6,
        "steps": [
          {
            "hour": 0,
            "colorKelvin": 2700
          },
          {
            "hour": 1, …
          },
          …
        ]
      },
                                    →
```

```
←
      "shortest": {
        "day": 21,
        "month": 12,
        "steps": [
          {
            "hour": 0,
            "colorKelvin": 2700
          },
          {
            "hour": 1,
            "colorKelvin": 2700
          },
          {
            "hour": 2,
            "colorKelvin": 2700
          },
          …
        ]
      },
      "enabled": true,
      "id": 1
    }
  ]
}
```

Einzelne Verläufe können mit dem `GET /automations/circadian/{_id}` Endpunkt abgefragt werden. Hierbei muss die Identifikationsnummer des Verlaufs eingesetzt werden.

4.3.3 Ändern von Circadianen Tageslichtverläufen

Um Circadiane Tageslichtverläufe zu ändern kann der `PUT /automations/circadian/{_id}` Endpunkt genutzt werden, wobei die Identifikationsnummer des Verlaufs eingesetzt werden muss. Die Struktur des „Request Body“ ist gleich wie die Struktur zum Hinzufügen neuer Verläufe (Abschnitt 4.3.1), allerdings sind bei Änderungen alle Parameter optional: nur diejenigen Parameter, die geändert werden sollen, müssen auch angegeben werden.

Dabei muss beachtet werden, dass der gesamte Parameter angegeben wird. Will man zum Beispiel das Datum des kürzesten Tags ändern, dann müssen auch alle Schritte der `shortest` Kurve neu übertragen werden.

```
"shortest": {
  "day": 22,
  "month": 12,
    "steps": [
      {
        "hour": 0,
        "colorKelvin": 2700
      },
      …
    ]
}
```

4.4 Trigger Actions

Trigger Actions sind automatische Weiterleitungen von DALI Befehlen und charakterisiert durch die folgenden Eigenschaften:

- eine Identifikationsnummer (`id`),
- einem optionalen Namen (`name`),
- einer Liste von Auslöser-Adressen, von denen Befehle weitergeleitet werden (`sources`),
- einer Liste von Ziel-Adressen, an die Befehle weitergeleitet werden (`targets`), und
- einem Parameter um den Verlauf zuzulassen (`enabled`)

Ziele-Adressen sind durch das `DeviceModel` definiert (Abschnitt 3.2). Auslöser-Adressen sind durch das `TriggerActionSource` Modell definiert, welches einen Typ (`type`) und optionale Parameter, die vom Typ abhängen, hat. Der Typ einer Auslöser-Adresse kann ein erfasstes Gerät (`device`), eine Gruppe (`group`), eine DALI Adresse (`d16gear`) oder eine DALI Gruppe (`d16group`) sein. Geräte und Gruppen benötigen eine zusätzliche Identifikationsnummer (`id`). DALI Adressen und DALI Gruppen benötigen eine zusätzliche Adresse (`address`) und eine Linie (`line`). Da das DALI-2 IoT nur an eine DALI-Linie angeschlossen werden kann muss der Linie-Parameter null (`0`) sein.

Der `d16gear` Typ erlaubt es eine Adresse als Auslöser für eine automatische Weiterleitung zu nutzen, die von keinem DALI Gerät genutzt wird. Zonen und Broadcasts werden derzeit nicht als Auslöser unterstützt.

```
{
  "type": "device",
  "id": 1
}
```

```
{
  "type": "group",
  "id": 1
}
```

```
{
  "type": "d16gear",
  "address": 1,
  "line": 0
}
```

```
{
  "type": "d16group",
  "address": 1,
  "line": 0
}
```

4.4.1 Hinzufügen von Trigger Actions

Der Endpunkt `POST /automations/triggerAction` kann genutzt werden um automatische Weiterleitungen hinzuzufügen. Dieser benötigt zumindest einen Aulöser und ein Ziel. Als Beispiel wird eine Trigger Action gezeigt, die Befehle von Gerät 1 an die Zone 1 weiterleitet.

```
{
  "enabled": true,
  "name": "",
  "sources": [
    {
      "type": "device",
      "id": 1
    }
  ],
  "targets": [
    {
      "type": "zone",
      "id": 1
    }
  ]
}
```

4.4.2 Abfragen von Trigger Actions

Alle bekannten Trigger Actions können mit dem Endpunkt `GET /automations/triggerActions` abgefragt werden.

```
{
  "triggerActions": [
    {
      "id": 1,
      "enabled": true,
      "name": "",
      "sources": [
        {
          "type": "device",
          "id": 1,
          "address": null,
          "line": null
        }
      ],
      "targets": [
        {
          "type": "zone",
          "id": 1
        }
      ]
    }
  ]
}
```

Einzelne Trigger Actions können mit dem Endpunkt `GET /automations/triggerAction/{_id}` abgefragt werden. Hierbei muss die Identifikationsnummer der Trigger Action eingesetzt werden.

4.4.3 Ändern von Trigger Actions

Um Trigger Actions zu ändern kann der `PUT /automations/triggerAction/{_id}` Endpunkt genutzt werden, wobei die Identifikationsnummer eingesetzt werden muss. Die Struktur des „Request Body“ ist gleich wie die Struktur zum Hinzufügen neuer Trigger Actions (Abschnitt 4.4.1), allerdings sind bei Änderungen alle Parameter optional: nur diejenigen Parameter, die geändert werden sollen, müssen auch angegeben werden.

```
{
  "name": "new name for the automation"
}
```

5 Allgemeine Einstellungen des DALI-2 IoT

5.1 Allgemeine Informationen des DALI-2 IoT

Allgemeine Informationen über das DALI-2 IoT können mit dem Endpunkt `GET /info` abgefragt werden. Die Antwort auf diese Abfrage enthält

- den einstellbaren Namen des DALI-2 IoT (`name`),
- die Software und Firmware Versionen des Geräts (`version`),
- die freigeschaltenen Features des Geräts (`tier` und `emergencyLight`),
- Fehlerzustände, die zum Beispiel ein Problem mit dem DALI Bus berichten (`errors`),
- eine Beschreibung des Hardware Interface (`descriptor`), und
- Informationen über das Gerät selbst, wie z.B. Serien- und Artikelnummern (`device`).

```
{
  "type": "info",
  "data": {
    "name": "dali-iot",
    "version": "v1.2.0/1.0.9",
    "tier": "plus",
    "emergencyLight": true,
    "errors": {},
    "descriptor": {
      "lines": 1,
      "bufferSize": 32,
      "tickResolution": 1978,
      "maxYnFrameSize": 32,
      "deviceListSpecifier": true,
      "protocolVersion": "1.0"
    },
    "device": {
      "serial": 1234567890,
      "gtin": 1234567890,
      "pcb": "9a",
      "articleNumber": 1234567890,
      "articleInfo": "",
      "productionYear": 2021,
      "productionWeek": 31
    }
```

```
  },
  "timeSignature": {
    "timestamp": 1644577937.2377043,
    "counter": 17
  }
}
```

Um den benutzerdefinierten Namen einzustellen kann der `PUT /info` Endpunkt genutzt werden. Hierbei muss der neue Name übertragen werden.

```
{
  "name": "Neuer Name für das Gerät"
}
```

5.2 Zeitzonen und Uhrzeit

Zeitzone und Uhrzeit müssen für einen korrekten Betrieb der zeitgesteuerten Automatisierungen eingestellt werden. Falls das DALI-2 IoT in einem Netzwerk mit Internetzugang betrieben wird, kann die Uhrzeit automatisch erkannt werden. Hierfür muss die korrekte Zeitzone eingestellt sein.

Um Uhrzeit und Zeitzone des DALI-2 IoT einzustellen kann der `POST /datetime` Endpunkt genutzt werden. Dabei können Zeitzone (`timezone`), automatischer Bezug der Zeit über das Internet (`automatic_time`), Datum (`date`) und Uhrzeit (`time`) angegeben werden. Die einzelnen Parameter sind optional. Nur Parameter, die auch angegeben wurden, werden geändert.

```
{
  "timezone": "Europe/Vienna",
  "automatic_time": true,
  "date": "28. October 2021",
  "time": "15:26"
}
```

Zeitzonen werden als Region/Ort angegeben. Um eine Liste an bekannten Zeitzonen auszugeben kann der Endpunkt `GET /datetime/timezones` genutzt werden.

```
{
  "timezones": [
    "Africa/Abidjan",
    "Africa/Accra",
    …
}
```

Wenn der automatische Bezug der Uhrzeit aktiviert ist, dann können Datum und Uhrzeit ausgelassen werden. Andernfalls müssen beide Werte eingestellt werden. Die eingestellten Zeitzonen und die aktuelle Uhrzeit können über den Endpunkt `GET /datetime` abgefragt werden.

5.3 Betriebsort

Der ungefähre Betriebsort muss für einen korrekten Betrieb der zeitgesteuerten Automatisierungen eingestellt werden, wenn Sonnenauf- oder -untergang zur Steuerung verwendet werden. Falls das DALI-2 IoT in einem Netzwerk mit Internetzugang betrieben wird, kann der Ort automatisch erkannt werden. Um den Ort automatisch zu erkennen und zu konfigurieren muss eine Abfrage an den Endpunkt `POST /location/detect` gesendet werden. Dieser Endpunkt benötigt keine zusätzlichen Angaben und gibt die ungefähren Koordinaten des erkannten Orts zurück.

```
{
  "lat": 48.20849,
  "lon": 16.37208
}
```

Wenn das DALI-2 IoT über keinen Internetzugang verfügt, dann müssen diese Koordinaten von Hand eingestellt werden, indem sie an den Endpunkt `POST /location` gesendet werden. Die eingestellten Koordinaten können über den Endpunkt `GET /location` abgefragt werden.

5.4 Netzwerkeinstellungen

Die Netzwerkeinstellungen des DALI-2 IoT können über `POST /ethernet` geändert werden. Dabei kann eingestellt werden ob ein DHCP-Server für die Zuweisung von IP-Adresse und anderen Parametern genutzt wird oder nicht (`dhcp`). Falls das DHCP-Protokoll nicht genutzt wird, dann müssen eine statische IP-Adresse (`ip_address`) und Subnetzmaske (`subnet_mask`) eingerichtet werden. Außerdem können Gateway (`gateway`) und DNS Nameserver (`nameservers`) angegeben werden.

```
{
  "dhcp": false,
  "ip_address": "10.0.0.73",
  "subnet_mask": "255.255.255.0",
  "gateway": "10.0.0.1",
  "nameservers": [
    "10.0.0.1"
  ]
}
```

Wichtig: Änderungen der Netzwerkeinstellungen sollten vorher notiert und mit besonderer Vorsicht vorgenommen werden. Wird eine statische IP-Adresse eingestellt, dann reagiert das DALI-2 IoT nur noch auf diese Adresse. Falls keine automatische Adresse über DHCP bezogen werden kann, dann nutzt das DALI-2 IoT als Rückgriff die Adresse 169.254.0.1 mit der Subnetzmaske 255.255.0.0.

Die Netzwerkeinstellungen können über den `GET /ethernet` Endpunkt abgefragt werden. Zusätzlich werden die MAC-Adresse des DALI-2 IoT und, falls das DHCP Protokoll genutzt wird, der Ablauf des derzeitigen DHCP Leasingobjekts (`dhcp_lease`), angegeben.

```
{
  "mac_address": "AA:BB:CC:DD:EE:FF",
  "settings": {
    "dhcp": false,
    "ip_address": "10.0.0.73",
    "subnet_mask": "255.255.255.0",
    "gateway": "10.0.0.1",
    "nameservers": [
      "10.0.0.1"
    ]
  },
  "dhcp_lease": null
}
```

5.5 Email

Email Einstellungen wurden in Version `1.3` hinzugefügt und werden für Benachrichtigungen bei automatischen Notlicht-Tests in einer zukünftigen Version genutzt.

Email Einstellungen können mit dem `PUT /email` geändert werden. Um Berichte über email zu versenden ist es nötig, die Einstellungen (`mailConfig`) einmalig einzurichten. Die Einstellungen für das Senden von emails enthalten die Details eines SMTP Servers (`server`, `port`, `security`), Logindaten (`username`, `password`), sowie den Namen (`senderName`) und die Email Adresse (`senderEmail`) des Senders. Drei verschiedene `security`-Modi werden für den Login für den

SMTP-Server unterstützt: Klartext (`"none"`), SSL (`"sslTls"`) und and STARTTLS (`"startTls"`). Es wird empfohlen einen eigenen, dem DALI-2 IoT zugehörigen, Email-Account einzurichten und ein separates „App“-Passwort für das Versenden von Nachrichten zu nutzen.

```
{
  "mailConfig": {
    "server": "smtp.example.com",
    "port": 25,
    "security": "startTls",
    "username": "username",
    "password": "password",
    "senderName": "Sender Name",
    "senderEmail": "sender@example.com"
  },
  "notifications": {…}
}
```

Zusätzlich zu den Einstellungen für den SMPT-Server gibt es Einstellungen für Benachrichtigungen (`notifications`). In den Benachrichtigungseinstellungen kann für jede Art von Notlicht-Test eingestellt werden, ob eine Benachrichtigung beim Erfolg des Tests (`sendOnSuccess`) oder bei einem Fehlerfall (`sendOnFailure`) gesendet werden soll. Benachrichtigungen werden an eine vorher eingestellte Liste von Empfängern gesendet.

```
{
  "mailConfig": {…},
  "notifications": {
    "functionTest": {
      "sendOnSuccess": true,
      "sendOnFailure": true
    },
    "functionTest": {
      "sendOnSuccess": true,
      "sendOnFailure": true
    },
    "functionTest": {
      "sendOnSuccess": true,
      "sendOnFailure": true
    },
    "mailReceivers": [
      "receiver@example.com"
    ]
  }
}
```

Email Einstellungen können getestet werden, indem eine Abfrage an den Endpunkt `POST /email` gesendet wird. Diese Abfrage benötigt keine zusätzlichen Eingaben. Der Endpunkt `GET /email` kann genutzt werden um die aktuellen Email-Einstellungen abzufragen.

6 Websocket Interface

6.1 Allgemeine Kommunikation

Ereignisse, wie zum Beispiel die Veränderung des Lichtpegel eines Geräts, werden in DALI IoT über Websocket-Verbindungen signalisiert. Hierdurch können mehrere Clients eine dauerhafte Verbindung mit dem DALI IoT aufrecht erhalten, vom DALI IoT ausgehend über Zustandsänderungen informiert werden und darauf reagieren.

Um die ausgesendeten Ereignisse zum Beispiel über ein simples Kommandozeilen-Programm auszugeben, reichen bereits wenige Zeilen Python-Code aus (`<IP_ADDRESS>` muss durch die IP Adresse des DALI IoT ersetzt werden).

```python
import asyncio
import websockets

async def receive_message(websocket):
    async for message in websocket:
        print(message)

async def run_client(url):
    try:
        async with websockets.connect(url) as websocket:
            wait_task = asyncio.create_task(
                receive_message(websocket)
            )
            await asyncio.wait([wait_task])
            wait_task.result()
    except:
        pass

# insert ip address or host name
asyncio.run(run_client("ws://<IP_ADDRESS>"))
```

Alle Websocket Ereignisse übertragen ein Paket im JSON Format, mit Feldern für den Ereignistyp (`type`), ein vom Ereignistyp abhängiges Datenfeld (`data`) und eine Signatur bestehend aus einem Zeitstempel und einem Zählstand, um Ereignisse zu unterscheiden (`timeSignature`). Derzeit werden folgende Ereignistypen unterstützt: DALI-Scan Fortschritt (`scanProgress`), Geräte-Ereignisse (`devices` und `devicesDeleted`), DALI Bus Monitor-Ereignisse (`daliMonitor`), Einblendungen allgemeiner Statusnachrichten (`messageFlash`) und Ereignisse zum Testen von Verbindungen (`ping`). Einzelheiten zu den jeweiligen Ereignistypen und deren Datenfeldern werden in den folgenden Abschnitten erklärt.

6.1.1 Begrüßungsnachricht (*info*)

Wenn eine Websocket-Verbindung erstmalig zustande kommt sendet das DALI-2 IoT eine Begrüßungsnachricht mit Informationen über das Gerät aus. Das `data` Feld dieser Nachricht ist äquivalent zur Antwort auf eine Abfrage des `GET /info` Endpunkts (Abschnitt 5.1).

```
{
  "type": "info",
```

```
  "data": {
    "name": "dali-iot",
    "version": "v1.2.0/1.0.9",
    "tier": "plus",
    "emergencyLight": true,
    "errors": {},
    "descriptor": {
      "lines": 1,
      "bufferSize": 32,
      "tickResolution": 1978,
      "maxYnFrameSize": 32,
      "deviceListSpecifier": true,
      "protocolVersion": "1.0"
    },
    "device": {
      "serial": 1234567890,
      "gtin": 1234567890,
      "pcb": "9a",
      "articleNumber": 1234567890,
      "articleInfo": "",
      "productionYear": 2021,
      "productionWeek": 31
    }
  },
  "timeSignature": {
    "timestamp": 1644577937.2377043,
    "counter": 17
  }
}
```


6.1.2 Ausblenden von Ereignistypen (*filtering*)

Jede Websocket-Verbindung zum DALI-2 IoT kann durch ein Websocket-Ereignis mit dem Typ „`filtering`“ festlegen, welche Ereignistypen das DALI-2 IoT über die Verbindung senden oder filtern soll. Hierfür muss vom Client ein `filtering`-Ereignis an das DALI-2 IoT gesendet werden. Jedem Ereignistyp wird ein Boolean-Wert zugeordnet, der angibt ob der Typ gefiltert (`true`) oder nicht gefiltert (`false`) werden soll. Gefilterte Ereignisse werden nicht mehr über die Verbindung gesendet, bis der Filter vom Client ausgehend wieder aufgehoben wird. Bei einer neuen Verbindung sind keine Filter konfiguriert.

```
{
  "type": "filtering",
  "data": {
    "daliMonitor": true,
    "fileUpload": false
  }
}
```

6.2 Direkter Zugang zum DALI Bus

6.2.1 DALI Bus Zustandsereignisse (*daliStatus*)

DALI Zustandsereignisse (daliStatus) werden vom Hardware Interface ausgesendet, wenn sich der Zustand des DALI Bus verändert. Der Zustandscode (status) ist ein Integer Code, der in Tabelle 2 erläutert wird.

```
{
  "type": "daliStatus",
```

```
  "data": {
    "status": 0,
    "line": 0
}
```

Tabelle 2: Status codes in `daliStatus` Ereignissen.

Status Value	Description
0	Der DALI Bus wird nicht mehr mit Strom versorgt.
1	Systemfehler im Hardware Interface.
2	Der DALI Bus wird mit Strom versorgt.
3	Der Sendepuffer des Hardware Interface ist voll.
4	Der Sendepuffer des Hardware Interface ist leer.
5	Die Stromversorgung des DALI Bus ist niedrig.
60	Ein Makro wurde durch Timeout beendet oder abgebrochen.
61	Ein Makro hat ein Zwischenereignis gesendet.
62	Ein Makro wurde mit einem Fehler beendet.
63	Ein Makro wurde erfolgreich beendet.

6.2.2 DALI Bus Monitor-Ereignisse (*daliMonitor*)

DALI Monitor-Ereignisse (`daliMonitor`) werden vom Hardware Interface ausgesendet, wenn Befehle oder Antworten auf Befehle am DALI Bus registriert werden. Diese Ereignissen enthalten

- den Zeit Tick des Hardware-Ereignisses (`tick_us`),
- einen Zeitstempel von der Registrierung des Ereignisses in der Software (`timestamp`),
- die Länge des DALI Befehls (`bits`),
- dessen Befehlsdaten (`data.data`), und
- die Linie, auf der das Ereignis registriert wurde (`data.line`).

Monitor-Ereignisse sind hardwarenah und enthalten deswegen DALI Befehle als Integer. Die Befehlsdaten hängen davon ab, ob es sich um einen Befehl oder eine Antwort auf einen Befehl handelt.

```
{
  "type": "daliMonitor",
  "data": {
    "tick_us": 86454,
    "timestamp": 1627631911.299144,
    "bits": 16,
    "data": [
      0,
      145
    ],
    "line": 0
  },
  "timeSignature": {
    "timestamp": 1627631911.3002446,
    "counter": 481
  }
}
```

Weiters unterscheiden sich die Daten nach der Art des Befehls: DALI, DALI-2 oder eDALI.

- DALI Befehle haben eine Länge von 16 Bits und zwei Werte im Datenfeld: Adresse und Opcode [IEC62386-102].
- DALI-2 Befehle haben eine Länge von 24 Bits und drei Werte im Datenfeld: Adresse, Instanzbyte und Opcode [IEC62386-103].
- eDALI Befehle haben eine Länge von 25 Bits und drei Werte im Datenfeld.
- Antworten haben eine Länge von 8 Bits, einen Wert im Datenfeld: die Antwort des DALI Gears auf einen Befehl [IEC62386-102], und ein zusätzliches Fehlerfeld (`data.framingError`) um fehlerhafte Frames zu kennzeichnen. Im Fehlerfall hat das Fehlerfeld den Wert `true` und das Datenfeld enthält einen Wert `null`. Im Normalfall enthält das Fehlerfeld `false` und das Datenfeld enthält die Antwort.


```
{
  "type": "daliMonitor",
  "data": {
    "tick_us": 86468,
    "timestamp": 1627631911.3125844,
    "bits": 8,
    "data": [
      255
    ],
    "line": 0,
    "framingError": false
  },
  "timeSignature": {
    "timestamp": 1627631911.3135417,
    "counter": 482
  }
}
```


6.2.3 Senden von DALI Befehlen (*daliFrame* und *daliAnswer*)

Der Nachrichtentyp `daliFrame` kann genutzt werden um DALI Befehle direkt an den DALI Bus zu senden. Antworten zu direkten DALI Befehlen nutzen den Nachrichtentyp `daliFrame` für Sendebestätigungen und Fehlermeldungen, sowie den `daliAnswer` Typ für Antworten vom DALI Bus. Antworten werden nur an die Verbindung gesendet, die ursprünglich eine `daliFrame` Nachricht übermittelt hat.

Eine daliFrame Nachricht enthält die Linie an die der Befehl gesendet werden soll, die Anzahl der Bits des Befehls, den Sendemodus und die DALI Daten des Befehls. Der Sendemodus enthält eine Angabe, ob der Befehl zwei mal gesendet werden soll, eine Angabe ob auf eine Antwort gewartet werden soll und die DALI Priorität des Befehls.


```
{
  "type": "daliFrame",
  "data": {
    "line": 0,
    "numberOfBits": 16,
    "mode": {
      "sendTwice": false,
      "waitForAnswer": true,
      "priority": 3
    },
    "daliData": [
      1, 145
```

```
    ]
  }
}
```

Eine daliFrame Nachricht wird pro gesendetem DALI Befehl an den Websocket Client zurückgesendet. Auf eine daliFrame-Anfrage mit Sendemodus `"twice": true` wird zwei mal geantwortet.

```
{
  "type": "daliFrame",
  "data": {
    "line": 0,
    "result": 0
}
```

Diese Antworten enthalten einen `result` Code, der angibt ob der DALI Befehl erfolgreich an den DALI Bus gesendt wurde, oder ein Fehler aufgetreten ist. Die konkreten Fehlercodes sind in Tabelle 3 aufgelistet.

Table 3: Rückgabewerte in `daliFrame` Antworten.

Rückgabewert	Beschreibung
0	Der Befehl wurde an den DALI Bus gesendet.
1	Der Befehl wurde aufgrund der Spannungsversorgung des DALI Bus nicht gesendet.
2	Der Befehl wurde aufgrund des DALI Initialize Modus nicht gesendet.
3	Der Befehl wurde aufgrund des DALI Quiescent Modus nicht gesendet.
4	Der Sendepuffer des DALI Interface ist voll.
5	Das DALI Interface unterstützt die angeforderte Linie nicht.
6	Der Befehl enthält einen Syntax-Fehler.
7	Der Befehl wurde aufgrund eines aktiven Makros nicht gesendet.
61	Der Befehl wurde aufgrund einer Kollision am DALI Bus nicht gesendet.
62	Der Befehl wurde aufgrund eines DALI Bus Fehlers nicht gesendet.
63	Der Befehl wurde aufgrund einer Zeitüberschreitung nicht gesendet.
100	Das DALI Interface hat keine Antwort zurückgegeben.

Für `daliFrame` Anfragen mit Sendemodus `"waitForAnswer": true` werden Antworten mit Typ `daliAnswer` zurückgesendet. Diese enthalten die Linie, auf der die Antwort erhalten wurde, einen Antwortcode und die Antwortdaten, die auf dem DALI Bus empfangen wurden. Der Antwortcode ist entweder `0` für keine Antwort, `8` für eine Antwort mit einem 8-Bit Wert, oder `63` für einen „framing“ Fehler. Solche Fehler können auftreten wenn eine Abfrage an mehrere Gears mit unterschiedlichen Rückgabewerten beantwortet wird.

```
{
  "type": "daliAnswer",
  "data": {
    "line": 0,
    "result": 8,
    "daliData": 255
}
```

6.3 Ereignisse

6.3.1 DALI-Bus Scan Fortschritt-Ereignisse (*scanProgress*)

Während eines Geräte-Scans werden laufend Fortschritt-Ereignisse (`scanProgress`) ausgesendet. Das Datenfeld ist ident mit dem `ScanModel`, welches über die Web API abgefragt werden kann (Abschnitt 3.4.1).


```
{
  "type": "scanProgress",
  "data": {
    "id": "8e2fdf64-bde7-4114-ad7c-ee4128ac986e",
    "progress": 0.78125,
    "found": 0,
    "status": "addressing"
  },
  "timeSignature": {
    "timestamp": 1627565631.0883,
    "counter": 11
  }
}
```


6.3.2 Geräte-Ereignisse (*devices* und *devicesDeleted*)

Wenn neue Geräte zu den bereits bekannten Geräten hinzugefügt werden, z.B. durch einen Geräte-Scan (Abschnitt 3.4.1), dann wird ein Geräte-Ereignis (`devices`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten Geräten und deren Zustände, welche ident sind mit den Rückgabewert von Geräteabfragen über die Web API (Abschnitt 3.4.2).


```
{
  "type": "devices",
  "data": {
    "devices": [
      {
        "id": 1,
        "name": "Dali #0",
        "info": "Dali #0",
        "type": "default",
        "features": {
          "switchable": {
            "status": false
          },
          "dimmable": {
            "status": 0.0
          },
          "scene": true,
          "colorRGB": {
            "status": {
              "r": 1.0,
              "g": 1.0,
              "b": 1.0
            }
          →
```

```
←
          },
          "colorKelvin": {
            "status": 2700
          },
          "colorXY": {
            "status": {
              "x": 0.0,
              "y": 0.0
            }
          },
          "saveToScene": true,
          "gotoLastActive": {}
        },
        "scenes": [],
        "groups": [0]
      }
    ]
  },
  "timeSignature": {
    "timestamp": 1627639253.1174,
    "counter": 217
  }
}
```


Wenn sich die Zustände von Geräten ändern, z.B. weil ein neuer Lichtpegel eingestellt wurde, dann wird ebenfalls ein Geräte-Ereignis (`devices`) ausgesendet. Dieses Ereignis unterscheidet sich von einem Ereignis bei neu hinzugefügten Geräten, indem nur die geänderten Eigenschaften übertragen werden. Wird ein Gerät mit der Identifikationsnummer 1 z.B. einer Gruppe 1 hinzugefügt, dann enthält das Datenfeld für dieses Gerät auch nur die neue Liste von Gruppen. Wenn ein Dimmer, der

vorher ausgeschaltet war auf 100 % gedimmt, dann verändern sich dessen „`switchable`“ und „`dimmable`“ Features.


```
{
  "type": "devices",
  "data": {
    "devices": [
      {
        "id": 1,
        "groups": [
          0,
          1
        ]
      }
    ]
  },
  "timeSignature": {
    "timestamp": 1627639378.9165,
    "counter": 333
  }
}
```

```
{
  "type": "devices",
  "data": {
    "devices": [
      {
        "id": 1,
        "features": {
          "switchable": {
            "status": true
          },
          "dimmable": {
            "status": 100.0
          }
        }
      }
    ]
  },
  "timeSignature": …
```


Geräte entfernt-Ereignisse (`devicesDeleted`) werden z.B. dann ausgesendet, wenn alle bekannten Geräte durch den Endpunkt `DELETE /devices` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Geräten, die entfernt werden.


```
{
  "type": "devicesDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


6.3.3 Zonenereignisse (*zones* und *zonesDeleted*)

Wenn neue Zonen erzeugt oder bestehende Zonen verändert werden, dann wird ein Zonen-Ereignis (`zones`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten oder veränderten Zonen, welche ident sind mit den Rückgabewert von Zonenabfragen über die Web API (Abschnitt 3.6.2).


```
{
  "type": "zones",
  "data": {
    "zones": [
      {
        "name": null,
        "targets": [
          {
            "type": "group",
            "id": 0
          },
          {
```

```
            "type": "device",
            "id": 1
          }
        ]
      }
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


Zonen entfernt-Ereignisse (`zonesDeleted`) werden ausgesendet wenn Zonen durch einen der Endpunkte `DELETE /zones` oder `DELETE /zone/{_id}` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Zonen, die entfernt wurden.


```
{
  "type": "zonesDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


6.3.4 Sequenz-Ereignisse (*sequences* und *sequencesDeleted*)

Wenn neue Sequenzen erzeugt oder bestehende Sequenzen verändert werden, dann wird ein Sequenzen-Ereignis (`sequences`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten oder veränderten Sequenzen, welche ident sind mit den Rückgabewert von Sequenzabfragen über die Web API (Abschnitt 4.1.2).


```
{
  "type": "sequences",
  "data": {
    "sequences": [
      {
        "name": "turn on after 2 seconds",
        "enabled": true,
        "loop": false,
        "repeat": 0,
        "steps": [
          {
            "enabled": true,
            "type": "features",
            "data": {
              "targets": [
```

```
                {
                  "type": "broadcast"
                }
              ],
              "features": {
                "switchable": true
              },
              "delay": 2
            }
          }
        ],
        "id": 1,
        "active": true
      }
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


Sequenzen entfernt-Ereignisse (`sequencesDeleted`) werden ausgesendet wenn Sequenzen z.B. durch den Endpunkt `DELETE /automations/sequence/{_id}` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Sequenzen, die entfernt wurden.


```
{
  "type": "sequencesDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


6.3.5 Scheduler Ereignisse (*schedulers* und *schedulersDeleted*)

Wenn neue Schedules erzeugt oder bestehende Schedules verändert werden, dann wird ein Scheduler-Ereignis (`schedulers`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten oder veränderten Schedules, welche ident sind mit den Rückgabewert von Schedule-Abfragen über die Web API (Abschnitt 4.2.2).


```
{
    "type": "schedulers",
    "data": {
        "schedulers": [
```

```
            {
                "id": 1,
                "name": "Turn off at 18:00",
                "enabled": true,
                "targets": [
                    {
                        "type": "device",
                        "id": 1
                    }
                ],
                "activePeriod": {},
                "activeMonths": {},
                "activeWeekdays": {},
                "activeDays": {},
                "recallMode": "timeOfDay",
                "recallTime": {
                    "hour": 18,
                    "minute": 0,
                    "second": 0
                },
                "action": {
                    "type": "features",
                    "data": {
                        "features": {
                            "switchable": false
                        }
                    }
                }
            }
        ]
    },
    "timeSignature": {
        "timestamp": 1644836407.57024,
        "counter": 21
    }
}
```


Schedules entfernt-Ereignisse (`schedulersDeleted`) werden ausgesendet wenn Schedules z.B. durch den Endpunkt `DELETE /automations/scheduler/{_id}` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Schedules, die entfernt wurden.


```
{
  "type": "schedulersDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
```

```
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```

6.3.6 Circadian Ereignisse (*circadians* und *circadiansDeleted*)

Wenn neue Tageslichtverläufe erzeugt oder bestehende Tageslichtverläufe verändert werden, dann wird ein Circadian-Ereignis (`circadians`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten oder veränderten Tageslichtverläufe, welche ident sind mit den Rückgabewert von Tageslichtverlauf-Abfragen über die Web API (Abschnitt 4.3.2).

```
{
    "type": "circadians",
    "data": {
        "circadians": [
            {
                "name": "New",
                "id": 1,
                "targets": [],
                "enabled": true,
                "longest": {
                    "day": 22,
                    "month": 6,
                    "steps": [
                        {
                            "hour": 0,
                            "colorKelvin": 2700.0
                        },
                        …
                    ]
                },
                "shortest": {
                    "day": 21,
                    "month": 12,
                    "steps": [
                        {
                            "hour": 0,
                            "colorKelvin": 2700.0
                        },
                        …
                    ]
                }
            }
        ]
    },
    "timeSignature": {
        "timestamp": 1644836898.3646882,
        "counter": 28
    }
}
```

Circadian entfernt-Ereignisse (`circadiansDeleted`) werden ausgesendet wenn Tageslichtverläufe z.B. durch den Endpunkt `DELETE /automations/circadian/{_id}` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Tageslichtverläufe, die entfernt wurden.


```
{
  "type": "circadiansDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```


6.3.7 Trigger-Actions (*triggerActions* und *triggerActionsDeleted*)

Wenn neue Trigger Actions erzeugt oder bestehende Trigger Actions verändert werden, dann wird ein Trigger Action-Ereignis (`triggerActions`) ausgesendet. Dieses Ereignis enthält als Datenfeld eine Liste an neu hinzugefügten oder veränderten Trigger Actions, welche ident sind mit den Rückgabewert von Trigger Action-Abfragen über die Web API (Abschnitt 4.4.2).


```
{
    "type": "triggerActions",
    "data": {
        "triggerActions": [
            {
                "id": 1,
                "enabled": true,
                "name": "New Link",
                "sources": [
                    {
                        "type": "device",
                        "id": 1
                    }
                ],
                "targets": [
                    {
                        "type": "zone",
                        "id": 1
                    }
                ]
            }
        ]
    },
    "timeSignature": {
        "timestamp": 1644837187.7067018,
```

```
        "counter": 35
    }
}
```

Trigger Action entfernt-Ereignisse (`triggerActionsDeleted`) werden ausgesendet wenn Trigger Actions z.B. durch den Endpunkt `DELETE /automations/triggerAction/{_id}` entfernt werden. Das Datenfeld enthält dann ein Feld (`deleted`) mit einer Liste an Identifikationsnummern von allen Trigger Actions, die entfernt wurden.

```
{
  "type": "triggerActionsDeleted",
  "data": {
    "deleted": [
      1,
      2
    ]
  },
  "timeSignature": {
    "timestamp": 1627639216.5944345,
    "counter": 169
  }
}
```

6.3.8 Änderung der Systemzeiteinstellungen (*datetime*)

Wenn die Systemzeit über die Einstellungen des DALI-2 IoT geändert wird, dann wird die neue Konfiguration, bestehend aus Zeitzone, Datum und Zeit, sowie der Einstellung, ob die Zeit automatisch aus dem Internet bezogen werden soll, als Ereignis ausgesendet.

```
{
  "type": "datetime",
  "data": {
    "timezone": "Europe/Vienna",
    "automatic_time": true,
    "date": "28. October 2021",
    "time": "10:42"
  },
  "timeSignature": {
    "timestamp": 1635410542.061734,
    "counter": 2
  }
}
```

6.3.9 Einblenden von Statusnachrichten (*messageFlash*)

Der Ereignistyp `messageFlash` wird genutzt um allgemeine Statusnachrichten zu übertragen. Diese Ereignisse richten sich an Benutzer und sollen auf Fehlerfälle, zum Beispiel einen nicht verbundenen DALI-Bus, hinweisen.

```
{
  "type": "messageFlash",
  "data": {
    "message": "Flash message text",
    "seconds": 30,
    "userDismissible": true
  },
  "timeSignature": {
```

```
    "timestamp": 1627565751.8057,
    "counter": 12
  }
}
```

6.3.10 Ereignisse für Verbindungstests (*ping*)

Wenn der API Endpunkt `POST /ping/echo` abgefragt wird, dann wird ein `ping`-Ereignis an alle Websocket-Verbindungen gesendet. Dieser Mechanismus kann für Verbindungstests genutzt werden.

Abfrage

```
{
  "echo": "test",
  "timeSignature":{
    "timestamp": 1635411825.11234,
    "counter": 4
  }
}
```

Ereignis

```
{
  "type": "ping",
  "data": {
    "echo": "test"
  },
  "timeSignature": {
    "timestamp": 1635411825.11206,
    "counter": 3
  }
}
```

7 Dokumentenverlauf

Revision	Änderungen	Datum
1.0	Erstausgabe	20.09.2021
2.0	Hinzugefügt: • `colorWAF` Feature • Zonen • Trigger Action Automation • Email Settings • Zusätzliche Angaben in Allgemeinen Informationen des DALI-2 IoT • Info, Bus Status, Zone, Email Settings and Automation Websocket Events Geändert: • Websocket Interface wurde nach General Settings verschoben.	14.02.2022
3.0	Hinzugefügt: • Zusätzliche `name` Eigenschaft in `discovery` Antworten. • Zusätzliche Angabe der Version `1.3` in Email Einstellungen. • Zusätzliche `daliTypes`-Eigenschaft bei DALI Geräten, ab Version `1.4`. • Zusätzliche Beschreibung der `daliFrame` und `daliAnswer` Websocket-Nachrichten, für direkten DALI Bus Zugriff ab Version `1.2`. Geändert: • Änderung der `info`-Eigenschaft bei DALI Geräten in explizite Angabe von `address` und `line`, ab Version `1.2`. • Aufteilung von Websocket Interface in Sektionen für generelle Kommunikation, DALI Bus und Ereignisse.	06.04.2022

8 Quellenverzeichnis

IEC62386-102: International Electrotechnical Commission, Digital addressable lighting interface - Part 102: General requirements - Control gear, 2014
IEC62386-103: International Electrotechnical Commission, Digital addressable lighting interface - Part 103: General requirements - Control devices, 2014