REST-API GUIDE

for getAir systems

To help end users connect
with their getAir IoT system

getAir®

Inhaltsverzeichnis

Willkommen

Dieser Leitfaden führt Sie durch die API-Aufrufe, die Sie benötigen, um mit Ihrem getAir System zu interagieren.

In diesem Leitfaden erfahren Sie:

- Wie Sie sich bei der API authentifizieren und sich mit Ihrem System verbinden.
- Wie Sie Werte Ihres Systems lesen und setzen.
- Wie Sie Ereignisse Ihres Systems abonnieren.

Zielgruppe

Dieses Dokument richtet sich an Besitzer von smarten getAir Lüftungs-Systemen, die ihre Geräte mithilfe von API-Tools steuern möchten. Es wird dabei vorausgesetzt, dass Sie:

- mit den grundlegenden REST-API-Prinzipien und Befehlszeilentools vertraut sind.
- Grundkenntnisse in gängigen Programmierkonzepten wie JSON, Objekte und Datentypen haben.

Geräte-Kompatibilität

Die Anleitungen in diesem Dokument gelten für die folgenden getAir Lüftungs-Systeme:

- ComfortControl Pro BT
- ComfortControl Pro

Voraussetzungen

Um den Beispielen folgen und die API verwenden zu können, benötigen Sie:

- eine Software für die Interaktion mit RESTful APIs, (z.B. Postman, cURL) oder - für fortgeschrittene Benutzer - eine Programmierumgebung mit der Sie Ihren eigenen Code schreiben können, um mit der API zu kommunizieren, wie bspw. Python oder JavaScript.
- Ein getAir-Konto, das mit Ihrem Smart-Gerät gekoppelt ist. Ein Konto anlegen und Ihr Gerät hinzufügen können Sie mit der getAir App für Android oder iOS.

Wichtige Hinweise

Mit dieser REST-API haben Sie vollen Zugriff auf die Einstellungen und Eigenschaften Ihres Smart-Home-Lüftungssystems. Dies bietet eine hohe Flexibilität, erfordert aber Vorsicht beim Vornehmen von Änderungen. Hier sind einige Dinge, die Sie beachten sollten:

- Die API ermöglicht uneingeschränkten Zugriff auf die Einstellungen Ihres Geräts, sodass Sie Änderungen vornehmen können, die sich auf die korrekte Funktionsweise Ihres Systems auswirken könnten. Wenn etwas falsch konfiguriert ist, kann dies die Funktion des Systems beeinträchtigen.
- Wir empfehlen, Änderungen nur vorzunehmen, wenn Sie sich sicher sind, dass Sie wissen, was Sie tun.
- Bitte beachten Sie, dass wir keine Verantwortung für Probleme übernehmen können, die durch benutzerdefinierte Konfigurationen verursacht werden, die über die API vorgenommen wurden.

Die API ist ein leistungsstarkes Werkzeug, setzen Sie es also mit Bedacht ein, um das Optimum aus Ihrem Smart-Home-System herauszuholen.

Grundlagen

Wir haben diesen Leitfaden um eine Reihe von exemplarischen Vorgehensweisen herum strukturiert, um die Funktionalität der API besser zu erläutern.

Bestimmte grundlegende Informationen zur API sollten bekannt sein, um eine reibungslose Nutzung zu gewährleisten. Dieser Abschnitt dient dazu, diese Details im Vorfeld zu erklären.

In diesem Abschnitt erfahren Sie:

- Wie API-Aufrufe aufgebaut sind.
- Was die wesentlichen Bestandteile eines cURL-Befehls sind, damit Sie diesen an Ihre bevorzugte Methode für die Arbeit mit APIs anpassen können.
- Die Struktur von Antwortmeldungen, einschließlich Fehlermeldungen.
- Wie die Daten innerhalb des Servers strukturiert sind.

API-Beispiele mit cURL verstehen

In diesem Leitfaden verwenden wir das Befehlszeilentool cURL (Client-URL), um API-Aufrufe zu veranschaulichen. Da cURL auf den meisten Betriebssystemen bereits vorinstalliert ist, können Sie die Beispiele im Folgenden direkt ausprobieren. Wenn Sie cURL nicht auf Ihrem System haben, können Sie es hier laden: `https://curl.se/download.html`

Wenn Sie diese Aufrufe in Ihr eigenes Projekt integrieren möchten, verwenden Sie möglicherweise ein anderes Tool oder eine andere Sprache. Trotzdem ist es sinnvoll, die Struktur der cURL-Befehle zu verstehen und diese dann nur daran anzupassen, wie Ihr Projekt mit APIs kommuniziert.

Betrachten wir zunächst einen allgemeinen cURL-Beispielbefehl. Der spezifische Inhalt kann zunächst ignoriert werden und dient nur zu Demonstrationszwecken.

API-Aufruf

```
curl --request POST '<<AUTH_URL>>/api/v1/authn'
     --header ‘Content-Type: application/json'
     --data '{
              "username": "<<BENUTZERNAME>>",
              "password": "<<PASSWORT>>"
             }'
```

Platzhalter (<< >>)

Wenn Sie diese spitzen Klammern sehen, ersetzen Sie sie durch den angegebenen Wert, z. B. die URL, den Benutzernamen oder das Kennwort, wenn Sie den Befehl verwenden.

cURL-Parameter

Der cURL-Befehl akzeptiert verschiedene Befehlszeilenargumente, von denen wir hauptsächlich die Folgenden verwenden werden:

Parameter	Bedeutung
--request	Der Request-Parameter gibt den Typ des HTTP-Requests an, z. B. GET oder PUT oder POST
--header	Gibt zusätzliche Header an, in diesem Beispiel, dass der Inhaltstyp im JSON-Format vorliegt.
--data	Enthält den Anforderungstext im JSON-Format, ein Format häufig zum Senden von Details wie Anmeldeinformationen oder anderen Daten verwendet wird.

Table 1: cURL-Befehlszeilenparameter

Grundlegendes zu Antwortnachrichten

Wenn Sie die API aufrufen, erhalten Sie immer eine Antwortnachricht. Jede Antwort enthält einen HTTP-Statuscode.

Hier ist eine Liste der möglichen Statuscodes, die die API zurückgeben kann, und was sie bedeuten.

HINWEIS: Die Authentifizierungs-API kann weitere Status-Codes zurückmelden, die hier nicht explizit aufgeführt sind.

HTTP-Statuscode	Bedeutung	Details
200	OK	Die Anforderung wurde vom Server erfolgreich verarbeitet, und die erwarteten Antwortdaten werden zurückgegeben.
204	Kein Inhalt.	Die Anforderung wurde erfolgreich verarbeitet, aber der Server gibt keinen Inhalt im Antworttext zurück.
400	Ungültige Anforderung	Der Server kann oder will die Anfrage aufgrund eines Client-Fehlers (z. B. fehlerhafte Anforderungssyntax, ungültige Anforderungsparameter) nicht verarbeiten.
401	Unbefugt	Der Client muss sich authentifizieren, um die angeforderte Antwort zu erhalten. Dies bedeutet in der Regel fehlende oder ungültige Authentifizierungsdaten.
404	Ressource nicht gefunden	Die angeforderte Ressource konnte auf dem Server nicht gefunden werden.
429	Zu viele Anfragen	Der Benutzer hat zu viele Anfragen in einem bestimmten Zeitraum gesendet (Ratenbegrenzung). Der Server drosselt weitere Anforderungen vom Client.
500	Vorgang fehlgeschlagen	Der Server ist auf eine unerwartete Bedingung gestoßen, die ihn daran hinderte, die Anforderung zu erfüllen.
503	Service nicht verfügbar	Der Server ist derzeit nicht in der Lage, die Anforderung zu verarbeiten, häufig aufgrund einer vorübergehenden Überlastung oder Wartung.

Table 2: http Status-Codes

Zusätzliche Antwort-Daten im JSON-Format

Wenn eine Antwort erklärungsbedürftige JSON-Daten enthält, werden wir sie im Detail aufschlüsseln. Um den Leitfaden übersichtlich und übersichtlich zu halten, zeigen wir nur die relevanten Teile großer JSON-Objekte und zeigen Beispielwerte an.

Unser Beispielaufruf von oben gibt die folgenden JSON-Daten zurück. Auch hier gilt: Machen Sie sich zunächst keine Gedanken über deren Bedeutung, wir werden diese wo erforderlich für die Antwort-Nachrichten später im Abschnitt "Exemplarische Vorgehensweisen" erklären:

```
{
      expiresAt: '2024-11-28T08:31:57.000Z',
      status: 'SUCCESS',
      sessionToken: '2129955IBwt-Q_z_dXdasgwrg2Twz0Bxm6BmVdgs9gp7JkE2_vQ',
      /* Zusätzliche Angaben zur besseren Lesbarkeit weggelassen */
}
```

Grundlegendes zur Datenstruktur in der API

In diesem Abschnitt erfahren Sie:

- Wie Daten innerhalb der API organisiert und benannt werden.
- Was ein Device, ein Service und ein Property ist.

Um effektiv mit der API interagieren zu können, ist es wichtig, die Schlüsselbegriffe zu verstehen, die wir zur Beschreibung der Daten verwenden. Das Ziel dieses Abschnitts ist es, Ihnen ein mentales Modell davon zu vermitteln, wie alles zusammenspielt, damit Sie sich besser in den spezifischen Details in den exemplarischen Vorgehensweisen zurechtfinden können.

HINWEIS: Diese Definitionen sind miteinander verbunden und bilden eine Hierarchie. Jeder Begriff baut auf dem nächsten auf.

Device

- Ein Gerät ("Device") stellt eine physische Hardware in Ihrem System dar, genauer gesagt: eine ComfortControl Pro oder eine ComfortControl Pro BT
- Jedes dieser Geräte bietet einen oder mehrere Dienste **("Services")** an.

Service

- Ein Service ist ein logischer Bestandteil des Smarten Geräts. Im Falle eines getAir Lüftungssystems sind die Services "System" und "Zone"
- Jeder Service besteht aus seiner Sammlung von **Properties**.

Property

- Eine Property ist die grundlegendste Einheit im System. Er stellt einen Datenpunkt für einen Dienst auf einem Gerät dar.
- Eine Beispieleigenschaft für den System-Service des getAir-Gerätes wäre z.B. "`temperature`"

Exemplarische Vorgehensweisen

Nachdem Sie die Grundlagen gemeistert haben, ist es an der Zeit, in die exemplarischen Vorgehensweisen einzutauchen. Die exemplarischen Vorgehensweisen sind in vier Abschnitte unterteilt:

- Zugriff auf Ihr Gerät erhalten
- Auslesen von Eigenschaften (“Properties”) des Geräts
- Ändern von Eigenschaften (“Properties”) des dem Gerät
- Abonnieren von Ereignissen (“Events”) über das Gerät

Jeder Abschnitt ist so konzipiert, dass er Sie durch eine bestimmte Funktion führt und Folgendes erklärt:

- Eine Beschreibung des Endpunkts.
- Wie der Endpunkt verwendet werden soll.
- Eine Demonstration des API-Aufrufs mit cURL.
- Bei Bedarf eine Beispielantwortnachricht, die zeigt, was der zurückgegebene JSON-Code enthält.

Sie können diese exemplarischen Vorgehensweisen entweder der Reihe nach durchgehen oder zu den exemplarischen Vorgehensweisen springen, die für Ihr Projekt am relevantesten sind. Beziehen Sie sich auf den Abschnitt "Grundlagen", wenn Fragen auftreten.

Viel Spaß beim Programmieren und viel Spaß beim Erkunden der API!

Zugriff auf Ihr Gerät erhalten

Bevor Sie über die API mit Ihrem Gerät interagieren können, müssen Sie sich authentifizieren. Dies geschieht in zwei Schritten:

Schritt 1: Anmelden

Zunächst müssen Sie sich mit Ihrem Benutzernamen und Passwort beim getAir Authentication Service anmelden, um ein "Session Token" abzurufen:

```
curl --request POST '<<AUTH_URL>>/api/v1/authn'
     --header 'Content-Type: application/json'
     --data '{
              "username": "<<BENUTZERNAME>>",
              "password": "<<PASSWORT>>"
             }'
```

Antwort

Wenn Ihr Benutzername und Ihr Passwort korrekt sind, erhalten Sie eine Antwort wie die Folgende. Der einzig relevante Teil ist das `sessionToken`.

```
{
      expiresAt: '2024-11-28T08:31:57.000Z',
      status: 'SUCCESS',
      sessionToken: '2129955IBwt-Q_z_dXdasgwrg2TVW6Qmjyadgs9gp7JkE2_vQ',
      /* Zusätzliche Angaben zur besseren Lesbarkeit weggelassen */
}
```

Schritt 2: Starten einer API-Sitzung

Nun rufen Sie die API wie folgt auf und geben das sessionToken an, das Sie zuvor erhalten haben

```
curl --request GET '<<API_URL>>/api/v1/iam/token?issuer=GETAIR_API&
                                    sessionToken=<<sessionToken>>'
     --header 'Content-Type': application/json'
```

Antwort

Wenn das sessionToken gültig ist und von der API akzeptiert wird, erhalten Sie ein weiteres Token als Antwort.

```
{
     "token": "7h15154n3x4mpl370k3n",
     "type": "TOKEN"
}
```

Dieses letzte Token wird benötigt, um Sie bei jedem nachfolgenden Aufruf der API zu identifizieren.

HINWEIS: Das Token ist 1 Stunde lang gültig. Danach müssen Sie sich erneut anmelden und eine neue Sitzung starten. Wenn das Token abgelaufen ist, werden Anfragen an die API alle mit eine 401 http Status-Code beantwortet.

Auslesen von Werten aus dem Gerät

In diesem Abschnitt zeigen wir Ihnen, wie Sie Daten von Ihrem Gerät auslesen können.

Struktur des API-Aufrufs

```
curl --request GET
     '<<API_URL>>/api/v1/devices/<<deviceId>>/services/<<service>>'
     --header 'Authorization: Bearer <<token>>'
```

Parameter	Beschreibung
API_URL	Die URL der getAir API (siehe Seite 12)
deviceId	Gibt das physische Gerät an, das Sie aktualisieren möchten. Sie können diesen Wert mithilfe von API-Aufrufen im Abschnitt „Auslesen von Werten aus dem Gerät“ auf Seite 8 ermitteln.
service	Gibt den spezifischen Dienst an, den Sie aktualisieren möchten. Siehe S. 12 für mögliche Werte
token	Das Token, das Sie von der API abgerufen haben

Tabelle 1

Beispiel für einen API-Aufruf

```
curl --request GET
     'api.example.cloud/api/v1/devices/001122334455/services/ServiceName'
```

```
      --header 'Authorization: Bearer 7H15154N3X4MPL370K3N'
```

Rückgabewert

```
{
      "brightness": 50,
      "speed": 4
}
```

Festlegen von Werten auf dem Gerät

In diesem Abschnitt konzentrieren wir uns darauf, wie Daten an die API gesendet werden, insbesondere durch Aktualisieren von Werten auf dem Gerät. Wir zeigen, wie Sie Ihre API-Aufrufe strukturieren, die Parameter verstehen und ein Beispiel, das Sie an Ihre eigenen Anwendungsfälle anpassen können.

Dieser Endpunkt ermöglicht es Ihnen, Eigenschaftswerte auf einem bestimmten Gerät zu aktualisieren.
Sie können diesen Aufruf z. B. verwenden, um den Lüfter ein- oder auszuschalten.

Struktur des API-Aufrufs

```
curl --request PUT
      '<<API_URL>>/api/v1/devices/<<deviceId>>/services/<<service>>'
      --header 'Content-Type: application/json'
      --header 'Authorization: Bearer <<token>>'
      --data '{
                  "<<property>>": "<<wert>>",
                  "<<property>>": "<<wert>>"
            }'
```

Parameter

Parameter	Beschreibung
API_URL	Die URL der getAir API (siehe Seite 12)
deviceId	Gibt das physische Gerät an, das Sie aktualisieren möchten. Sie können diesen Wert mithilfe von API-Aufrufen im Abschnitt "Daten lesen" ermitteln.
service	Gibt den spezifischen Dienst an, den Sie aktualisieren möchten. Dieser Wert kann auch über die API abgerufen werden.
token	Das Token aus dem Authentifizierungs-Schritt 2
property	Der Name einer Eigenschaft des Service (Siehe Seite 12). Es können mehrere Eigenschaften in einem Aufruf setzen. Die Eigenschaften müssen durch Komma getrennt aufgeführt werden
wert	Der gewünschte Ziel-Wert der Eigenschaft, entsprechend dem korrekten Datentyp entweder mit " " (bei strings) oder ohne bei numerischen Werten.

Tabelle 2

Beispiel für einen API-Aufruf

```
curl --request PUT
'api.example.cloud/api/v1/devices/001122334455/services/ServiceName'
     --header 'Content-Type: application/json'
     --header 'Authorization: Bearer 7h15154n3x4mpl370k3n'
     --data '{
             "speed": 2.0
             }'
```

HINWEIS: Das Gerät lässt es möglicherweise nicht zu, dass Sie ein Property auf einen beliebigen Wert festlegen können. Der Wert, der in der Antwort zurückgegeben wird, spiegelt wider, auf welchen Wert bzw. welche Werte tatsächlich gesetzt wurden.

Abonnieren von Ereignissen

Ereignisse benachrichtigen Sie in Echtzeit über Änderungen in Ihrem System. Auf diese Weise müssen Sie nicht ständig Werte bei der API abfragen.

Mithilfe von WebSockets können Sie diese Ereignisse abonnieren und auf sie reagieren, sobald sie auftreten. Sie können dies z. B. verwenden, um Benachrichtigungen zu erhalten, wenn sich der Status eines Geräts ändert.

Um eine WebSocket-Verbindung aufzubauen, müssen Sie zunächst ein sogenanntes Ticket generieren. Dieses Ticket fungiert als temporärer Schlüssel, um die Notwendigkeit eines Authorization-Headers zu umgehen, der von einigen Browsern und Bibliotheken während des WebSocket-Upgrade-Prozesses nicht unterstützt wird.

API-Aufruf

```
curl  --request GET
      --location '<<API_URL>>/api/v1/events/ticket'
      --header 'Authorization: Bearer <<TOKEN>>'
```

Antwort

Der Server antwortet mit einem JSON-Objekt, das Ihre Ticketzeichenfolge enthält.

```
{
   "ticket": "ef5c8f"
}
```

Beachten Sie, dass das Ticket nach der Generierung nur **30 Sekunden** lang aktiv ist.

Sobald Sie das Ticket haben, können Sie eine WebSocket-Verbindung starten. Ersetzen Sie <<TICKET>> durch Ihre Ticketzeichenfolge.

```
"wss://<<API_URL>>/api/v1/events/<<TICKET>>"
```

Die WebSocket-Verbindung abonniert nun Echtzeit-Ereignisbenachrichtigungen. Diese Benachrichtigungen umfassen verschiedene Ereignisse, z. B. Änderungen an Ihren Geräten, Aktualisierungen von Diensten oder andere systemweite Aktionen.

Ereignis-Format

Wenn Sie eine WebSocket-Verbindung abonniert haben, erhalten Sie Ereignisse des Geräts als JSON-Strukturen zurück.
Dabei sind diese immer wie im folgenden Beispiel aufgebaut:

```
{
   "type": "OBJECT_UPDATED",
   "flakeAddress": 6277,
   "deviceIdentifier": "00112233445566",
   "serviceIdentifier": "00000501-0a45-0202-0000-ab0000910002",
   "updates": [
     {
        "tag":"value",
        "modifier": { /* ausgelassen für bessere Lesbarkeit */ },
        "propId": 16900,
        "type": "UINT16",
        "value": 5465
      }
   ],
   "timestamp":1732984334848
}
```

JSON-Parameter

Parameter	Beschreibung
type	Die Art des Ereignisses. Relevant ist hier nur „OBJECT_UPDATED“, was bedeutet, dass sich Werte auf dem Gerät geändert haben.
deviceIdentifier	Die eindeutige Identifikations-Nummer des Geräts.
serviceIdentifier	Der Service, der das Ereignis meldet. In Websocket-Ereignissen werden alphanumerische UUIDs statt der Klarnamen übertragen. Diese UUIDs entnehmen Sie für Ihr Gerät bitte den Tabellen auf Seite 12)
propId	Der numerische Bezeichner des Properties. Die Zuordnung von Klarnamen zu numerischen Bezeichnern finden Sie auf Seite 12
type	Der Typ des Property-Wertes.
value	Der neue Wert des Properties
timestamp	UNIX-Zeitstempel des Ereignis-Zeitpunkts

Table 3: Rückgabewerte von Websocket-Ereignissen

Beachten Sie, dass wir cURL in diesem Beispiel nicht verwendet haben. Das liegt daran, dass Sie über dieses Tool keine Schnittstelle zu Websockets herstellen können. Abhängig von Ihrer Programmierumgebung müssen Sie einen dedizierten WebSocket-Client oder eine dedizierte WebSocket-Bibliothek verwenden.

Referenzmaterialien

In diesem Abschnitt finden Sie:

- API-URLs
- Referenztabellen für Gerätedienste/-eigenschaften.

URLs

Platzhalter	URL (Englisch)
AUTH_URL	https://auth.getair.eu
API_URL	https://be01.ga-cc.de

Table 4: URLs

Referenztabelle für Gerätewerte

Die Tabelle auf den nächsten Seiten beschreibt die Eigenschaften der Services in Ihrem System.

Service „System“

Service-Name **System**	Service-UUID **abbc9241-4886-407f-8e36-e26d0b64776c**

Service-Properties

Name	ID (hex)	ID	Typ	Ein-heit	Schreib-schutz	Beschreibung
system-type	0x2000	8192	uint8		Ja	1: SmartFan 2: EasyFan 3: ComfortControl Pro BT 4: ComfortControl Pro
system-version	0x2001	8193	uint16		Ja	Systemversion (intern)
num-zones	0x2002	8194	uint8		Ja	Anzahl der Lüftungszonen
system-id	0x2003	8195	binär		Ja	Eindeutige Hardware-Kennung
runtime	0x2005	8197	uint32	h	Ja	Gesamtlaufzeit in Stunden
modelock	0x2007	8199	bool		Ja	Zeigt an, ob der „Wohnungswirtschaftsmodus“ aktiviert ist. Falls der Wert „true“ ist, kann die Lüftungsstufe 0 nicht gewählt werden.
notification	0x2008	8200	string		Ja	Benachrichtigung, z.B. über ein Firmware-Update
notify-time	0x2009	8201	datetime		Ja	Unix-Zeitstempel der Benachrichtigung
humidity	0x2040	8256	float	%	Ja	Relative Luftfeuchtigkeit
air-pressure	0x2041	8257	float	hPa	Ja	Luftdruck
temperature	0x2042	8258	float	0,5 °C	Ja	Temperatur in Schritten von einem halben Grad. Um die Temperatur in Grad Celsius zu erhalten, multiplizieren Sie den Wert mit 2

indoor-air-quality	0x2043	8259	float	Ja	Luftqualität in Innenräumen: 0 bis 50 – Ausgezeichnet 51 bis 100 – Gut 101 bis 150 – Leicht verschmutzt 151 bis 200 – Mäßig verschmutzt 201 bis 250 – Stark verschmutzt 251 bis 350 - Stark verschmutzt 351 bis 500 – Extrem verschmutzt
iaq-accuracy	0x2044	8260	uint8	Ja	Genauigkeit der Luftqualität in Innenräumen: 1: Der Sensor befindet sich in der Anfangskalibrierungsphase 2: Der Sensor wird neu kalibriert 3: Der Sensor ist kalibriert, die Daten sind genau
fw-app-version	0x2100	8448	uint8	Ja	Version der Firmware-App
fw-app-version-str	0x2101	8449	string	Ja	Firmware-App-Version als „semver“ string
boot-time	0x220A	8714	datetime	Ja	Unix-Zeitstempel des letzten Bootvorgangs
supports-password	0x2030	8240	bool	Ja	Das Gerät unterstützt das Festlegen eines Kennworts
supports-auto-update	0x2031	8241	bool	Ja	Das Gerät unterstützt automatische Firmware-Updates
auto-update-enabled	0x2103	8451	bool	Ja	Automatische Firmware-Updates sind aktiviert
time-profile-X-name (x= 1-10)	0x2051 - 0x205A	8271 - 8282	string	Nein	Benutzerdefinierte Namen der Zeit-Profile
time-profile-x-data (x = 1-10)	0x2061 - 0x206A	8289 - 8298	binär	Nein	Daten von Zeitprofilen

Table 5: System-Service Properties

Service „Zone“

Service-Name **Zone** Service-UUID **abbc9241-4886-407f-8e36-e26d0b64776d**

Service-Properties

Name	ID (hex)	ID	Typ	Einheit	Schreibschutz	Beschreibung
name	0x2004	8196	string		Nein	Ein benutzerdefinierter Name für die Lüftungszone
runtime	0x2005	8197	uint32	h	Ja	Lüftungslaufzeit in Stunden
last-filter-change	0x2006	8198	uint32	h	Nein	Laufzeit seit dem letzten Filterwechsel
speed	0x2011	8209	float		Nein	Lüfterdrehzahl in der Lüftungszone (0-4)
zone-index	0x2019	8217	uint8		Ja	Index der Zone, wenn das Gerät mehrere Zonen unterstützt.
mode	0x2020	8218	string		Nein	Modi: "ventilate" - Normales Lüften "ventilate_inv" - Inverses Lüften

						"ventilate_hr" - Normales Lüften mit Wärmerückgewinnung (WRG) "night" – Nachtmodus "rush" – Normales Stoßlüften "rush_inv" – Inverses Stoßlüften "rush_hr" – Stoßlüften mit WRG "auto" – Automatik-Modus
mode-deadline	0x2021	8225	datetime		Nein	Unix-Zeitstempel der Deadline für den aktuellen Modus
target-temp	0x2030	8240	float	°C	Nein	Zieltemperatur
target-hmdty-level	0x2031	8241	string		Nein	Soll-Luftfeuchtigkeit im Auto-Modus: "thirty-fifty" - 30-50% Luftfeuchtigkeit "fourty-sixty" - 40-60% Luftfeuchtigkeit "fitiy-seventy" - 50-70% Luftfeuchtigkeit
auto-mode-voc	0x2032	8242	bool		Nein	Zeigt an, ob der VOC-Automodus aktiviert ist
auto-mode-silent	0x2033	8243	bool		Nein	Gibt an, ob der automatische Modus aktiviert ist
humidity	0x2040	8256	float	%	Ja	Aktuelle Luftfeuchtigkeit
temperature	0x2042	8258	float	°C	Ja	Aktuelle Temperatur
hmdty-outdoors	0x2045	8261	float	%	Ja	Luftfeuchtigkeit im Freien
temperature-outdoors	0x2046	8262	float	°C	Ja	Außentemperatur
time-profile	0x2050	8272	uint8		Nein	Aktuell aktives Zeitprofil. 0 = kein Zeitprofil aktiv.

Table 6: Zone-Service Properties

Zeitprofil Binärformat

HINWEIS: Zeitprofile sollten im Normalfall über die Apps gesetzt werden. Das Setzen über die API ist fortgeschrittenen Benutzern vorbehalten, daher wird im Folgenden vorausgesetzt, dass das grundsätzliche Vorgehen beim Kodieren von Binärwerten bekannt ist.

Der Binärwert der Zeitprofile besteht aus bis zu 70 Blöcken á 4 Bytes der Form

Byte 1	**Byte 2**	**Byte 3**	**Byte 4**
(Bit 0..7)	(Bit 8..15)		
Start (UINT16 LE)		**Mode (UINT8)**	**scaledSpeed (UINT8)**

wobei:

- **start** ein UINT16 ist, der die Minute der Woche angibt, zu der der Modus aktiv werden soll (d. h. wie viele Minuten seit Beginn Montag, 0 Uhr vergangen sind). Der Wert muss im Bereich von 0 bis 10079 liegen.

- **mode** ein UINT8 ist, der wie folgt codiert werden muss.

Luftrichtung \ Modus	Normal	Nacht	Stoßlüftung	Automatik
Normal	0 (0x00)	16 (0x10)	32 (0x20)	48 (0x30)
Invers	1 (0x01)		33 (0x21)	
Wechsel für Wärmerückgewinnung	2 (0x02)		34 (0x22)	

- **scaledSpeed** ein UINT8 ist, der die Geschwindigkeit des Lüfters im Modus darstellt, jedoch skaliert und gerundet nach folgender Regel:

  ```
  scaledSpeed = round(speed * 10)
  ```

  Es soll dabei kaufmännisch gerundet werden. Da die Lüftergeschwindigkeit zwischen 0,5 und 4,0 liegt, sollte die skalierte Geschwindigkeit immer eine Ganzzahl im Bereich von 5 bis 40 sein. Für andere Modi wird dieser Wert ignoriert.

get Air®

Krefelder Straße 670, 41066 Mönchengladbach

www.getair.eu

info@getair.eu