

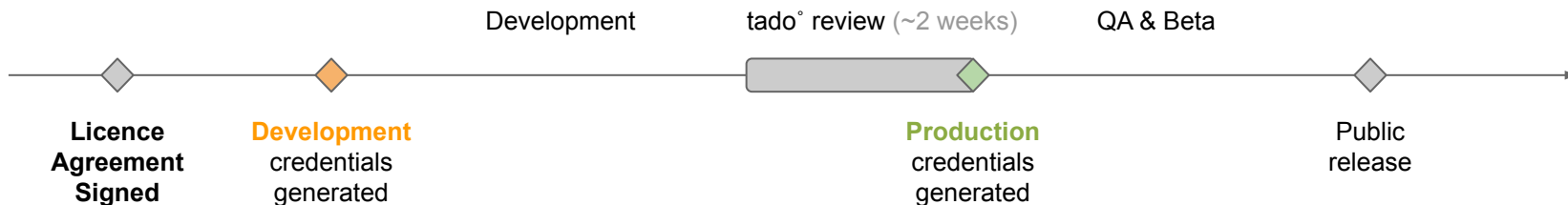
tado° Control API

Version history on the next page

Version	Status	Date	Authors	Explanation
1.0.0	Final	2021-11-30	Timo Streckfuß	Explicit v1.0.0 to indicate that the API is stable
0.18.0	Final	2021-11-30	Timo Streckfuß, Alexander Kalach	Update AC capabilities
0.17.0	Final	2020-03-31	Timo Streckfuß	Clarify review process
0.16.0	Final	2020-01-23	Timo Streckfuß	Clarify scope of removed properties in API evolution
0.15.0	Final	2020-01-14	Timo Streckfuß	Remove incorrect deprecation description from AC capabilities
0.14.0	Final	2019-08-30	Timo Streckfuß	Correctly state relation between hooks and customer account
0.13.0	Final	2019-07-26	Timo Streckfuß	Add details about API evolution
0.12.0	Final	2019-05-08	Timo Streckfuß	Added details to review phase and API Gotchas
0.11.0	Final	2018-04-27	Michal Knizek	Add inside temperature and humidity measurement webhooks
0.10.0	Final	2018-03-14	Michal Knizek	Add open window detection to Zone State
0.9.0	Final	2018-01-05	Timo Streckfuß	Auth Partner Portal added
Older versions on the next page				

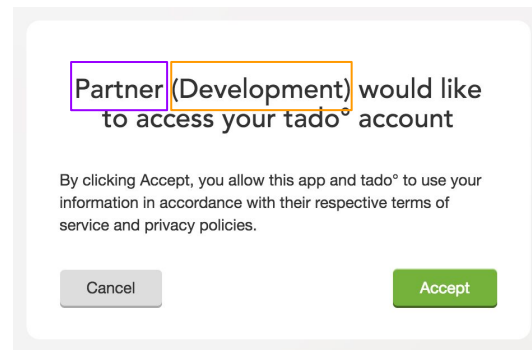
Version	Status	Date	Authors	Explanation
0.8.0	Final	2017-11-21	Michal Knizek, Michael Marino	Add notes about auth request usage
0.7.0	Final	2017-10-19	Michal Knizek	Mark "Non Thermostatic Control" AC modes as deprecated
0.6.0	Final	2017-09-04	Michal Knizek	Use auth.tado.com as authentication server
0.5.5	Final	2017-08-07	Michal Knizek	Document authentication errors
0.5.4	Final	2017-05-23	Michal Knizek	Update AC capabilities
0.5.3	Final	2017-03-21	Michal Knizek	Document token max size
0.5.2	Final	2017-03-01	Michal Knizek	Unregister Webhook endpoint
0.5.1	Final	2017-01-27	Johannes Schwarz	Removed sleep tado mode
0.5.0	Final	2017-01-11	Michal Knizek	Add AC Setting & Capabilities API
0.4.4	Final	2016-12-21	Michal Knizek	Add missing response_type parameter
0.4.3	Final	2016-12-16	Michal Knizek	Available OAuth Scopes
0.4.2	Final	2016-12-07	Michal Knizek, Johannes Schwarz	API Development Roadmap for Partner + Auth Credential Request Process
0.4.1	Final	2015-11-11	Michal Knizek	Zone state structure update
0.4.0	Final	2015-11-03	Michal Knizek	Setting Webhook
0.3.0	Final	2015-11-03	Michal Knizek	Multizone API Endpoints
0.1.0	Editor's Draft	2015-01-28	Michal Knizek	Created

API Development Roadmap for Partner



Authentication Credentials Request:

- tado° will generate **development credentials** for a partner. These credentials:
 - must not** be used in production for end customers
 - will contain “**(Development)**” in the client name
 - can be used to control up to 10 tado° accounts
 - can be revoked by tado° on short notice
- Once the partner is done with the integration, tado° will coordinate on how the review can be performed and review the integration on the scheduled time slot. If the review does not uncover any critical issues, **production credentials** will be generated by tado°. The partner should plan for spare time between the review and the public release to address these issues.



Sample Credentials:

Client Name: "Partner (Development)"
Client ID: partner-development
Client Secret: jhkjdshdsfSDSDSfjkjdsh

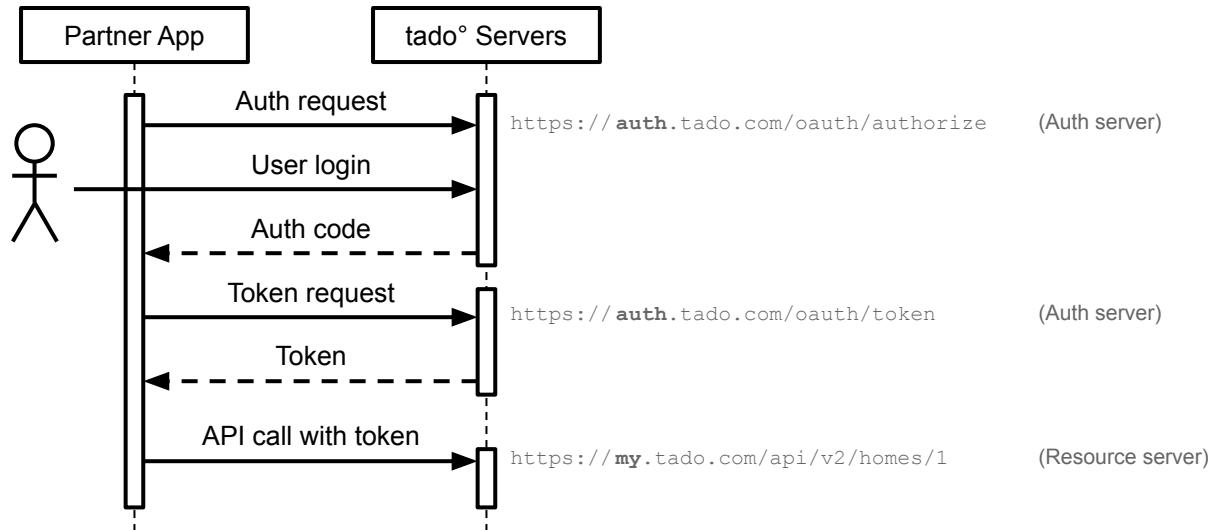
API Authentication and Authorization

Authentication and Authorization process overview



Authentication and Authorization

tado° APIs use the [OAuth 2.0](#) web application flow for authentication and authorization.



Auth request

GET `https://auth.tado.com/oauth/authorize`

Parameter	Value	Description
client_id	String	Client ID assigned by tado
redirect_uri	URI	Determines where the response is sent to. Allowed values have to be pre-defined via the Partner Portal .
scope	Space separated list	Identifies the tado° API that the application is requesting.
state	Any string	Random string to mitigate against CSRF
response_type	String	code (as defined in the OAuth 2.0 specification)

Example:

```
GET https://auth.tado.com/
oauth/authorize?
client_id=partner-app&
redirect_uri=https://app.partner.com&
scope=identity:read&
state=bNRET9wpUXEnDdtKfXF&
response_type=code
```

Response

Redirect to `<redirect_uri>?state=<state>&code=<code>`

Parameter	Value	Description
state	Any string	Random string to mitigate against CSRF
code	String	The authorization code

Example:

```
https://app.partner.com?
state=bNRET9wpUXEnDdtKfXF&
code=6RptQB
```


Auth request

GET `https://auth.tado.com/oauth/authorize`

Important Security Notes:

`redirect_uri` **must** be https, **cannot** contain query parameters, and **will be** matched exactly (no wildcards). Allowed redirect URLs may be viewed/updated in the [Partner Portal](#) using the provided admin account.

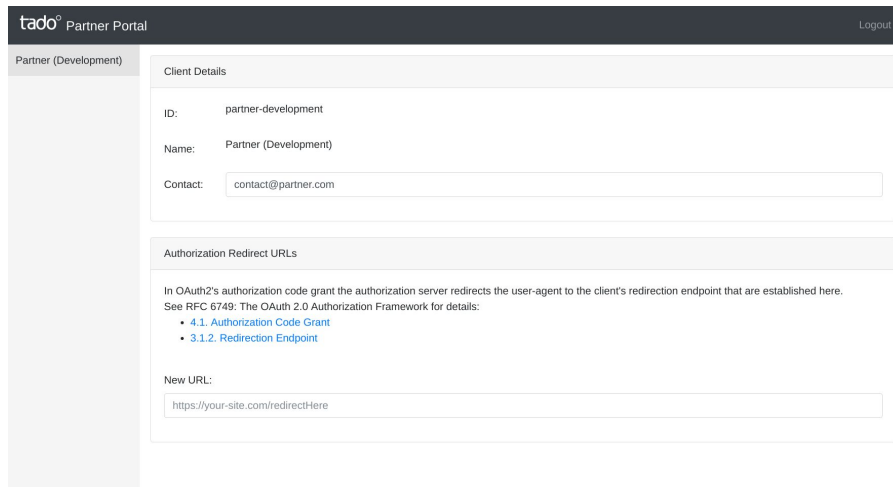
`state` (in the request) should be used to verify the subsequent response. This can be done e.g. through persistence or by signing the string. It should also be used to associate the received `code` with the user who initiated the authorization code flow.

`state` (in the response) **must be** verified by the server to ensure validity of the response.

Partner Portal

The Partner Portal allows for managing the **account contact** and the **accepted redirect_uris**.

Login with the provided admin account at <https://auth.tado.com/app/partner/#/login>



The screenshot displays the 'tado° Partner Portal' interface. On the left, a sidebar lists 'Partner (Development)'. The main content area is divided into two sections: 'Client Details' and 'Authorization Redirect URLs'.

Client Details

ID:	partner-development
Name:	Partner (Development)
Contact:	<input type="text" value="contact@partner.com"/>

Authorization Redirect URLs

In OAuth2's authorization code grant the authorization server redirects the user-agent to the client's redirection endpoint that are established here. See RFC 6749: The OAuth 2.0 Authorization Framework for details:

- [4.1. Authorization Code Grant](#)
- [3.1.2. Redirection Endpoint](#)

New URL:

Token request

POST `https://auth.tado.com/oauth/token`

Parameter	Value	Description
client_id	String	Client ID assigned by tado°
client_secret	String	Client secret assigned by tado°
redirect_uri	URI	Has to match redirect_uri from Auth request
code	String	The authorization code
grant_type	String	<code>authorization_code</code> (as defined in the OAuth 2.0 specification)

Example:

```
POST https://auth.tado.com/
  oauth/token
{
  "client_id": "partner-app",
  "client_secret": "top-secret",
  "redirect_uri": "https://app.partner.com",
  "code": "876GHT8",
  "grant_type": "authorization_code"
}
```

Response

JSON

Parameter	Value	Description
access_token	String	The access token (up to 2KB)
refresh_token	String	The refresh token for getting new access tokens (up to 2KB).
expires_in	Integer	Lifetime of the access token in seconds
token_type	String	Always value <code>Bearer</code>

Example:

```
{
  "access_token": "DjT9vaeJKnQhaPf",
  "refresh_token": "GfFgz7cqhqEqNfv",
  "expires_in": 600,
  "token_type": "Bearer"
}
```

Refresh token request

POST `https://auth.tado.com/oauth/token`

Parameter	Value	Description
client_id	String	Client ID assigned by tado°
client_secret	String	Client secret assigned by tado°
refresh_token	String	The refresh token acquired from auth request
grant_type	String	<code>refresh_token</code> (as defined in the OAuth 2.0 specification)

Example:

```
POST https://auth.tado.com/
  oauth/token
{
  "client_id": "partner-app",
  "client_secret": "top-secret",
  "refresh_token": "GfFgz7cqhQEqNfv",
  "grant_type": "refresh_token"
}
```

Response JSON

Parameter	Value	Description
access_token	String	The access token (up to 2KB)
refresh_token	String	The refresh token for getting new access tokens (up to 2KB). The new refresh token has to be used in the next token refresh request.
expires_in	Integer	Lifetime of the access token in seconds
token_type	String	Always value <code>Bearer</code>

Example:

```
{
  "access_token": "DjT9vaeJKnQhaPf",
  "refresh_token": "Hkosdfkasdpokl",
  "expires_in": 600,
  "token_type": "Bearer"
}
```

API call authentication `https://my.tado.com/<api>`

Header	Value	Description
Authorization	<code>Bearer <access_token></code>	Auth header with access token

Example:

GET `https://my.tado.com/api/v2/homes/1`

Scopes

Access to tado° API resources is limited by defined scopes. Currently PARTNER integration can access these scopes:

- `identity:read` - access user details
- `home.details:read` - access home & customer details
- `home.operation:read` - access current operation of home (ie. set point temperature, inside temperature, humidity, ...)
- `home.operation.overlay:write` - manually control home operation (overrides user's schedule)
- `home.webhooks` - manage webhooks

Authentication errors

Authentication errors are following the [OAuth 2.0 specification](#).

Example:

```
{  
  "error": "invalid_client",  
  "error_description": "Bad client credentials"  
}
```

API Resources

API base URL `<base_url>`

`https://my.tado.com/api/v2`

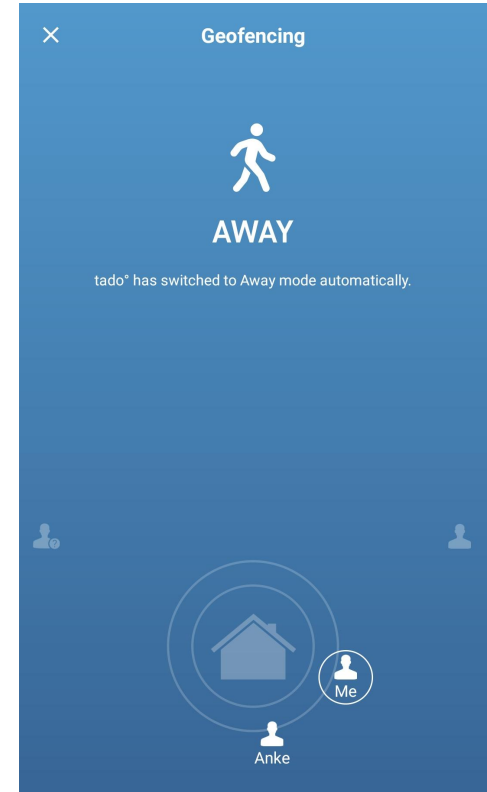
Common error codes

HTTP status	Response	Description
2xx		Default HTTP semantics (201, 204, ...)
400	<code>{"errors": [{"code": "<code>" , "title": "<title>"}]}</code>	Bad request, wrong type of parameters, etc.
401	<code>{"errors": [{"code": "unauthorized" , "title": "<title>"}]}</code>	User authentication failed
403	<code>{"errors": [{"code": "accessDenied" , "title": "<title>"}]}</code>	Authenticated user has no access rights to the requested entity
404	<code>{"errors": [{"code": "notFound" , "title": "<title>"}]}</code>	Requested entity was not found
422	<code>{"errors": [{"code": "<code>" , "title": "<title>"}]}</code>	Semantic application error
500	<code>{"errors": [{"code": "exception" , "title": "<title>"}]}</code>	Unexpected server error

User and Home

A tado° home is the central place for customers to operate and configure their heating and/or cooling and is uniquely identified via its ID.

Each resident of the home has a separate user account that is connected to the home. Residents can for example control the temperature or change configuration. Their geolocation is tracked individually.



Current user

GET <base_url>/me

No parameters

The information returned by this endpoint can also be obtained by decoding the JWT Token

Example:

GET <base_url>/me

Response JSON

Parameter	Value	Description
name	String	Full name of the user
email	String	Email of the user
username	String	Username of the user
homes	List<Object>	List of Homes of the user, with their IDs and names
locale	String	Locale string in format of Java Locale.toString()

Example:

```
{
  "name": "John Doe",
  "email": "john@doe.com",
  "username": "john@doe.com",
  "homes": [{
    "id": 1234,
    "name": "John's home"
  }, ...],
  "locale": "en"
}
```

Home

GET <base_url>/homes/<home_id>

No parameters

Example:

GET <base_url>/homes/1234

Response

JSON

Parameter	Value	Description
id	Number	ID of the home
name	String	User defined name for the home
dateTimeZone	String	Time zone ID of the home time zone
partner	String	tado° Partner if applicable

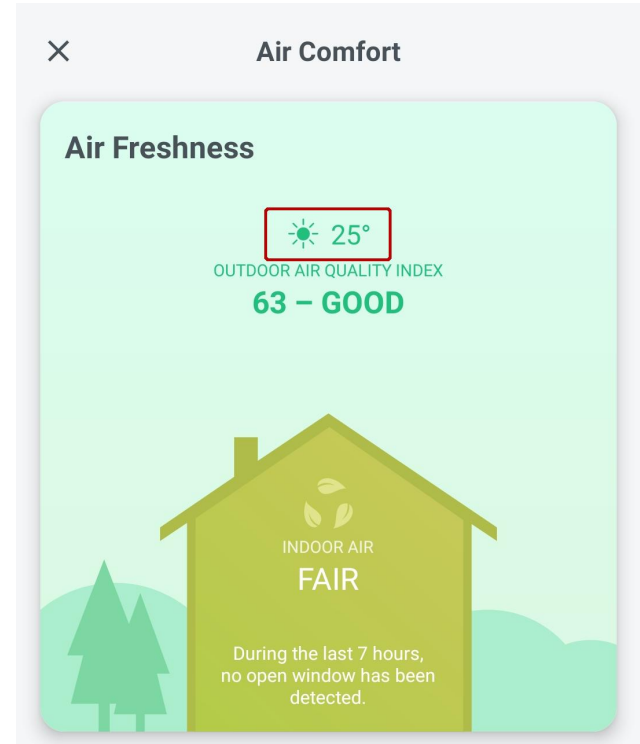
Example:

```
{
  "id": 1234,
  "name": "John's home",
  "dateTimeZone": "Europe/London",
  "partner": "PARTNER"
}
```

Weather

The current weather information at the customer's home location is an integral part of the predictive heating and cooling control developed at tado°.

The current weather, as known to tado°, can be retrieved via the weather API endpoint. This information is for example displayed in the Air Comfort Section of the Apps



Weather

GET <base_url>/homes/<home_id>/weather

No parameters

Example:

GET <base_url>/homes/1234/weather

Response

JSON

Parameter	Value	Description
solarIntensity	object	timestamp and percentage of current solar intensity
outsideTemperature	object	timestamp and celsius/fahrenheit values of current outside temperature
weatherState	object	timestamp and value of current weather conditions: WIND, THUNDERSTORM, SUN, RAIN, SCATTERED_RAIN, SNOW, SCATTERED_SNOW, RAIN_SNOW, SCATTERED_RAIN_SNOW, HAIL, RAIN_HAIL, NIGHT_CLOUDY, NIGHT_CLEAR, FREEZING, FOGGY, DRIZZLE, CLOUDY, CLOUDY_PARTLY, CLOUDY_MOSTLY

Example:

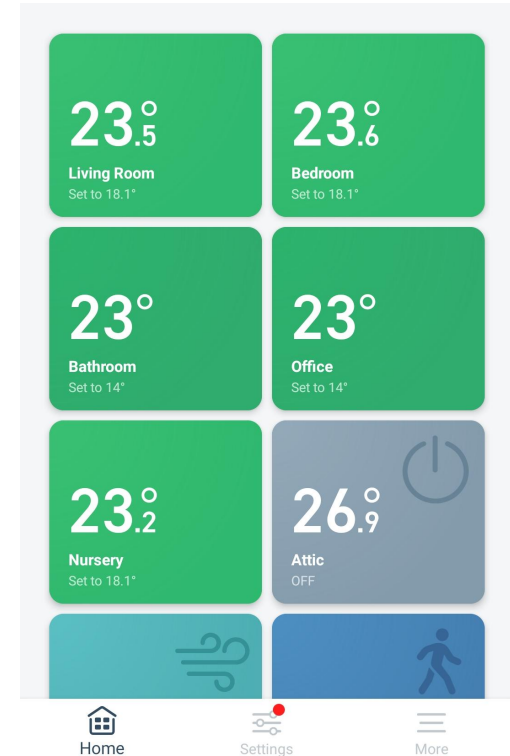
```
{
  "solarIntensity": {
    "type": "PERCENTAGE",
    "percentage": 49,
    "timestamp": "2016-01-25T14:21:58.207Z"
  },
  "outsideTemperature": {
    "celsius": 12,
    "fahrenheit": 53.6,
    "timestamp": "2016-01-25T14:21:58.207Z",
    "type": "TEMPERATURE"
  },
  "weatherState": {
    "value": "SUN",
    "timestamp": "2016-01-25T14:18:52.129Z"
  }
}
```

Zones

A zone, referred to as a room in tado° documentation, represents an area where climate can be controlled individually. Typically this correlates to individual rooms, e.g. Living Room, or building levels, like the First Floor.

tado° currently supports three zone types that refer to whether it can control heating, an air conditioning or the preparation of hot water.

Zones have their own state, can be controlled independently and follow their own schedule.



Zones

GET <base_url>/homes/<home_id>/zones

No parameters

Example:

GET <base_url>/homes/1234/zones

Response

JSON

List of zones. Each zone is represented by the following JSON object

Parameter	Value	Description
id	Number	ID of the zone (unique within one home)
name	String	Zone name
type	String	Zone type can be either HEATING, AIR_CONDITIONING or HOT_WATER

Example:

```
[
  {
    "id": 0,
    "name": "Hot water",
    "type": "HOT_WATER"
  },
  {
    "id": 1,
    "name": "Living room",
    "type": "AIR_CONDITIONING"
  },
  {
    "id": 2,
    "name": "Ground Floor",
    "type": "HEATING"
  },
  {
    "id": 3,
    "name": "Second Floor",
    "type": "HEATING"
  }
]
```

Zone State

No parameters

Response JSON

GET

<base_url>/homes/<home_id>/zones/<zone_id>/state

Example:

GET <base_url>/homes/1234/zones/1/state

Example:

```
{
  "tadoMode": "HOME",
  "preparation": {
    "tadoMode": "HOME"
  },
  "geolocationOverride": false,
  "overlay": null,
  "setting": {
    "type": "HEATING",
    "power": "ON",
    "temperature": {
      "celsius": 22.0,
      "fahrenheit": 71.6
    }
  },
  "openWindow": { /* ... */ },
  "link": {
    "state": "ONLINE"
  },
  "sensorDataPoints": {
    "insideTemperature": {
      "type": "TEMPERATURE",
      "celsius": 21.6,
      "fahrenheit": 70.9,
      "timestamp": "2015-07-24T09:18:17.884Z"
    },
    "humidity": {
      "type": "PERCENTAGE",
      "percentage": 73.5,
      "timestamp": "2015-07-24T09:18:17.884Z"
    }
  }
}
```

Parameter	Value	Description
tadoMode	String	One of HOME, AWAY
preparation	Object	Contains preparation target if tado is preparing for an upcoming mode, otherwise null.
preparation.tadoMode	String	One of HOME, AWAY
geolocationOverride	Boolean	Whether the geolocation is detected (through the geolocations of the app users) or overridden (by "always active" blocks of the block schedule)
overlay	Object	Manual Control
setting	Object	Current setting of the zone
openWindow	Object	Indicates, that open window is currently detected
link	Object	Link object of current zone connection
sensorDataPoints	Map	Map of sensor data points such as insideTemperature and humidity

Generic Zone Setting

Parameter	Value	Description
type	String	Tado system type <code>HEATING</code> , <code>AIR_CONDITIONING</code> or <code>HOT_WATER</code>

Heating Zone Setting

Parameter	Value	Description
power	String	Heating system power <code>ON/OFF</code>
temperature.celsius	Number	Temperature in Celsius
temperature.fahrenheit	Number	Temperature in Fahrenheit

Hot Water Zone Setting

Parameter	Value	Description
power	String	Hot water heater power <code>ON/OFF</code>
temperature.celsius	Number	Temperature in Celsius (only if temperature can be controlled, see Capabilities)
temperature.fahrenheit	Number	Temperature in Fahrenheit (only if temperature can be controlled, see Capabilities)

AC Zone Setting

Parameter	Value	Description
power	String	Heating system power ON/OFF
mode	String	One of COOL, HEAT, DRY, FAN, AUTO
temperature.celsius	Number	Temperature in Celsius
temperature.fahrenheit	Number	Temperature in Fahrenheit
fanSpeed (Deprecated)	String	One of LOW, MIDDLE, HIGH, AUTO
swing (Deprecated)	String	One of ON, OFF
fanLevel	String	One of SILENT, LEVEL1, LEVEL2, LEVEL3, LEVEL4, LEVEL5, AUTO
verticalSwing	String	One of OFF, ON, UP, MID_UP, MID, MID_DOWN, DOWN, AUTO
horizontalSwing	String	One of OFF, ON, LEFT, MID_LEFT, MID, MID_RIGHT, RIGHT, AUTO
light	String	One of ON, OFF

Open Window

Parameter	Value	Description
detectedTime	String	Time when open window was detected. ISO8601 datetime (e.g.: 2015-09-28T15:03:20Z) with second precision.
durationInSeconds	Number	The number of seconds that the open window was configured to last.
expiry	String	ISO8601 datetime (e.g.: 2015-09-28T15:03:20Z) with second precision.
remainingTimeInSeconds	Number	The number of seconds until expiry time.

Link

Parameter	Value	Description
state	String	ONLINE/OFFLINE
reason.code	String	Reason message key intended for i18n
reason.title	String	Short english error description, should not be presented to the user

Sensor Data Point

Parameter	Value	Description
type	String	Data type of the value
timestamp	DateTime	Measurement timestamp

Sensor Data Point: Temperature (used for inside temperature measurements)

Parameter	Value	Description
celsius	Number	Temperature in Celsius
fahrenheit	Number	Temperature in Fahrenheit

Sensor Data Point: Percentage (used for humidity measurements)

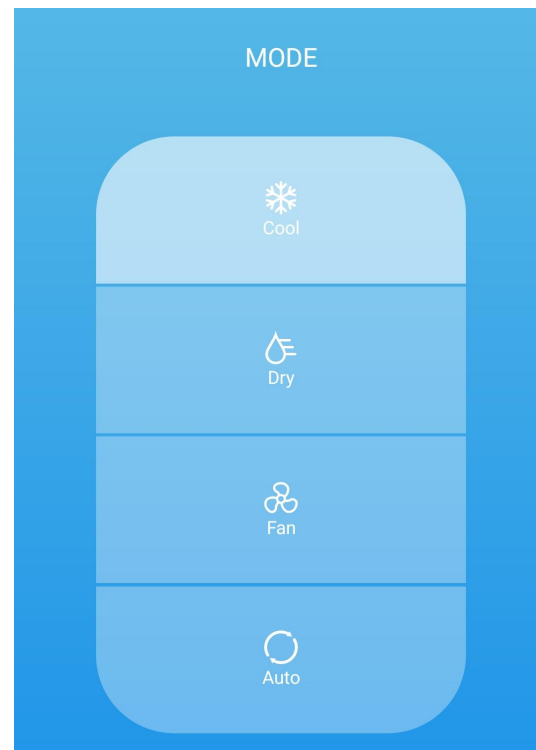
Parameter	Value	Description
percentage	Number	Relative value (0-100)

Zone Capabilities

Depending on the zone type and configuration, a zone has a set of capabilities that define how the climate can be affected. This includes how state is shown as well as how the zone can be controlled.

Examples are:

- Granularity of temperature setting (Full Degrees vs. 0.1 Degrees)
- Supported Modes for ACs (e.g. Cool Mode, Heat Mode)
- Support for setting the temperature of Hot Water preparation



Zone Capabilities

GET <baseurl>/homes/<home_id>/zones/<zone_id>/capabilities

No parameters

Example:

Response

JSON

GET <base_url>/homes/1234/zones/1/capabilities

Parameter	Value	Description
type	String	Tado system type HEATING, AIR_CONDITIONING or HOT_WATER.

Heating Zone Capabilities

Parameter	Value	Description
temperatures.celsius	Object	Temperature range in Celsius.
temperatures.fahrenheit	Object	Temperature range in Fahrenheit.

Temperature Range

Parameter	Value	Description
min	Number	Minimum temperature.
max	Number	Maximum temperature.
step	Number	Minimum step size (such as 1 or 0.1).

Hot Water Zone Capabilities

Parameter	Value	Description
canSetTemperature	Boolean	Whether tado° can control hot water temperature.
temperatures.celsius	Object	Temperature range in Celsius. Only set if <code>canSetTemperature</code> is <code>true</code> .
temperatures.fahrenheit	Object	Temperature range in Fahrenheit. Only set if <code>canSetTemperature</code> is <code>true</code> .

AC Zone Capabilities

Parameter	Value	Description
COOL	Object	Capabilities in COOL mode
HEAT	Object	Capabilities in HEAT mode
DRY	Object	Capabilities in DRY mode
FAN	Object	Capabilities in FAN mode
AUTO	Object	Capabilities in AUTO mode

AC Mode Capabilities

Parameter	Value	Description
temperatures.celsius	Object	Temperature range in Celsius.
temperatures.fahrenheit	Object	Temperature range in Fahrenheit.
fanSpeeds (Deprecated)	Array<String>	AC system available fan speeds (LOW, MIDDLE, HIGH, AUTO)
swings (Deprecated)	Array<String>	AC system available swing modes (ON, OFF)
fanLevel	Array<String>	AC system available fan levels (SILENT, LEVEL1, LEVEL2, LEVEL3, LEVEL4, LEVEL5, AUTO)
verticalSwing	Array<String>	AC system available vertical swings (OFF, ON, UP, MID_UP, MID, MID_DOWN, DOWN, AUTO)
horizontalSwing	Array<String>	AC system available horizontal swings (OFF, ON, LEFT, MID_LEFT, MID, MID_RIGHT, RIGHT, AUTO)
light	Array<String>	AC system available light mode (ON, OFF)

Zone Capabilities (I)

Heating Zone

```
{
  "type": "HEATING",
  "temperatures": {
    "celsius": {
      "min": 5,
      "max": 25,
      "step": 0.1
    },
    "fahrenheit": {
      "min": 41,
      "max": 77,
      "step": 0.1
    }
  }
}
```

Hot Water Zone

```
{
  "type": "HOT_WATER",
  "canSetTemperature": true,
  "temperatures": {
    "celsius": {
      "min": 30,
      "max": 70,
      "step": 1
    },
    "fahrenheit": {
      "min": 86,
      "max": 158,
      "step": 1
    }
  }
}
```

AC Zone

```
{
  "type": "AIR_CONDITIONING",
  "AUTO": {},
  "COOL": {
    "temperatures": {
      "celsius": {
        "min": 18,
        "max": 30,
        "step": 1
      },
      "fahrenheit": {
        "min": 64,
        "max": 86,
        "step": 1
      }
    }
  },
  "fanSpeeds": [
    "AUTO", "HIGH", "LOW"
  ],
  "DRY": {},
  "FAN": {},
  "HEAT": {}
}
```

Zone Capabilities (II)

AC Zone (full capabilities)

```
{
  "type": "AIR_CONDITIONING",
  "COOL": {
    "temperatures": {
      "celsius": {
        "min": 18,
        "max": 30,
        "step": 1
      },
      "fahrenheit": {
        "min": 64,
        "max": 86,
        "step": 1
      }
    },
    "fanLevel": ["SILENT", "AUTO"],
    "verticalSwing": [
      "OFF", "MID", "ON"
    ],
    "horizontalSwing": [
      "OFF", "MID", "ON"
    ],
    "light": ["OFF", "ON"],
  },
  ...
}
```

AC Zone (partial capabilities)

```
{
  "type": "AIR_CONDITIONING",
  "COOL": {
    "temperatures": {
      "celsius": {
        "min": 18,
        "max": 30,
        "step": 1
      },
      "fahrenheit": {
        "min": 64,
        "max": 86,
        "step": 1
      }
    },
    "fanLevel": [
      "SILENT", ..., "AUTO"
    ],
    "light": [
      "OFF", "ON"
    ],
  },
  ...
}
```

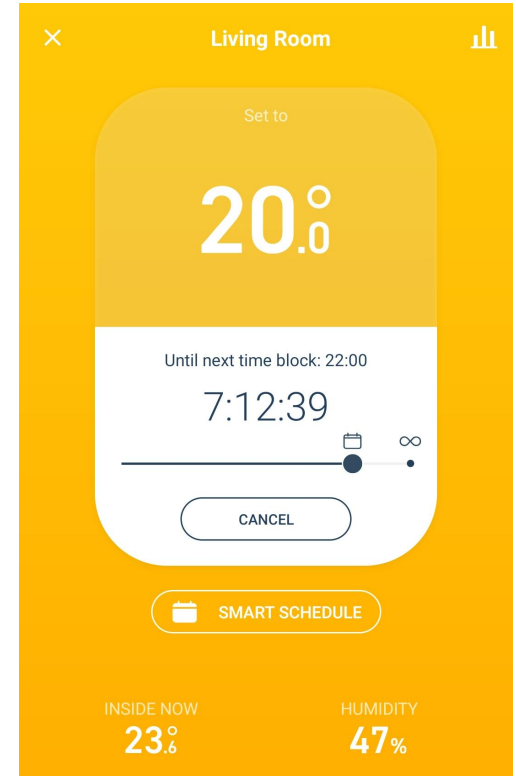
AC Zone (deprecated)

```
{
  "type": "AIR_CONDITIONING",
  "COOL": {
    "temperatures": {
      "celsius": {
        "min": 18,
        "max": 30,
        "step": 1
      },
      "fahrenheit": {
        "min": 64,
        "max": 86,
        "step": 1
      }
    },
    "fanSpeeds": [
      "AUTO", "HIGH", "LOW"
    ],
  },
  "HEAT": {}
}
```

Overlays

In regular operation tado° follows the schedule that has been set by the customer. To make temporary adjustments to the climate control, the customer can create a Manual Control - called Overlay in the API.

The overlay shown in the picture for example sets the temperature in the living room to 20 degrees until the next time block in the schedule is reached. There are more termination conditions to limit the duration of an overlay or have it stay active until manually ended.



Overlay

No parameters

GET <baseurl>/homes/<home_id>/zones/<zone_id>/overlay

Example:

GET <base_url>/homes/1234/zones/1/overlay

Response JSON

Parameter	Value	Description
setting	Object	Overlay setting
termination	Object	Overlay termination condition - determines how is the overlay removed

Example:

```
{
  "setting": {
    "type": "HEATING",
    "power": "ON",
    "temperature": {
      "celsius": 22.0,
      "fahrenheit": 71.6
    }
  },
  "termination": {
    "type": "MANUAL"
  }
}
```

Example 2:

```
{
  "setting": {
    "type": "HEATING",
    "power": "OFF"
  },
  "termination": {
    "type": "TADO_MODE"
  }
}
```

Termination Condition: Manual

Overlay is removed manually by user or external tool (Partner app, IFTTT, ..).

Parameter	Value	Description
type	String	Constant "MANUAL"

Termination Condition: Tado Mode

Overlay is removed automatically when tado mode (HOME, AWAY) changes due to schedule settings or geolocation update.

Parameter	Value	Description
type	String	Constant "TADO_MODE"

Termination Condition: Timer

Overlay is removed after timeout.

Parameter	Value	Description
type	String	Constant "TIMER"
durationInSeconds	Number	The number of seconds that the overlay should last/was configured to last. Max 24h.
expiry	String	ISO8601 datetime (e.g.: 2015-09-28T15:03:20Z) with second precision. Read only - should not be sent with the PUT request

Overlay

No parameters

JSON Body:

Parameter	Value	Description
setting	Object	Contains setting object of the current set point. Temperature object should contain only one temperature unit .
termination	Object	Overlay termination condition - determines how is the overlay removed

PUT <baseurl>/homes/<home_id>/zones/<zone_id>/overlay

Example:

PUT <base_url>/homes/1234/zones/1/overlay

```
{
  "setting": {
    "type": "HEATING",
    "power": "ON",
    "temperature": {
      "celsius": 22.0
    }
  },
  "termination": {
    "type": "MANUAL"
  }
}
```

Response JSON

Same as [GET Overlay](#)

Overlay

No parameters

DELETE

<baseurl>/homes/<home_id>/zones/<zone_id>/overlay

Example:

DELETE <base_url>/homes/1234/zones/1/overlay

Response

HTTP status 204

Health Check

GET <baseurl>/health

No parameters

Example:

```
GET <base_url>/health
```

Response JSON

Parameter	Value	Description
state	String	healthy/unhealthy
message	String	if unhealthy, more detailed error message
timestamp	DateTime	Last health update

Example:

```
{
  "state": "healthy",
  "timestamp": "2015-07-20T16:23:12Z"
}
```

Health check always returns HTTP status 200. Any HTTP error (4xx, 5xx) or timeout means that API is not available.

Home notifications (I)

- register HTTP callbacks (aka “webhooks”) for receiving events about homes
 - **tadoMode** - any time the tado° mode of any zone changes
 - **overlayType** - any time the overlay of any zone is activated or deactivated
 - **setting** - any time the current setting (or any part of it e.g. power, temperature, fan speed, ...) changes
 - **insideTemperature** - any time the temperature measured in a zone changes by a significant amount
 - **humidity** - any time the humidity measured in a zone changes by a significant amount
 - *temperature crossed specified value - to be further specified*

Home notifications (II)

- hooks
 - associated with customer account
 - APIs for register/unregister and list hooks
- (authenticated) HTTP POST callback

Register hook

POST <base_url>/homes/<home_id>/hooks

No parameters

JSON Body:

Parameter	Value	Description
events	List<String>	the events to be notified about
url	String	The URL to POST to when notifying
auth.basic	Object	Username and password for HTTP Basic Auth (optional)
auth.signature.secret	String	A secret string for authenticating the callbacks (optional)

Example:

```
POST <base_url>/homes/1234/hooks
{
  "events": ["tadoMode"],
  "url": "https://www.example.com/hook",
  "auth": {
    "basic": {
      "username": "user",
      "password": "pass"
    },
    "signature": {
      "secret": "secret"
    }
  }
}
```

Response

JSON, Location header

Parameter	Value	Description
id	Number	Hook id
events	List<String>	the events to be notified about
url	String	The URL to POST to when notifying

Example:

Location: <base_url>/home/<home_id>/hooks/<id>

```
{
  "id": 12,
  "events": ["tadoMode"],
  "url": "https://www.example.com/hook"
}
```

same info can be retrieved with GET <base_url>/homes/<home_id>/hooks/<id>

all hooks can be listed with GET <base_url>/homes/<home_id>/hooks

HTTP callback

POST <CALLBACK_URL>

Headers:

Header	Description
Tado-Home-Event	the occurred event
Authorization	(Preemptive) Basic authentication header created from auth.basic
X-Hub-Signature	base64 digest of an HMAC-SHA1 to verify authenticity of request, using the hooks auth.signature.secret as the key

Body

JSON

Parameter	Value	Description
timestamp	ISO8601 string	when the event occurred.
home	Object	info about the home for which the event occurred

Example:

POST <CALLBACK_URL>

Tado-Home-Event: tadoMode

X-Hub-Signature: sha1=9c2cead7a6e2731...

```
{
  "timestamp": "2015-08-10T14:53:52.829Z",
  "tadoMode": "HOME",
  "home": {
    ... // details about the home
  },
  "zone": {
    "name": "living room",
    "id": 2
  }
}
```

No retries, if POST fails.

Unregister hook `DELETE <base_url>/homes/<home_id>/hooks/<id>`

No parameters
Empty body

Example:

```
DELETE <base_url>/homes/1234/hooks/12
```

Response

HTTP status 204

Additional HTTP callback payload (I)

In addition to timestamp and home, there is additional, event-type dependent payload sent:

- **tadoMode** event payload properties:
 - [zone](#): Data about the zone for which the tado° mode changed
 - **tadoMode**: New tado mode (HOME, AWAY)
- **overlayType** event payload properties:
 - [zone](#): Data about the zone for which the overlay type changed
 - **overlayType**: The new type of overlay (MANUAL, TADO_MODE, TIMER, or null, if none)
- **setting** event payload properties:
 - [zone](#): Data about the zone for which the settings changed
 - [setting](#): New setting which is sent to the heating/AC/hot water control
 - [zoneState](#): The complete state of the zone for which the setting changed. Same as when issuing a GET to /homes/ID/zones/ID/state. See more above.

Additional HTTP callback payload (II)

- **insideTemperature** event payload properties:
 - **zone**: Data about the zone for which a significant temperature change was detected
 - **insideTemperature**: The latest measured temperature in the zone in the same format as the *insideTemperature* property of *sensorDataPoints* in a **zoneState** response.
- **humidity** event payload properties:
 - **zone**: Data about the zone for which a significant humidity change was detected
 - **humidity**: The latest measured humidity in the zone in the same format as the *humidity* property of *sensorDataPoints* in a **zoneState** response.

Exemplary setting HTTP payload

```
{
  "timestamp": "2015-08-10T14:53:52.829Z",
  "home": { "id": 42, "name": "John's home" },
  "zone": { "name": "living room", "id": 2, "type": "HEATING" },
  "setting": {
    "type": "HEATING",
    "power": "ON",
    "temperature": { "celsius": 22, "fahrenheit": 71.6 }
  },
  ...
}
```

```
...
"zoneState": {
  "tadoMode": "HOME",
  "geolocationOverride": false,
  "overlayType": null, // deprecated
  "overlay": null,
  "preparation": null,
  "setting": {
    "type": "HEATING",
    "power": "ON",
    "temperature": { "celsius": 22, "fahrenheit": 71.6 }
  },
  "link": { "state": "ONLINE" },
  "sensorDataPoints": {
    "insideTemperature": {
      "type": "TEMPERATURE",
      "celsius": 21.6,
      "fahrenheit": 70.9,
      "timestamp": "2015-07-24T09:18:17.884Z"
    },
    "humidity": {
      "type": "PERCENTAGE",
      "percentage": 73.5,
      "timestamp": "2015-07-24T09:18:17.884Z"
    }
  }
}
}}
```


API Gotchas

- **Settings** used in **Overlays** have to conform to the **Zone capabilities**. For example:
 - **Zones** that only contain Smart Radiator Thermostat allow temperature ranges with a **step** size 1. A request to create an **overlay** with a temperature of 18.5 will fail
 - For **AC Zones**, all supported settings need to be part of the **overlay** settings. If an **AC Zone** supports **fanSpeeds**, a request to create an overlay without **fanSpeeds** will fail

Cross cutting aspects

- **media type:** application/json
- some APIs provide **ETags** that can be used for conditional requests
- right now, there are **no CORS headers**
- **timestamps** are in the ISO8601 format:

YYYY-MM-DDTHH:MM:SSZ
- there is **no rate limiting** in place right now
- **redirects** should be followed (301, 302, 307)

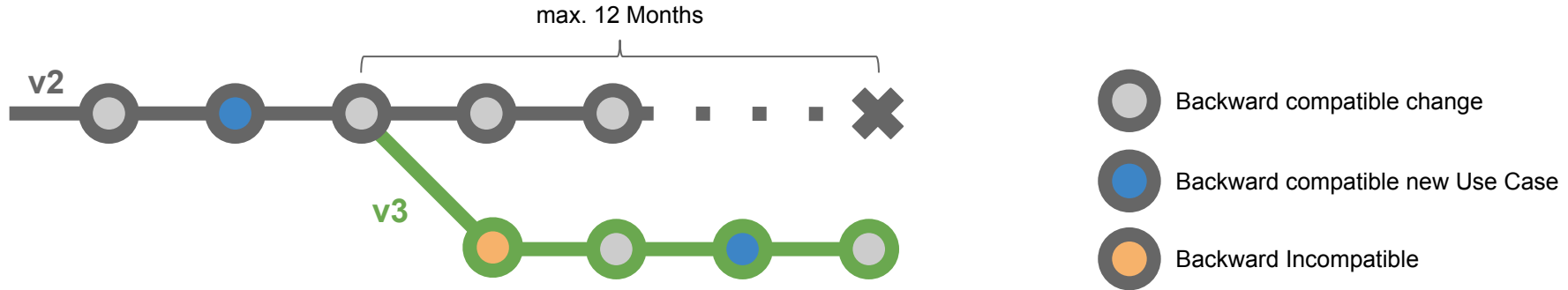
evolution of the API (I)

- **backward compatible** changes
 - might happen without notice. That includes
 - added properties on JSON objects
 - added URL endpoints, added methods for existing URLs
 - added ENUM values

evolution of the API (II)

- **backward incompatible** changes
 - result in a new version for the **whole API**. These changes include:
 - removed properties, that are documented in this document, on JSON objects
 - removed URL endpoints or removed methods on endpoints
 - changed (enumerated) values for constants (e.g. “MANUAL”)

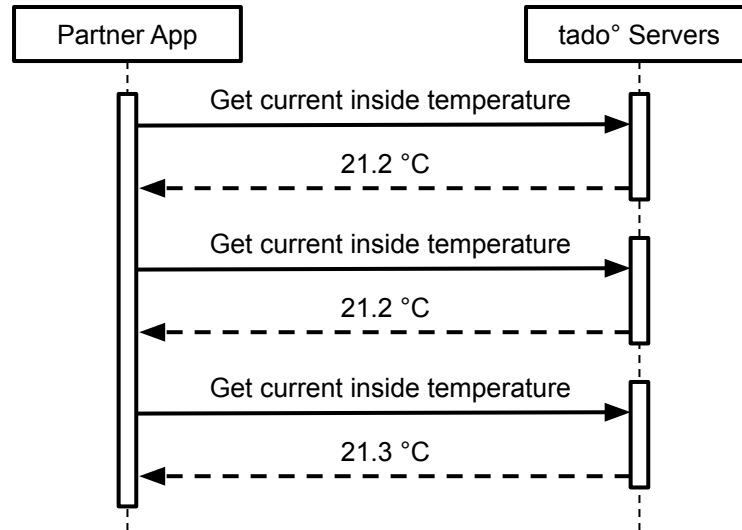
evolution of the API (III)



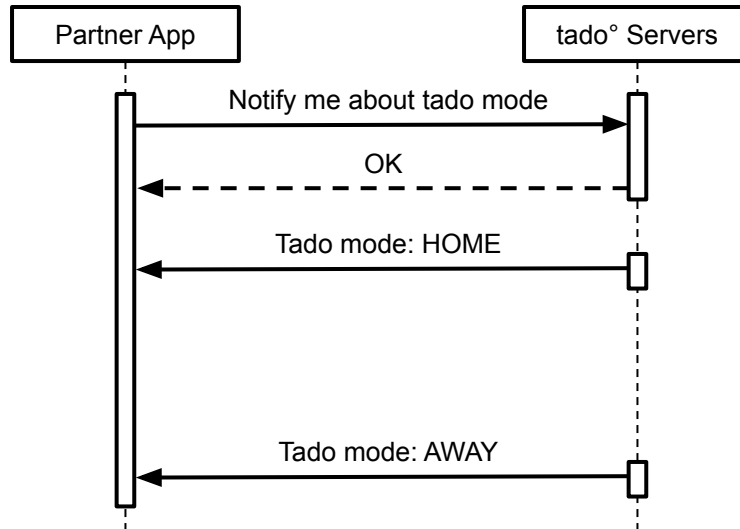
- Additional use cases to current version (backward compatible) will be added to this documentation and communicated to partners to allow partners at their sole discretion to also add the new use case on their platform
- After a new (backward incompatible) version is introduced, the old version will be supported for a maximum of 12 months to give partners time to migrate to the new API

BACKUP

Regular API interaction



Webhook interaction



SCOPES

=====

identity

identity:read

identity:write

home

home:read

home:write

home.overlay

home.overlay:read

home.overlay:write

Zones

GET <base_url>/homes/<home_id>/zones

No parameters

Example:

GET <base_url>/homes/1234/zones

Response

JSON

List of zones. Each zone is represented by the following JSON object

Parameter	Value	Description
id	Number	ID of the zone
name	String	Zone name
type	String	Zone type can be either HEATING, AIR_CONDITIONING or HOT_WATER

Example:

```
[
  {
    "id": 4567,
    "name": "Living room",
    "type": "AIR_CONDITIONING"
  },
  {
    "id": 4568,
    "name": "Ground Floor",
    "type": "HEATING"
  },
  {
    "id": 13715,
    "name": "Second Floor",
    "type": "HEATING"
  },
  {
    "id": 13717,
    "name": "Hot water",
    "type": "HOT_WATER"
  }
]
```

Zone Configuration - Hot Water

No parameters

Response JSON

GET <base_url>/homes/<home_id>/zones/<zone_id>/config

Parameter	Value	Description
type	String	tado system type HEATING, AIR_CONDITIONING or HOT_WATER
temperatureControl	Boolean	Whether tado° can control hot water temperature

Example:

GET <base_url>/homes/1234/zones/4567/config

Example:

```
{
  "type": "HOT_WATER",
  "temperatureControl": true
}
```

Zone State - Hot Water

No parameters

Response JSON

GET <base_url>/homes/<home_id>/zones/<zone_id>/state

Parameter	Value	Description
tadoMode	String	One of HOME, AWAY
preparation	Object	Contains preparation target if tado is preparing for an upcoming mode, otherwise null.
preparation.tadoMode	String	One of HOME, AWAY
geolocationOverride	Boolean	Whether the geolocation is detected (through the geolocations of the app users) or overridden (by "always active" blocks of the block schedule)
overlayType	String	MANUAL, TADO_MODE, TIMER or null if no overlay is present
setting	Object	Current setting of the zone
link	Object	Link object of current zone connection
sensorDataPoints	Map	Map of sensor data points such as waterTemperature (optional) for hot water

Example:

GET <base_url>/homes/1234/zones/4567/state

Example:

```
{
  "tadoMode": "HOME",
  "preparation": {
    "tadoMode": "HOME"
  },
  "geolocationOverride": false,
  "overlayType": null,
  "setting": {
    "type": "HOT_WATER",
    "power": "ON",
    "temperature": {
      "celsius": 60.0,
      "fahrenheit": 140.0
    }
  },
  "link": {
    "state": "ONLINE"
  },
  "sensorDataPoints": {
    "waterTemperature": {
      "type": "TEMPERATURE",
      "celsius": 56.2,
      "fahrenheit": 133.16,
      "timestamp": "2015-07-24T09:18:17.884Z"
    }
  }
}
```

Generic Zone Setting

Parameter	Value	Description
type	String	tado system type <code>HEATING</code> , <code>AIR_CONDITIONING</code> or <code>HOT_WATER</code>

Hot Water Zone Setting

Parameter	Value	Description
power	String	Hot water heater power <code>ON/OFF</code>
temperature.celsius	Number	Temperature in Celsius (only if temperature can be controlled)
temperature.fahrenheit	Number	Temperature in Fahrenheit (only if temperature can be controlled)

Overlay - Hot Water

No parameters

GET <baseurl>/homes/<home_id>/zones/<zone_id>/overlay

Response JSON

Parameter	Value	Description
type	String	MANUAL, TADO_MODE, TIMER
setting	Object	Overlay setting

Example 2:

```
{
  "type": "MANUAL",
  "setting": {
    "type": "HOT_WATER",
    "power": "OFF"
  }
}
```

Example:

GET
<base_url>/homes/1234/zones/4567/overlay

Example:

```
{
  "type": "MANUAL",
  "setting": {
    "type": "HOT_WATER",
    "power": "ON",
    "temperature": {
      "celsius": 60.0,
      "fahrenheit": 140.0
    }
  }
}
```

Example 3 (temperature not supported):

```
{
  "type": "MANUAL",
  "setting": {
    "type": "HOT_WATER",
    "power": "ON"
  }
}
```

Overlay - Hot Water

PUT <baseurl>/homes/<home_id>/zones/<zone_id>/overlay

No parameters

JSON Body:

Parameter	Value	Description
type	String	MANUAL, TADO_MODE, TIMER
setting	Object	Contains setting object. Temperature object should contain only one temperature unit.

Example:

PUT <base_url>/homes/1234/zones/4567/overlay

```
{
  "type": "MANUAL",
  "setting": {
    "type": "HOT_WATER",
    "power": "ON",
    "temperature": {
      "celsius": 65.0
    }
  }
}
```

Example:

```
{
  "type": "MANUAL",
  "setting": {
    "type": "HOT_WATER",
    "power": "ON",
    "temperature": {
      "celsius": 65.0,
      "fahrenheit": 149.0
    }
  }
}
```

Response JSON

Parameter	Value	Description
type	String	MANUAL, TADO_MODE, TIMER
setting	Object	Contains setting object.

Overlay - Hot Water

No parameters

```
DELETE <baseurl>/homes/<home_id>/zones/<zone_id>/overlay
```

Example:

```
DELETE <base_url>/homes/1234/zones/4567/overlay
```

Response

HTTP status 204

Home notifications

- register HTTP callbacks (aka “webhooks”) for receiving events about homes
 - tadoMode - any time the tado° mode of any zone changes
 - overlayType - any time the overlay of any zone is activated or deactivated
 - insideTemperature - any time a new inside temperature is measured in any zone (only changed bigger than 0.1K will be reported)
 - setting - any time the setting of any zone changes
 - link - any time device connects to the server / disconnects from the server
- hooks
 - associated with OAuth refresh token
 - APIs for register/unregister and list hooks
- (authenticated) HTTP POST callback

Register hook

POST <base_url>/homes/<home_id>/hooks

No parameters

JSON Body:

Parameter	Value	Description
events	List<String>	the events to be notified about
url	String	The URL to POST to when notifying (can contain query parameters)
auth.basic	Object	Username and password for HTTP Basic Auth (optional)
auth.signature.secret	String	A secret string for authenticating the callbacks (optional)

Example:

```
POST <base_url>/homes/1234/hooks
{
  "events": ["insideTemperature",
             "setting"],
  "url": "https://www.example.com/hook",
  "auth": {
    "basic": {
      "username": "user",
      "password": "pass"
    },
    "signature": {
      "secret": "secret"
    }
  }
}
```

Response

JSON, Location header

Parameter	Value	Description
id	Number	Hook id
events	List<String>	the events to be notified about
url	String	The URL to POST to when notifying

Example:

Location: <base_url>/home/<home_id>/hooks/<id>

```
{
  "id": 12,
  "events": ["insideTemperature",
             "setting"],
  "url": "https://www.example.com/hook"
}
```

same info can be retrieved with GET <base_url>/homes/<home_id>/hooks/<id>

all hooks can be listed with GET <base_url>/homes/<home_id>/hooks

HTTP callback

POST <CALLBACK_URL>

Headers:

Header	Description
Tado-Home-Event	the occurred event
Authorization	(Preemptive) Basic authentication header created from auth.basic
X-Hub-Signature	base64 digest of an HMAC-SHA1 to verify authenticity of request, using the hooks auth.signature.secret as the key

Body JSON

Parameter	Value	Description
home	Object	info about the home for which the event occurred
<payload>	...	other attributes, based on event - for tadoMode event: <ul style="list-style-type: none">- tadoMode property with new tado° mode- zone property for which the tado° mode changed

Example:

POST <CALLBACK_URL>

Tado-Home-Event: insideTemperature

X-Hub-Signature: sha1=9c2cead7a6e2731...

```
{
  "timestamp": "2015-08-10T14:53:52.829Z",
  "insideTemperature": {
    "type": "TEMPERATURE",
    "celsius": 21.6,
    "fahrenheit": 70.9,
    "timestamp": "2015-08-10T14:53:52.829Z"
  },
  "home": {
    ... // details about the home
  }
  "zone": {
    "name": "living room",
    "id": 44453
  }
}
```

No retries, if POST fails.

OAuth 2 details

Environments

We are providing access to two environments:

Staging

API base URL: [<https://cidevelop.tado.com/api/v2>]

Staging environment should be used for development of early prototypes and for testing the API.

Production

API base URL: [<https://my.tado.com/api/v2>]

Production environment should be used for integration, acceptance testing, testing with tado° hardware and for end customers.

Authentication

tado° API uses (OAuth 2)[<https://tools.ietf.org/html/rfc6749#section-4.1>] for API authentication. For integration with PARTNER these flows are available:

* (Authorization Code Grant)[<https://tools.ietf.org/html/rfc6749#section-4.1>]

Scopes

Access to tado° API resources is limited by defined scopes. Currently PARTNER integration can access these scopes:

- * `identity:read` - access user details
- * `home.details:read` - access home & customer details
- * `home.operation:read` - access current operation of home (ie. set point temperature, inside temperature, humidity, ...)
- * `home.operation.overlay:write` - manually control home operation (overrides user's schedule)
- * `home.webhooks` - manage webhooks

Client details