# Der "ultimative" ioBroker Lovelace Leitfaden ;)

Dieses Dokument entstand nach längerer Suche/Recherche/Ausprobieren und soll zukünftigen Benutzern als Hilfsmittel dienen einen schnellen aber doch auch tiefgreifenden Einstieg in Lovelace zu finden. Die Bezeichnung "ultimativ" im Leitfaden ist eher "scherzhaft" gemeint, da zum Zeitpunkt der Erstellung dieses Dokuments "bruchstückenhafte" Dokumentationen existieren, die der Autor nun versucht hat in diesem Leitfaden "ultimativ" zusammenzuführen. Daher erhebt dieser Leitfaden keine absolute Vollständigkeit und Richtigkeit und kann garantiert kontinuierlich erweitert werden.

## Versionshistorie

1.0 - 12.08.2020 - Ersterstellung durch Tirador (der Autor über Nachrichtenfunktion im ioBroker-Forum erreichbar)
1.1 - 19.08.2020 - Kleine Ergänzungen
1.2 - 30.08.2020 - Überabeitung diverser Punkte nach Rückmeldung von Garfonso
1.3 - 19.09.2020 - Call-Service ergänzt (Danke at **KNXbroker**)

## Inhaltsverzeichnis

# Einleitung Lovelace

Lovelace ist die Standard-Visualisierung der Heimautomatisierungssoftware "Home Assistant" und wurde auf ioBroker als eigenständiger Adapter portiert. Damit stehen die wesentlichen Lovelace-Funktionen nun auch unmittelbar für ioBroker zur Verfügung.

Lovelace ermöglicht es eine individualisierte Visualisierung zu erstellen, die folgende Features hat:

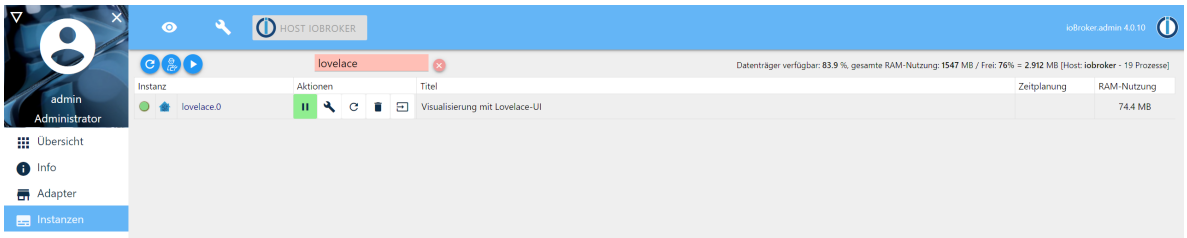
- Browserbasierte Visualisierung
- Responsive Design (Auflösungsunabhängig, funktioniert auf Handy, Tablet, Desktop)
- Moderne Oberflächen
- Schnelle Reaktionszeiten
- Konfiguration im Browser durch Bedieneroberfläche
- Erweiterte Konfiguration durch YAML-Syntax
- Themes
- Erweiterbare Karten / Custom Cards

Lovelace eignet sich gegenüber den anderen Visualisierungen von ioBrokern besonders, weil es sehr schnelle Ergebnisse erzielt und eine weitestgehende Konfigurationen ohne Details in Programmiersprachen erfordert.

Demo von Lovelace (basierend auf Home Assistant).

# Installation

Die stabile Version von Lovelace für ioBroker kann über die ioBroker-Administrationsseite installiert werden.
Die Installation erfolgt analog (wie für andere Adapter) über die Adapter-Seite.  In der Standardauslieferung von ioBroker erhält man damit die letzte stabile veröffentlichte Version.
Nach der Installation sollte der Adapter als Instanz unter Instanzen auftauchen. Der Adapter sollte "grün" sein (fehlerfreier Start).




Anmerkung:

Installation einer durch den Benutzer ausgewählten Version:

1. Unter "Adapter" den Expertenmodus aktivieren.
2. Beim Lovelace-Adapter ganz hinten auf "Bestimmte Version installieren" klicken.
3. Dort kann man sich die gewünschte Version aussuchen.

# Konfiguration

## Instanz-Konfiguration in ioBroker

### Initiale Konfigurationseinstellungen

IP / Port: Vorgabe von welchen IPs und unter welchem Port die Visualisierung von Lovelace erreichbar ist.

Historische Instanz: hier kann ausgewählt werden woher Lovelace die Historisierung der Entitäten ermitteln soll. Eine Konfiguration ist sinnvoll, wenn man in der Visualisierung von Lovelace die integrierten Diagrammverläufe und historischen Zustände von Entitäten nutzen möchte. Beispiele für historische Instanzen sind "influxdb.0".

Beispiel: Adapterkonfiguration Lovelace-Adapter

image-20200814153438554

## Entitäten

Lovelace arbeitet nicht mit den in ioBroker verwendeten "Objekten" bzw. "Datenpunkten", wie man es in anderen Visualisierungen (wie ioBroker VIS) kennt, sondern baut auf der Datenstruktur "Entität"  (engl. Entity) von Home Asistant auf. Eine Entität repräsentiert ein Gerät / Service beziehungsweise Objekt, dass eine definierte Struktur hat und aus mehreren Eigenschaften besteht.

Die folgenden Entitäten werden durch den Lovelace-Adapter von ioBroker unterstützt:

- Binary Sensor Entity: Sensor mit zwei Eigenschaften (An/Aus bzw. True/False). Ein solche Entität ist entsprechend ein Kontakt (Offen/Geschlossen, Fenster Auf/zu etc.)
- Light Entity: Repräsentiert ein Licht mit den Eigenschaften An/Aus, Helligkeit, Sättigung, Farbe, Farbtemperatur

- Media Player Entity
- Sensor Entity
- Switch Entity
- Weather Entity
- Lock Entity: Repräsentiert ein Schloss (geöffnet, geschlossen)
- Climate Entity: Klimasteuerung für Temperaturkontrolle, Feuchtigkeit, Ventilatoren und Klimageräten

Hinweis: Die folgenden Entitäten sind NICHT im Lovelace Adapter unterstützt:

- Vacuum Entity: Staubsauger
- Water Heater Entity: Heizung
- Fan Entity: Ventilator
- Remote Entity
- Air Quality

## Erzeugen von Entitäten

Damit Lovelace etwas visualisieren kann sind entsprechende "Entitäten" von ioBroker zur Verfügung zu stellen. Dafür sind die Datenpunkte von ioBroker in Entitäten zu überführen. Hierfür existieren die folgenden Wege:

- Automatische Erzeugung von Entitäten
- Manuelle Zuordnung von Entitäten
- Erzeugung von Entitäten über den Adapter "TODO"

Die Möglichkeiten sind in den folgenden Abschnitten behandelt.

**Automatische Erzeugung von Entitäten**

Der Lovelace-Adapter kann automatisiert Entitäten erkennen, typisieren und anlegen. Allerdings hat dies zwei Voraussetzungen, die der Benutzer sicherstellen muss.

Jedem Objekt in ioBroker sind zwingend IMMER die folgenden Dinge zuzuordnen:

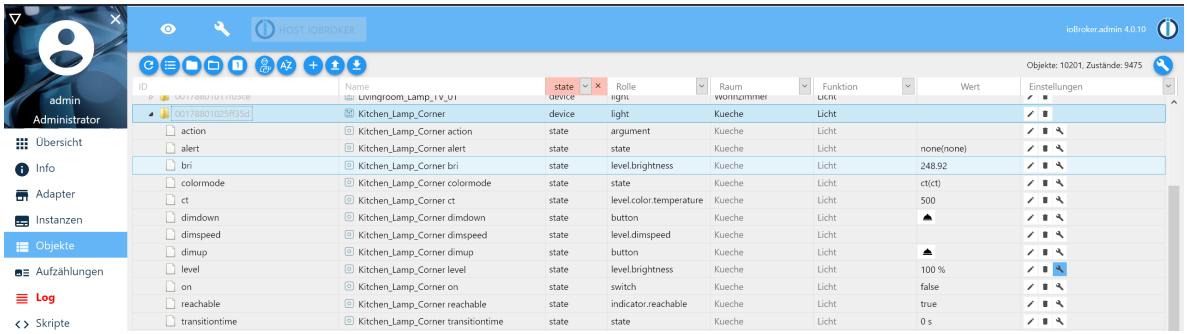
1. Raum
2. Funktion

Bei Objekten, die nicht nur aus einem Datenpunkt bestehen sollte immer der ganze Objektbaum (ab dem Ordner) Raum und Funktion zugeordnet werden. Sofern man Räume und Funktionen erweitert, sollte man darauf achten keine Sonderzeichen  in den Räumen und Funktionen zu verwenden. Insbesondere sollte man daher die folgenden Zeichen gänzlich vermeiden:

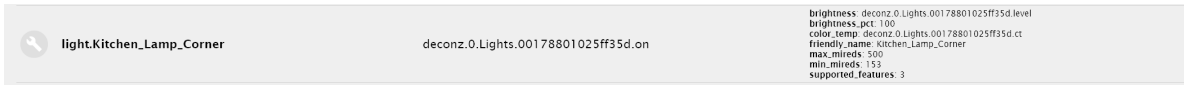
- Deutsche Umlaute / Sonderzeichen: ä,ü,ö, ß, ..
- Leerzeichen
- Unterstrich _
- Strich -

Beispiel: Erkennung einer Entität "Licht".

Dem folgenden Objektzweig "Kitchen_Lamp_Corner" wurde der Raum "Kueche" und Funktion "Licht" zugeordnet:




Aus den zugeordneten Raum "Kueche" und Funktion "Licht" leitet ioBroker die folgende Entität ab:




Anschließend kann das Licht direkt in den Cards in Lovelace verwendet werden.

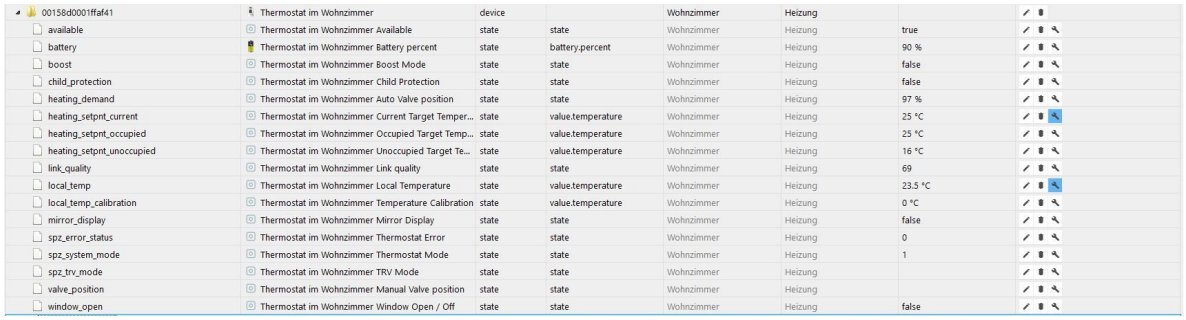
So wie im Beispiel erkennbar, kann ioBroker anhand des zugeordneten Raums und der Funktion die Entitäten automatisch erzeugen. Dafür nutzt ioBroker intern den sogenannten "TypeDetector". Der "TypeDetector" versucht aus den Eigenschaften der Datenpunkte, die den gleichem Raum/Funktionen zugeordnet sind ein oder mehrere Entitäten automatisch abzuleiten. In einem zweiten Schritt baut der Lovelace-Adapter dann auf die vom TypeDetector erkannten Eigenschaften eine Überleitung zu den "Home Assistant"-Entitäten. In den meisten Fällen funktioniert dies gut. Es kann jedoch Fälle geben, in denen weitere Eigenschaften an den Datenpunkten notwendig sind, damit die Erkennung sicher funktioniert. Beeinflussen kann man die bessere Erkennung durch den "TypeDetector" über die zugeordnete "Rolle" auf einem Datenpunkt. Eine "Rolle" definiert eine semantische Bedeutung für den Datenpunkt.

Beispiel: Rollen für Sensoren werden unterschieden in folgende Eigenschaften:

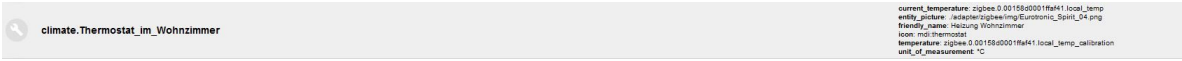
- sensor.alarm - Alarmsensor
- sensor.alarm.fire - Feueralarm
- sensor.alarm.flood - Wasseralarm
- sensor.rain - Regen erkannt
- sensor.window - Fenster geöffnet
- ...

Eine Auflistung aller "Rollen" von ioBroker findet sich hier und hier.

Beispiel: An diesem konkreten Beispiel eines Heizungsthermostat wird aufgezeigt, wie durch die Veränderung der "Rolle" von Datenpunkten die automatische Erkennung der Entität beeinflusst werden kann. Ein Thermostat besitzt die folgenden Datenpunkte. Es wurde der Raum "Wohnzimmer" und die Funktion "Heizung" zugeordnet.



Dennoch wird das Thermostat leider nicht automatisch als Entität "Thermostat" erkannt. Hingegen wird der Thermostat als folgende Entität erkannt:




Es ist erkennbar, dass der Thomastat nur eine Temperatur-Eigenschaft besitzt, aber kein Steuerungselement (um die Temperatur zu beeinflussen). Für das konkrete Beispiel ist zu identifizieren, welche Datenpunkte die Steuerung der Heizung übernehmen.

**heating_setpnt_current** ist die Soll-Temperatur.
**local_temp** ist die Ist-Temperatur.

Die zugewiesenen Rollen der Datenpunkte sind "value.temperature". Dadurch ist es dem TypeDetector nicht möglich zu erkennen, welcher Datenpunkt die Soll oder Ist-Temperatur ist. Anschließend sind die Rollen eindeutig vorzugeben:

Die Rolle von *heating_setpnt_current* ist von **value.temperature** auf **level.temperature** zu ändern.
Die Rolle von *local_temp* ist von **value.temperature** auf **value.current_temperature** ändern.

Anschließend kann die Erkennung neu durchgeführt werden durch Lovelace.

Hinweis: Sofern Rollen von einem Adapter nicht korrekt vorgegeben werden sollte ein Issue / Bug für den entsprechenden Adapter gemeldet werden, damit sich das Verhalten global verbessert. Die vorgeschlagenen Änderungen der Rollen wirken sich meist nur temporär aus und können durch den Adapter jederzeit wieder verändert werden.

Beispiel 2: Richtige Darstellung von Fenstern für "offen" / "geschlossen" / "gekippt". Hierfür sind folgende Einstellungen notwendig:

- Rolle des Datenpunkts für den Fenstersensor ist auf "value.window" zu setzen.
- Datentyp auf "number"
- Im RAW Objekt des Datenpunkts ist das "states" Feld richtig zu zu setzen. Mit den States kann eine Übersetzung der Werte des Datenpunkts in einen String erfolgen. Damit können beliebige Zustände übersetzt werden (je nach vorliegendem Objekttyp)

  Beispiel:

```
{
...
  "common": {
    "name": "Bad Oben Fenster",
    "role": "value.window",
    "type": "number",
    "read": true,
    "write": false,
    "states": {
      "0": "closed",
      "1": "open",
      "2": "tilted"
    },
  },
...
}
```

Hinweis: Sofern Funktionen, Räume oder Rollen geändert wurden können die Entitäten neu geladen werden in der Lovelace-Adapterkonfiguration im Segment "Entitäten" durch den Button "Entitäten neu laden". Damit die Entitäten auch in Lovelace Visualisierung zur Verfügung stehen ist die Seite im Browser neu zu laden. Alternativ kann auch der Adapter neu gestartet werden.

Hinweis 2: Sofern Entitäten umbenannt werden gibt es noch eine Schwachstelle im Adapter. Entitäten werden nicht gelöscht -> bis zum vollständigen Neustart des Adapters bleiben sie auch unter dem alten Namen erhalten. Es ist daher zu empfehlen in solchen Konstellation den Lovelace-Adapter komplett neuzustarten.

### Manuelle Zuordnung von Entitäten

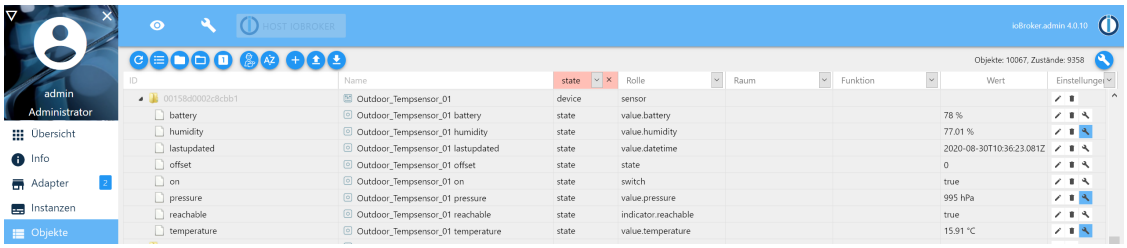
Entitäten können auch manuell angelegt werden. Das Verfahren eignet sich nur für genau jeweils EINEN Datenpunkt, der damit als Entität für Lovelace freigegeben wird. Es ist nicht möglich komplexere Entitäten, die aus mehreren Datenpunkten bestehen über dieses Verfahren zu generieren. Die manuelle Zuordnung von Entitäten eignet sich damit in der Regel nur für einfache Datenpunkte, die vollständig in sich den Objektzustand der Entität beschreiben.

Beispiele sind:

- Sensoren (Temperatur, Luftfeuchte, Fenster- oder Türkontakte)
- Einfache Lichter (An/Aus)
- Einfache Schalter (An/Aus)
- Eingabefelder (Zahlen, Text, Boolescher Wert)
- ...

Die manuelle Zuordnung einer Entität für Lovelace erfolgt wie folgt:

1. Aufruf "Objekte" in der ioBroker Admin-Oberfläche.
2. Auswahl "Datenpunkt" der als Entität angelegt werden soll.
3. Klick auf das Symbol "Schraubenschlüssel" (ganz rechts)




4. Es öffnet sich ein Dialog "Einstellungen von ...". Dort kann man "Einstellungen für Lovelace auswählen" und folgende Eigenschaften bestimmen:
    1. Aktiviert: die Box ist anzuhaken.
    2. Entitätstyp: Zuordnung eines Entitätstyps. Es stehen folgende Entitäten zur Auswahl:
        - **automatisch**: Automatische Erkennung des Entitätstyps durch Lovelace und den TypeDetector. Der TypeDetector und Lovelace müssen den Typ kennen/unterstützen. Ist dies der Fall dann wird auch die richtige "device_class" in der Entität gesetzt und aus z.B. on/off kann auch "Offen / Geschlossen" werden oder Lovelace weiß, dass ein Sensor eine Temperatur ist usw.
        - **Sensor**: Bei Sensoren die Datenwerte ausgeben und mehrere Zustände besitzen kann man diesen Entitätstyp wählen
          Beispiele: Temperatursensor, Feuchtigkeitssensor, Batteriezustand in Prozent, etc.

- **binary_sensor**: Verwendung für Sensoren, die binäre Zustände verwenden ("An" und "Aus", 0 oder 1, true oder false etc.).
  Beispiele: Kontakten (Tür geöffnet / geschlossen), Wassermelder (Wasser, Kein Wasser), Alarmmelder (Alarm, Kein Alarm).
- **Licht (light)**: Verwendung für einfache Lichter (An/Aus oder einem Datenpunkt für die Dimmfunktion eines Lichts).
- **Eingabe - wahrheitswert (input_boolean)**: Boolesches Eingabefeld.  Für normale Eingabefelder gelten Voraussetzungen: siehe GitHub.
- **Eingabe - Zahl (input_number)**:  Eingabefeld für Zahlenwerte. Für diesen Entitätstyp ist es Pflicht in der "RAW-Konfiguration" des Datenpunkts einen minimalen und maximalen Wert `min` and `max` vorzugeben. Optional kann das Attribut `step` ergänzt werden. Die Darstellung als Slider kann beeinflusst werden, indem das Attribtut `mode` auf 'number' gesetzt wird.

  Beispiel:

  ```
  common: {
      custom: {
          "lovelace.0": {
              "enabled": true,
              "entity": "input_number",
              "name": "Shutter" // this is a name how the entity will
  be called. In this case "alarm_control_panel.simulateAlarm"
              "mode": "number", // default presentation is slider
          }
      }
  }
  ```

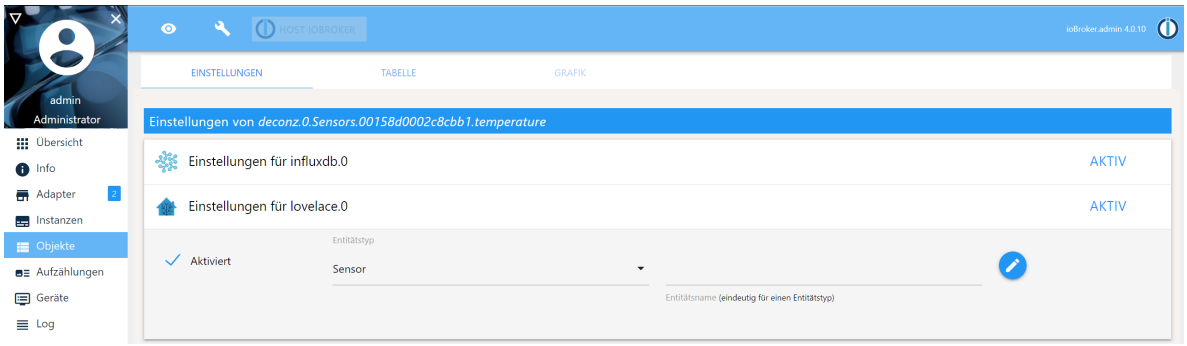
- **Eingabe mit Optionen (input_select)**: Eingabefeld als "Dropdown" mit einer Liste von Vorgabewerten. Die Vorgabewerte (Optionen) werden im Datenpunkt des Objekts durch eine "states"-Liste definiert. Die Konfiguration erreicht man in der ioBroker-Adminoberfläche durch bearbeiten des Datenpunkts auf dem Segment "RAW (Nur Experten)". Beispiel: Diese states-Liste definiert drei Eingabeoptionen "select 1", "select 2" und "select 3"

  ```
  "common": {
      "type": "string",
      "states": {
        "1": "select 1",
        "2": "Select 2",
        "3": "select 3"
      },
      "custom": {
        "lovelace.0": {
          "enabled": true,
          "entity": "input_text",
          "name": "test_input_select"
        }
      }
  ```

...

3. Entitätsname: Vorgabe eines Alias-Namens, der in Lovelace verwendet wird. Der Entitätsname muss systemweit eindeutig sein. Der Entitätsname kann auch später in der Visualisierung von Lovelace über die YAML-Notation überschrieben werden und braucht daher nicht zwangsläufig bereits hier gesetzt werden.

Beispiel: Konfiguration Lovelace-Entitätstyp, Entitätsname




**Erzeugung von Entitäten über den"devices"-Adapter**

Es ist über den devices-Adapter möglich eigene Objekte/Geräte in ioBroker zu erstellen, die Eigenschaften aus mehreren Quellen besitzen. Diese selbst erstellten Devices bestehen wie andere Geräte in ioBroker aus mehreren Datenpunkten und befinden sich unter dem Namensraum alias.0.x.

Um diese Geräte in Entitäten zu überführen kann man anschließend die "Automatische Erzeugung von Entitäten" anwenden (siehe oben).

Anmerkung: An dieser Stelle hat die Anleitung noch viel "Luft" nach oben. Sofern jemand Praxisbeispiele hat, kann er diese gerne rückmelden!

## Anzeige von Entitäten

Die von ioBroker zur Verfügung gestellten Entitäten können in der Adapterkonfiguration der Lovelace-Instanz unter dem Segment "Entitäten" eingesehen werden.

Äußerst hilfreich ist der Button "Attibute anzeigen", der zu den einzelnen Entitäten entsprechend auch die Attributseigenschaften darstellt. Damit kann man erkennen, ob Entitäten richtig erkannt und angelegt wurden.

## Unterstützung von ioBroker Types durch Lovelace Entitäten

Dieser Abschnitt dient als technischer Hinweis für versierte Benutzer. Alle Anfänger sollten diesen Abschnitt initial überspringen. Die nachfolgende Tabelle versucht eine Aufstellung der folgenden Eigenschaften zeilenweise zu liefern

- ioBroker Datentyp
- Home Asistant Datentyp/Entität mit möglichen Werten
- Unterstützung der Entität durch den ioBroker Lovelace-Adapter

| Beschreibung | Type ioBroker | Suchpattern Rolle ioBroker | Entität Lovelace / HASS | Werte Lovelace / HASS | ioBroker Lovelace Unterstützung |
|---|---|---|---|---|---|
| Boolean | - | - | input_boolean | on, off | Ja |
| Schalter | - | - | switch | on, off | Ja |
| Binär Sensor | - | - | binary_sensor | on, off | Ja |
| Alarmkontrollpanel | - | - | alarm_control_panel | armed, disarmed | Ja (indirekt per Skript) |
| Medienplayer | mediaPlayer | Siehe Code | | | Ja |
| Wettervorhersage | weatherForecast | Siehe Code | | | Ja |
| RGB-Licht | rgb | Siehe Code | | | Ja |
| HUE-Licht | hue | Siehe Code | | | Ja |
| CT-Licht | ct | Siehe Code | | | Ja |
| Warnung | warning | Siehe Code | | | Nein |
| Klima | airCondition | Siehe Code | | | Nein |
| Thermostat | thermostat | Siehe Code | | | Ja |
| Rolladen | blinds | Siehe Code | | | Ja |
| Schlosss | lock | | | unlocked, locked | Ja |
| Bewegung | motion | /^state.motion$ ^sensor.motion$/ | | | Ja |
| Fenster | window | /^state(.window) ^sensor(.window)?/ | | | Ja |
| Fenster mit Kipp | windowTilt | /^state?$ ^value(.window)?$/ | | | Ja |
| Feueralarm | fireAlarm | /^state?$ ^sensor(.alarm)?.fire/ | | | Nein |
| Tür | door | /^state?$ ^state(.door)?$ ^sensor(.door)?/ | | | Ja |
| Dimmer | dimmer | | | | Ja |
| Licht | light | | | on, off | Ja |
| Laustärke | volume | | | | Nein |
| Standort 1 | location_one | | | | Nein |
| Standort | location | | | | Ja |
| Lautstärkegruppe | volumeGroup | | | | Nein |
| Schieberegler | levelSlider | | | | Nein |
| Anschluss | socket | | | | Ja |
| Button | button | | | | Ja |
| Sensor | buttonSensor | | | | Nein |
| Temperatur | temperature | | | | Ja |
| Luftfeuchte | humidity | | | | Nein |
| Bild | image | | | | Ja |
| Information | info | | | | Nein |

## Visualisierung Lovelace

Wenn der Adapter gestartet ist, kann die Visualisierung von Lovelace über das Icon "Adapter - Webseite öffnen" auf der Instanz geöffnet werden. Dafür wird die Visualisierung in einem neuen Browserfenster geöffnet (Beispieladresse: http://iobroker:8091/lovelace/default_view).

Die Visualisierung ist in den Konfigurationsmodus zu versetzen. Dies erfolgt durch folgende Schritte:

1. Klick auf "..." oben rechts.
2. Klick auf "Benutzeroberfläche konfigurieren"

Im Anschluss kann man die einzelnen "Ansichten" der Lovelace-Oberfläche konfigurieren.

Der Aufbau der Visualisierung von Lovelace ist zweistufig und besteht im wesentlichen aus:

- Ansichten (engl. Views)
- Karten (engl. Cards)

In den folgenden Abschnitten wird detailliert auf die Funktionen eingegangen.

## Ansichten

**"Ansichten" (engl. Views)** dienen der Navigation und nehmen sogenannte "Karten" (engl. Cards) der Oberfläche auf.

Beispielsweise kann es folgende Ansichten geben:

- Home
- Ansichten nach Funktionen (wie Lichter, Türen/Fenster, Wetter Alarmanlage, Staubsaugerroboter, etc.)
- Ansichten aufgeteilt nach Räume (Wohnzimmer, Küche, Büro, Keller etc.)

Anlage / Änderung / Verschieben von Ansichten
Standardmäßig gibt es nur die Ansicht "Home". Eine bestehende Ansicht kann über das "Stift-Symbol" editiert werden. Über das "+"-Symbol in der oberen Leiste können weitere Ansichten hinzugefügt werden. Über die Symbole "<-" (Pfeil nach links) oder "->" (Pfeil nach rechts) können die Ansichten nachträglich verschoben werden.

Eine Ansicht hat die folgenden Eigenschaften:

- Titel (optional):
- Symbol (optional): Das Icon ist in der Notation "mdi:" vorzugeben.

  Es können alle "Material Design Icons" verwendet werden.
  Ein Icon kann über die Webseite http://materialdesignicons.com/ gesucht/ermittelt werden.

  *Beispiel*: Das Icon lightbulb-on ist das Symbol für ein eingeschaltetes Licht. Es ist auf der Materialdesignicons-Webseite unter folgendem Link http://materialdesignicons.com/icon/lightbulb-on zu finden. Um Das Icon in Lovelace zu verwenden ist es wie folgt unter der Eigenschaft "Symbol (optional)" vorzugeben: "mdi:lightbulb-on"
- Url (optional): ...

## Karten

**"Karten" (engl. Cards)** sind Elemente die in einer Ansicht verwendet werden. Eine Karte kann über das runde "+"-Symbol unten rechts auf der aktuell ausgewählten Ansicht hinzugefügt werden. Anschließend öffnet sich ein Dialog der dem Anwender die Auswahl über die folgenden Karten ermöglicht:

- Elemente
- Bedingte Elemente

- Alarmpanel
- Entity Button
- Glance
- IFRAME
- Markdown
- Picture Elements
- Pflanzen Status
- Thermostat
- Entity Filter
- History Graph
- Licht
- Mediensteuerung
- Picture Entity
- Sensor
- Vertikaler Stapel
- Manual Card
- Gauge
- Horizontaler Stapel
- Karte
- Picture
- Picture Glance
- Einkaufsliste
- Wettervorhersage

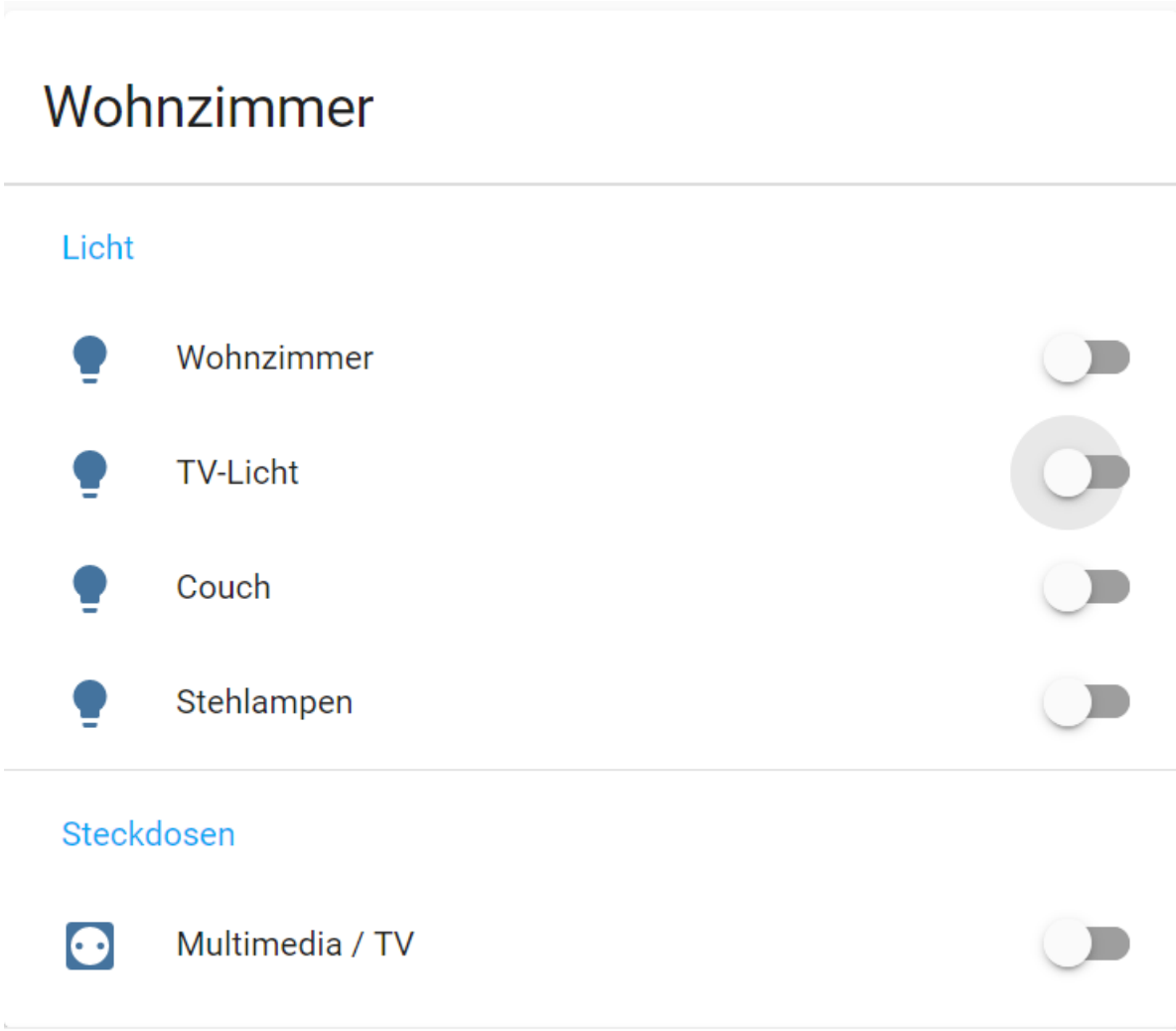
**Entity / Entitäten / Elemente**

Die Entity (oder im deutschen: Elemente) Card ermöglicht es Elemente darzustellen.

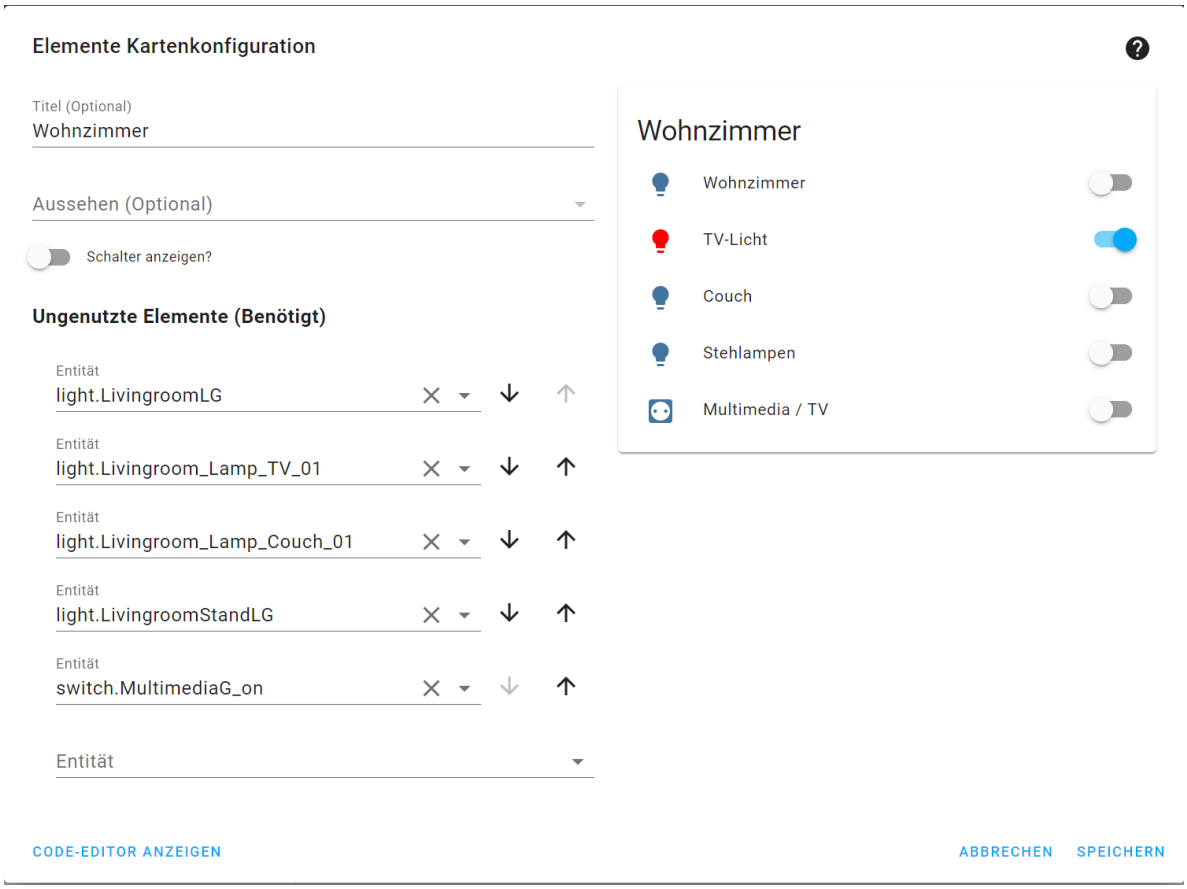
Die Kartenkonfiguration bietet mehrere Optionen:

- Titel (Optional): Vorgabe eines Titels für die Karte
- Aussehen (Optional):
- Schalter anzeigen?: Anzeige eines globalen Schalters (für die ganze Karte zur Steuerung aller einzelnen Schalter)
- Ungenutzte Elemente (benötigt): Auflistung der Elemente auf der Karte

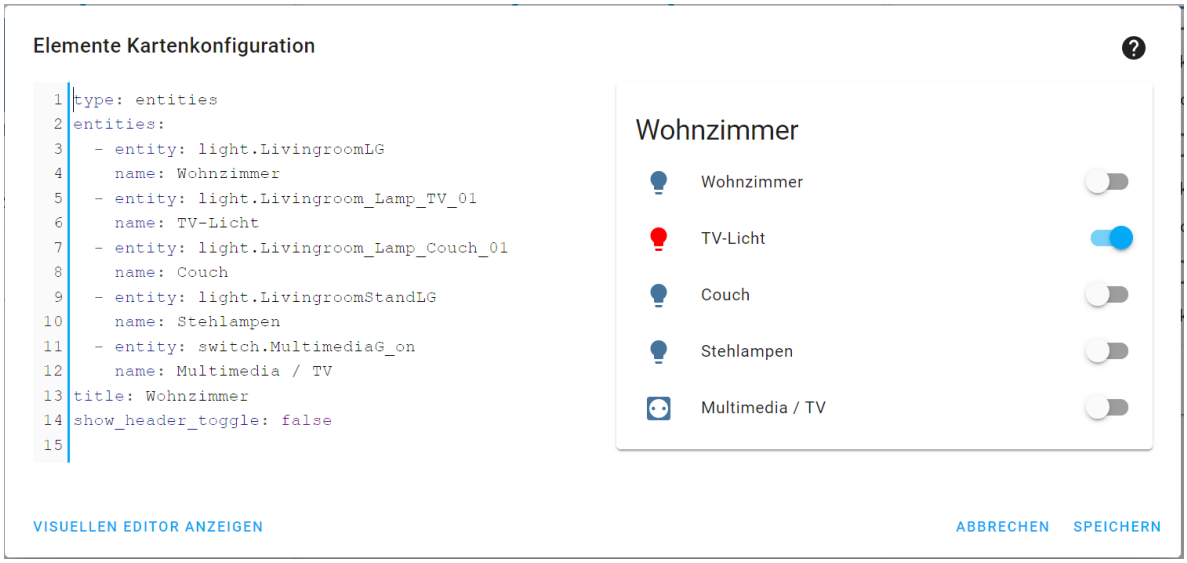
Beispiel: Wohnzimmer-Karte für Lichtsteuerung und Steckdosen-Steuerung




Konfiguration der "Wohnzimmer"-Karte für die Steuerung von Licht und Steckdosen:



Über die Option "CODE-EDITOR ANZEIGEN" kann man den korrespondierenden YAML-Code bearbeiten:




In dieser Karte wurde über das "name"-Attribut eine sprechende Bezeichnung für den Benutzer vorgegeben.

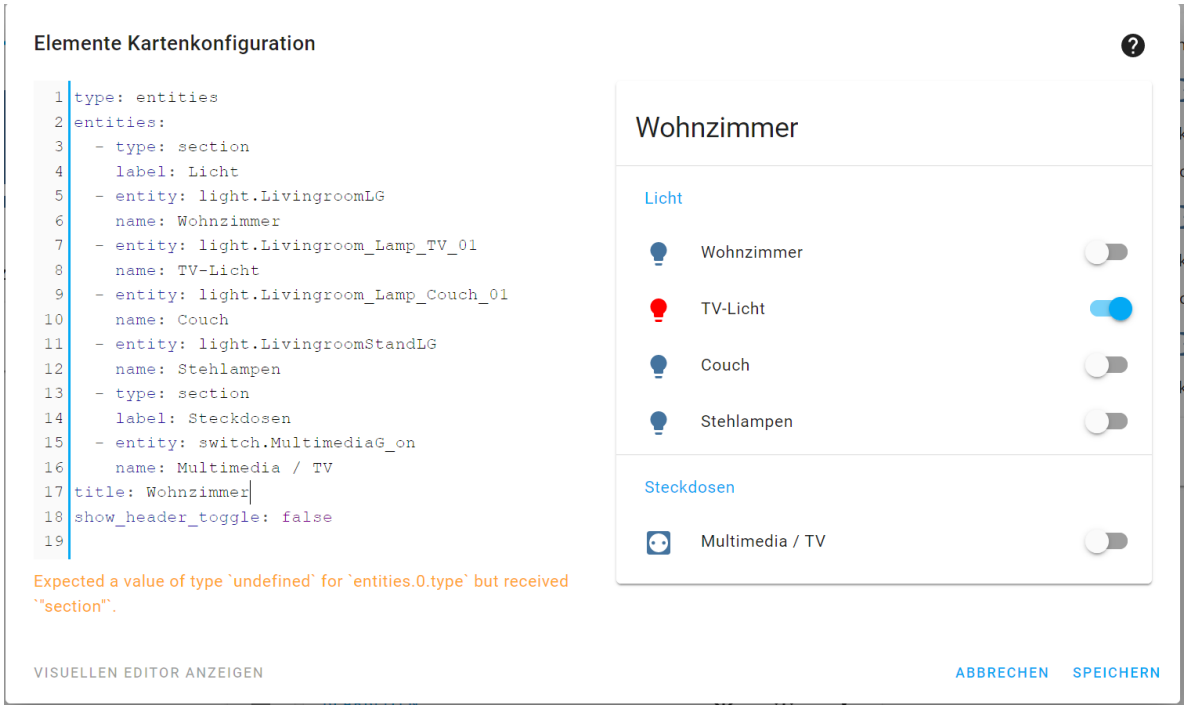
```
type: entities
entities:
  - entity: light.LivingroomLG
    name: Wohnzimmer
  - entity: light.Livingroom_Lamp_TV_01
    name: TV-Licht
  - entity: light.Livingroom_Lamp_Couch_01
    name: Couch
  - entity: light.LivingroomStandLG
    name: Stehlampen
  - entity: switch.MultimediaG_on
    name: Multimedia / TV
title: Wohnzimmer
show_header_toggle: false
```

Weitergehend kann man die Lichter und die Steckdose auch in eigenen Bereichen mit Überschriften gruppieren.

Hierfür ist der YAML-Code um die Attribute section und label zu erweitern:

```
...
  - type: section
    label: Licht
...
```

Beispiel: Die Lichter haben nun eine Überschrift "Licht". Die Steckdosen eine Überschrift "Steckdosen"



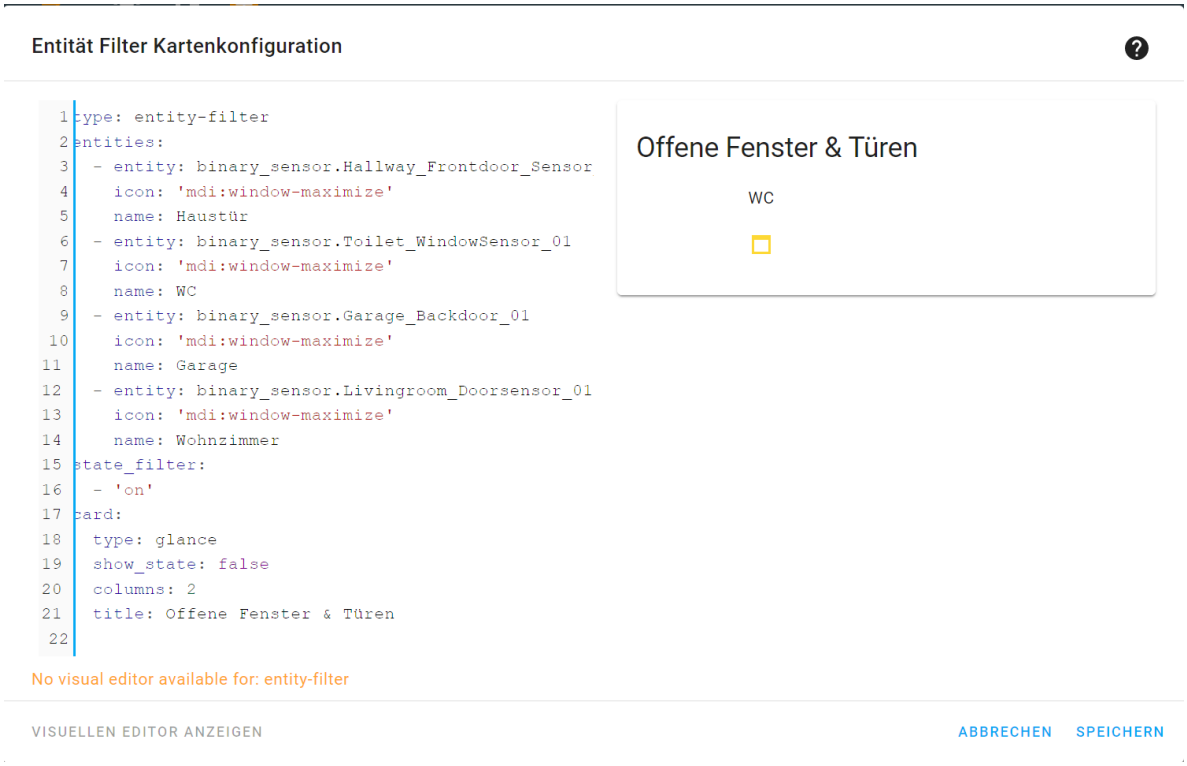

Vollständiger YAML-Code für dieses Beispiel:

```
type: entities
entities:
  - type: section
    label: Licht
  - entity: light.LivingroomLG
    name: Wohnzimmer
  - entity: light.Livingroom_Lamp_TV_01
    name: TV-Licht
  - entity: light.Livingroom_Lamp_Couch_01
    name: Couch
  - entity: light.LivingroomStandLG
    name: Stehlampen
  - type: section
    label: Steckdosen
  - entity: switch.MultimediaG_on
    name: Multimedia / TV
title: Wohnzimmer
show_header_toggle: false
```

### Entity Filter

Die Entity Filter-Karte ermöglicht es analog der Karte Elemente "Entities" anzuzeigen. Jedoch können diese zusätzlich mit Bedingungen versehen werden unter welchen die Entities angezeigt werden sollen.

Beispiel: Anzeige aller geöffneten Fenster & Türen in einer "Glance"-Card mit zweispaltiger Ansicht. Durch die card-Eigenschaft `show_state: false` wird der Wert des offenen Fensters nicht dargestellt.




Beispiel-YAML-Code

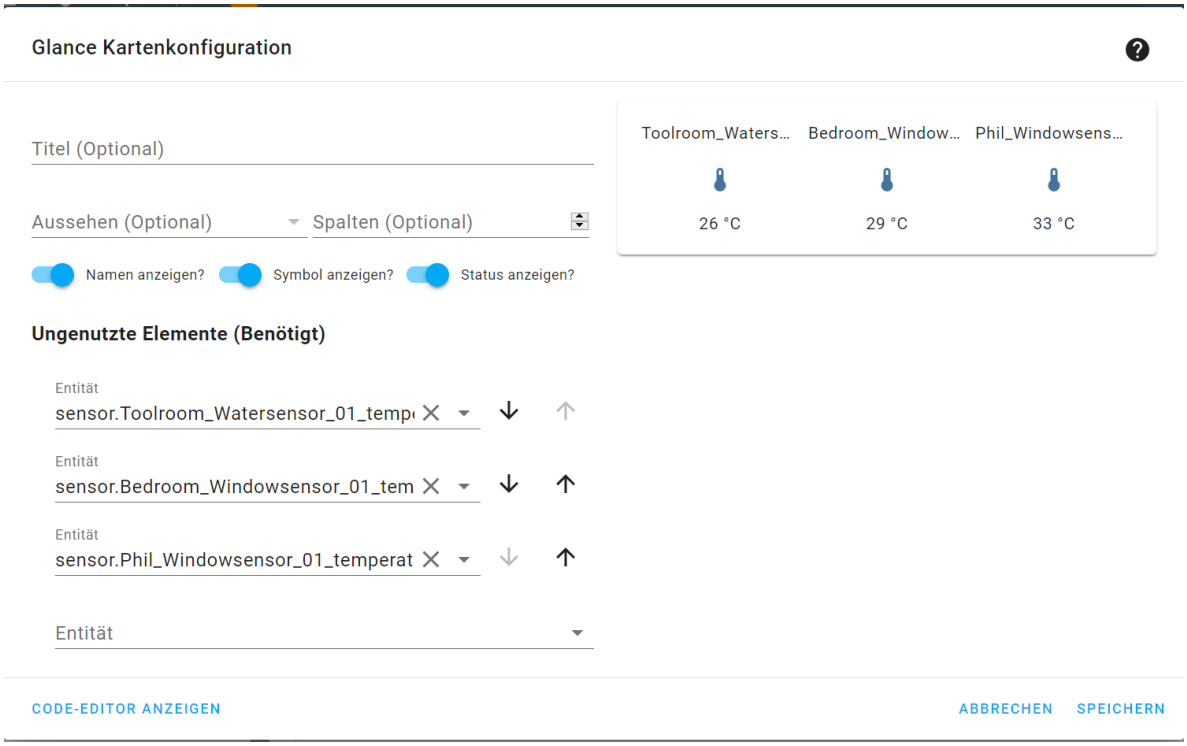
```yaml
type: entity-filter
entities:
  - entity: binary_sensor.Hallway_Frontdoor_Sensor_01
    icon: 'mdi:window-maximize'
    name: Haustür
  - entity: binary_sensor.Toilet_WindowSensor_01
    icon: 'mdi:window-maximize'
    name: WC
  - entity: binary_sensor.Garage_Backdoor_01
    icon: 'mdi:window-maximize'
    name: Garage
  - entity: binary_sensor.Livingroom_Doorsensor_01
    icon: 'mdi:window-maximize'
    name: Wohnzimmer
  - entity: binary_sensor.Bath_Window_Sensor_01
    icon: 'mdi:window-maximize'
    name: Bad
  - entity: binary_sensor.Bedroom_Windowsensor_01
    icon: 'mdi:window-maximize'
    name: Schlafzimmer
  - entity: binary_sensor.Phil_Windowsensor_01
    icon: 'mdi:window-maximize'
    name: Philipp
  - entity: binary_sensor.Eric_Windowsensor_01
    icon: 'mdi:window-maximize'
    name: Erik
  - entity: binary_sensor.Roof_Windowsensor_01
    icon: 'mdi:window-maximize'
    name: Büro
state_filter:
  - 'on'
card:
  type: glance
  show_state: false
```

```
  columns: 2
  title: Offene Fenster & Türen
```

### Glance

Die Glance-Card ermöglicht es mehrere Entities nebeneinander ein einer Karte darzustellen.

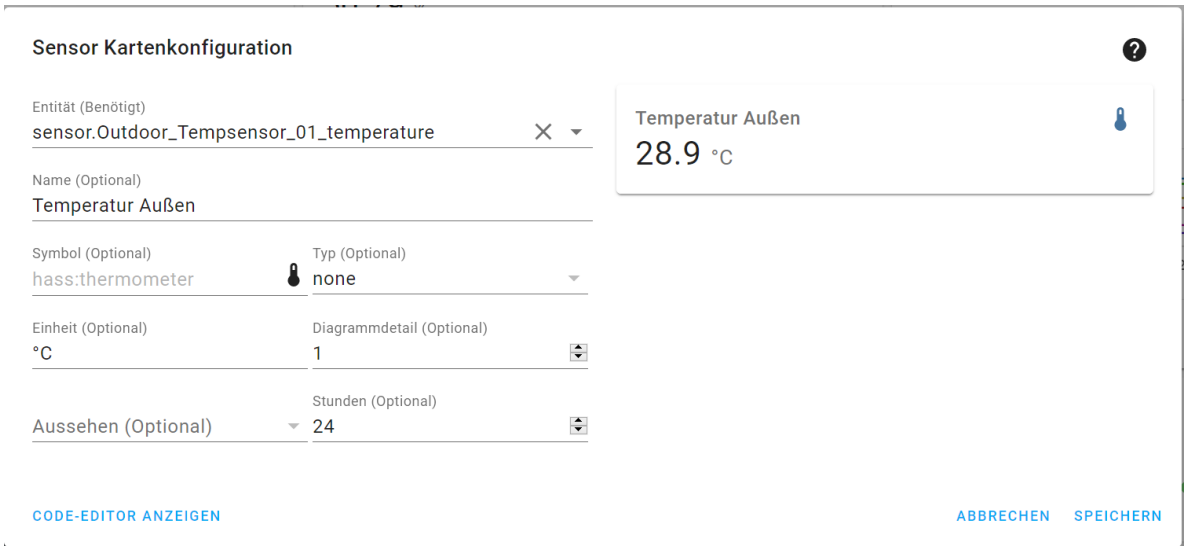
Beispiel:



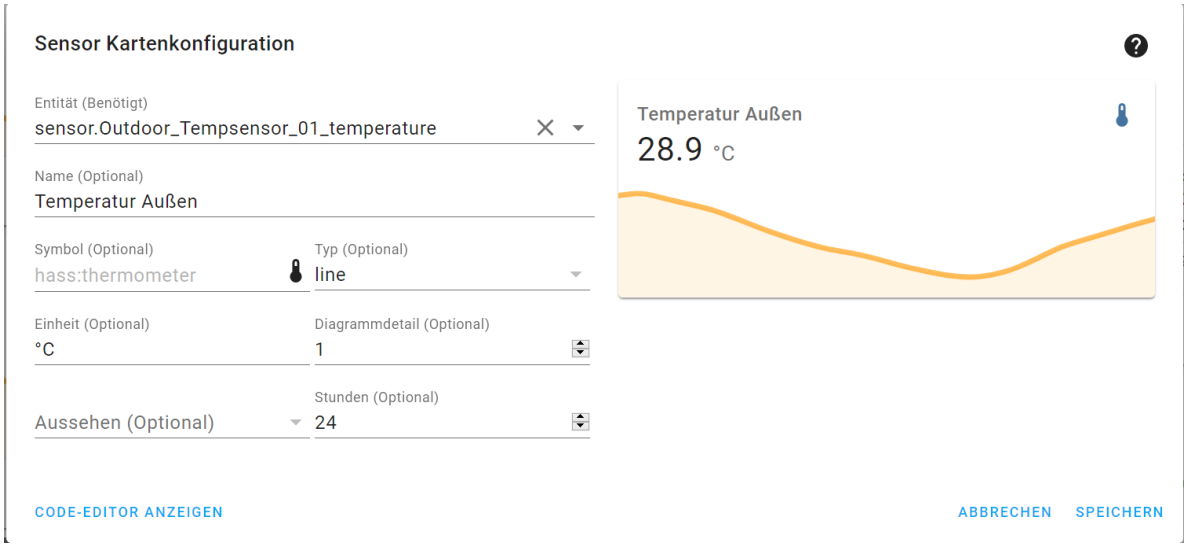

### Sensor

Die Sensor-Card ermöglicht es Sensorwerte darzustellen.

Beispiel: Darstellung eines Temperatursensors



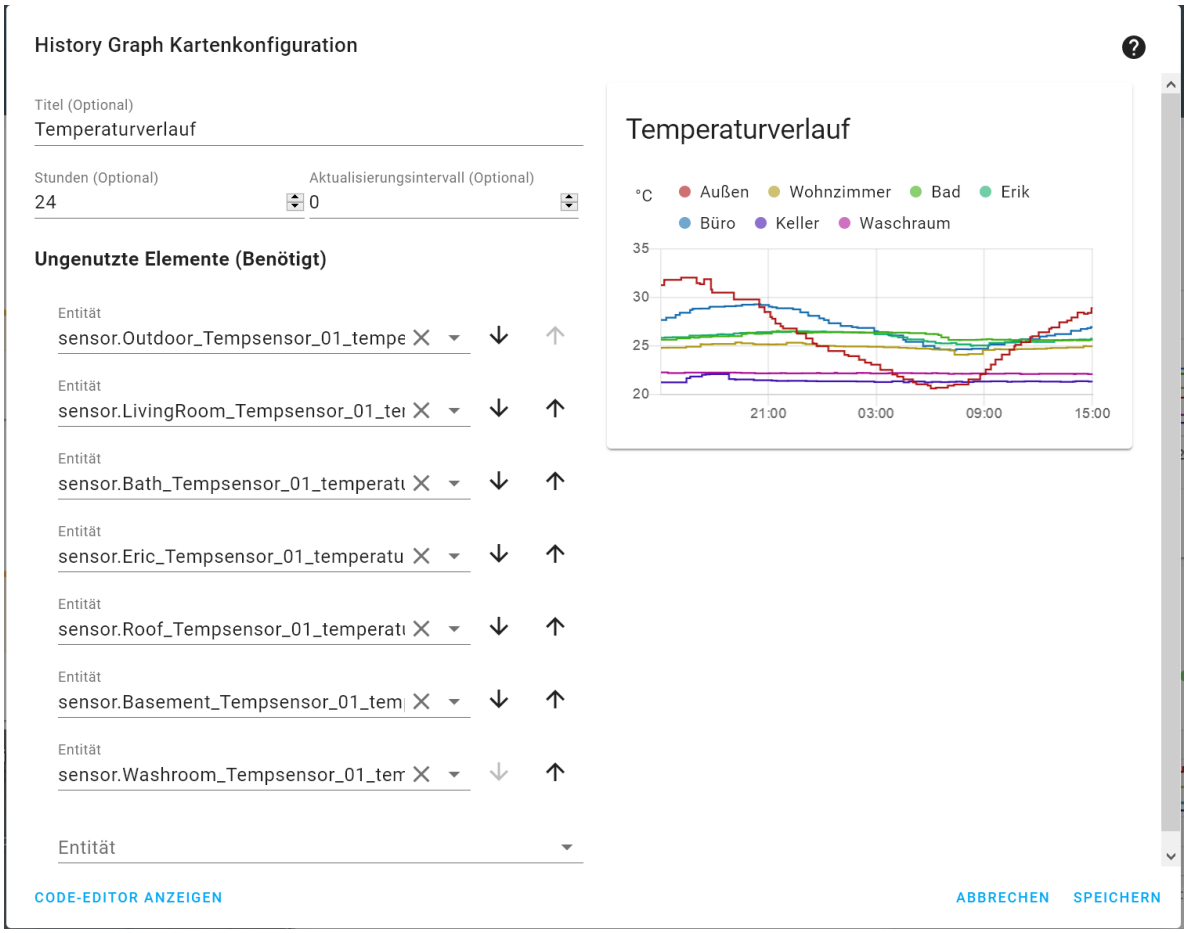
Sofern der Datenpunkt historisiert wird und im Lovelace-Adapter die Historisierung konfiguriert wurde, kann man auch auf die Historie in dieser Card zurückgreifen. Um den Diagrammverlauf des Sensors darzustellen kann der "Typ" auf "line" gesetzt werden. Entsprechend erhält man die folgende Darstellung:



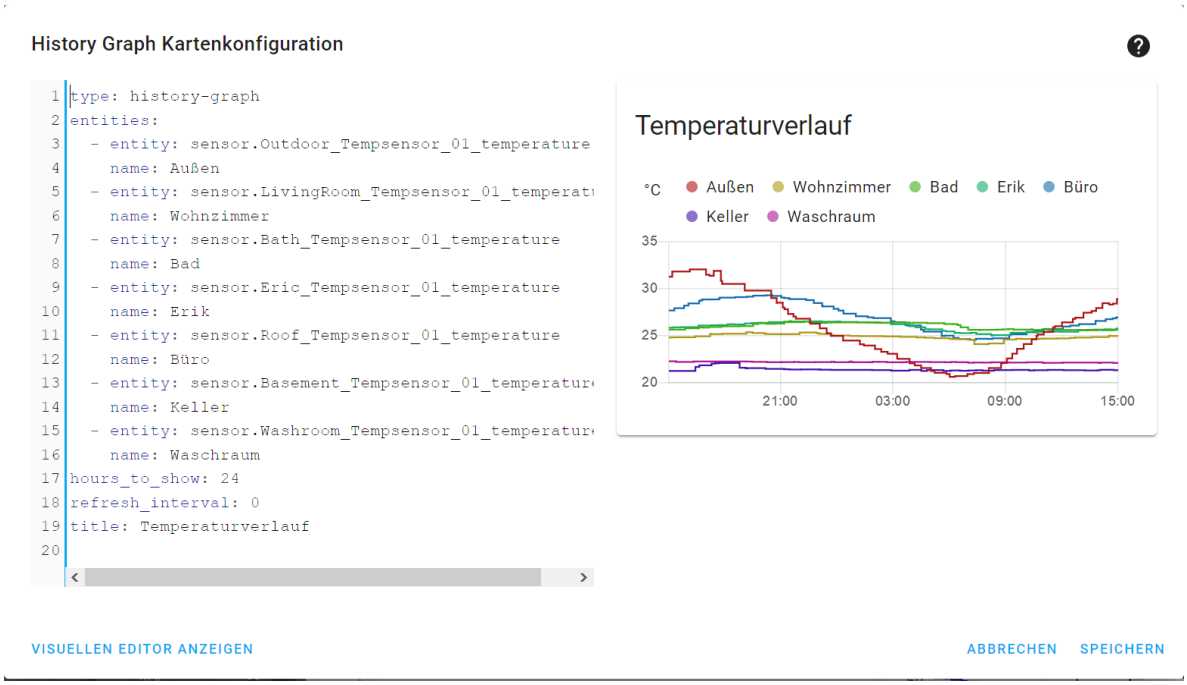

## History Graph

Über die History Graph Card können mehrere Sensoren in einem xy-Diagramm dargestellt werden. Voraussetzung sind auch hier das die Datenpunkte historisiert sind und im Lovelace-Adapter die Historisierung konfiguriert wurde, kann man auch auf die Historie in dieser Card zurückgreifen.

Beispiel: Darstellung mehrerer Temperatursensoren



Im "CODE-EDITOR ANZEIGEN"-Modus kann auch hier per YAML eine sprechende Bezeichnung für die Entitäten vorgegeben werden:

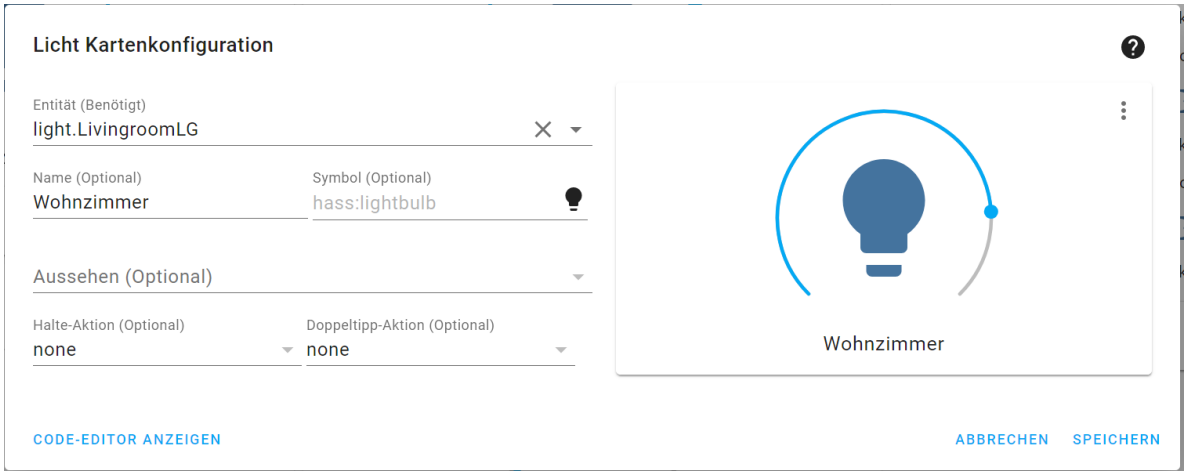



```
type: history-graph
entities:
  - entity: sensor.Outdoor_Tempsensor_01_temperature
    name: Außen
  - entity: sensor.LivingRoom_Tempsensor_01_temperature
    name: Wohnzimmer
  - entity: sensor.Bath_Tempsensor_01_temperature
    name: Bad
  - entity: sensor.Eric_Tempsensor_01_temperature
    name: Erik
  - entity: sensor.Roof_Tempsensor_01_temperature
    name: Büro
  - entity: sensor.Basement_Tempsensor_01_temperature
    name: Keller
  - entity: sensor.Washroom_Tempsensor_01_temperature
    name: Waschraum
hours_to_show: 24
refresh_interval: 0
title: Temperaturverlauf
```

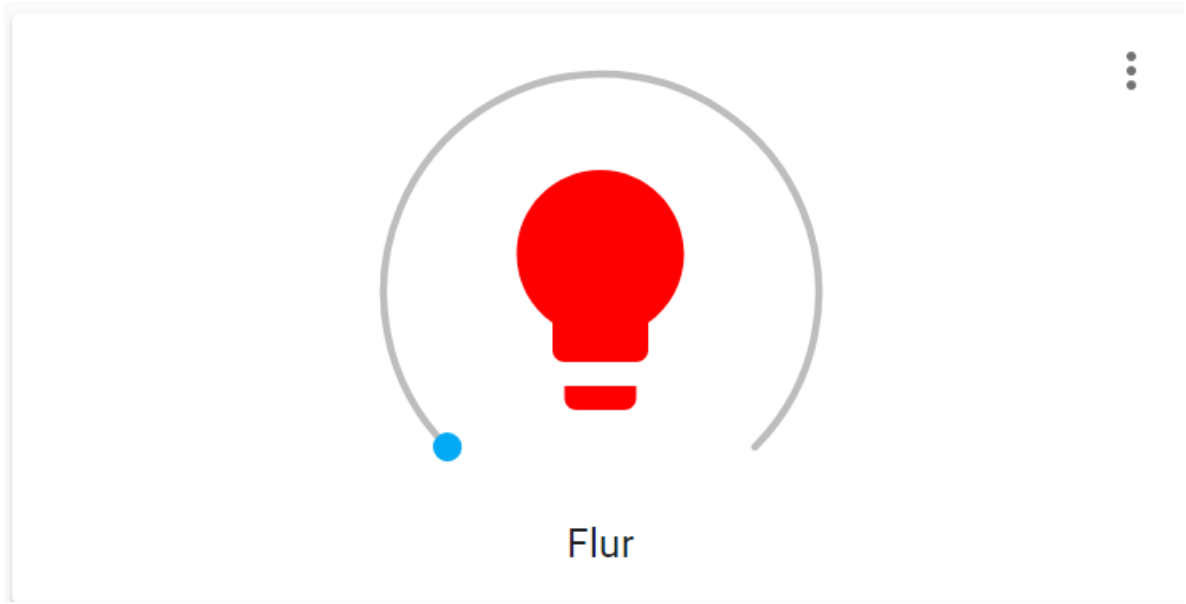
**Lichter / Light**

Die Light Card ermöglicht es eine Dimmfunktion inkl. An/Aus-Schalter für Lichter darzustellen:

Beispiel:




Beispiel: Sofern es sich um eine RGB-Lampe handelt wird zusätzlich auch die Lichtfarbe visualisiert.




Beispiel: Über die "..."-Option der Karte kann man eine erweiterte Steuerung des Lichts vornehmen. Die Anzeige ist variabel und abhängig davon, ob das Licht eine Steuerung der Helligkeit, Farbtemperatur und Lichtfarbe ermöglicht oder nicht.

# Hallway_Lamp_01

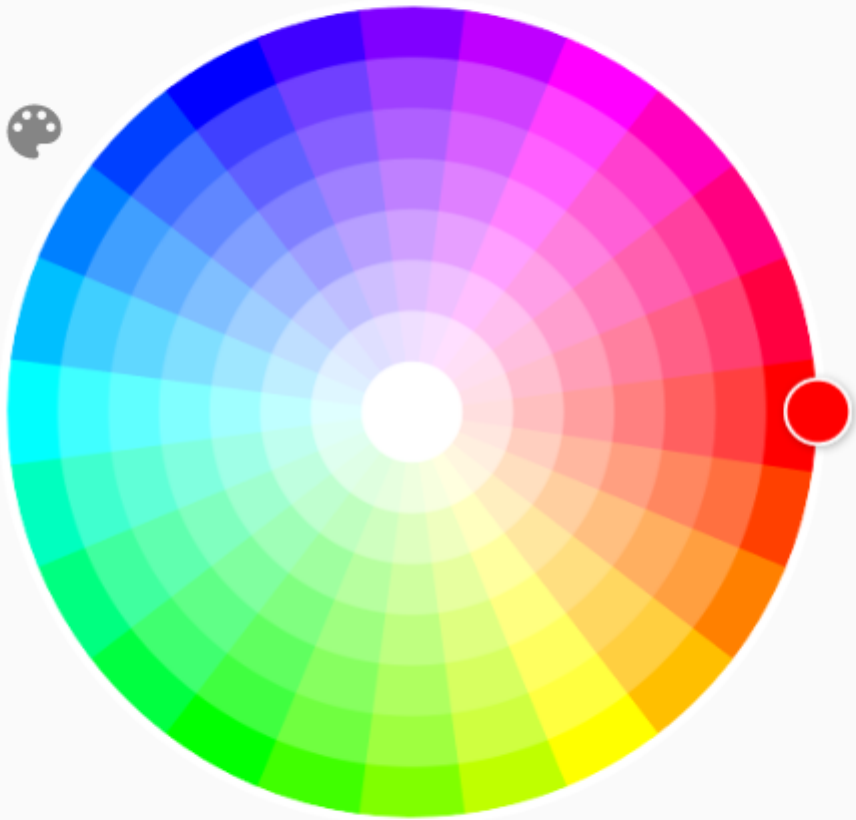
Hallway_Lamp_01 An

Vor 2 Stunden

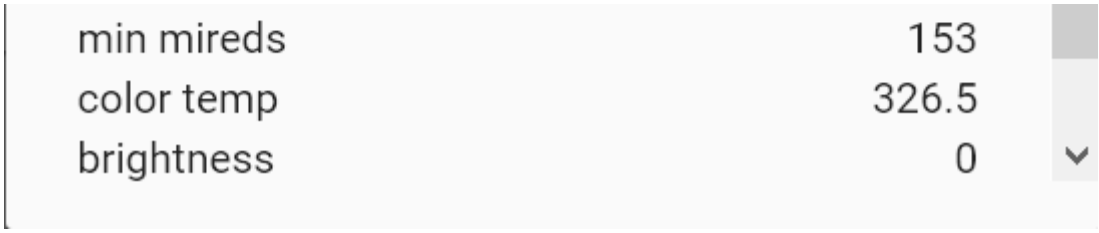
Kein Zustandsverlauf gefunden.

Helligkeit

Farbtemperatur



max mireds 500




**IFRAME / Webpage Card**

Über die IFRAME bzw. Webpage Card ist es möglich eine autonome URL als Karte in die Visualisierung aufzunehmen. Standardmäßig blendet Lovelace hier die URL für den Wetterdienst windy.com vor.

Interessante IFRAME-Anwendungen:

- Wettervorhersage windy.com
- Wettervorhersage meteoblue
- Einbindung Grafana-Charts
- ...

Optionen für die Konfigurationen des IFRAME:

- URL (benötigt)
- Titel (optional)
- Seitenverhältnis (Optional)

**Wettervorhersage**

Über die Wettervorhersagen-Karte kann nativ eine Wetterkarte eingebunden werden.

Beispiel:


Eine Wettervorhersage klappt mit den ioBroker-Adaptern yr und daswetter.

Für die Konfiguration ist die Funktion "Wetter" (enum.functions.weather) und der Raum "Any" (enum.rooms.any) anzulegen.

Konfiguration daswetter-Adapter:

Der komplette Objektbaum muss die `Funktion=Weather` und `Raum=Any` zugeordnet bekommen:

- daswetter.0.NextDays.Location_1

Konfiguration yr-Adapter:

Der komplette Objektbaum muss die `Funktion=Weather` und `Raum=Any` zugeordnet bekommen:

- yr.0.forecast

Hinweis: Die Wetter-Karte funktioniert nur in der letzten DEVELOPMENT-Version aktuell richtig. Zum Teil kommt es dazu, dass keine Icons angezeigt (oder nur der Pfad zu den Icons angezeigt) werden oder das das Datum nicht richtig dargestellt (Meldung "invalid date") wird.

Konfiguration AccuWeather-Adapter:

Hinweis: Der AccuWeather-Adapter funktioniert nicht richtig. Er wird vom "TypeDetector" nicht richtig erkannt. Dies setzt eine Modifikation des TypeDetectors voraus (siehe GitHub).
"Mutige" können den TypeDetector wie folgt tauschen:

1. accuweather.0.Summary Raum "Any" und Function "Wetter" zuordnen.
2. ioBroker.type-detector für AccuWeather : /opt/iobroker/node_modules/iobroker.lovelace/node_modules/iobroker.type-detector/index.js ersetzen durch : https://github.com/algar42/ioBroker.type-detector/blob/master/index.js
3. Lovelace neustarten.
4. Für AccuWeather ist eine eigene Card zu installieren.
5. Die Card ist wie folgt über YAML einzubinden:

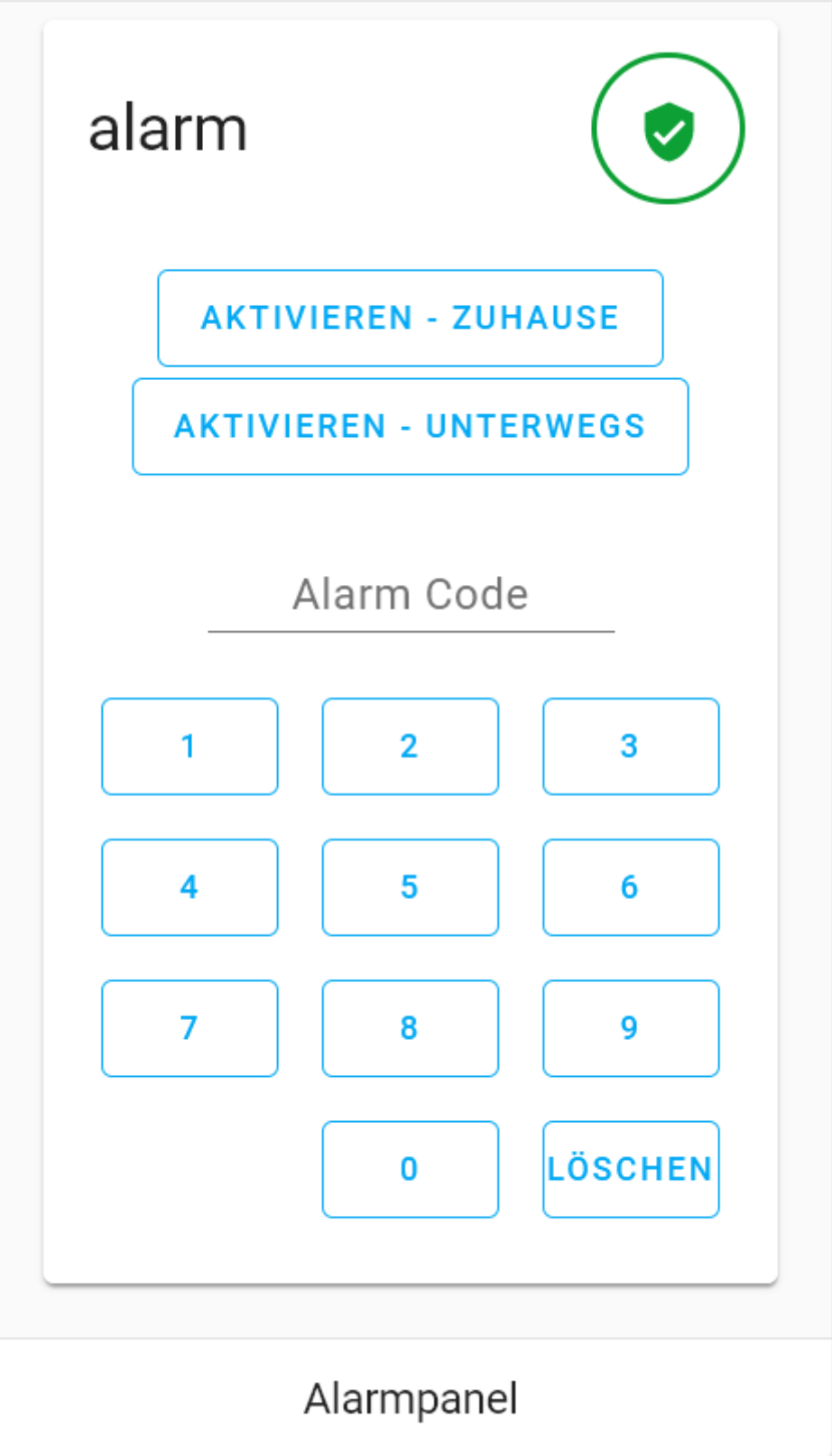
```
type: 'custom:weather-card'
entity: weather.Weather_Summary

name: Berlin
windunit: km/h
pressureunit: mmHg
tempunit: C
waterunit: mm
convertspeedtoms: km/h

convertpressuretomm: false

convertemptof: false
```

Hinweis: Die accueweather-Karte ist mit der aktuellsten Lovelace-Adapterversion nicht mehr richtig "hübsch" formatiert.

**Alarmpanel**

Das Alarmpanel wird nicht als Entität-Standardmäßig unterstützt, kann aber durch ein ioBroker-Skript emuliert werden.

Details auf der ioBroker.lovelace GitHub-Seite.



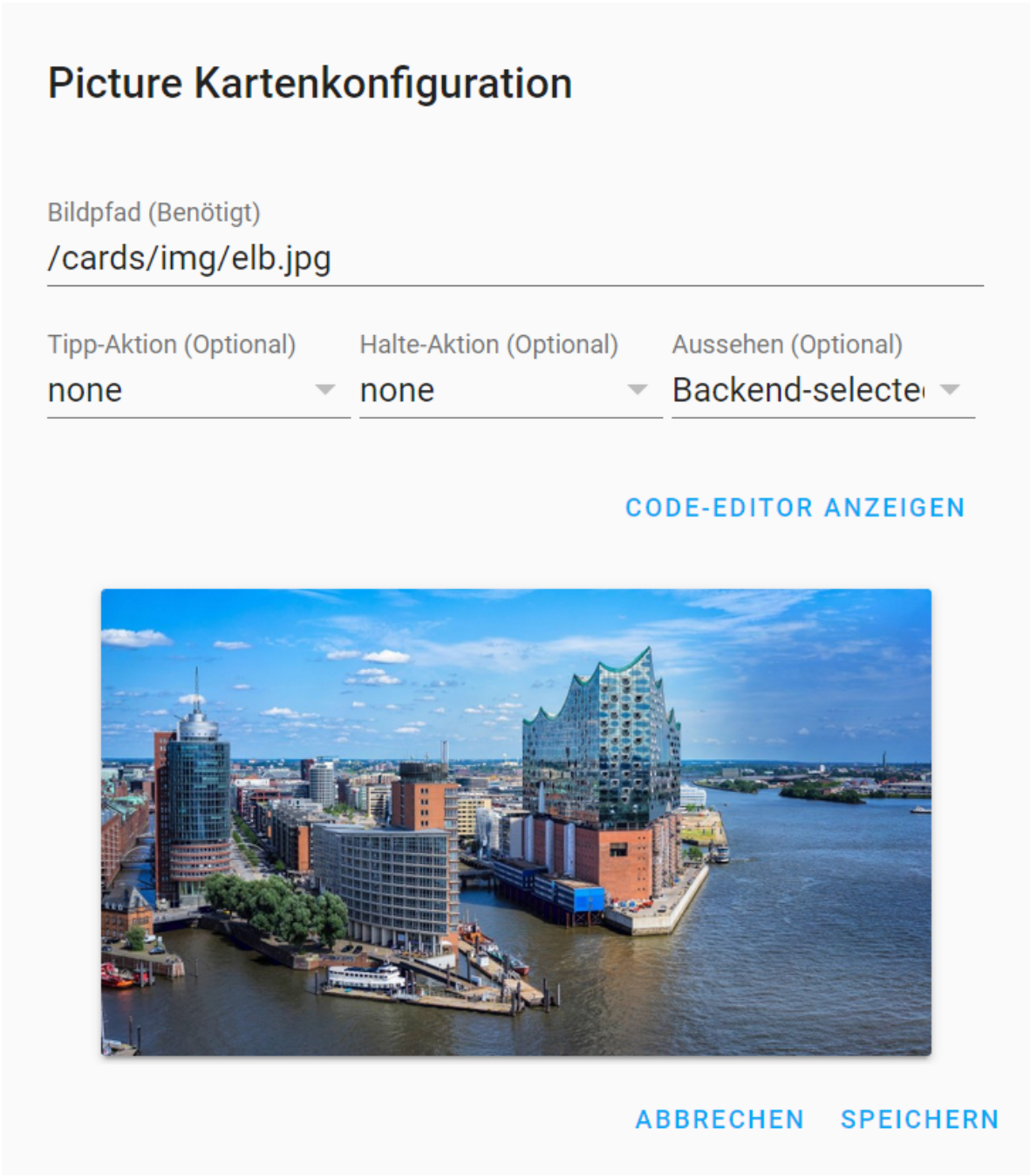

Alarmpanel

**Picture / Bild**

Die Picture Card ermöglicht es ein Bild darzustellen.

Statische Bilder können im ioBroker-Verzeichnis hinterlegt werden an der gleichen Stelle, wo die Dateien für die Custom-Cards liegen. Bilder können, wie die Custom Cards auch, per Drag & Drop auf der  Konfigurationsseite der Lovelace-Adapterkonfiguration hochgeladen werden.

Um das Bild in ioBroker zu verwenden, ist der komplette Pfad inkl. card-Verzeichnis anzugeben zum Bild:




Beispiel in YAML-Code:

```
type: picture
tap_action:
  action: none
hold_action:
  action: none
image: /cards/elb.jpg
```

Die Bilder werden dann intern in das Verzeichnis "/opt/iobroker/iobroker-data/files/lovelace.0/cards" gespeichert.

Beispiel 2 : Anzeige einer Webcam / Cam. Voraussetzung Anlage einer Entity "Camera"

```
type: picture-entity
entity: camera.Einfahrt
aspect_ratio: 50%
show_state: false
show_name: false
hold_action:
action: more-info
tap_action:
   action: navigate
   navigation_path: kamera
```

Dieser Abschnitt benötigt weitere Beispiele!

### Alternative Ablage Bilder über das Dateisystem

Alternativ können Dateien auch auf dem Dateisystem im "cards"-Ornder abgelegt werden. Um eine Datei dort abzulegen, sollte Sie mit dem ioBroker write-Befehl gespeichert werden:

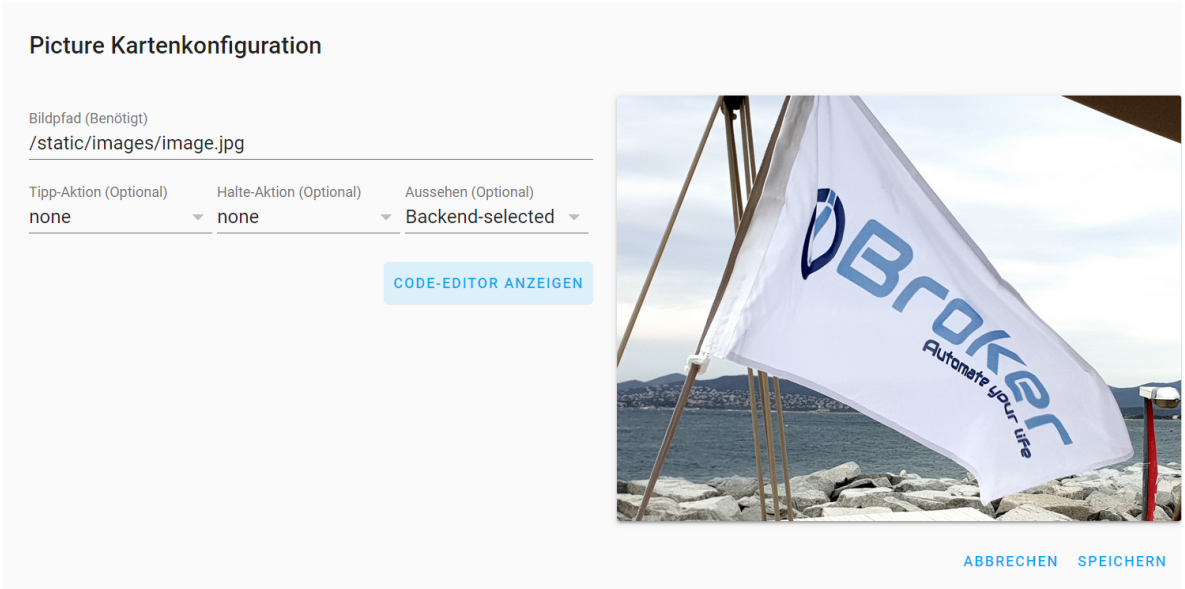
```
iobroker file write </verzeichnisVomBild/>bild.jpg
/lovelace.0/cards/img/bild.jpg
```

Wichtig: Bei den Verzeichnissen / Dateinamen keine Sonderzeichen und nur Kleinschreibung verwenden!

**Alternativer Speicherort:** Bilder können auch in einem separaten Verzeichnis (also nicht im cards-Verzeichnis) abgelegt werden. Diese Vorgehensweise ist nicht empfohlen, weil der Lovelace-Adapter die Dateien dort zurücksetzen kann bei Aktualisierung der Lovelace-Version.

```
/opt/iobroker/node_modules/iobroker.lovelace/hass_frontend/static/images
```

In Lovelace kann über das Picture-Element das Bild wie folgt referenziert werden:




---

Hinweis: sofern dort Unterordner angelegt werden, sollten diese alle für den ioBroker-Benutzer Berechtigung haben:

```
chown iobroker:iobroker <MeinBildVerzeichnis>
```

## Eigene Karten / Custom Cards

Prinzipiell ist es möglich auf einen großen Umfang von frei erstellten / eigenen Karten (engl. "Custom Card") von Home Assistant / Lovelace zurückzugreifen. Eine "Custom Card" wird zum Teil als Quelldateien in den jeweiligen GitHub-Projekten zur Verfügung gestellt und muss entsprechend in eine .js-Datei kompiliert werden. Die .js-Dateien können anschließend in den ioBroker-Adapter installiert werden.

Damit die Karten funktionieren müssen Sie zur Lovelace-Adapterversion vom Versionsstand passen. Sofern eine Karte in einer neuen Lovelace-Adapterversion nicht mehr funktioniert, sollte als erstes geprüft werden, ob es eine neue Version der Karte gibt und die Karte entsprechend aktualisiert werden.

Hinweis: Es kann Karten geben, die auf Entitäten oder Funktionen zurückgreifen (wie eine Karte für Pflanzen oder Staubsauger), die im aktuellen Lovelace-Adapter nicht implementiert sind. Siehe hierzu auch den Abschnitt Entitäten.

Eine Liste der erfolgreich getesteten Karten findet sich weiter unten.

**Installation Custom Cards über die ioBroker-Adminoberfläche**

Diese frei erstellten / eigenen Karten (engl. Custom Cards von Home Assistant / Lovelace) können bequem über die Admin-Oberfläche von ioBroker installiert werden. Dafür ist die Adapterkonfiguration des Lovelace-Adapters zu öffnen. Auf dem Segment "Kundenspezifische Karten" kann die .JS-Datei per Drag&Drop eingespielt werden. Anschließend ist der Lovelace-Adapter neuzustarten.

Hinweis: Sollte es im Firefox-Browser nicht klappen, einfach mal mit Chrome probieren ;)

**Alternative Installation Custom Cards über die Shell**

Alternativ können eigene Karten auch über die Shell über den folgenden Befehl eingespielt werden:

```
iobroker file write PATH_TO_FILE\bignumber-card.js /lovelace.0/cards/
```

Anschließend ist der Lovelace-Adapter neuzustarten.

**Erfolgreich getestete Karten**

Eine Liste der Karten, die erfolgreich durch Benutzer in Foren gemeldet wurden:

- bignumber-card
- simple-thermostat
- thermostat (die Dateien .js und .lib.js werden benötigt)
- mini-graph-card
- mini-media-player
- button entity row
- flex table
- auto entities
- card mod

- slider entity row
- slideshow

IOBroker-spezifische Karten:

Tankerkoenig Lovelace Card for ioBroker

[Vorlage] Lovelace: ical Kalender Karte

## YAML-Basics

Dieser Abschnitt ist im Aufbau (TODO).

## Call-Service

Die Services selbst sind hier grob erklärt:
https://www.home-assistant.io/docs/scripts/service-calls/

Für detailierte Dokumentation wird auf die Entwicklertools verwiesen:
https://www.home-assistant.io/docs/tools/dev-tools/

Die Entwicklertools sind allerdings nur auf einer lokalen HASS Installation einsehbar. Deshalb habe ich mir kurzerhand eine VM mit Hyper-V aufgesetzt: https://www.home-assistant.io/hassio/installation/

Anmerkung falls das jemand anderes nachmacht: Die VM wie in der Anleitung starten und anschließend auf dem Host (Windows) einfach diese Adresse im Browser eingeben:http://homeassistant.local:8123/

Nun haben wir eine lokale HASS Installation und können unter Entwickleroptionen die detaillierten Service Beschreibungen einsehen.

Wie sich bei der nachfolgenden Dokumentation gezeigt hat, sind viele Services sehr HomeAssistant spezifisch. Nachfolgend die wichtigsten Services, mit welchen wahrscheinlich 95% aller Anforderungen erfüllt werden können. **Ich habe nicht alle Services im IOBroker getestet. Bitte um Info sollte irgendwas nicht funktionieren, dann werde ich es vermerken.**

#### HOMEASSISTANT SERVICES

https://www.home-assistant.io/docs/scripts/service-calls/

Es gibt folgende allgemeingültige Services, welche auf beliebige Entitys angewendet werden können (sofern von diesen unterstützt). Beispiele sind Switch, Light, etc.

| Service | Beschreibung | Service Data |
|---|---|---|
| homeassistant.turn_on | Entity einschalten | entity_id |
| homeassistant.turn_off | Entity ausschalten | entity_id |
| homeassistant.toggle | Entity umschalten | entity_id |
| homeassistant.update | Entity(s) aktualisieren | entity_id(s) |

Statt **homeassistant** kann auch das jeweilige Entity verwendet werden (wie z.B. **switch.turn_off**)

Beispiel: Zentraler Ausschaltknopf

```
type: button
name: Alles aus
tap_action:
  action: call-service
  service: homeassistant.turn_off
  service_data:
    entity_id: switch.Zentral_Reset_Wohnen
entity: switch.Zentral_Reset_Wohnen
```

#### INPUT_SELECT SERVICES

https://www.home-assistant.io/integrations/input_select/

Folgende Services werden von INPUT_SELECT zur Verfügung gestellt:

| Service | Beschreibung | Service Data |
|---|---|---|
| input_select.select_next | Nächste Option | entity_id |
| input_select.select_previous | Vorherige Option | entity_id |
| input_select.set_options | Optionen definieren | entity_id; options |
| input_select.select_option | Option auswählen | entity_id; option |
| input_select.reload | Entitys neu laden | |

Beispiel: Fernsehsender auswählen

```
type: button
name: ARD anschauen
entity_id: input_select.TV_Sender
tap_action:
  action: call-service
  service: input_select.select_option
  service_data:
    entity_id: input_select.TV_Sender
    option: ARD
```

#### INPUT_NUMBER SERVICES

https://www.home-assistant.io/integrations/input_number/

Folgende Services werden von INPUT_NUMBER zur Verfügung gestellt:

| Service | Beschreibung | Service Data |
|---|---|---|
| input_number.decrement | Wert verringern | entity_id |
| input_number.increment | Wert erhöhen | entity_id |
| input_number.set_value | Wert setzen | entity_id; value |
| input_number.reload | Entitys neu laden | |

Beispiel: Rollladen beschatten

```
type: button
name: Rollladen beschatten
entity_id: input_number.Rollladen_Position
tap_action:
  action: call-service
  service: input_number.set_value
  service_data:
    entity_id: input_number.Rollladen_Position
    value: 80
```

#### INPUT_TEXT SERVICES

https://www.home-assistant.io/integrations/input_text/

Folgende Services werden von INPUT_TEXT zur Verfügung gestellt:

| Service | Beschreibung | Service Data |
|---|---|---|
| input_text.set_value | Wert setzen | entity_id; value |
| input_text.reload | Entitys neu laden | |

Beispiel:  Passwort zurücksetzen

```
type: button
name: Passwort zurücksetzen
entity_id: input_text.Passwort
tap_action:
  action: call-service
  service: input_text.set_value
  service_data:
    entity_id: input_text.Passwort
    value: 1234567890
```

## Themes

Lovelace unterstützt sehr variabel die Anpassung von Farben, Hintergründen etc.

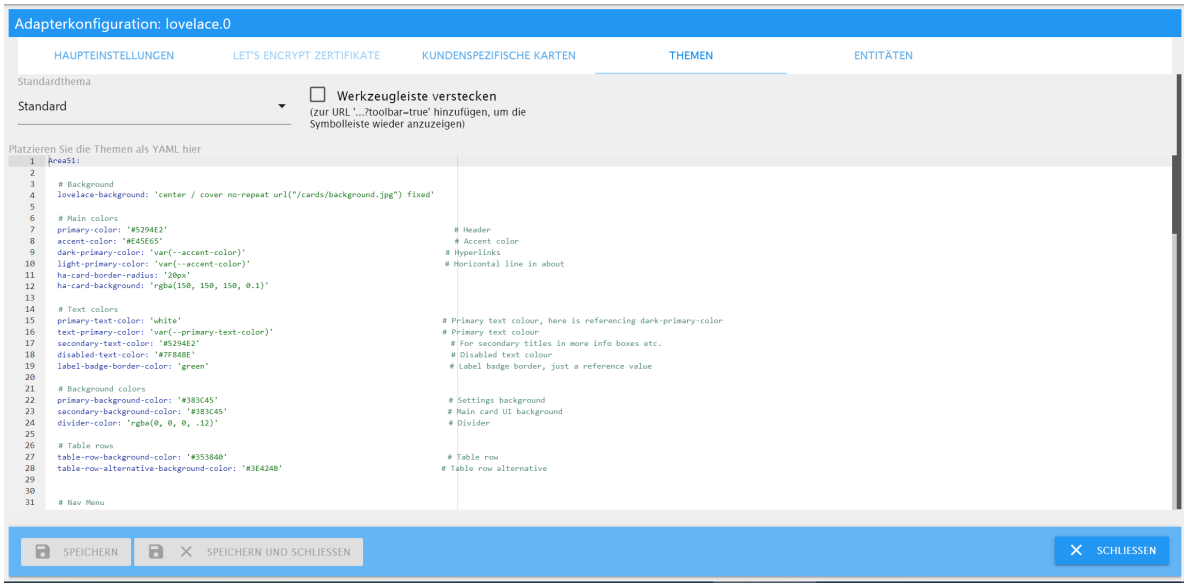
Es sind die folgenden Dinge auszuführen:

**1.) Vorgabe YAML-Theme**

In der Lovelace-Adapterkonfiguration kann unter dem Segment "THEMEN" eine Vorgabe eines Themes-YAML gemacht werden.
Beispiele für Themen (siehe unten).

Hinweis: Der Code-Editor ist nicht im Firefox-Browser sichtbar. Bitte benutze Chrome!

Die YAML-Theme-Vorgabe kann im Code Editor gepastet werden:




### 2.) Optional: Einbettung von Hintergrundbildern

Hintergrundbilder können im Theme wie folgt referenziert werden:

```
  # Background
  lovelace-background: 'center / cover no-repeat url("/cards/background.jpg")
fixed'
```

Ein entsprechendes Hintergrundbild kann per Drag&Drop in der Adapterkonfiguration von Lovelace im Segment "Kundenspezifische Karten" hochgeladen werden.

### 3.) Setzen des Standardthemas

Das Standardthema ist auf das entsprechende Theme zu setzen. Anschließend ist der Speichern-Button zu betätigen.

### Themes-Beispiele

Theme Beispiele: Area51-Theme mit einem Hintergrundbild (background.jpg) und midnight-Theme

```
Area51:

  # Background
  lovelace-background: 'center / cover no-repeat url("/cards/background.jpg")
fixed'

  # Main colors
  primary-color: '#5294E2'
# Header
  accent-color: '#E45E65'
  # Accent color
  dark-primary-color: 'var(--accent-color)'
  # Hyperlinks
  light-primary-color: 'var(--accent-color)'
# Horizontal line in about
```

```
  ha-card-border-radius: '20px'
  ha-card-background: 'rgba(150, 150, 150, 0.1)'

  # Text colors

  primary-text-color: 'white'
# Primary text colour, here is referencing dark-primary-color
  text-primary-color: 'var(--primary-text-color)'
  # Primary text colour
  secondary-text-color: '#5294E2'
  # For secondary titles in more info boxes etc.
  disabled-text-color: '#7F848E'
# Disabled text colour
  label-badge-border-color: 'green'
  # Label badge border, just a reference value

  # Background colors

  primary-background-color: '#383C45'
  # Settings background
  secondary-background-color: '#383C45'
  # Main card UI background
  divider-color: 'rgba(0, 0, 0, .12)'
  # Divider

  # Table rows
  table-row-background-color: '#353840'
  # Table row
  table-row-alternative-background-color: '#3E424B'
  # Table row alternative


  # Nav Menu
  paper-listbox-color: 'var(--primary-color)'
  # Navigation menu selection hoover
  paper-listbox-background-color: '#2E333A'
  # Navigation menu background
  paper-grey-50: 'var(--primary-text-color)'
  paper-grey-200: '#414A59'
  # Navigation menu selection

  # Paper card
  paper-card-header-color: 'var(--accent-color)'
# Card header text colour
  paper-card-background-color: 'rgba(150, 150, 150, 0.1)'
  # Card background colour
  paper-dialog-background-color: '#434954'
# Card dialog background colour
  paper-item-icon-color: 'var(--primary-text-color)'
# Icon color
  paper-item-icon-active-color: '#F9C536'
  # Icon color active
  paper-item-icon_-_color: 'green'
  paper-item-selected_-_background-color: '#434954'
  # Popup item select
  paper-tabs-selection-bar-color: 'green'
```

```
  # Labels
  label-badge-red: 'var(--accent-color)'
# References the brand colour label badge border
  label-badge-text-color: 'var(--primary-text-color)'
  # Now same as label badge border but that's a matter of taste
  label-badge-background-color: '#2E333A'
  # Same, but can also be set to transparent here
  label-badge-background: 'rgba(255, 255, 255, 0.1)'

  # Switches
  paper-toggle-button-checked-button-color: 'var(--accent-color)'
  paper-toggle-button-checked-bar-color: 'var(--accent-color)'
  paper-toggle-button-checked-ink-color: 'var(--accent-color)'
  paper-toggle-button-unchecked-button-color: 'var(--disabled-text-color)'
  paper-toggle-button-unchecked-bar-color: 'var(--disabled-text-color)'
  paper-toggle-button-unchecked-ink-color: 'var(--disabled-text-color)'

  # Sliders
  paper-slider-knob-color: 'var(--accent-color)'
  paper-slider-knob-start-color: 'var(--accent-color)'
  paper-slider-pin-color: 'var(--accent-color)'
  paper-slider-active-color: 'var(--accent-color)'
  paper-slider-container-color: 'linear-gradient(var(--primary-background-
color), var(--secondary-background-color)) no-repeat'
  paper-slider-secondary-color: 'var(--secondary-background-color)'
  paper-slider-disabled-active-color: 'var(--disabled-text-color)'
  paper-slider-disabled-secondary-color: 'var(--disabled-text-color)'


  # Google colors
  google-red-500: '#E45E65'
  google-green-500: '#39E949'

midnight:
  # Main colors
  primary-color: '#5294E2'
# Header
  accent-color: '#E45E65'
  # Accent color
  dark-primary-color: 'var(--accent-color)'
  # Hyperlinks
  light-primary-color: 'var(--accent-color)'
# Horizontal line in about

  # Text colors
  primary-text-color: '#FFFFFF'
  # Primary text colour, here is referencing dark-primary-color
  text-primary-color: 'var(--primary-text-color)'
  # Primary text colour
  secondary-text-color: '#5294E2'
  # For secondary titles in more info boxes etc.
  disabled-text-color: '#7F848E'
# Disabled text colour
  label-badge-border-color: 'green'
  # Label badge border, just a reference value

  # Background colors
```

```
  primary-background-color: '#383C45'
  # Settings background
  secondary-background-color: '#383C45'
  # Main card UI background
  divider-color: 'rgba(0, 0, 0, .12)'
  # Divider

  # Table rows
  table-row-background-color: '#353840'
  # Table row
  table-row-alternative-background-color: '#3E424B'
  # Table row alternative

  # Nav Menu
  paper-listbox-color: 'var(--primary-color)'
  # Navigation menu selection hoover
  paper-listbox-background-color: '#2E333A'
  # Navigation menu background
  paper-grey-50: 'var(--primary-text-color)'
  paper-grey-200: '#414A59'
  # Navigation menu selection

  # Paper card
  paper-card-header-color: 'var(--accent-color)'
# Card header text colour
  paper-card-background-color: '#434954'
# Card background colour
  paper-dialog-background-color: '#434954'
# Card dialog background colour
  paper-item-icon-color: 'var(--primary-text-color)'
# Icon color
  paper-item-icon-active-color: '#F9C536'
  # Icon color active
  paper-item-icon_-_color: 'green'
  paper-item-selected_-_background-color: '#434954'
  # Popup item select
  paper-tabs-selection-bar-color: 'green'

  # Labels
  label-badge-red: 'var(--accent-color)'
# References the brand colour label badge border
  label-badge-text-color: 'var(--primary-text-color)'
  # Now same as label badge border but that's a matter of taste
  label-badge-background-color: '#2E333A'
  # Same, but can also be set to transparent here

  # Switches
  paper-toggle-button-checked-button-color: 'var(--accent-color)'
  paper-toggle-button-checked-bar-color: 'var(--accent-color)'
  paper-toggle-button-checked-ink-color: 'var(--accent-color)'
  paper-toggle-button-unchecked-button-color: 'var(--disabled-text-color)'
  paper-toggle-button-unchecked-bar-color: 'var(--disabled-text-color)'
  paper-toggle-button-unchecked-ink-color: 'var(--disabled-text-color)'

  # Sliders
  paper-slider-knob-color: 'var(--accent-color)'
  paper-slider-knob-start-color: 'var(--accent-color)'
  paper-slider-pin-color: 'var(--accent-color)'
```

```
  paper-slider-active-color: 'var(--accent-color)'
  paper-slider-container-color: 'linear-gradient(var(--primary-background-
color), var(--secondary-background-color)) no-repeat'
  paper-slider-secondary-color: 'var(--secondary-background-color)'
  paper-slider-disabled-active-color: 'var(--disabled-text-color)'
  paper-slider-disabled-secondary-color: 'var(--disabled-text-color)'

  # Google colors
  google-red-500: '#E45E65'
  google-green-500: '#39E949'
```

### Import / Export der Konfiguration

Die komplette Konfiguration der Lovelace-Visualisierung kann wie folgt exportiert werden.

Die Visualisierung ist in den Konfigurationsmodus zu versetzen. Dies erfolgt durch folgende Schritte:

1. Klick auf "..." oben rechts.
2. Klick auf "Benutzeroberfläche konfigurieren"

Anschließend kann der "RAW-Konfigurationseditor" wie folgt erreicht werden:

1. Klick auf "..." oben rechts.
2. Klicke auf "RAW-Konfigurationseditor"

Es öffnet sich nun ein Editor, der alle Ansichten und Karten enthält. Damit liegt die Basis für den Export / Import von Konfigurationen vor.

Export: Die Konfiguration kann durch markieren aller Inhalte und Kopieren in die Zwischenablage (und Einfügen in einer Datei) exportiert werden.

Import: Die Importquelle kann durch Einfügen aus der Zwischenablage eingefügt werden. Entsprechende Änderungen an der Konfiguration sind durch den Button "SPEICHERN" zu sichern.

Alternative Möglichkeit Import/Export:

Die komplette Konfiguration steht in der "objects.json" zur Verfügung.

Mit einem simplen JSON -> YAML Konverter kann man die Daten in YAML aufbereiten!

## Beispiele / Anwendungsfälle

ioBroker-Forum: Zeigt her eure Lovelace-Visualisierung

IoBroker + TR064 + lovelace – Anruferkennung per Popup auf Dashboard

Beispielkonfiguration von Scrounger

Beispielkonfiguration von allgrind

Beispielkonfiguration von WW1983

Farbe vom Icon je nach Zustand setzen

# FAQ

## Wie kann ich die Toolbar verstecken?

Um die Toolbar zu verstecken kann man die Option "Werkzeugleiste verstecken" in der Lovelace-Adapterkonfiguration auf dem Segment "Themen" setzen. Um die Toolbar dynamisch wieder anzuzeigen, kann man an die URL den folgenden Parameter anhängen: `?toolbar=true`

## Custom element doesn't exist: xy-card.

Sofern dieser Fehler in der Visualisierung von Lovelace erscheint wurde eventuell die .JS-Datei nicht richtig übertragen oder es handelt sich nicht um eine .JS-Datei!

## Kann ich statt An/Aus andere Bezeichnungen für die Werte von Entitäten ausgeben?

Kann Manchmal möchte man statt "An" und "Aus" eine andere Bezeichnung. Bei einem Wassermelder sollte nicht "An" und "Aus" stehen, sondern "Wasser erkannt" oder "trocken". Die Änderung von Werten ist  leider mit den Standard-Karten nicht möglich. Eine Idee ist es den Wert komplett auszublenden und bei binären Sensoren (Fensterkontakten) nur die Werte eines Bereichs über die Entity Filter Card anzuzeigen (z.B. nur die offenen Fenster/Türen, die Alarmmelder mit Alarm etc.).

Mit der Custom Card "Banner-Card" kann man eine Transformierung der Werte durch YAML-Code durchführen:

Beispiel: YAML-Code für einen Fensterkontakt mit der Banner-Card.

```
entity: xy..
map_state:
  'on': geschlossen
  'off': geöffnet
```

Mit der Custom Card "Button-Card" ist ebenso eine Transformierung der Werte durch YAML-Code möglich:

Beispiel: YAML-Code für einen Haustürsensor mit der Button-Card.

```
type: 'custom:button-card'
entity: sensor.kontakt_haustuere
name: Terasse Rechts
label: |
  [[[
    if (entity.state == true)
      return "geöffnet";
    else
      return "geschlossen";
  ]]]
layout: icon_name_state2nd
show_state: false
show_name: true
```

```
show_label: true
icon: 'mdi:door-closed'
styles:
  label:
    - font-size: small
    - font-style: italic
    - color: '#a1a1a1'
```

### Kann man die Werte die durch einen Schalter gesetzt werden beeinflussen?

Das kann man über alias lösen:

```
read = "val == 'on' ? true : false";
write = "val ? 'on' : 'off'";
```

## Links / Verweise

- ioBroker.net
- GitHub Lovelace Adapter
  - Issues
- Home Assistant
- Threads ioBroker-Forum
  - Test Adapter lovelace v0.2.x
  - Test Adapter lovelace v1.2.x