ioBroker documentation
(UNOFFICIAL) Documentation

Matthias Kleine

11.01.2022

Erste Schritte

Tipp: Diese Dokumentation ist Open Source und hier zu finden: GitHub. Falls Du einen Fehler findest oder Inhalte ergänzen möchtest, bist Du herzlich dazu eingeladen!

Willkommen auf den Seiten der **inoffiziellen** ioBroker-Dokumentation. Diese Dokumentation wurde von Matthias Kleine, haus-automatisierung.com ins Leben gerufen und soll als alternative Informationsquelle zur offiziellen Variante stehen.

Die Dokumentation ist ausschließlich auf Deutsch verfügbar.

KAPITEL 1

Was ist der ioBroker?

Wie der Name schon vermuten lässt, handelt es sich beim ioBroker um eine Software-Lösung, welche es ermöglicht, verschiedene Smart-Home Systeme miteinander zu verknüpfen und zwischen diesen Systemen zu vermitteln. Aktuell können über 250 verschiedene Systeme integriert werden.

So kann man zum Beispiel Geräte mit Alexa ansteuern, welche selbst keine Alexa-Integration bieten. Gleiches gilt für HomeKit oder auch Google Home. Aber auch ohne Cloud-Lösungen im rein lokalen Betrieb macht der ioBroker Sinn. Du möchtest auf einem KNX-Taster die Leistung der per Modbus angebundenen PV-Anlage sehen? Kein Problem. Du möchtest mit einem Philips Hue Button einen HomeMatic-Aktor ansteuern? Auch kein Problem.

Mit dem ioBroker kannst Du also über die Grenzen der Hardware-Hersteller hinausgehen und Systeme miteinander kombinieren, welche nicht direkt miteinander sprechen können.

Je länger Du mit ioBroker arbeitest, desto mehr Möglichkeiten wirst Du entdecken! Es lässt sich nahezu jede Lösung anbinden, welche eine Schnittstelle bereitstellt. Der ioBroker steht dann praktisch in der Mitte all dieser Systeme und vermittelt - daher auch der Name **io** (input / output) **Broker** (Vermittler).

Bemerkung: ioBroker ist eine reine Software-Lösung, welche über viele Interfaces/Schnittstellen und diversen Protokollen mit unterschiedlicher Hardware und Software kommunizieren kann.

1.1 Hardware

Generell lässt sich der ioBroker auf jedem Betriebssystem installieren, auf welchem auch nodejs läuft.

Tipp: Achte auf den Energiebedarf Deiner eingesetzten Hardware. ioBroker macht nur Sinn, wenn die Anwendung rund um die Uhr (24/7) verfügbar ist und läuft. Selbst wenige Watt machen über das gesamte Jahr betrachtet einen Unterschied.

1.1.1 Raspberry Pi

Als einfachster Einstieg in die ioBroker-Welt wird nach wie vor der Raspberry Pi empfohlen. Dabei handelt es sich um einen kleinen Einplatinen-Computer, welcher sehr klein ist, wenig Leistung im Dauerbetrieb benötigt und ausreichend Ressourcen für den Betrieb von ioBroker bietet.

Mit der Version 4 des Raspberry Pi wurden erstmals Modelle mit mehr als 1 GB Arbeitsspeicher vorgestellt. Da der ioBroker beim Betrieb mehrerer Adapter relativ viel Arbeitsspeicher benötigt, ist ein System mit **mindestens 2 GB RAM** empfohlen. Die Variante mit 4 GB oder 8 GB bietet noch einmal deutlich mehr Reserven.

Raspberry Pi 4 (mindestens 2 GB RAM, besser 4 GB)

- Reichelt **
- Amazon **
- Conrad **

Raspberry Pi Gehäuse

- Reichelt **
- Amazon **
- Conrad **

Raspberry Pi Netzteil

- Reichelt **
- Amazon **
- Conrad **

Micro SD-Karte

- Reichelt **
- Amazon **
- Conrad **

** Link zu einer Produktempfehlung. Wenn Du über einen dieser Links etwas kaufst, bekomme ich eine kleine Provision vom Shop (Affiliate-Link).

Tipp: Verwende als Image auf jeden Fall Raspberry Pi OS Lite und NICHT die Desktop-Version. Warum? Weil die Desktop-Variante viel mehr Ressourcen braucht und extrem viel Overhead mitbringt, welcher nicht gebraucht wird. Die Desktop-Version ist nur erforderlich, wenn Du eine Maus, Tastatur und einen Monitor direkt an dem Raspberry Pi anschließen möchtest um diesen (wie der Name sagt) als Desktop-Computer zu verwenden.

Wie genau ein Raspberry Pi 4 für die Installation von ioBroker vorbereitet wird, erfährst Du in diesem Video:

1.1.2 Proxmox

Neben einer nativen Installation, ist auch die Installation in einer virtuellen Maschine oder einem Container möglich. Dafür sollte dann aber etwas leistungsstärkere Hardware wie ein Intel NUC® oder ähnliches verwendet werden.

1.2 Installation

Generell lässt sich der ioBroker auf jedem Betriebssystem installieren, welchs auch nodejs unterstützt.

1.2.1 Fertige Images

In der Vergangenheit wurden z.B. für den Raspberry Pi fertige Images angeboten, welche aber heute nicht mehr verwendet werden sollten! Eine manuelle Installation ist immer zu bevorzugen.

1.2.2 Manuelle Installation (Linux)

Bemerkung: In älteren Anleitungen liest man häufig, dass noch weitere Pakete (wie nodejs) vorher auf dem System installiert werden müssen. Das ist aber nicht mehr nötig, da sich der ioBroker-Installer selbst um die Abhängigkeiten kümmert und diese installiert!

Unter Linux lässt sich der ioBroker mit nur einem einzigen Befehl installieren:

```
curl -sLf https://iobroker.net/install.sh | bash -
```

Dieser Befehl bereitet das komplette System vor und richtet alles nötige ein:

- `nodejs` wird in der aktuell empfohlenen Version heruntergeladen und installiert
- Ein neuer System-Benutzer wird erstellt (ioBroker)
- Der ioBroker wird an die richtige Stelle im System installiert
- Es werden die wichtigsten Adapter installiert und Instanzen hinzugefügt (Admin, Discovery, ...)
- Ein Autostart für ioBroker wird eingerichtet

Danach kann über den Standard-Port 8083 im Browser Deiner Wahl die Admin-Oberfläche aufgerufen werden.

1.2.3 Manuelle Installation (Windows)

Natürlich kann der ioBroker auch unter Windows installiert werden. Ich persönliche sehe Windows aber als Desktop-Betriebsystem, welches nicht als Server verwendet werden sollte. Eine Installation unter Linux ist daher aus meiner Sicht immer zu bevorzugen.

1.2.4 Docker-Image

Das offizielle Docker-Image (von buanet) findest Du hier: Official Docker Image for ioBroker

Auf den GitHub-Seiten ist außerdem erklärt, wie man mit dem Image arbeitet.

1.3 Ressourcen

Neben dieser Dokumentation gibt es natürlich noch viele weitere offizielle und inoffizielle Wege, mehr über den ioBroker zu lernen. Beginnen wir mit den offensichtlichen Punkten:

- Offizielles Forum
- Offizielle Dokumentation

Ich würde Dir wärmstens empfehlen, Dich im Forum zu registrieren. Dort sind sehr viele kompetente Menschen unterwegs, welche gerne bei Problemen und Fragen helfen.

1.3.1 Facebook-Gruppen

- ioBroker SmartHome und IoT
- ioBroker Vis

1.3.2 YouTube-Kanäle

- haus-automatisierung.com
- verdrahtet
- EddyD's SmartHome
- Marcels Custom Shop
- Svens ioBroker Tutorials (inaktiv)
- nurChris (inaktiv)
- TeamSpeak SmartHome (inaktiv)

1.3.3 Discord-Server

- Mehr Details im Forum
- Discord

1.4 Architektur

Der ioBroker ist modular aufgebaut und besteht aus mehreren Komponenten. In der Mitte finden wir das Herzstück des Systems: Den `js-controller`.

Dieser Controller verwaltet alles im System. Von dort werden die einzelnen Instanzen gestartet und mit diesen kommuniziert.

1.4.1 Adapter

Ein Adapter ist ein Stück Software, welches eine bestimmte Aufgabe übernimmt. Diese Aufgabe kann alles mögliche sein.

Es gibt daher verschiedenste Adapter, welche zum Beispiel

- mit Hardware kommunizieren (Philips Hue, KNX, HomeMatic, Loxone, ...)
- Daten aus dem Internet abrufen (Wetter, Verkehr, ...)
- Logiken bereitstellen (Szenen, Regeln, ...)

- Schnittstellen in den ioBroker von außen öffnen / Cloud-Verbindungen (Zugriff von unterwegs)
- verschiedene Visualisierungsoberflächen bereitstellen (Steuerung per Tablet)
- den Sonnenstand berechnen
- Feiertage für dein Bundesland ermitteln
- die Administrationsoberfläche bereitstellen (admin)
- Backups erstellen
- Verbindungen zu Datenbanken herstellen
- uvm.

Die Liste der verfügbaren Adapter ist extrem lang und umfasst über 400 verschiedene Integrationen.

Diese Liste ist Dein Baukasten. Was Du brauchst, installierst Du dazu. Was Du nicht brauchst, lässt Du weg. Ganz einfach.

Bemerkung: Nach der Installation eines Adapters wird automatisch eine neue Instanz erstellt, um Dir das Leben leichter zu machen.

Damit Du einfacher den richtigen Adapter für eine Aufgabe finden kannst, sind Adapter in Kategorien eingeteilt:

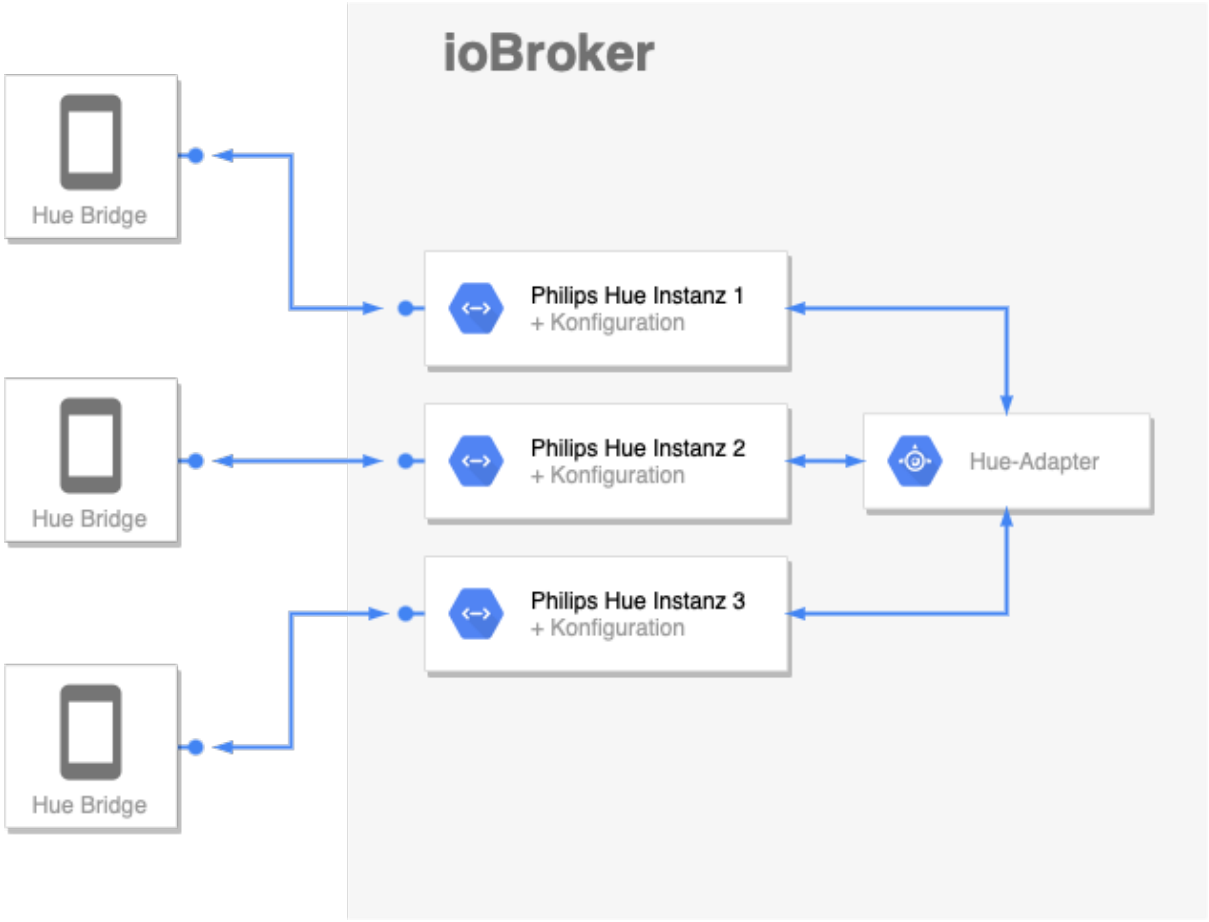
- Allgemein
- Aufbewahrung
- Visualisierung
- Energie
- Logik
- …

1.4.2 Instanzen

Eine Instanz ist am Ende der eigentliche **Prozess**, welcher für einen Adapter gestartet wird. Für jeden Adapter können beliebig viele Prozesse (Instanzen) gestartet werden.

Nehmen wir das Beispiel Textverarbeitung. *Microsoft Word* ist also der Adapter und wird einfach nur einmalig installiert. Du kannst aber dann beliebig viele Word-Dokumente parallel öffnen, ohne das Programm Word mehrfach installieren zu müssen. Diese Prozesse sind dann die Instanzen. Was genau diese Prozesse darstellen, hängt von der geöffneten Word-Datei ab - diese Datei wäre dann im ioBroker-Umfeld die Instanz-Konfiguration.

Warum sollte man mehrere Instanzen für Adapter erstellen? In der Praxis kommt das relativ selten vor. Aber angenommen Du hast mehrere Hue-Bridges von Philips im Haus. Dann würde jede Instanz die Kommunikation mit genau einer Hue-Bridge übernehmen. Du bräuchtest also genauso viele Instanzen des Hue-Adapters wie Du Hue-Bridges zu Hause hast.



Ein weiteres Beispiel wären die Wetterdaten von verschiedenen Orten. Möchtest Du bei Dir zu Hause die Regenwahrscheinlichkeit ermitteln, installierst Du einen Wetter-Adapter und konfigurierst eine Instanz auf deinen Wohnort. Wenn Du dann noch ein Ferienhaus hast, erstellst Du eine weitere Instanz, welche auf den entsprechenden Ort konfiguriert wird und somit von dort die Daten liefert.

Da es also **mehrere Instanzen vom gleichen Adapter** geben kann, werden diese Instanzen durchnummeriert. Standardmäßig startet die Numerierung bei Null. In diesem Beispiel also:

- `hue.0`
- `hue.1`
- `hue.2`

Von den meisten Adaptern wirst Du aber nur eine einzelne Instanz haben, welche die Nummer Null (0) bekommt. Nur in seltenen Fällen werden mehrere Instanzen von einem Adapter erstellt.

Bemerkung: Während Adapter „nur“ die Software bereitstellen, enthalten Instanzen die spezifische Konfiguration. Der Adapter gibt dabei nur vor, WAS konfiguriert werden kann. Die exakte Konfiguration wird in der jeweiligen Instanz gespeichert (wie zum Beispiel der Wohnort im Wetter-Adapter).

Einen Adapter zu installieren braucht also erstmal nur Ressourcen auf der Festplatte. Eine neue Instanz davon zu erstellen braucht dann weitere Ressourcen wie CPU-Zeit oder Arbeitsspeicher.

Jede Instanz wird dabei (normalerweise) in einem eigenen Prozess gestartet. Das hat zur Folge, dass der Prozess das restliche System nicht beeinflussen kann. Sollte also ein Adpater abstürzen oder nicht mehr laufen, funktioniert der Rest trotzdem weiter! Dieses Vorgehen braucht zwar deutlich mehr Ressourcen als bei vergleichbaren Lösungen, aber sorgt für viel mehr Stabilität. Um Ressourcen zu sparen, kann (zu lasten der Stabilität) der *Compact Mode* aktiviert werden.

Selbst die Admin-Oberfläche ist eine Instanz eines Adapters, welche genau wie alle anderen Instanzen mit dem

`js-controller` spricht. Theoretisch könnte man also den ioBroker auch komplett ohne den Admin-Adapter betreiben. Dieser bietet einfach nur eine möglichst komfortable Oberfläche um dein System zu verwalten.

1.4.3 js-controller

Kommen wir noch einmal zurück zum angesprochenen `js-controller`. Dieser startet die Prozesse der einzelnen Instanzen und verwaltet die Kommunikation mit diesen.

Bleiben wir bei dem Beispiel von Philips Hue. Nachdem also eine neue Instanz für den installierten Adapter gestartet wurde, legt der Adapter Objekte an, welche Deine Räume, Szenen und Lampen repräsentieren, welche Deine Philips Hue Bridge kennt. Diese Objekte kann er aber nicht selbst anlegen, sondern nur den `js-controller` darum bitten dies zu tun. Also sendet der Hue-Adapter eine Nachricht an diesen Prozess, und übermittelt die Informationen für das Anlegen der Objekte.

Genauso können sich Instanzen beim `js-controller` registrieren, dass diese bestimmte Informationen abrufen möchten. Angenommen die Visualisierung möchte immer den aktuellen Status von Philips Hue Lampen auf einer Webseite darstellen. In dem Fall würde der Adapter der Visualisrung den `js-controller` bitten, bei jeder Änderung eines Status im Philips Hue Adapter informiert zu werden. So wird vermieden, dass jede Instanz über jede Änderung im System informiert wird. Ansonsten würden ohne Ende irrelevante Nachrichten durch das System gesendet. Und das vermeidet dieses Konzept.

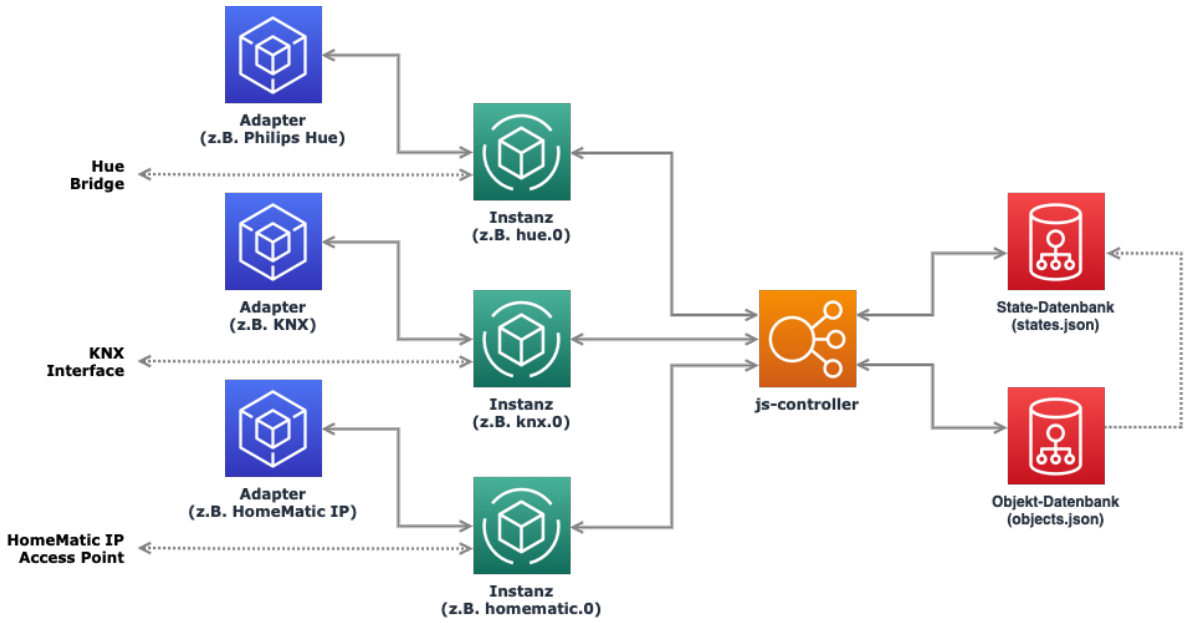
Bemerkung: Falls Du einige Begriffe hier noch nicht verstanden haben solltest, werden diese in den anderen Grundlagen-Dokumentation noch im Detail erklärt!

GitHub-Repository vom js-controller

1.5 Daten-Speicherung

Um Daten zu speichern, muss ein Objekt und ein Zustand existieren. Das Objekt ist statisch und enthält viele Meta-Daten. Zusätzlich gibt es den dynamischen Zustand (auch „State“ genannt), welcher den aktuellen Wert hält. Beides zusammen nennt sich Datenpunkt.

Die Objekte und Zustände werden in Datenbanken vorgehalten, welche der `js-controller` verwaltet. Über diesen können Daten aus den Datenbanken abgefragt oder geändert werden.



Bemerkung: Als Anwender muss man nur sehr wenige Datenpunkte selber anlegen. Die meisten Datenpunkte werden von den einzelnen Adaptern automatisch angelegt.

1.5.1 Objekt

Ein Objekt beschreibt, welche Informationen genau gespeichert werden können. Es handelt sich bei einem Objekt hauptsächlich Meta-Informationen, welche den Datenpunkt beschreiben. Erforderlich sind:

- `_id` - Eindeutige **ID**
- `type`- Typ des Objektes (`device`, `channel`, `state`, ...) - alle Typen (mit Beispielen) findest Du hier: *Objekte*

Daneben gibt es noch weitere (optionale) Informationen, welche das Objekt genauer definieren.

- Name
- Einheit (z.B. °C oder kWh)
- Beschreibung
- erlaubter Minimalwert und Maximalwert
- Lese- und Schreibrechte
- ...

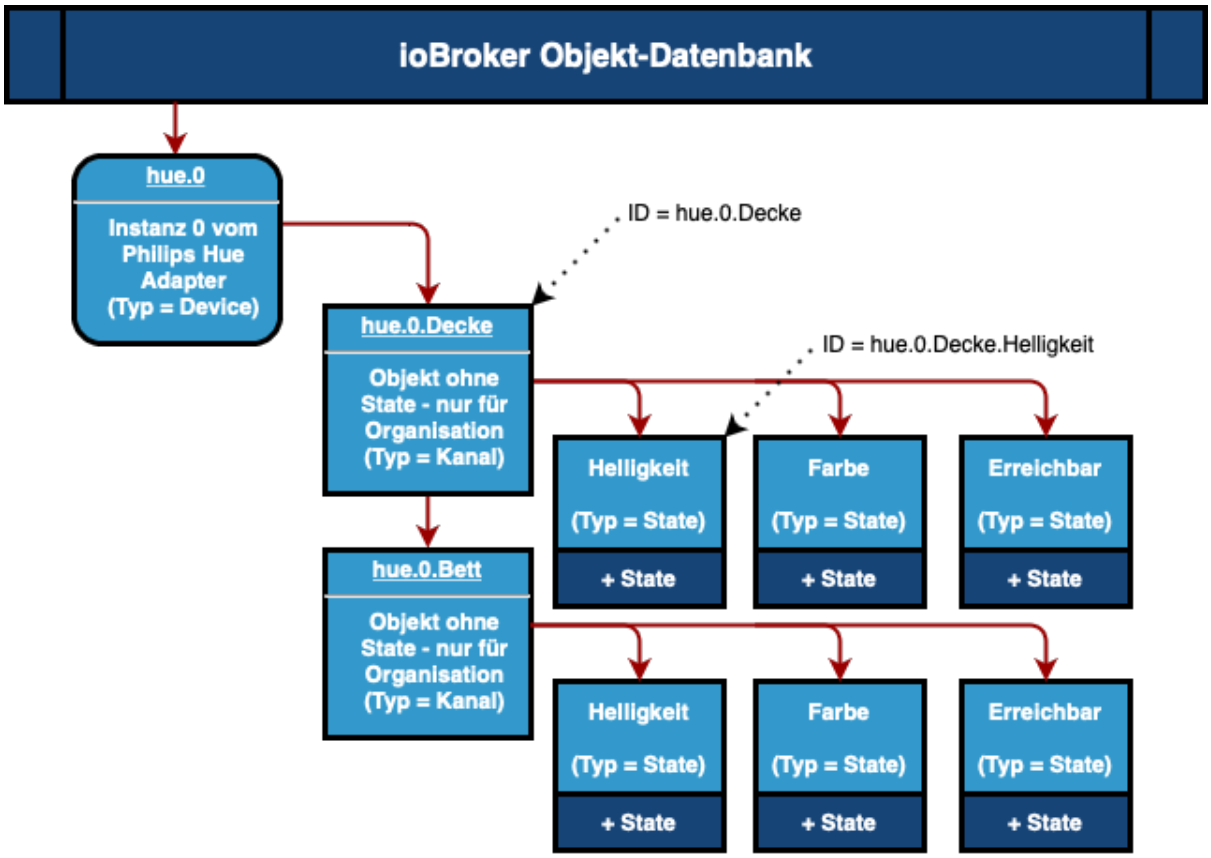
Objekte werden dabei in einer Hierarchie abgebildet. Also in einer Baumstruktur - genau wie in vielen Dateisystemen. **Als Trennzeichen der Ebenen wird dabei der Punkt benutzt.**

Angenommen Du hast eine Philips Hue Bridge und hast den Philips Hue-Adapter installiert. Dann würde für diese Instanz automatisch alle nötigen Objekte für die Steuerung der angelernten Lampen, Strips usw. anlegen.

Der Namespace (siehe unten) dieser Instanz lautet dann `hue.0`. Hier siehst Du schon das erste Trennzeichen. Das erste Objekt auf der Root-Ebene heißt also `hue`. Danach folgt ein weiteres, welches wie die Instanznummer heißt.

Unter diesem Objekt werden dann weitere Objekte angelegt, welche alles Mögliche abbilden können. Dabei werden die Informationen so granular wie möglich abgebildet. So gibt es zum Beispiel für jede angelernte Lampe ein weiteres Objekt, welches dann wieder Objekte darunter enthält. So wird eine logische Hierarchie aufgebaut. Stell Dir das wie deine Urlaubsfotos vor, welche Du auch in verschiedene Ordner auf deiner Festplatte ablegst. Alle Fotos aus einem Urlaub kommen zusammen in einen Ordner. Und so ist das mit den Objekten auch. Alles, was zum Beispiel eine einzelne Lampe kann, wird als einzelne Objekte unter ein gemeinsames Objekt gepackt.

Bemerkung: Nicht jeder Datenpunkt hat zwingend einen zugehörigen Zustand. Aus organisatorischen Gründen kann man auch Objekte anlegen, welches nur für die Struktur dienen. Diese Objekte sind vom Typ „Kanal“ bzw. Englisch „Channel“.



Die **ID** ist dabei ein eindeutiger Schlüssel zu einem Objekt. Dieser ist vergleichbar mit dem absoluten Pfad in einem Dateisystem, welcher zu genau einem Ziel führt. In der obigen Grafik siehst Du die verschiedenen IDs einiger Datenpunkte. Über diese ID kommunizierst Du mit dem Objekt.

Objekte werden unter Linux als JSON (Text, UTF-8) in der folgenden Datei abgelegt:

`/opt/iobroker/iobroker-data/objects.json`

Diese Datei nennt man auch Objekt-Datenbank. Mehr Details (für Entwickler) unter *Objekte*.

1.5.2 State (Zustand)

Bemerkung: Nur Objekte vom Typ `state` haben auch einen zugehörigen Zustand. **Nicht jedes Objekt hat einen Zustand, aber jeder Zustand ein Objekt!**

Ein `state` ist der eigentliche Wert eines Datenpunktes. Neben dem Wert werden aber auch hier noch weitere Informationen vorgehalten, wie zum Beispiel:

- `val` - Der aktuell gespeicherte Wert
- `ack` - Bestätigt-Flag, ob der (neue) Wert vom Adapter bzw. Ziel akzeptiert wurde. Siehe auch *Logik / Automation*
- `ts` - Unix Timestamp (Zeitstempel in Millisekunden) wann der Zustand zuletzt **aktualisiert** wurde
- `lc` - Unix Zimestamp (Zeitstempel in Millisekunden) wann der Zustand zuletzt **geändert** wurde (last change)
- ...

Es handelt sich also im Gegensatz zum Objekt um dynamische Daten, welche sich ständig ändern können.

Bemerkung: Die meisten dieser Informationen sind für Dich als Anwender nicht interessant. Du arbeitest zu 99% nur mit dem Wert `val` eines Zustandes. Dennoch solltest Du wissen, dass neben dem Wert noch mehr Informationen gespeichert werden.

Das zugehörige Objekt gibt dabei vor, wie der Zustand aussehen darf. Also in welchem Datentyp der Wert vorgehalten wird, ob der Zustand nur gelesen werden darf oder auch geschrieben werden kann, uvm.

Es ist besonders wichtig zu verstehen, was es mit bestätigten Zuständen auf sich hat (siehe `ack`). Dabei hilft Dir dieses Video:

Zustände werden im Standard als JSON (Text, UTF-8) in der folgenden Datei abgelegt:

`/opt/iobroker/iobroker-data/states.json`

Diese Datei nennt man auch State-Datenbank. Mehr Details (für Entwickler) unter *States*.

1.5.3 Datenpunkt

Wenn man von einem Datenpunkt spricht, ist die Kombination aus Objekt mit dem zugehörigen Zustand gemeint.

Die Kombination von Objekte und Zustand ist die einzige Möglichkeit in ioBroker Daten abzulegen. Alles wird in dieser Struktur abgebildet. Selbst die Konfiguration von Instanzen wird in Datenpunkten gespeichert. Diese findest Du z.B. im System-Namespace (siehe unten).

1.5.4 Namespace

Damit die Objekte im System in einer logischen Struktur abgelegt werden, gibt es sog. Namespaces (Namensräume). So wird vermieden, dass nicht jeder Adapter seine Daten an eine andere Stelle in der Hierarchie speichert. Außerdem werden auf diese Weise doppelte Namen vermieden und als Entwickler kann man sich in seinem Namespace „frei bewegen“.

Bleiben wir beim Beispiel Philips Hue, welches schon ötfter in dieser Dokumentation herhalten musste. Erstellst Du eine Instanz vom Hue-Adapter, lautet der Namespace für diesen Adapter `hue.0`. Du erinnerst Dich: Die Null steht dabei für die erste Instanz, da von einem Adpater mehrere Instanzen erstellt werden können. Alles, was der Adapter nun an Objekten bereitstellt, ist in diesem Namespace zu finden. Löschst Du die Instanz, wird der Namespace ebenfalls gelöscht.

Bemerkung: Als Anwender solltest Du keine eigenen Objekte in Namespaces von Adaptern oder vom System ablegen! Wenn Du eigene Objekte erstellen möchtest, tu dies bitte im Namespace **0_userdata**

Weiterhin gibt es den (reservierten) Namespace `system.` für das System. Dort ist unter anderem folgendes enthalten:

- `system.config` - Systemkonfiguration (Sprache, Datumsformat, Verwahrungsort, ...) - siehe *System-Config*
- `system.host.<hostname>`` - js-controller-Prozess (Uptime, Ressourcen, ...)
- `system.repositories` - Liste der vefügbaren Adpater
- `system.certificates` - Konfigurierte Zertifikate
- `system.meta.` - Meta-Informationen
- `system.user.` - Alle Benutzer des Systems
- `system.group.` - Alle Benutzer-Gruppen des Systems
- `system.adapter.<adapter-name>` - Standard-Konfiguration des Adapters für neue Instanzen
- `system.adapter.<adapter-name>.<instanz-nummmer>` - Informationen zur einzelnen Instanz (Uptime, Ressourcen, ...)

Gefahr: Ändere keine Informationen in dem System-Namespace, wenn Du nicht genau weißt, was Du tust. Als normaler Anwender gibt es keinen Grund, dort etwas ändern. Diese Informationen sind nur für Entwickler relevant! Im Admin-Adapter sieht man diese Objekte auch nur dann, wenn der Expertenmodus angeschaltet ist.

1.5.5 Speicherort

Im Standard arbeitet der ioBroker mit dem Dateisystem (`files`) als Speicherort für die Objekt- und Zustands-Datenbank. Dies kann aber auch umkonfiguriert werden, sodass stattdessen z.B. Redis zum Speichern der Daten genutzt wird. Dabei handelt es sich um einen Dienst, welcher zusätzlich auf dem System installiert werden muss.

TODO

1.6 CLI (Command Line Interface)

Das Command Line Interface von ioBroker ist sehr mächtig. Hierüber kann im Prinzip alles gesteuert werden, was über den ioBroker Admin als Weboberfläche auch möglich ist - und noch mehr. Es können also zum Beispiel neue Adapter installiert oder aktualisiert werden, Backups verwaltet werden, Dateirechte angepasst werden und vieles mehr!

```
iobroker [command]

Commands:
iobroker setup                                          Setup ioBroker
iobroker start                                          Starts the js-controller
iobroker stop                                           stops the js-controller
iobroker restart                                        Restarts js-controller
iobroker debug <adapter>[.<instance>]                   Starts a Node.js␣
↪debugging session for the adapter instance
iobroker info                                           Shows the host info
iobroker logs [<adapter>]                               Monitor log
iobroker add <adapter> [desiredNumber]                  Add instance of adapter
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
iobroker install <adapter>                                    Installs a specified␣
↪adapter
iobroker rebuild <adapter>|self                               Rebuilds a specified␣
↪adapter
iobroker url <url> [<name>]                                   Install adapter from␣
↪specified url, e.g. GitHub
iobroker del <adapter>                                        Remove adapter from␣
↪system
iobroker del <adapter>.<instance>                             Remove adapter instance
iobroker update [<repositoryUrl>]                             Update repository and␣
↪list adapters
iobroker upgrade                                              Upgrade management
iobroker upload                                               Upload management
iobroker object                                               Object management
iobroker state                                                State management
iobroker message <adapter>[.instance] <command> [<message>]   Send message to adapter␣
↪instance/s
iobroker list <type> [<filter>]                               List all entries, like␣
↪objects
iobroker chmod <mode> <file>                                  Change file rights
iobroker chown <user> <group> <file>                          Change file ownership
iobroker touch <file>                                         Touch file
iobroker rm <file>                                            Remove file
iobroker file                                                 File management
iobroker user                                                 User commands
iobroker group                                                group management
iobroker host <hostname>                                      Set host to given␣
↪hostname
iobroker set <adapter>.<instance>                             Change settings of␣
↪adapter config
iobroker license <license.file or license.text>               Update license by given␣
↪file
iobroker cert                                                 Certificate management
iobroker clean <yes>                                          Clears all objects and␣
↪states
iobroker backup                                               Create backup
iobroker restore <backup name or path>                        Restore a specified␣
↪backup
iobroker validate <backup name or path>                       Validate a specified␣
↪backup
iobroker status [all|<adapter>.<instance>]                    Status of ioBroker or␣
↪adapter instance
iobroker repo [<name>]                                        Show repo information
iobroker uuid                                                 Show uuid of the␣
↪installation
iobroker unsetup                                              Reset license,␣
↪installation secret and language
iobroker fix                                                  Execute the installation␣
↪fixer script, this updates your ioBroker installation
iobroker multihost                                            Multihost management
iobroker compact                                              compact group management
iobroker plugin                                               Plugin management
iobroker version [<adapter>]                                  Show version of js-
↪controller or specified adapter
```

Weiterhin ist es möglich, zu (fast) jedem Befehl weitere Hilfen aufzurufen. Dafür wird einfach die Option `--help` angehängt:

```
iobroker user --help

Commands:
iobroker user add <user>      Add new user
iobroker user del <user>      Delete user
iobroker user passwd <user>   Change user password
iobroker user enable <user>   Enable user
iobroker user disable <user>  Disable user
iobroker user get <user>      Get user
iobroker user check <user>    Check user password

Options:
    --help     Show help  [boolean]
-v, --version  Show version  [boolean]
```

1.6.1 Beispiele (System)

- `iobroker start`: Startet das gesamte ioBroker-System (`js-controller` und alle Instanzen)
- `iobroker stop`: Stoppt das gesamte ioBroker-System (`js-controller` und alle Instanzen)
- `iobroker version`: Version des `js-controller`
- `iobroker uuid`: UUID des Systems - siehe *System-UUID*
- `iobroker upgrade self`: Aktualisiert den `js-controller`

1.6.2 Beispiele (Backup)

- `iobroker backup`: Erstellt ein neues Backup anhand der Konfiguration im `backitup` Adapter

1.6.3 Beispiele (Adapter)

- `iobroker upgrade --yes`: Aktualisiert alle Adapter, ohne für jeden Adapter nachzufragen, ob die Version wirklich installiert werden soll.
- `iobroker upgrade ical`: Aktualisiert einen einzelnen Adpater (in diesem Fall ical) auf die aktuellste Version
- `iobroker upgrade ical@1.11.2`: Aktualisiert einen einzelnen Adpater (in diesem Fall ical) auf die Version `1.11.2`
- `iobroker add wled@0.6.0`: Installiert die Version `0.6.0` des WLED Adapters und erstellt eine neue Instanz
- `iobroker add wled`: Installiert die aktuellste Version des WLED Adapters und erstellt eine neue Instanz
- `iobroker del wled.0`: Löscht die Instanz `wled.0`
- `iobroker del wled`: Deinstalliert den WLED Adapter
- `iobroker status iot.0`: Prüft, ob eine Instanz des IoT Adapters läuft
- `iobroker version iot`: Version des installierten IoT Adapters
- `iobroker update -u`: Listet alle Adapter mit verfügbaren Updates auf
- `iobroker update -a`: Listet alle verfügbaren Adapter auf

1.6.4 Beispiele (User)

- `iobroker user passwd admin`: Passwort vom User `admin` ändern (z.B. falls es vergessen wurde)

1.6.5 Beispiele (Objekte)

- `iobroker object get system.config`: Liefert den Inhalt des Objektes - siehe *Daten-Speicherung*

1.6.6 Beispiele (Zustände)

- `iobroker state get admin.0.info.updatesNumber`: Liefert den kompletten Zustand als JSON
- `iobroker state getvalue admin.0.info.updatesNumber`: Liefert nur den Wert des Zustandes - siehe *Daten-Speicherung*

1.7 ACL (Access Control List)

TODO

1.8 System-Config

Die aktuelle System-Konfiguration (welche auch über den Admin konfigurierbar ist), wird im Objekt `system.config` (siehe *Objekte*) gespeichert.

1.9 System-UUID

Jede Installation bekommt automatisch eine zufällige UUID zugewiesen, welche für unterschiedliche Aktionen genutzt wird. Dieses kann beispielsweise über das *CLI (Command Line Interface)* ausgelesen werden:

```
iobroker uuid
```

Diese UUID wird unter anderem für folgende Informationen/Dienste genutzt:

- *Statistics*
- *Adapter-Ratings*

1.10 Logik / Automation

Wie Du schon gelernt hast, gibt es verschiedene Adapter, welche durch konfigurierte Instanzen über den `js-controller` Daten ablegen / entgegen nehmen können.

TODO

1.11 Compact Mode

Die grunlegende *Architektur* vom ioBroker sieht vor, dass jede Instanz eines Adapters in einem eigenen Prozess gestartet wird. Da dies aber sehr viele Ressourcen benötigt, wurde der sogenannte Compact Mode entwickelt. In diesem Modus werden sämtliche Instanzen von Adaptern, welche diesen Modus unterstützen, im gleichen Prozess gestartet, wie der `js-controller`.

TODO

- ExtSource: https://forum.iobroker.net/topic/18338/experimentell-js-controller-compact-mode
- ExtSource: https://forum.iobroker.net/topic/32789/anleitung-für-adapter-entwickler-compact-mode-testen

1.12 ioBroker-Domains

1.12.1 iobroker.com

- Unternehmens-Webseite und Shop

1.12.2 iobroker.dev

- Developer Portal

1.12.3 iobroker.net

- Cloud-Verbindung (Cloud-Adapter) und Lizenzen
- Installations-Script (Redirect auf iobroker.live) - siehe *Installation*
- Installations-Fixing-Script (Redirect auf iobroker.live)

1.12.4 www.iobroker.net

- Offizielle Website und Dokumentation
- Adapter-Liste

1.12.5 download.iobroker.net

- Adapter-Liste (alt) (veraltet, sollte nicht mehr genutzt werden)
- Repository stable (Redirect auf repo.iobroker.live)
- Repository beta (Redirect auf repo.iobroker.live)

1.12.6 sentry.iobroker.net

- Sentry-Report Management - siehe *Sentry*

1.12.7 forum.iobroker.net

- Offizielles Forum

1.12.8 rating.iobroker.net

- URL für neue Bewertung (POST) - siehe *Adapter-Ratings*
- URL für aktuelle Bewertungen (GET) - siehe *Adapter-Ratings*

1.12.9 weblate.iobroker.net

- Übersetzungs-Tool (Web-based continuous localization)

1.12.10 translator.iobroker.in

- Übersetzungs-Tool (dev) - siehe *Adapter-Entwicklung*

1.12.11 adapter-check.iobroker.in

- Adapter-Checker (dev) - siehe *Adapter-Entwicklung*

1.12.12 iobroker.pro

- Cloud-Verbindung (IoT-Adapter) und Abonnements - siehe *IoT-Adapter*

1.12.13 iobroker.live

- Installations-Script
- Installations-Fixing-Script
- GitHub Badge (stable Version)
- GitHub Badge (Installationen)
- News für den Admin 5

1.12.14 repo.iobroker.live

- Repository stable - siehe *Repositories*
- Repository hash stable - siehe *Repositories*
- Repository beta - siehe *Repositories*
- Repository hash beta - siehe *Repositories*

1.12.15 iobroker.link

- Login

1.13 Repositories

Welche Adapter zur Verfügung stehen, wird in sogenannten Repositories hinterlegt. Die Liste an verfügbaren Repositories kann man selbst ändern, um zum Beispiel ein externes Repository zu nutzen. In den allermeisten Fällen wird dies aber niemand machen, sondern nur die Standard-Repositories nutzen.

Generell gibt es zwei verschiedene Adapter-Listen (Repositories), welche vom ioBroker-Team angeboten werden:

- `stable` (früher auch `default` genannt) - wird täglich aktualisiert und hier bereitgestellt: `http://download.iobroker.net/sources-dist.json`
- `beta` (früher auch `latest` genannt) - wird täglich aktualisiert und hier bereitgestellt: `http://download.iobroker.net/sources-dist-latest.json`

1.13.1 Pflege der Listen

Beide Listen werden in diesem GitHub Repository (ioBroker.repositories) gepflegt.

- `stable` = `sources-dist-stable.json`
- `beta` bzw. `latest` = `sources-dist.json`

Im `stable` werden getestete Adapter aufgenommen. Dort wird neben dem Repository auch eine genaue Version mit angegeben. Ein Eintrag sieht dort zum Beispiel so aus:

```
"admin": {
    "meta": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/io-
↪package.json",
    "icon": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/admin/
↪admin.png",
    "type": "general",
    "version": "5.1.25"
}
```

Wie Du siehst, ist vom Admin-Adapter in diesem Beispiel aktuell die Version `5.1.25` als stabil (`stable`) definiert.

Es kann gut sein, dass auf npm mittlerweile neue Versionen vergeben wurden und diese auch veröffentlicht ist. Diese Version bekommt man als Nutzer angeboten, wenn man das `beta` Repository wählt.

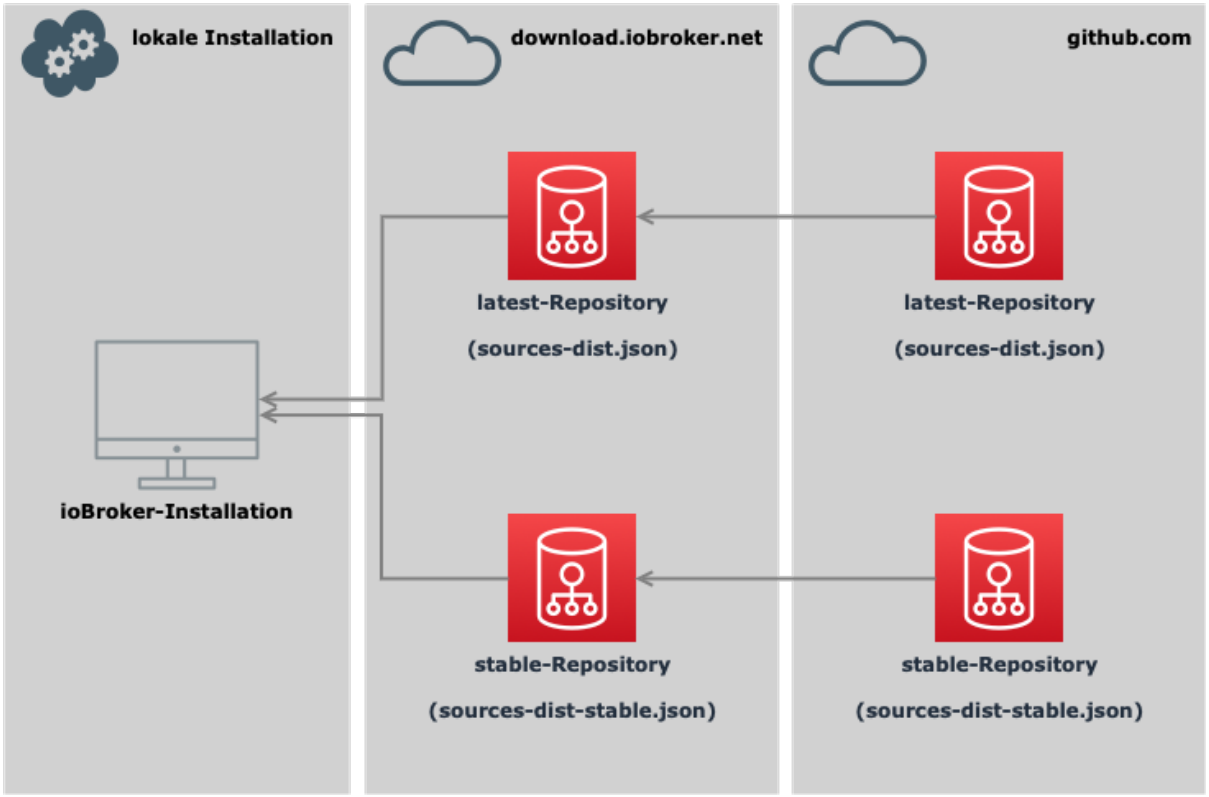
Im Gegensatz dazu hat der Eintrag im `beta` Repository keine definierte Versionsnummer:

```
"admin": {
    "meta": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/io-
↪package.json",
    "icon": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/admin/
↪admin.png",
    "type": "general"
}
```

Bei dem `beta` Repository wird automatisch immer die letzte freigegebene Version zum Update angeboten (von npm).

Dieses Vorgehen hat den Vorteil, dass man als Adapter-Entwickler genau steuern kann, welche Nutzer welche Version angeboten bekommen. So können neue Versionen zwar veröffentlicht werden, aber „`stable`-Nutzer“ werden erst später auf eine neue Version gebracht, wenn diese von vielen „`beta`-Nutzern“ bereits getestet wurde.

Bemerkung: Es kann vorkommen, dass einige Adapter zwar im beta-Repository vorhanden sind, aber noch nicht im stable-Repository zu finden sind (weil noch in Entwicklung bzw. noch keine stabile Version verfügbar ist)!



1.13.2 Bereitgestellte Daten

Die offiziellen Repositories werden regelmäßig aktualisiert und auf einem separaten Webserver bereitgestellt. Hier wird der Eintrag mit weiteren Informationen aus der *io-package.json* angereichert.

Unser Beispiel-Eintrag für den Admin-Adapter sieht dann wie folgt aus (zur Übersichtlichkeit wurden Übersetzungen in weitere Sprachen aus dem JSON gelöscht):

```
"admin": {
    "name": "admin",
    "version": "5.1.25",
    "titleLang": {
        "en": "Admin",
        "de": "Admin"
    },
    "title": "Admin",
    "connectionType": "local",
    "dataSource": "push",
    "news": {
        "5.1.25": {
            "en": "Corrected some errors reported via sentry and the github issues",
            "de": "Einige Fehler, die Ã¼ber Wache und die Github-Probleme gemeldet␣
↪wurden, korrigiert"
        },
        "5.1.24": {
            "en": "Corrected some errors reported via sentry and the github issues",
            "de": "Einige Fehler, die Ã¼ber Wache und die Github-Probleme gemeldet␣
↪wurden, korrigiert"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
        },
        "5.1.23": {
            "en": "Corrected some errors reported via sentry",
            "de": "Einige Fehler behoben, die Ã¼ber die Wache gemeldet wurden"
        },
        "5.1.22": {
            "en": "Corrected some errors reported via sentry",
            "de": "Einige Fehler behoben, die Ã¼ber die Wache gemeldet wurden"
        },
        "5.1.21": {
            "en": "Corrected some errors reported via sentry",
            "de": "Einige Fehler behoben, die Ã¼ber die Wache gemeldet wurden"
        },
        "5.1.20": {
            "en": "Corrected some errors reported via sentry",
            "de": "Einige Fehler behoben, die Ã¼ber die Wache gemeldet wurden"
        }
    },
    "desc": {
        "en": "The configuration of ioBroker via Web-Interface",
        "de": "Die Konfiguration von ioBroker Ã¼ber das Web-Interface"
    },
    "docs": {
        "en": "docs/en/admin.md",
        "ru": "docs/ru/admin.md",
        "de": [
            "docs/de/admin.md",
            "docs/de/admin/tab-adapters.md",
            "docs/de/admin/tab-instances.md",
            "docs/de/admin/tab-objects.md",
            "docs/de/admin/tab-states.md",
            "docs/de/admin/tab-groups.md",
            "docs/de/admin/tab-users.md",
            "docs/de/admin/tab-events.md",
            "docs/de/admin/tab-hosts.md",
            "docs/de/admin/tab-enums.md",
            "docs/de/admin/tab-log.md",
            "docs/de/admin/tab-system.md"
        ],
        "pt": "docs/pt/admin.md",
        "nl": "docs/nl/admin.md",
        "es": "docs/es/admin.md",
        "fr": "docs/fr/admin.md",
        "it": "docs/it/admin.md",
        "pl": "docs/pl/admin.md",
        "zh-cn": "docs/zh-cn/admin.md"
    },
    "materialize": true,
    "mode": "daemon",
    "platform": "Javascript/Node.js",
    "loglevel": "info",
    "icon": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/admin/
↪admin.png",
    "messagebox": true,
    "enabled": true,
    "extIcon": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/
↪admin/admin.png",
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
    "keywords": [
        "setup",
        "config",
        "update",
        "upgrade",
        "system",
        "konfiguration",
        "administration",
        "einrichtung",
        "wartung"
    ],
    "compact": true,
    "readme": "https://github.com/ioBroker/ioBroker.admin/blob/master/README.md",
    "authors": [
        "bluefox &lt;bluefox@ccu.io&gt;",
        "hobbyquaker &lt;hq@ccu.io&gt;"
    ],
    "dependencies": [
        {
            "js-controller": ">=3.2.16"
        }
    ],
    "type": "general",
    "license": "MIT",
    "logTransporter": true,
    "stopBeforeUpdate": true,
    "wwwDontUpload": true,
    "nogit": true,
    "welcomeScreenPro": {
        "link": "admin/index.html",
        "name": "Admin",
        "img": "admin/img/admin.png",
        "color": "pink",
        "order": 5,
        "localLinks": "_default",
        "localLink": true
    },
    "localLinks": {
        "_default": {
            "link": "%protocol%://%bind%:%port%",
            "pro": true
        }
    },
    "plugins": {
        "sentry": {
            "dsn": "https://9d2aaf29332a4999b133c693f43203b9@sentry.iobroker.net/18"
        }
    },
    "jsonConfig": true,
    "adminUI": {
        "config": "json"
    },
    "node": ">=10.0.0",
    "meta": "https://raw.githubusercontent.com/ioBroker/ioBroker.admin/master/io-
↪package.json",
    "published": "2014-12-04T18:45:44.907Z",
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
    "versionDate": "2021-08-15T12:14:58.829Z",
    "stars": 232,
    "stat": 49433,
    "issues": 118,
    "score": 1,
    "weekDownloads": 6687,
    "repoTime": "2021-10-05T02:19:59.616Z",
    "latestVersion": "5.1.25"
}
```


1.13.3 Einstellungen im ioBroker

Der ioBroker kann zwar mehrere Repositories verwalten (zum Beispiel über den Admin-Adapter), aber nur ein einzelnes Repository kann aktiv sein.

Das aktive Repository wird dabei im Objekt `system.config` im Attribut `common.activeRepo` hinterlegt. Siehe *System-Config*.

1.13.4 Update-Prozess

Das konfigurierte/aktive Repository wird regelmäßig geprüft. Dafür wird die jeweils angegebene URL geändert, sodass stattdessen eine Hash-Datei abgerufen wird.

```
urlOrPath = urlOrPath.replace(/\.json$/, '-hash.json');
```

So wird also z.B. statt `http://download.iobroker.net/sources-dist.json` erstmal `http://download.iobroker.net/sources-dist-hash.json` abgerufen. Aktuell hat die Datei folgenden Inhalt:


```
{
    "hash": "a3276c4275647354fa9f81748dde7941",
    "date": "2021-10-04T14:21:02.483Z",
    "name": "sources-dist.json"
}
```


Dieser Hash wird mit dem aktuellen Hash in `system.repositories` verglichen. Sollte der Hash abweichen, wird die eigentliche JSON-Datei geladen. Dies wurde so gelöst, um den Traffic von tausenden anfragenden Systemen zu reduzieren.

1.13.5 Links

- Repository

1.14 IoT-Adapter

Der IoT-Adapter stellt eine Schnittstelle zu den ioBroker-Servern bereit. Damit ist ein wichtiger Bestandteil, wenn man möglichst einfach eine Schnittstelle nach außen erstellen möchte. Diese ist zum Beispiel für Dienst wie Alexa, Google Home, IFTTT oder auch Yandex Alisa erforderlich.

Zusätzlich kann man durch die offene Schnittstelle z.B. auch Geofencing realisieren oder andere beliebige Status-Meldungen aus dem Netz an die lokale ioBroker-Installation durchreichen.

In jedem Fall ist ein Konto auf der Webseite iobroker.pro erforderlich. Dieses Kontodaten werden dann im installierten IoT-Adapter hinterlegt.

1.15 Adapter-Ratings

Seitdem der Admin-Adapter in Version 5 zur Verfügung steht, können die Anwender einzelne Adapter bewerten und auch Bewertungen von anderen Nutzern ansehen. Dabei sieht man in der Adapter-Liste direkt eine Sterne-Bewertung (1-5 Sterne). Mit einem Klick auf die Bewertung können Details abgefragt werden.

- Bewertungen pro Adapter
- In der Adapter-Liste sichtbar
- Details per Klick sichtbar (Kommentare von anderen Nutzern)
- Neue Bewertungen für installierte Adapter möglich

1.15.1 Neue Bewertung

POST-Request: `https://rating.iobroker.net/vote`

Payload:

```
{
    "uuid": "xxx",
    "adapter": "wled",
    "version": "0.6.3",
    "rating": 4,
    "comment": "Wirklich gut gemacht",
    "lang": "de"
}
```

Hier wird ebenfalls die *System-UUID* mitgeliefert.

Response:

```
{
    "adapter": "javascript",
    "rating": {
        "r": 4.51,
        "c": 21
    },
    "5.2.13": {
        "r": 5,
        "c": 1
    }
}
```

1.15.2 Auslesen

GET-Request: `https://rating.iobroker.net/adapter/wled?uuid=xxx` - siehe *System-UUID*

Response:

```
{
    "rating": {},
    "comments": [
        {
            "ts": "2021-08-29T18:14:26.000Z",
            "comment": "Perfekt, aber ich bekommen seit dieser Version immer die␣
↪Meldungen has to be one of type  string ,  number ,  boolean  but received type ␣
↪object",
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
            "version": "0.5.6",
            "uuid": false,
            "rating": 5,
            "lang": "de"
        }
    ],
    "adapter": "wled"
}
```


1.16 Sentry

Konfiguration in der *io-package.json* (`common.pugins.sentry`)

TODO

1.16.1 Links

- Plugin Repository

1.17 Statistics

Falls konfiguriert, sendet der ioBroker regelmäßig Nutzungsstatistiken an das ioBroker-Team. So kann ermittelt werden, welche Adapter besonders häufig im Einsatz sind, welche Sprache die Nutzer am meisten einstellen und aus welchem Land die Nutzer sind.

1.17.1 Modus

Welche Daten übermittelt werden, hängt vom jeweils konfigurierten Modus ab.

- normal (`normal`)
- ohne Stadt (`no-city`)
- erweitert (`extended`)

Hier ein Beispiel, was genau gesendet wird. Dies entspricht dem Modus „erweitert“ (`extended`):


```
{
    "uuid": "xxx",
    "language": "de",
    "country": "Germany",
    "hosts": [
        {
            "version": "3.3.18",
            "platform": "Javascript/Node.js",
            "type": "linux"
        }
    ],
    "node": "v12.22.6",
    "arch": "arm",
    "adapters": {
        "admin": {
            "version": "5.1.25",
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
            "platform": "Javascript/Node.js"
        },
        "discovery": {
            "version": "2.7.0",
            "platform": "Javascript/Node.js"
        },
        "backitup": {
            "version": "2.1.17",
            "platform": "Javascript/Node.js"
        },
        "feiertage": {
            "version": "1.0.17",
            "platform": "javascript/Node.js"
        }
    },
    "statesType": "file",
    "objectsType": "file",
    "model": "ARMv7 Processor rev 3 (v7l)",
    "cpus": 4,
    "mem": 4025200640,
    "ostype": "Linux",
    "city": "Custom City"
}
```


Weiterhin gibt es die Option „ohne Stadt“ (`no-city`), welche das letzte Attribut (city) nicht mitsendet.

Im Modus „normal“ (`normal`) werden weniger Informationen übertragen:


```
{
    "uuid": "xxx",
    "language": "de",
    "hosts": [
        {
            "version": "3.3.18",
            "platform": "Javascript/Node.js",
            "type": "linux"
        }
    ],
    "node": "v12.22.6",
    "arch": "arm",
    "adapters": {
        "admin": {
            "version": "5.1.25",
            "platform": "Javascript/Node.js"
        },
        "discovery": {
            "version": "2.7.0",
            "platform": "Javascript/Node.js"
        },
        "backitup": {
            "version": "2.1.17",
            "platform": "Javascript/Node.js"
        },
        "feiertage": {
            "version": "1.0.17",
            "platform": "javascript/Node.js"
        }
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    },
    "statesType": "file",
    "objectsType": "file"
}
```

Es wird darum gebeten, den erweiterten Modus zu aktivieren.

Es wird in allen Fällen die *System-UUID* der Installation mit übermittelt. Außerdem wird gesendet, wie Du deine Daten speicherst. Siehe auch *Daten-Speicherung*.

1.17.2 Backend

Die Daten werden vom `js-controller` an *http://download.iobroker.net/diag.php* gesendet (POST-Request mit JSON-Payload an `data`).

```
curl -v -X POST -d 'data={"uuid": "xxx","language": "de","hosts": [{"version": "3.3.18
↪","platform": "Javascript/Node.js","type": "linux"}],"node": "v12.22.6","arch": "arm
↪","adapters": {"admin": {"version": "5.1.25","platform": "Javascript/Node.js"},
↪"discovery": {"version": "2.7.0","platform": "Javascript/Node.js"},"backitup": {
↪"version": "2.1.17","platform": "Javascript/Node.js"},"feiertage": {"version": "1.0.
↪17","platform": "javascript/Node.js"}},"statesType": "file","objectsType": "file"}'␣
↪http://download.iobroker.net/diag.php
```

1.18 Adapter-Entwicklung

In diesem Abschnitt fasse ich für Dich zusammen, was es bei der Entwicklung von neuen Adaptern zu beachten gibt. Auf dem Weg zum eigenen/neuen Adapter gibt es jede Menge Werkzeuge, welche Dir das Leben leichter machen und Dich sehr schnell zum Ziel bringen werden!

1.18.1 Neuer Adapter

Bevor Du einen neuen Adapter entwickelst, schau am besten nach, ob dieser schon existiert oder angefangen wurde.

Offene Anfragen für Adapter findest Du hier.

Wenn Du einen neuen Adapter entwickeln möchtest, empfiehlt es sich die Erstellung mit Hilfe des Create-Adapter-Tools durchzuführen. Dieses Tool erstellt Dir auf Basis eines Templates ein neues Projekt, mit welchem Du direkt durchstarten kannst.

Das Tool findest Du auf GitHub.

Um einen neuen Adapter zu erstellen, führst Du folgenden Befehl aus:

```
npx @iobroker/create-adapter
```

Nachdem das Programm ausgeführt wird, werden Dir einige Fragen gestellt, wie Du gerne arbeiten möchtest. Das Ergebnis sieht beispielsweise so aus:

```
npx: Installierte 147 in 31.103s

=====================================================
Welcome to the ioBroker adapter creator v1.32.0!
=====================================================
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
You can cancel at any point by pressing Ctrl+C.

Let's get started with a few questions about your project!
✓ Please enter the name of your project: · documentation
✓ Which title should be shown in the admin UI? · Documentation
✓ Please enter a short description: · An example adapter for the ioBroker␣
↪documentation
✓ Enter some keywords (separated by commas) to describe your project: ·␣
↪documentation,learn,development
✓ If you have any contributors, please enter their names (seperated by commas): ·

Nice! Let's get technical...
✓ How detailed do you want to configure your project? · yes
✓ Which features should your project contain? · adapter
✓ Which additional features should be available in the admin? · No items were␣
↪selected
✓ Which category does your adapter fall into? · storage
✓ When should the adapter be started? · daemon
✓ From where will the adapter get its data? · local
✓ How will the adapter receive its data? · push
✓ Do you want to indicate the connection state? · yes
✓ Which language do you want to use to code the adapter? · JavaScript
✓ Use React for the Admin UI? · yes
✓ Which of the following tools do you want to use? · ESLint, type checking
✓ Do you prefer tab or space indentation? · Space (4)
✓ Do you prefer double or single quotes? · single
✓ How should the main adapter file be structured? · yes

Almost done! Just a few administrative details...
✓ Please enter your name (or nickname): · Matthias Kleine
✓ What's your name/org on GitHub? · klein0r
✓ What's your email address? · info@haus-automatisierung.com
✓ Which protocol should be used for the repo URL? · SSH
✓ Initialize the GitHub repo automatically? · yes
✓ Which license should be used for your project? · MIT License
✓ Which continuous integration service should be used? · gh-actions
✓ Do you want to receive regular dependency updates through Pull Requests? · yes
```

Danach werden automatisch alle nötigen Dateien erstellt und Du kannst direkt mit der Entwicklung starten!

Natürlich könntest Du auch alle Dateien manuell anlegen - aber das ist nicht zu empfehlen und bedeutet viel mehr Arbeit!

1.18.2 GitHub Repository

Tipp: Ich würde generell empfehlen, den Quellcode für den Adapter auf GitHub zu veröffentlichen. Natürlich würden andere Plattformen wie Amazon Code Commit oder Bitbucket genauso funktionieren, allerdings arbeitet der Großteil der ioBroker-Community mit GitHub.

Wichtig ist, dass Du den Namenskonventionen für ein neues Repository folgst. Das Repository heißt dabei `ioBroker.<deinadapter>`. **Achte auf Groß- und Kleinschreibung!** Das B von ioBroker wird im Repository-Namen groß geschrieben! Der komplette Rest wird klein geschrieben!

Gültige Namen für Dein neues **Repository** wären also zum Beispiel:

- `ioBroker.admin`

- `ioBroker.javascript`
- `ioBroker.luftdaten`
- `ioBroker.octoprint`

Warnung: Achte darauf, dass der von Dir gewählte Name für einen Adapter noch nicht vergeben ist! Die oben genannten Beispiele sind alle schon vorhanden. Ansonsten kannst Du deinen Adapter später nicht veröffentlichen / in die Adapter-Liste mit aufnehmen.

Beschäftige Dich also auf jeden Fall mit diesen Themen:

- `git commit`
- `git push`
- Remote repositories
- Branches
- Tags
- SSH Key Authentication

1.18.3 Übersetzungen

Generell ist es sinnvoll, direkt von Anfang an deinen neuen Adapter in mehrere Sprachen zu übersetzen. Die „Basis-Sprache“ sollte Englisch sein. Von dort wird in andere Sprachen übersetzt. Damit Du das nicht manuell machen musst, gibt es vom ioBroker-Team ein Tool, welches Dir einen Englischen Text in andere Sprachen übersetzt und im richtigen Format für den ioBroker zurückliefert.

Zum ioBroker Translator.

Gibst Du dort zum Beispiel `today` ein, liefert Dir das Programm folgende Übersetzungen im JSON-Format:

```
{
    "today": {
        "en": "today",
        "de": "heute",
        "ru": "",
        "pt": "hoje",
        "nl": "vandaag",
        "fr": "aujourd'hui",
        "it": "oggi",
        "es": "hoy dia",
        "pl": "dzisiaj",
        "zh-cn": ""
    }
}
```

Diese Informationen kannst Du direkt in deinem Adapter verwenden. Achte darauf, dass alle Texte übersetzt sind.

Bemerkung: Natürlich ist es so, dass (wie üblich) die erstellen Übersetzungen nicht immer einwandfrei sind. Häufig ist z.B. die Deutsche Übersetzung einfachf falsch oder ergibt keinen Sinn. Kontrolliere noch einmal manuell, ob die Texte korrekt sind. Je mehr Sprachen, desto besser!

Alle Texte **müssen** in die folgenden Sprachen übersetzt werden:

- Englisch (en)
- Deutsch (de)

Alle Text **sollten** zusätzlich auch diese Sprachen übersetzt werden:

- Russisch (ru)
- Portugisisch (pt)
- Niederländisch (nl)
- Französisch (fr)
- Italienisch (it)
- Spanisch (es)
- Polnisch (pl)
- Chinesisch (zh-cn)

1.18.4 npm

Sobald es einen Release deines Adapters gibt, solltest Du eine Versionsnummer vergeben. Achte dabei auf semantische Versionierung!

Die erste Version deines Adapters wird also höchstwahrscheinlich die `0.0.1` sein.

Generell werden nodejs-Pakete über `npm` veröffentlicht. Dieser Paketmanager kümmert sich um deine Abhängigkeiten im Projekt und von dort werden auch die Pakete bei der Installation des Adapters geladen.

Tipp: Es gibt im Adapter-Creator-Tool (siehe oben) verschiedene Scripts, welche Dir automatisch beim Erstellen eines neuen Releases das Paket auf npm.js veröffentlichen. Dafür musst Du ein Token erstellen, welches im GitHub-Repository hinterlegt wird.

Beschäftige Dich also auf jeden Fall mit diesen Themen:

- semantische Versionierung
- npmjs.org
- `package.json`
- publish von neuen npm Paketen

Bemerkung: Generell haben GitHub und npmjs erstmal nichts miteinadner zu tun. Das sind zwei unterschiedliche Plattformen. GitHub hilft Dir bei der Entwicklung und Issue-Tracking, während npm das fertige Pakete vorhält und an die Nutzer ausliefert. Über diverse Integrationsmöglichen greifen diese beiden Plattformen aber ineinander und vereinfachen den Workflow.

Der Name deines Paketes für npm unterscheidet sich dabei vom Namen des Repository! Hier wird das „B“ in ioBroker nicht mehr groß geschrieben! Der Paket-Name enthält also nur Kleinbuchstaben.

Gültige Namen für Dein neues **npm Paket** wären also zum Beispiel:

- `iobroker.admin`
- `iobroker.javascript`
- `iobroker.luftdaten`
- `iobroker.octoprint`

Solltest Du den Adapter mit dem oben genannten Tool erstellt haben, wird dies bereits automatisch berücksichtigt!

1.18.5 Adapter prüfen

Für einen Adapter gibt es eine Liste an Regeln, an welche Du Dich halten solltest. Entspricht Dein Adapter nicht diesen Anforderungen, wird er nicht in die offizielle Liste der verfügbaren Adapter aufgenommen!

Diese Regeln einzuhalten ist relativ einfach, da Dir der `ioBroker Adapter Checker` genau sagt, was noch getan werden muss bzw. falsch läuft.

Sobald Du also eine erste Version von deinem Adapter fertig hast, Du alles in GitHub-Repository gepusht hast und Dein Paket auf npmjs veröffentlich wurde, kannst Du den Adapter-Checker starten:

Zum ioBroker Adapter-Checker.

Dort fügst Du die URL von deinem GitHub-Repository ein.

Wichtig ist, dass alle Haken grün sind.

Tipp: Prüfe schon während der Entwicklung regelmäßig, ob dein Adapter den Anforderungen entspricht.

Generell gilt, dass auch hier die Entwicklung weiter geht. Es werden mehr Prüfungen hinzugefügt oder andere entfernt. Wenn dein Adapter also heute alle Tests besteht, muss das bei der nächsten Version nicht mehr unbedingt so sein.

Das Repository vom Adapter-Checker kann mit neuen Regeln erweitert werden (siehe `index.js`).

1.18.6 Adapter veröffentlichen

Möchtest Du deinen Adapter nun anderen zur Verfügung stellen, solltest Du diesen erst von erfahrenen Nutzern testen lassen. Erstelle dazu einen neuen Foren-Beitrag mit der Bitte um einen Test.

Danach kannst Du einen Pull-Request im GitHub Repository (ioBroker.repositories) erstellen, indem Du Deinen Adapter dort hinzufügst. Mehr Details hier: *Repositories*

Bemerkung: Bitte beachte, dass Adapter abgelehnt werden, wenn nicht alle Adapter-Checks (siehe oben) erfüllt sind.

1.18.7 Links

- Create-Adapter
- Adapter-Checker
- Release-Script von AlCalzone
- Adapter-Examples

1.19 Objekte

Bemerkung: Lies zuerst die Grundlagen zur *Daten-Speicherung*.

Objekte können beispielsweise über das *CLI (Command Line Interface)* ausgelesen werden.

`_id`
: Eindeutige ID

  Type string

`type`
Typ des Objektes. Gültige Werte sind:

- `state` - Zustand. Das übergeornete Objekte sollte vom Typ `channel`, `device`, `instance` oder `host` sein. Siehe *States*
- `channel` - „Kanal“ um mehrere Zustände darunter zu strukturieren. Das übergeornete Objekte sollte vom Typ `device` sein.
- `device` - „Gerät“ um mehrere Zustände oder Kanäle darunter zu strukturieren. Das übergeornete Objekte sollte vom Typ `instance` sein.
- `enum` - „Liste“ mit vordefinierten Werten in `common.members`. that points to the states, channels, devices or files.
- `host` - Ein „Host“, welcher einen `js-controller` Prozess ausführt. Beispielsweise `system.host.raspberrypi-iobroker`.
- `adapter` - Die Standard-Konfiguration von einem Adaper. Beispielsweise `system.adapter.admin`.
- `instance` - Die Konfiguration der einzelnen Instanz. Beispielsweise `system.adapter.admin.0`. Das übergeornete Objekte sollte vom Typ `adapter` sein.
- `meta` - Sich selten ändernde Meta-Informationen wie zum Beispiel die *System-UUID* unter `system.meta.uuid`.
- `config` - Konfigurationen. Beispielsweise `system.config` oder `system.repositories`
- `script` - Skripte unter `script.js.*`
- `user` - Benutzer des Systems. Beispielsweise `system.user.admin`
- `group` - Benutzer-Gruppen des Sytems. Beispielsweise `system.group.administrator`
- `chart` - Diagramm
- `folder` - Verzeichnis. Beispielsweise `system.host.raspberrypi-iobroker.notifications`

Type string

`common.type`
Typ der gespeicherten Daten

- `mixed` - Kann einen beliebigen Wert annehmen (nicht empfohlen)
- `number` - Numerische Werte
- `string` - Zeichenketten
- `boolean` - true / false
- `array` - Liste von Werten
- `object` - Objekt
- `json` - ???
- `file` - ???
- `multistate` - Auswahlmöglichkeiten (Enum)

Eine Ausnahme bilden die Objekte mit `type` = `meta`. Diese können hier noch den Type `meta.user` oder `meta.folder` bekommen.

> **Warnung:** Falls der Typ `array`, `object` oder `mixed` lautet, muss der Wert als String mit `JSON.stringify()` gespeichert werden.

Type string

Default `mixed`

`common.role`
Rolle des zugehörigen State (`type` = `state`), welche festlegt, wie der Wert im Frontend (Admin) dargestellt werden soll.

Type string

`common.read`
Legt fest, ob der zugehörige State (`type` = `state`) gelesen werden darf. Siehe *States*

Type boolean

`common.write`
Legt fest, ob der zugehörige State (`type` = `state`) geschrieben werden darf. Siehe *States*

Type boolean

`common.name`
(optional) Name des Objektes - wird im Frontend (wie dem Admin) dargestellt. **Es ist empfohlen, diesen Wert zu setzen!**

Type string

`common.custom`
(optional) Zusatzkonfiguration weiterer Adapter für das Objekt. Wird zum Beispiel für Datenbank-Adapter genutzt. Je Eintrag ist das `enabled` Attribut erforderlich.

```
"custom": {
    "influxdb.0": {
        "enabled": true,
        "storageType": "",
        "aliasId": "",
        "changesOnly": true,
        "debounce": "1000",
        "changesRelogInterval": "0",
        "changesMinDelta": "0"
    },
    "history.0": {
        "enabled": true,
        "aliasId": "",
        "changesOnly": true,
        "debounce": 1000,
        "changesRelogInterval": 0,
        "changesMinDelta": 0,
        "maxLength": 960,
        "retention": 31536000
    }
}
```

Type object

`native`
Eigenschaften des Zielsystems (z.B. eine ID eines Gerätes)

Type object

- ``common.min (optional)
- ``common.max (optional)
- ``common.step (optional) - increase/decrease interval. E.g. 0.5 for thermostat
- ``common.unit (optional)

- ``common.def (optional - the default value)
- ``common.defAck (optional - if common.def is set this value is used as ack flag, js-controller 2.0.0+)
- ``common.desc (optional, string or object) - description, object for multilingual description
- ``common.states (optional) attribute of type number with the object of possible states {‚value': ‚valueName', ‚value2': ‚valueName2', 0: ‚OFF', 1: ‚ON'} or (supported up from admin5) an states array, like [‚Start', ‚Flight', ‚Land']
- ``common.workingID (string, optional) - if this state has helper state WORKING. Here must be written the full name or just the last part if the first parts are the same with actual. Used for HM.LEVEL and normally has value „WORKING"

1.19.1 Typ Config (Beispiel)

```
iobroker object get system.config
```

Beispiel-Ausgabe:


```
{
    "_id": "system.config",
    "type": "config",
    "common": {
        "name": {
            "en": "System configuration",
            "de": "Systemkonfiguration",
            "ru": " ",
            "pt": "Configuração do sistema",
            "nl": "Systeem configuratie",
            "fr": "Configuration du système",
            "it": "Configurazione di sistema",
            "es": "Configuración del sistema",
            "pl": "Konfiguracja systemu",
            "zh-cn": ""
        },
        "city": "Custom City",
        "country": "Germany",
        "longitude": 8.111,
        "latitude": 51.111,
        "language": "de",
        "tempUnit": "°C",
        "currency": "€",
        "dontDelete": true,
        "dateFormat": "DD.MM.YYYY",
        "isFloatComma": true,
        "licenseConfirmed": true,
        "defaultHistory": "",
        "expertMode": false,
        "defaultLogLevel": "info",
        "activeRepo": "stable",
        "diag": "extended",
        "tabs": [
            "tab-intro",
            "tab-info",
            "tab-adapters",
            "tab-instances",
            "tab-objects",
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
            "tab-log",
            "tab-scenes",
            "tab-javascript",
            "tab-text2command-0",
            "tab-node-red-0"
        ],
        "tabsVisible": [
            {
                "name": "tab-intro",
                "visible": true
            },
            {
                "name": "tab-adapters",
                "visible": true
            },
            {
                "name": "tab-instances",
                "visible": true
            },
            {
                "name": "tab-objects",
                "visible": true
            },
            {
                "name": "tab-enums",
                "visible": true
            },
            {
                "name": "tab-logs",
                "visible": true
            },
            {
                "name": "tab-users",
                "visible": true
            },
            {
                "name": "tab-hosts",
                "visible": true
            },
            {
                "name": "tab-files",
                "visible": true
            },
            {
                "name": "tab-backitup-0",
                "visible": true
            }
        ],
        "defaultNewAcl": {
            "object": 1636,
            "state": 1636,
            "file": 1632,
            "owner": "system.user.admin",
            "ownerGroup": "system.group.administrator"
        }
    },
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
    "acl": {
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator",
        "object": 1604
    },
    "native": {
        "secret": "971640e8df0885faf7d49c90e38423fc65425b2b861d5e7b"
    },
    "from": "system.adapter.admin.0",
    "user": "system.user.admin",
    "ts": 1633096344214
}
```


1.19.2 Typ State (Beispiel)

```
iobroker object get hue.0.Deckenlampe.bri
```


```
{
    "_id": "hue.0.Deckenlampe.bri",
    "type": "state",
    "common": {
        "name": "Deckenlampe.bri",
        "read": true,
        "write": true,
        "type": "number",
        "role": "level.dimmer",
        "min": 0,
        "max": 254,
        "def": 254
    },
    "native": {
        "id": "3"
    },
    "from": "system.adapter.hue.0",
    "user": "system.user.admin",
    "ts": 1604080553077,
    "acl": {
        "object": 1636,
        "state": 1636,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    }
}
```

1.19.3 Typ Host (Beispiel)

```
iobroker object get system.host.raspberrypi-iobroker
```


```
{
    "_id": "system.host.raspberrypi-iobroker",
    "type": "host",
    "common": {
        "name": "raspberrypi-iobroker",
        "title": "JS controller",
        "installedVersion": "3.3.18",
        "platform": "Javascript/Node.js",
        "cmd": "/usr/bin/node  /opt/iobroker/node_modules/iobroker.js-controller/
↪controller.js",
        "hostname": "raspberrypi-iobroker",
        "address": [
            "172.16.0.120",
            "fe80::46f4:a0bb:45c7:6fd7"
        ],
        "type": "js-controller"
    },
    "native": {
        "process": {
            "title": "iobroker.js-controller",
            "versions": {
                "node": "12.22.6",
                "v8": "7.8.279.23-node.56",
                "uv": "1.40.0",
                "zlib": "1.2.11",
                "brotli": "1.0.9",
                "ares": "1.17.2",
                "modules": "72",
                "nghttp2": "1.41.0",
                "napi": "8",
                "llhttp": "2.1.3",
                "http_parser": "2.9.4",
                "openssl": "1.1.1l",
                "cldr": "37.0",
                "icu": "67.1",
                "tz": "2019c",
                "unicode": "13.0"
            },
            "env": {
                "NODE": "$(which node)",
                "PWD": "/",
                "LOGNAME": "iobroker",
                "HOME": "/home/iobroker",
                "LANG": "de_DE.UTF-8",
                "INVOCATION_ID": "82481d3eabae4b618e7be1b24552c984",
                "USER": "iobroker",
                "SHLVL": "0",
                "JOURNAL_STREAM": "8:21058",
                "PATH": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
↪",
                "_": "/usr/bin/node"
            }
        },
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
        "os": {
            "hostname": "raspberrypi-iobroker",
            "type": "Linux",
            "platform": "linux",
            "arch": "arm",
            "release": "5.10.63-v7l+",
            "endianness": "LE",
            "tmpdir": "/tmp"
        },
        "hardware": {
            "cpus": [
                {
                    "model": "ARMv7 Processor rev 3 (v7l)",
                    "speed": 1500
                },
                {
                    "model": "ARMv7 Processor rev 3 (v7l)",
                    "speed": 1500
                },
                {
                    "model": "ARMv7 Processor rev 3 (v7l)",
                    "speed": 1500
                },
                {
                    "model": "ARMv7 Processor rev 3 (v7l)",
                    "speed": 1500
                }
            ],
            "totalmem": 4025200640,
            "networkInterfaces": {
                "lo": [
                    {
                        "address": "127.0.0.1",
                        "netmask": "255.0.0.0",
                        "family": "IPv4",
                        "mac": "00:00:00:00:00:00",
                        "internal": true,
                        "cidr": "127.0.0.1/8"
                    },
                    {
                        "address": "::1",
                        "netmask": "ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff",
                        "family": "IPv6",
                        "mac": "00:00:00:00:00:00",
                        "internal": true,
                        "cidr": "::1/128",
                        "scopeid": 0
                    }
                ],
                "eth0": [
                    {
                        "address": "172.16.0.120",
                        "netmask": "255.255.0.0",
                        "family": "IPv4",
                        "mac": "e4:5f:01:5d:01:31",
                        "internal": false,
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
                        "cidr": "172.16.0.120/16"
                    },
                    {
                        "address": "fe80::46f4:a0bb:45c7:6fd7",
                        "netmask": "ffff:ffff:ffff:ffff::",
                        "family": "IPv6",
                        "mac": "e4:5f:01:5d:01:31",
                        "internal": false,
                        "cidr": "fe80::46f4:a0bb:45c7:6fd7/64",
                        "scopeid": 2
                    }
                ]
            }
        }
    },
    "from": "system.host.raspberrypi-iobroker",
    "ts": 1633374149865,
    "acl": {
        "object": 1636,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    }
}
```


1.19.4 Typ Script (Beispiel)

```
iobroker object get script.js.Büro.Licht_einschalten
```


```
{
    "common": {
        "name": "Licht einschalten",
        "expert": true,
        "engineType": "Blockly",
        "engine": "system.adapter.javascript.0",
        "source": "on({id: \"zigbee.0.00158d00020f4ab5.click\"...",
        "debug": false,
        "verbose": false,
        "enabled": true
    },
    "type": "script",
    "from": "system.adapter.admin.0",
    "user": "system.user.admin",
    "ts": 1628941638315,
    "_id": "script.js.Büro.Licht_einschalten",
    "acl": {
        "object": 1636,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    }
}
```

1.19.5 Typ User (Beispiel)

```
iobroker object get system.user.admin
```


```
{
    "type": "user",
    "common": {
        "name": "Matthias Kleine",
        "password": "pbkdf2$10000$021943a847a4e2c20b...",
        "dontDelete": true,
        "enabled": true
    },
    "native": {},
    "_id": "system.user.admin",
    "acl": {
        "object": 1636,
        "state": 1636,
        "file": 1632,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    },
    "enums": {},
    "from": "system.adapter.admin.0",
    "user": "system.user.admin",
    "ts": 1633095538813
}
```


1.19.6 Typ Group (Beispiel)

```
iobroker object get system.group.administrator
```


```
{
    "_id": "system.group.administrator",
    "type": "group",
    "common": {
        "icon": "...",
        "name": {
            "en": "Administrator",
            "de": "Administrator"
        },
        "description": {
            "en": "Can do everything with System",
            "de": "Darf alles mit dem System machen"
        },
        "members": [
            "system.user.admin"
        ],
        "dontDelete": true,
        "acl": {
            "object": {
                "list": true,
                "read": true,
                "write": true,
                "delete": true
            },
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
            "state": {
                "list": true,
                "read": true,
                "write": true,
                "create": true,
                "delete": true
            },
            "users": {
                "list": true,
                "read": true,
                "write": true,
                "create": true,
                "delete": true
            },
            "other": {
                "execute": true,
                "http": true,
                "sendto": true
            },
            "file": {
                "list": true,
                "read": true,
                "write": true,
                "create": true,
                "delete": true
            }
        }
    },
    "acl": {
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator",
        "object": 1604
    },
    "from": "system.host.raspberrypi-iobroker.cli",
    "ts": 1633092016342
}
```


1.19.7 Typ Folder (Beispiel)

```
iobroker object get system.host.raspberrypi-iobroker.notifications
```


```
{
    "type": "folder",
    "common": {
        "name": {
            "en": "Notifications",
            "de": "Benachrichtigungen"
        }
    },
    "native": {},
    "_id": "system.host.raspberrypi-iobroker.notifications",
    "acl": {
        "object": 1636,
        "state": 1636,
```


(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
        "file": 1632,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    }
}
```


1.19.8 Typ Meta (Beispiel)

```
iobroker object get system.meta.uuid
```


```
{
    "type": "meta",
    "common": {
        "name": "uuid",
        "type": "uuid"
    },
    "ts": 1633092016485,
    "from": "system.host.raspberrypi-iobroker.tools",
    "native": {
        "uuid": "23b1992b-8d91-a4fc-b201-2bd851bdc807"
    },
    "_id": "system.meta.uuid",
    "acl": {
        "object": 1636,
        "state": 1636,
        "file": 1632,
        "owner": "system.user.admin",
        "ownerGroup": "system.group.administrator"
    }
}
```


1.20 States

Bemerkung: Lies zuerst die Grundlagen zur *Daten-Speicherung*.

States können beispielsweise über das *CLI (Command Line Interface)* ausgelesen werden:

```
iobroker state get admin.0.info.updatesNumber
```

1.20.1 Beispiel


```
{
    "val": 0,
    "ack": true,
    "ts": 1633428163294,
    "lc": 1633092122629
    "q": 0,
    "from": "system.adapter.admin.0",
    "user": "system.user.admin"
}
```

1.20.2 Eigenschaften

val
Der aktuell gespeicherte Wert

Type string

ack
Bestätigt-Flag, ob der (neue) Wert vom Adapter bzw. Ziel-System akzeptiert wurde

Type boolean

ts
Unix Timestamp (Zeitstempel in Millisekunden) wann der Zustand zuletzt **aktualisiert** wurde

Type number

lc
Unix Zimestamp (Zeitstempel in Millisekunden) wann der Zustand zuletzt **geändert** wurde (last change)

Type number

q
Qualität

```
0x00 - 00000000 - good (can be undefined or null)
0x01 - 00000001 - general bad, general problem
0x02 - 00000010 - no connection problem

0x10 - 00010000 - substitute value from controller
0x20 - 00100000 - substitute initial value
0x40 - 01000000 - substitute value from device or instance
0x80 - 10000000 - substitute value from sensor

0x11 - 01000001 - general problem by instance
0x41 - 01000001 - general problem by device
0x81 - 10000001 - general problem by sensor

0x12 - 00010010 - instance not connected
0x42 - 01000010 - device not connected
0x82 - 10000010 - sensor not connected

0x44 - 01000100 - device reports error
0x84 - 10000100 - sensor reports error
```

Type number

from
Instanz, welche die Änderung durchgeführt hat (z.B. `system.adapter.admin.0`) (optional)

Type string

user
Benutzer, welcher die Änderung durchgeführt hat (z.B. `system.user.admin`) (optional)

Type string

c
Kommentar (optional)

Type string

expire
: Zeit in Sekunden, wann der Wert auf null gesetzt wird (optional) In diesem Beispiel ist der Wert `0` (numerisch Null).

  Type number

Wenn man nur den Wert (also die Eigenschaft `val`) auslesen möchte, geht das per *CLI (Command Line Interface)* wie folgt:

```
iobroker state getvalue admin.0.info.updatesNumber
```

1.21 io-package.json

Jeder Adapter enthält neben der `package.json` für npm noch eine `io-package.json`. Hier werden sämtliche Meta-Informationen für den Adpater hinterlegt.

1.21.1 Beispiel

Hier eine Beispiel-Datei aus dem Luftdaten-Adapter. Eine Beschreibung der einzelnen Eigenschaften folgt weiter unten.

```
{
    "common": {
        "name": "luftdaten",
        "version": "2.1.1",
        "news": {
            "2.0.3": {
                "en": "Fixed error logging",
                "de": "Fehler-Logging behoben"
            },
            "2.0.2": {
                "en": "Added timeout option",
                "de": "Option für Timeout-Limit hinzugefügt"
            },
            "2.0.1": {
                "en": "Minor bug fixes",
                "de": "Kleinere Bugfixes"
            },
            "2.0.0": {
                "en": "Updated admin interface to maintain multiple sensors in one␣
↪instance",
                "de": "Benutzeroberfläche angepasst, um mehrere Sensoren in einer␣
↪Instanz verwalten zu können"
            }
        },
        "title": "Luftdaten.info",
        "titleLang": {
            "en": "Luftdaten.info",
            "de": "Luftdaten.info"
        },
        "desc": {
            "en": "Loads current air quality data from a local or remote sensor",
            "de": "Lädt aktuelle Luftqualitätsdaten eines lokalen oder Cloud-Sensors"
        },
        "authors": [
            "Matthias Kleine <info@haus-automatisierung.com>"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)


```
        ],
        "keywords": [
            "web",
            "weather",
            "air",
            "quality"
        ],
        "license": "MIT",
        "platform": "Javascript/Node.js",
        "icon": "luftdaten.png",
        "extIcon": "https://raw.githubusercontent.com/klein0r/ioBroker.luftdaten/
↪master/admin/luftdaten.png",
        "enabled": true,
        "readme": "https://github.com/klein0r/ioBroker.luftdaten/blob/master/README.md
↪",
        "loglevel": "info",
        "mode": "schedule",
        "allowInit": true,
        "schedule": "*/30 * * * *",
        "type": "weather",
        "compact": true,
        "connectionType": "cloud",
        "dataSource": "poll",
        "adminUI": {
            "config": "json"
        },
        "dependencies": [
            {
                "js-controller": ">=3.3.0"
            }
        ],
        "globalDependencies": [
            {
                "admin": ">=5.1.19"
            }
        ],
        "plugins": {
            "sentry": {
                "dsn": "https://baf35e4e423d409bbec94cb01b55257e@sentry.iobroker.net/
↪103"
            }
        }
    },
    "native": {
        "requestTimeout": 10
    },
    "objects": [
    ]
}
```

1.21.2 Eigenschaften (erforderlich)

common.name
Name des Adapters (darf nicht „ioBroker“ enthalten)

Type string

common.version
Aktuelle Version des Adapters (muss mit der Version der *package.json* übereinstimmen)

Type string

common.platform
Die Plattform, auf welcher der Adapter programmiert wurde

Type string

Default `Javascript/Node.js`

common.titleLang
Titel des Adapters (übersetzt in mehrere Sprachen)

```
"titleLang": {
    "en": "Luftdaten.info",
    "de": "Luftdaten.info"
}
```

Type object

common.desc
Beschreibung, was der Adapter machen soll (übersetzt in mehrere Sprachen)

```
"desc": {
    "en": "Loads current air quality data from a local or remote sensor",
    "de": "Lädt aktuelle Luftqualitätsdaten eines lokalen oder Cloud-Sensors"
}
```

Type object

common.news
Liste mit Infos zu den verschiedenen Versionen (Updatehistorie / Changelog)

```
"news": {
    "2.0.3": {
        "en": "Fixed error logging",
        "de": "Fehler-Logging behoben"
    },
    "2.0.2": {
        "en": "Added timeout option",
        "de": "Option für Timeout-Limit hinzugefügt"
    },
    "2.0.1": {
        "en": "Minor bug fixes",
        "de": "Kleinere Bugfixes"
    },
    "2.0.0": {
        "en": "Updated admin interface to maintain multiple sensors in one␣
↪instance",
        "de": "Benutzeroberfläche angepasst, um mehrere Sensoren in einer␣
↪Instanz verwalten zu können"
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
    }
}
```

Type object

common.mode
Modus des Adapters

- `none` - Der Adapter wird nicht gestartet
- `daemon` - Separat laufender Prozess
- `subscribe` - Wird gestartet, wenn der State `system.adapter.<adapter-name>.<instanz-nummmer>.alive` auf `true` gesetzt wird. Wird automatisch beendet, wenn der State auf `false` geändert wird. Der State wird automatisch auf `false` gesetzt, wenn der Prozess beendet wurde.
- `schedule` - Wird nach dem in `common.schedule` festgelegten Zeitplan automatisch gestartet
- `once` - Wird jedes Mal automatisch gestartet, wenn das `system.adater.<adapter-name>.<instanz-nummmer>`-Objekt geändert wird
- `extension` - ???

Type string

1.21.3 Eigenschaften (Allgemein)

common.enabled
Legt fest, ob die Instanz gestartet werden soll, oder nicht

Type boolean

Default `true`

common.tier
Legt fest, in welcher Reihenfolge die Adapter gestartet werden

- `1` - Logik
- `2` - API und andere Daten
- `3` - alle anderen

Type number

Default `3`

common.messagebox
`true` wenn Nachrichten per `sendTo()` an den Adapter erlaubt sind. Siehe *Messagebox*

Type boolean

Default `false`

common.readme
URL zur Readme-Datei (z.B. HTTP-URL zur README.md auf GitHub)

Type string

common.docs

Type object

common.authors
Liste mit Entwicklern des Adapters

```
"authors": [
    "Matthias Kleine <info@haus-automatisierung.com>"
]
```

Type array

`common.license`
Lizenz des Adapters (z.B. MIT). Gültige Werte sind im Schema zu finden (Link siehe unten)

Type string

`common.type`
Typ/Kategorie des Adapters

- `alarm` - Sicherheitssysteme, Alarmanlagen, ...
- `climate-control` - Klimasteuerung, Heizung, Luftfilter, ...
- `communication` - Kommunikation mit anderen Adaptern (REST Api)
- `date-and-time` - Kalender, Ferien, Feiertage, ...
- `energy` - PV-Anlage, Verbrauchsdaten, ...
- `metering` - Energiemessung
- `garden` - Rasenmähroboter, Bewässerung, ...
- `general` - Allgemeine Adapter wie Admin
- `geoposition` - Position von Objekten oder Personen
- `hardware` - Allgemeine Hardware-Schnittstellen (z.B. für ESP8266, ESP32)
- `health` - Gesundheitsdaten wie Blutdruck, Blutzucker, ...
- `household` - Küchengeräte, Haushaltsgeräte, Staubsaugerroboter, ...
- `infrastructure` - Netzwerktechnik, Drucker, Scanner, Telefone, ...
- `iot-systems` - Weitere IoT-Geräte, welche nicht in die anderen Kategorien passen
- `lighting` - Beleuchtung
- `logic` - Logikmodule für eigene Regeln oder Szenen
- `messaging` - Nachrichtendienste wie Telegram oder E-Mail
- `misc-data` - Export und Import von Daten
- `multimedia` - Fernseher, Receiver, Beamer, ...
- `network` - Ping, ...
- `protocols` - Generische Protokolle (wie MQTT)
- `storage` - Daten-Speicherung wie mySQL oder InfluxDB
- `utility` - Weitere Tools wie Backup-Adapter
- `visualization` - Visualisierungs-Adapter
- `visualization-icons` - Zusätzliche Icons für die Visualisierung
- `visualization-widgets` - Weitere Widgets für die Visualisierung
- `weather` - Wetterdaten

Type string

common.unsafePerm
Legt fest, ob das Adapter-Paket mit dem `--unsafe-perm` Parameter für npm installiert werden muss. Siehe npm Dokumentation

Type boolean

common.plugins
Liste von Plugins (z.B. *Sentry*)

Type object

common.plugins.sentry
Konfiguration des Sentry-Plugins. Siehe *Sentry*

```
"plugins": {
    "sentry": {
        "dsn": "https://xxx@sentry.iobroker.net/xxx"
    }
}
```

Type object

common.availableModes
Werte für `common.mode` (falls mehr als ein Wert erlaubt ist)

Type array

common.blockly
Legt fest, ob der Adapter eigene Blockly-Bausteine mitbringt (`admin/blockly.js` erforderlich)

Type boolean

Default `false`

common.connectionType
Definiert die Qulle der Adapter-Daten. Wird im Admin ab Version 5 dargestellt und dient als Information für den Nutzer

- `local` - Die Kommunikation findet lokal / im eigenen Netzwerk statt (z.B. mit dem Gerät direkt per HTTP)
- `cloud` - Für den Adapter ist eine aktive Internetverbindung erforderlich. Die Daten werden z.B. vom Server des Herstellers abgerufen.

Type string

common.dataSource
Legt fest, wie Daten geholt werden

- `poll` - Die Daten werden regelmäßig abgefragt (z.B. per Zeitplan)
- `push` - Das Gerät liefert die Daten selbstständig zum Adapter
- `assumption` - Der genaue Status ist nicht definiert

Type string

common.compact
Legt fest, ob der Adapter im *Compact Mode* gestartet werden kann

Type boolean

Default `false`

common.dataFolder
Verzeichnis-Pfad, in welchem der Adapter seine Daten ablegt (relativ zu `iobroker-data`). Die Variable `%INSTANCE%` kann ebenfalls im Pfad genutzt werden

Type string

common.dependencies
Liste von Abhängigkeiten auf dem lokalen System, welche für diesen Adapter notwendig sind

```
"dependencies": [
    {
        "js-controller": ">=3.3.0"
    }
]
```

Type array

common.eraseOnUpload
Löscht alle existierenden Daten im Adapter-Verzeichnis vor einem Upload

Type boolean

common.extIcon
URL zur Icon-Datei für die Admin-Übersicht (z.B. PNG-Datei auf GitHub)

Type string

common.getHistory
Legt fest, ob der Adapter den `getHistory` Befehl unterstützt

Type boolean

common.globalDependencies
Liste von Abhängigkeiten im gesamten ioBroker-System (Multihost-Betrieb). Siehe basics-multihost

```
"globalDependencies": [
    {
        "admin": ">=5.1.19"
    }
]
```

Type array

common.icon
Pfad zum lokalen Icon des Adapters (nach Installation). Sollte im Unterverzeichnis `admin` liegen

Type string

common.keywords
Liste von Schlüsselwörtern, um den Adapter über die Suche (besser) finden zu können

```
"keywords": [
    "web",
    "weather",
    "air",
    "quality"
]
```

Type array

common.localLinks
Konfiguration für Intro-Tab

```
"localLinks": {
    "_default": {
        "link": "%protocol%://%bind%:%port%",
        "pro": true
    }
}
```

Type object

common.loglevel
Standard Log-Level neuer Instanzen. Empfohlen: `info`

- `silly` - Alles
- `debug` - Debug-Nachrichten
- `info` - Informationen
- `warn` - Warnungen
- `error` - Fehler

Type string

common.logTransporter
Legt fest, ob der Adapter die Log-Einträge von anderen Adaptern entgegen nehmen kann (um sie z.B. wo anders zu speichern)

Type boolean

common.noIntro

Type boolean

common.noRepository

Type boolean

common.nogit
Legt fest, ob eine Installation direkt von GitHub verboten werden soll

Type boolean

common.nondeletable
Legt fest, ob ein Adapter gelöscht oder aktualisiert werden kann. Falls `true`, kümmert sich der `js-controller` um diese Aufgaben

Type boolean

Default `false`

common.onlyWWW
Legt fest, ob der Adapter nur weitere HTML-Dateien bereitstellt und keine Logik enthält (wie zum Beispiel Widget-Adapter für `VIS`)

Type boolean

common.osDependencies.darwin
Liste mit erforderlichen MacOS-Paketen für diesen Adapter

Type array

common.osDependencies.linux
Liste mit erforderlichen Linux-Paketen für diesen Adapter

Type array

`common.osDependencies.win32`
Aktuell nicht genutzt, da Linux keinen Paket-Manager hat

Type array

`common.os`
Liste mit unterstützten Betriebssystemen

- `darwin` - Mac OS X
- `linux` - Linux
- `win32` - Windows

type string / array

`common.preserveSettings`
Liste mit Attributen, welche nicht automatisch gelöscht werden sollen (z.B. `history`)

Type string / array

`common.restartAdapters`
Liste mit Adaptern, welche neugestartet werden sollen, nachdem dieser Adapter installiert wurde (z.B. `["vis"]`)

Type array

`common.serviceStates`

Type string / boolean

`common.singletonHost`
Legt fest, ob es nur eine einzelne Instanz pro Host geben darf

Type boolean

Default `false`

`common.singleton`
Legt fest, ob es nur eine einzelne Instanz im gesamten ioBroker-System geben darf (Multihost-Betrieb). Siehe basics-multihost

Type boolean

Default `false`

`common.stopBeforeUpdate`
Legt fest, ob die Instanzen vor einem Update gestoppt werden müssen

Type boolean

`common.stopTimeout`
Wartezeit in Millisekunden, bis der Adapter angehalten wird

Type number

Default `500`

`common.subscribable`
???

Type boolean

`common.subscribe`
???

Type string

`common.supportCustoms`
Legt fest, ob es zusätzliche Einstellungen für jeden Datenpunkt gibt (`admin/custom.html` erforderlich)

Type boolean

common.supportStopInstance
Legt fest, ob der Adapter das `stopInstance` Signal unterstützt. Siehe *Messagebox*

Type boolean

common.wakeup
Legt fest, ob die Instanz gestartet werden soll, wenn ein Wert in `system.adapter.<adapter-name>.<instanz-nummmer>.wakeup` geschrieben wird.

Type boolean

common.webByVersion

Type boolean

common.webExtendable
Legt fest, ob der Webserver dieses Adapters mit Plugins erweitert werden kann (z.B. `simple-api`)

Type boolean

common.webExtension
Relativer Pfad der Web-Extension (z.B. `lib/simpleapi.js`)

Type string

common.webPreSettings

Type object

common.webservers
Liste mit Webservern, welche Inhalte aus dem www-Verzeichnis des Adapters liefern

Type array

common.welcomeScreen

Type array

common.welcomeScreenPro

Identisch zu `common.welcomeScreen`, allerdings für Zugriff über die ioBroker-Cloud

```
   "welcomeScreenPro": {
       "link": "admin/index.html",
       "name": "Admin",
       "img": "admin/img/admin.png",
       "color": "pink",
       "order": 5,
       "localLinks": "_default",
       "localLink": true
   }

:type: object
```

1.21.4 Eigenschaften (Schedule)

common.schedule
CRON-Definition, wann die Instanzen gestartet werden sollen (kann vom Benutzer angepasst werden)

Type string

common.allowInit
Legt fest, ob ein Adapter auch außerhalb des definierten Zeitplanes gestartet wird (z.B. nach Änderung der Instanz-Konfiguration)

Type boolean

1.21.5 Eigenschaften (Daemon)

common.restartSchedule
CRON-Definition, wann die laufenden Instanzen neugestartet werden sollen (kann vom Benutzer angepasst werden)

Type string

1.21.6 Eigenschaften (Admin)

common.adminColumns
Custom attributes, that must be shown in the admin in the object browser. Like: Type is a type of the attribute (e.g. string, number, boolean) and only needed if edit is enabled. objTypes is a list of the object types, that could have such attribute. Used only in edit mode too

```
[
    {
        "name": {
            "en": "KNX address"
        },
        "path": "native.address",
        "width": 100,
        "align": "left"
    },
    {
        "name": "DPT",
        "path": "native.dpt",
        "width": 100,
        "align": "right",
        "type": "number",
        "edit": true,
        "objTypes": [
            "state",
            "channel"
        ]
    }
]
```

Type array

common.adminTab.fa-icon
Font-Awesome Icon für das Tab

Type string

common.adminTab.ignoreConfigUpdate

Type boolean

common.adminTab.link

Type string

common.adminTab.name
Titel des Tabs (übersetzt in mehrere Sprachen)

Type object

common.adminTab.singleton
Legt fest, ob nur ein Tab für alle Instanzen angezeigt werden soll

Type boolean

`common.adminUI`
Legt fest, wie die Konfiguration im Admin erfolgen soll

Type object

`common.adminUI.config`
Legt fest, wie die Konfiguration für die Admin-Oberfläche aufgebaut ist

- `none`
- `materialize` (`admin/index_m.html` erforderlich - ab Admin Version 4)
- `json` (`admin/jsonConfig.json` erforderlich - ab Admin Version 5)

Type string

`common.adminUI.custom`

- `none`
- `materialize` (`admin/custom_m.html` erforderlich - ab Admin Version 4)
- `json` (`admin/jsonCustom.json` erforderlich - ab Admin Version 5)

Type string

`common.adminUI.tab`

- `html`
- `materialize`

Type string

Weitere Optionen

`objects`
Liste von Objekten, welche für den Adapter erstellt werden sollen

Type array

`instanceObjects`
Liste von Objekten, welche für jede Instanz automatisch erstellt werden

Type array

`protectedNative`
Liste von Attributen, welche nur vom Adapter selbst lesbar sind (z.B. `["password"]`). Siehe *Daten-Verschlüsselung*

Type array

`encryptedNative`
Liste vo automatisch verschlüsselten Attributen. Siehe *Daten-Verschlüsselung*

Type array

`native`
Liste von vordefinierten Attributen, welche z.B. in der Admin-Konfiguration überschrieben werden können

Type object

`notifications`
Liste von Objekten zur Konfiguration zur Konfiguration des internen Notification-Systems. Siehe *Notifications*

Seit js-controller 3.2.0

Type array

1.21.7 Eigenschaften (deprecated)

Diese Eigenschaften sind für aktuelle Adapter mit dem Admin 5 nicht mehr relevant

- `common.title` - Langer Name des Adapters für Admin-Version 2, 3 und 4
- `common.npmLibs` - Ersetzt durch Abhängigkeiten in der `package.json`
- `common.main` - Ersetzt durch `main` in der `package.json`
- `common.localLink` - Ersetzt durch `common.localLinks`
- `common.engineTypes` - Ersetzt durch `engine` in der `package.json`
- `common.config.height` - Standard-Höhe für den Konfigurations-Dialog für Admin 2
- `common.config.minHeight` - Mindest-Höhe für den Konfigurations-Dialog für Admin 2
- `common.config.width` - Standard-Breite für den Konfigurations-Dialog für Admin 2
- `common.config.minWidth` - Mindest-Breite für den Konfigurations-Dialog für Admin 2
- `common.materialize` (boolean) - Legt fest, ob der Adapter die Admin-Oberfläche für Admin-Version 3 und 4 bereitstellt
- `common.materializeTab` (boolean) - Legt fest, ob der Adapter ein eigenes Tab für Admin-Version 3 und 4 bereitstellt
- `common.noConfig` (boolean) - Definiert, ob Instanzen konfiguriert werden können (ab Admin 5 sollte `adminUI.config = none` verwendet werden)

1.21.8 Links

- Schema-Datei
- Offizielle Doku

1.22 Daten-Verschlüsselung

Normalerweise werden alle Einstellungen in Adaptern unverschlüsselt abgelegt. Das heißt, dass zum Beispiel die Instanz-Konfiguration einfach mit in die Objects-Datenbank geschrieben wird. Im Klartext.

Gerade sensible Informationen sollten daher verschlüsselt abgelegt werden.

Seit `js-controller` Version 3.0.0 gehört eine Verschlüsselung zum Standard und kann sehr einfach verwendet werden.

Gefahr: Da auch die Instanz-Einstellungen nur Objekte sind, kann jeder Wert im System von jedem anderen Adapter auch gelesen werden. Es wäre also ein leichtes, einen Adapter oder ein Script zu bauen, welches konfigurierte Benutzerdaten, Zugangstoken oder Passwörter anderer Adapter ausliest und diese ins Internet sendet. Dieser Gefahr sollte man sich als Nutzer generell bewusst sein.

1.22.1 Zugriff verbieten

Eine kleine Hürde bietet die Funktion, dass der Zugriff auf sensible Konfigurations-Attribute von anderen Adaptern aus verboten wird. Das heißt, dass diese Info einfach nicht ausgeliefert wird. Der Admin-Adapter ist dabei die Ausnahme (immerhin müssen die Daten noch konfigurierbar bleiben).

Dazu kann in der *io-package.json* mit einem neuen Attribute festgelegt werden, dass diese Information gefiltert wird, wenn ein anderer Adapter das Objekt liest.

```
"protectedNative": [
    "user",
    "password"
]
```

In diesem Beispiel sind also die beiden Attribute `user` und `password` nicht mehr von anderen Adaptern einfach so lesbar.

Warnung: Dieses Vorgehen bietet natürlich keine 100% Sicherheit, denn ein anderer Adapter kann nach wie vor die Objekt-Datei von der Festplatte lesen und das Attribut dort extrahieren.

1.22.2 Daten verschlüsseln

Wie Du schon weißt, werden Objekte als JSON in der Datei `/opt/iobroker/iobroker-data/objects.json` abgelegt. Jede Eigenschaft wird dabei einfach im Klartext dort gespeichert. Hat nun jemand Zugriff auf diese Datei, können die Daten einfach gelesen werden. Passwörter, API-Token, Benutzernamen usw. liegen also im Klartext in dieser Datei.

Um die Daten nicht mehr im Klartext zu speichern, kann man einzelne Attribute verschlüsseln lassen. Dies geschieht ebenfalls über ein Attribut in der `io-package.json` des einzelnen Adapters.

```
"encryptedNative": [
    "password"
]
```

In diesem Beispiel wird also das Passwort verschlüsselt in der Objekt-Datenbank abgelegt und ist nicht mehr im Klartext lesbar.

Warnung: Alle Adapter verschlüsseln ihre Informationen mit dem gleichen Schlüssel und dem gleichen Verfahren. Es ist also dennoch möglich, dass andere Adapter diese Informationen auszulesen und die Daten zu entschlüsseln.

Generell ist es sinnvoll, beide Attribute zu kombinieren (wie hier gezeigt). Also auch verschlüsselte Informationen nicht einfach auslesen zu lassen.

Die Verschlüsselung und Entschlüsselung passiert komplett automatisch. Die Informationen müssen also nicht manuell im Code entschlüssel werden.

1.22.3 Links

- Offizielle Doku

1.23 Messagebox

Konfiguration in der *io-package.json* (`common.messagebox`)

- `common.supportStopInstance` (boolean)

TODO

1.24 Notifications

Seit js-controller 3.2.0

Konfiguration in der *io-package.json* (`notifications`)

```
async this.registerNotification('system', null, 'test');
```

1.25 DEV-Environment

Um ioBroker-Adapter zu entwickeln, gibt es mehrere Wege. Der einfachste und schnellste ist der `dev-server`. Dieses Paket wird vom ioBroker-Team bereitgestellt und ermöglicht eine schnelle und einfache Installation der ioBroker-Umgebung. Dieser ermöglicht es, den Adapter mit unterschiedlichen Versionen des `js-controller` zu testen.

```
npm install --global @iobroker/dev-server
```

1.25.1 Beispiel

```
git clone git@github.com:klein0r/ioBroker.birthdays.git
cd ioBroker.birthdays/
dev-server setup --jsController 3.3.18 --admin 5.1.28
dev-server watch
```

Für eine ausführliche Dokumentation und alle verfügbaren Parameter, schaue bitte in das GitHub-Repository.

1.25.2 Links

- GitHub-Repo DEV-Server

1.26 Adapter-Entwicklung

In diesem Dokument werden einige Code-Beispiele gesammelt, welche häufig bei der *Adapter-Entwicklung* benötigt werden.

> **Gefahr:** Diese Code-Beispiele sind NICHT für JavaScript-Code im JavaScript-Adapter gedacht! Dort werden teilweise andere Funktionen bzw. unterschiedliche Signaturen genutzt.

1.26.1 Objekte

Wie Objekte genau aufgebaut sind, und welche Eigenschaften sie unterstützten, erfährst Du auf der Seite *Objekte*.

Neues Objekt erstellen im eigenen Namespace

Gerenell ist es empfehlenswert, Objekte über die `instanceObjects` in der *io-package.json* anzulegen. Solltest Du beim Adapter-Start schon wissen, welche Objekte Du später haben möchtest, nutze diese Funktion.

```
await this.setObjectNotExistsAsync(deviceName, {
    type: 'device',
    common: {
        name: 'Any name',
        type: 'string',
        role: 'text'
    },
    native: {}
});
```

Objekt im eigenen Namespace löschen (rekursiv)

```
this.delObjectAsync(id, {recursive: true}, () => {
    this.log.debug('object deleted: ' + id);
});
```

1.26.2 States

Wie States genau aufgebaut sind, und welche Eigenschaften sie unterstützten, erfährst Du auf der Seite _development-states.

Wert schreiben

```
await this.setStateAsync('myState', {val: newValue, ack: true});
```

Alternativ kann man den neuen Wert auch einzeln übergeben. Allerdings würde ich empfehlen, immer ein komplettes State-Objekt zu übergeben, da dies ansonsten intern aufgebaut wird. Sollte `newValue` (versehentlich) ein Objekt sein, wird es als fertiges State-Objekt interpretiert, welchem dann wichtige Eigenschaften fehlen werden.

```
await this.setStateAsync('myState', newValue, true);
await this.setStateAsync('myState', newValue, true, (err) => {
    if (err) this.log.error(err);
});
```

this.getCertificates(publicName, privateName, chainedName, callback) this.getCertificatesAsync(publicName, privateName, chainedName, callback)

this.restart()

1.26.3 Links

- adapter.js
- Offizielle Doku

1.27 Logging

Bei der Adapter-Entwicklung sollte man darauf achten, dass man aussagekräftige Logs schreibt. Dabei gilt es jeweilige Log-Level zu berücksichtigen. Je nach eingestelltem Loglevel auf der aktuellen Instanz, werden unterschiedlich viele Informationen geloggt.

Das Standard-Loglevel eines Adapters / neuer Instanzen wird in der Konfiguration in der *io-package.json* festgelegt (`common.loglevel`). Dieses Level kann später vom Benutzer auf jeder einzelnen Instanz angepasst werden!

```
this.log.debug('debug message'); // log message with debug level
this.log.info('info message');   // log message with info level (enabled by default␣
↪for all adapters)
this.log.warn('warning');        // log message with info warn
this.log.error('error');         // log message with info error
```

1.28 Testing

Um verschiedene Tests für den eigenen ioBroker-Adapter vorzubereiten, gibt es das Paket @iobroker/testing. Auf der GitHub-Seite sind einige Beispiele vorhanden.

Dabei werden folgende Typen unterschieden:

- Package-Tests prüfen, ob das Paket den generellen Anforderungen vom ioBroker-Team entspricht (ähnlich Adapter-Checker)
- Unit-Tests (laufen ohne einen `js-controller`) und arbeiten mit Mocks
- Integration-Tests arbeiten mit einer `js-controller`-Instanz, welche automatisch gestartet wird

1.28.1 Package-Tests

Diese Tests führen Prüfungen der `package.json` und der `io-package.json` durch. Datei `test/package.js`:

```
const path = require('path');
const { tests } = require('@iobroker/testing');

// Validate the package files
tests.packageFiles(path.join(__dirname, '..'));
```

1.28.2 Integration-Tests

TODO

1.28.3 Unit-Test

Diese Tests führen allgemeine Unit-Tests durch. Datei `test/unit.js`:

```
const path = require('path');
const { tests } = require('@iobroker/testing');

// Run unit tests - See https://github.com/ioBroker/testing for a detailed␣
↪explanation and further options
tests.unit(path.join(__dirname, '..'));
```

1.28.4 Tests lokal ausführen

Um alle Tests lokal auszuführen, können diese über npm angestoßen werden.

```
npm run test:js
npm run test:integration
npm run test:package
npm run test:unit
```

Damit diese Befehle funktionieren, muss die `package.json` des Adapters entsprechende Einträge enthalten:

```
"scripts": {
    "test:js": "mocha --config test/mocharc.custom.json \"{!(node_modules|test)/**/*.
↪test.js,*.test.js,test/**/test!(PackageFiles|Startup).js}\"",
    "test:package": "mocha test/package --exit",
    "test:unit": "mocha test/unit --exit",
    "test:integration": "mocha test/integration --exit",
    "test": "npm run test:js && npm run test:package"
}
```

1.28.5 GitHub Actions

TODO

1.28.6 Links

- GitHub-Repo testing
- GitHub-Repo testing-action-check (GitHub Actions)
- GitHub-Repo testing-action-adapter (GitHub Actions)
- GitHub-Repo testing-action-deploy (GitHub Actions)
- test-and-release.yml Template

KAPITEL 2

License

MIT License

Copyright (c) 2021 Matthias Kleine <info@haus-automatisierung.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the „Software“), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED „AS IS“, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Stichwortverzeichnis

Sonderzeichen

A

C

E

F

I

L

N

O

P

Q

T

U

V