Dein **Hauptskript** ist das zentrale Steuerelement unseres Mülltonnen-Systems. Es verarbeitet automatisch **Füllstandsmeldungen** und **RFID-Scans**, erstellt **Abhollisten** und **Leerungslisten**, und sorgt für die **Anzeigen in Jarvis**.

**Was macht das Script?**

**1. Definition von Dateien & Topics**

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

* Nutzt 3 Dateien als Datenbasis:
  + **Kundendaten.json** → enthält alle bekannten Tonnen + Kunden
  + **AbholListe.json** → Mülltonnen, die zur Abholung gemeldet wurden
  + **Leerungen.json** → Tonnen, die geleert wurden
* Lauscht auf MQTT-Nachrichten für Füllstand und Leerung

**2. Erzeugung von ioBroker-States für Jarvis**

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Diese Datenpunkte zeigen dir:

| **Datenpunkt** | **Inhalt** |
| --- | --- |
| abholListeText | Lesbare Liste aller angemeldeten Abholungen |
| abholListeJson | Die gleiche Liste als JSON |
| leerungslisteText | Lesbare Liste aller durchgeführten Leerungen |
| leerungslisteJson | JSON-Liste der Leerungen |
| kunden.info | Letzter erkannter Kunde mit Leerungszeit |

**3. Füllstandsmeldung verarbeiten**

**Trigger: MQTT-Nachricht Muelleimer.Fuellstand**



**Ablauf:**

* Parse die Nachricht, z. B.: { "muelleimer\_id": "123abc", "status": "voll" }
* Prüfe, ob RFID in Kundendaten.json bekannt ist
* Wenn der Status voll ist:
  + Eintrag in AbholListe.json erzeugen, **aber nur**, wenn **dieser RFID nicht bereits innerhalb von 30 Tagen** gemeldet wurde

**Neu:**

* gemeldet\_am: lesbares Datum
* timestamp: für sicheren Zeitvergleich

**4. RFID-Scan verarbeiten (Leerung)**

**Trigger: MQTT-Nachricht Muelleimer.Leerung**



**Ablauf:**

* Parse RFID, z. B. { "muelleimer\_id": "123abc" }
* Prüfe, ob der Kunde existiert
* Nur speichern, wenn die letzte bekannte Meldung status = voll war
* Dann:
  + Eintrag in Leerungen.json schreiben
  + kunden.info-State aktualisieren (für Jarvis)
  + **Geleerte Tonne aus AbholListe.json entfernen**

**5. Anzeige alle 30 Sekunden aktualisieren**



* Liest AbholListe.json und Leerungen.json
* Erzeugt aus beiden jeweils:
  + eine **formatierte Textliste** für Jarvis
  + und eine **JSON-Liste** für weitere Verarbeitung oder Anzeige

**Zusammengefasst:**

| **Komponente** | **Funktion** |
| --- | --- |
| **MQTT-Listener Füllstand** | Erkennt volle Tonnen, meldet sie zur Abholung |
| **MQTT-Listener Leerung** | Erkennt RFID-Scan bei Entleerung, protokolliert die Leerung |
| **Duplikatschutz** | Keine Mehrfacheinträge für dieselbe Tonne in kurzer Zeit |
| **State-Ausgabe** | Anzeige der Daten in Jarvis (Text und JSON) |
| **kunden.info** | Letzter Kunde mit leerungsdatum (optional für Logging oder Verlauf) |

# Grafische Ablaufskizze

Ein Bild, das Text, Screenshot, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

| **Funktion** | **Zweck** |
| --- | --- |
| **Automatische Anmeldung** | Tonnen melden sich bei "voll" selbst |
| **Duplikat-Schutz** | Nur 1 Eintrag pro Tonne alle 30 Tage |
| **Automatische Leerung** | Bei RFID-Scan nur, wenn vorher als "voll" erkannt |
| **Listenpflege** | Abholliste wird bei Leerung automatisch bereinigt |
| **Jarvis-Anzeige** | Abholliste & Leerungen lesbar und als JSON |