SubLine
Watch

Konfigurationsdatei

Dokumentation zur config.json

SubLine Watch v1.4.0

Version 1

www.subline.watch

Inhaltsverzeichnis

Einleitung

Dieses Dokument beschreibt den Aufbau der Konfigurationsdatei `config.json`. Es stellt ein ergänzendes Dokument zum Produktdatenblatt, zum Anwendungshandbuch sowie zum Administratorhandbuch dar.

Dieses Dokument ist für Sie nur relevant, wenn Sie die Konfigurationsdatei manuell oder durch eigene Tools automatisiert bearbeiten oder erstellen möchten.

Beim Bearbeiten der `config.json` ist folgendes zu bechten:

- Stellen Sie beim Bearbeiten sicher, dass das Administrator Tool nicht Ihre manuelle Konfiguration überschreibt. Wechseln Sie im Administator Tool in das Hauptmenü oder beenden Sie das Tool, bevor Sie mit der manuellen Bearbeitung beginnen.
- Prüfen Sie Ihre manuelle Bearbeitung anschließend mit dem Administrator Tool auf Richtigkeit, indem Sie in der Bearbeitungsansicht auf entsprechende Hinweise achten.

Aufbau der Konfigurationsdatei

Die `config.json` besteht aus einem JSON-Array, welches alle Konfigurationen in Form von JSON-Objekten beinhaltet:

```
[
   {
      …Konfiguration 1…
   },
   {
      …Konfiguration 2…
   }
]
```

Aufbau eines Konfigurationsobjektes

Jede Konfiguration bzw. jedes Konfigurationsobjekt verfügt über folgende Eigenschaften:

Eigenschaft	**Type**	**Beschreibung**
`name`	`string`	Name der Konfiguration
`alarmSettings`	`object` (optional)	Globale Alarmeinstellungen
`alarmSettings.volume`	`number` (optional)	Lautstärke der Alarmtöne von 0 (aus) bis 100 (maximale Lautstärke, Standard). Nur Ganzzahlen sind zulässig.
`alarmSettings.vibration`	`boolean` (optional)	Vibrationsfolge bei Alarmierung ein- (`true`, Standard) oder ausschalten

		(`false`).
`mqttBroker`	`object`	Informationen zum MQTT-Broker sowie zur MQTT-Verbindung
`mqttBroker.host`	`string`	IP-Adresse des MQTT-Brokers
`mqttBroker.port`	`number`	Port des MQTT-Brokers
`mqttBroker.user`	`string` (optional)	Benutzername zur Authentifizierung am MQTT-Broker, falls notwendig. Nur zusammen mit `mqttBroker.password` wirksam.
`mqttBroker.password`	`string` (optional)	Passwort zur Authentifizierung am MQTT-Broker, falls notwendig. Nur zusammen mit `mqttBroker.user` wirksam.
`mqttBroker.secure`	`boolean` (optional)	Wenn eine TLS-gesicherte Verbindung gewünscht ist, muss dieser Wert auf `true` gesetzt werden. Ggf. muss hierzu ein Schlüsselspeicher importiert werden (siehe Administratorhandbuch).
`mqttBroker.ignoreHostnameVerification`	`boolean` (optional)	Ist dieser Wert auf `true` gesetzt, wird keine Hostnamen-Überprüfung des TLS-Zertifikats vorgenommen. Dies stellt ein Sicherheitsrisiko dar und wird nicht empfohlen. Dieser Wert wird nur berücksichtigt, wenn `mqttBroker.secure` auf `true` gesetzt ist. Dieser Wert hat keinen Effekt, wenn die vollständige Überprüfung des Zertifikats deaktiviert ist (`mqttBroker.ignoreCertificateVerification=true`).
`mqttBroker.ignoreCertificateVerification`	`boolean` (optional)	Ist dieser Wert auf `true` gesetzt, wird keine Überprüfung des TLS-Zertifikats vorgenommen. Dies stellt ein Sicherheitsrisiko dar und wird nicht empfohlen. Dieser Wert wird nur berücksichtigt, wenn `mqttBroker.secure` auf `true` gesetzt ist.
`mqttBroker.keepAlive`	`number` (optional)	MQTT “keep alive”-Zeitdauer. Ist dieser Wert nicht gesetzt, wird der Standardwert von 30 Sekunden genommen.
`mqttBroker.cleanSession`	`boolean`	Ist eine persistente MQTT-Verbindung

	(optional)	gewünscht, muss dieser Wert auf `false` gesetzt werden (Empfehlung). Andernfalls wird von einer “clean session” ausgegangen.
`machines`	`array`	Sammlung aller Maschinen(-Objekte). Die Reihenfolge der Objekte bestimmt die Reihenfolge der Maschinen in der Maschinenübersichts-Ansicht.
`machines[].name`	`string`	Name der Maschine bzw. der Sammlung von Datenpunkten
`machines[].data`	`array`	Sammlung aller Datenpunkte (Objekte) der Maschine. Die Reihenfolge der Objekte bestimmt die Reihenfolge in der Maschinen-Ansicht.
`machines[].data[].topic`	`string`	MQTT-Topic des Datenpunktes. Es sollten nur Zeichen verwendet werden, welche von der MQTT-Spezifikation als empfehlenswert deklariert wurden, mit Ausnahme folgender Zeichen: • < • > • = (diese Zeichen nicht verwenden)
`machines[].data[].name`	`array`	Name des Datenpunktes
`machines[].data[].unit`	`string` (optional)	Einheit des Datenpunktes
`machines[].data[].format`	`object` (optional)	Formatbeschreibung des MQTT-Payloads zur Extraktion des tatsächlichen Wertes. Bei Weglassen der Eigenschaft wird keine Extraktion durchgeführt und der gesamte MQTT-Payload wird als Rohwert interpretiert.
`machines[].data[].format.type`	`string`	Angabe, um welches Format es sich handelt. Zulässige Werte: • `json` Bei der Angabe eines unbekannten Formates wird keine Extraktion der Daten durchgeführt und der gesamte MQTT-Payload wird als Rohwert verstanden.
`machines[].data[].format.data`	`array`	Für den Formattyp `json` gilt Folgendes: Es handelt sich um ein Array aus Strings, welche die Kaskade der verschachtelten

		JSON-Objekte hin zum eigentlichen Wert beschreiben. Bis auf das letzte Element besteht das Array aus JSON-Objektnamen, beginnend mit dem äußersten Objekt. Der letzte Eintrag ist der Name der Eigenschaft, in der sich der tatsächliche Wert befindet.
		
		Es ist als eine Art “Wegbeschreibung” durch das JSON-Objekt zum eigentlichen Wert zu verstehen.
		
		Beispiel eines MQTT-Payloads:
		`{`
		`  "id": "gh346h10",`
		`  "data": {`
		`  "name": "Tempertur XY",`
		`  "value": 13.55`
		`  }`
		`}`
		
		Die dazugehörige Wegbeschreibung:
		`["data", "value"]`

`machines[].data[].decimal`	`number` (optional)	Anzahl der maximalen Dezimalstellen des Datenpunktes. SubLine Watch unterstützt Werte bis maximal 9 Dezimalstellen. Bei einem Wert von `-1` wird der Wert des Datenpunktes als Text verarbeitet. Ist kein Wert gesetzt, wird der Wert des Datenpunktes als Ganzzahl verarbeitet (`decimal=0`). Weicht die Angabe der maximalen Dezimalstellen von den Dezimalstellen des Rohwertes ab, versucht SubLine Watch den Wert auf die gewünschten Dezimalstellen zu runden.
`machines[].data[].qos`	`number` (optional)	MQTT “Quality of Service”-Level. Zulässige Werte `0` (Standard, empfohlen), `1`, `2`.
`machines[].data[].assignment`	`boolean` (optional)	Indikator für einen speziellen Datenpunkt (`assignment=true`), der es Benutzern ermöglicht, über Alarmmeldungen der übergeordneten Maschine, eine Selbstzuweisung durchzuführen. Über das im Datenpunkt angegebene MQTT-Topic wird die eigene MQTT-Client-ID veröffentlichen und damit anderen Benutzern signalisiert, dass die Verantwortung der

		übergeordneten Maschine übernommen wurde. Dieser Datenpunkt wird nicht in der Maschinen-Übersicht der SubLine Watch App angezeigt. Ist `assignment` nicht gesetzt oder `false`, handelt es sich um einen regulären Datenpunkt.
`machines[].data[].alarms`	`array` (optional)	Alarmdefinitionen des Datenpunktes
`machines[].data[].alarms[].message`	`string`	Alarm-Text
`machines[].data[].alarms[].type`	`string` (optional)	Alarmtyp. Gültige Werte sind `error`, `warn`, und `info`. Ungültige Werte oder das Weglassen der Eigenschaft spezifiziert den Alarmtyp als Fehler (`error`).
`machines[].data[].alarms[].conditions`	`array`	Sammlung von Bedingungen (`strings`), die alle erfüllt sein müssen, damit der Alarm ausgelöst wird.
`machines[].data[].translation`	`string` (optional)	Anzuwendende Übersetzungstabelle
`translations`	`array` (optional)	Übersetzungstabellen
`translations[].id`	`string`	Eindeutige Bezeichnung der Übersetzungstabelle. Gültige Zeichen: • a-z (Kleinbuchstaben ohne Umlaute) • 0-9 • _ (Unterstrich)
`translations[].values`	`array`	Übersetzungswerte
`translations[].values[].value`	`string`	Rohwert, auf den die Übersetzung angewendet werden soll.
`translations[].values[].color`	`string` (optional)	Farbmarkierung als #RRGGBB-Wert. Alternativ können auch folgende Namen verwendet werden: `red`, `blue`, `green`, `black`, `white`, `gray`, `cyan`, `magenta`, `yellow`, `lightgray`, `darkgray`, `grey`, `lightgrey`, `darkgrey`, `aqua`, `fuchsia`, `lime`, `maroon`, `navy`, `olive`, `purple`, `silver`, und `teal`.
`translations[].values[]`	`string`	Anzuzeigender Text (Übersetzung).

`.text`	(optional)	

Alarmbedingungen

Jeder Alarm wird von mindestens einer Alarmbedingung ausgelöst. Sind mehrere Alarmbedingungen für einen Alarm definiert, müssen alle Bedingungen erfüllt sein, damit der Alarm ausgelöst wird.

Eine Bedingung ist durch einen Text definiert:

```
…
"alarms": [
    {
        "message": "Alarm! Wert1 ist größer als Wert2!”
        "conditions": ["wert1 > wert2"]
    }
]
…
```

Eine Bedingung zeichnet sich durch zwei Operanden (`wert1` und `wert2`) aus, welche durch einen Operator (`>`) voneinander getrennt sind.

Folgende Operanden sind zulässig:

- (Statische) Ganzzahlen (z. B. `13`)
- (Statische) Dezimalzahlen (z. B. `13.33`)
- (Statische) Texte (z. B. `fault 123`)
- Referenzen auf andere Datenpunkt-Werte, welche durch das MQTT-Topic identifiziert werden (z. B. `my_mqtt_topic/machine_1/temperature`). Diese Datenpunkt-Werte werden dann entweder als Ganzzahl, Dezimalzahl oder Text interpretiert (je nach `decimal`-Eigenschaft des Datenpunktes).
    - Sind mehrere Datenpunkte mit dem gleichen MQTT-Topic definiert, wird der Wert des zuerst gefundenen Datenpunktes als Referenz genommen. Dieser Spezialfall sollte vermieden werden, da es sich um eine nicht eindeutige Zuordnung handelt.
    - Wurde kein Datenpunkt mit dem entsprechenden MQTT-Topic gefunden, wird der vermeintliche “MQTT-Topic”-Text als statischer Text interpretiert. Durch diese Vorgehensweise kann SubLine Watch zwischen einem Text und einer Referenz eindeutig unterscheiden, was auf den ersten Blick nicht immer eindeutig zu sein scheint:
      “`ist/das/ein/topic = oder_ein_statischer_text`”

Weiterhin sind folgende Operatoren zulässig:

- `<` (kleiner als)
- `<=` (kleiner als oder gleich)
- `=` (gleich)
- `!=` (ungleich)
- `>=` (größer als oder gleich)

- `>` (größer als)

Wird ein Text mit einem anderen Operator als "`=`" oder "`!=`" verglichen, gilt diese Bedingung als nicht erfüllt und der Alarm kann nicht ausgelöst werden.

Beispiele für Alarm-Bedingungen eines Datenpunktes mit dem MQTT-Topic `mytopic/data1`, welcher die Rohwerte aus dem MQTT-Payload als Ganzzahlen interpretiert (`decimal=0`):

`mytopic/data1 > 5`	Bedingung ist erfüllt, wenn der eigene Wert größer ist als 5.
`mytopic/data1 = 20`	Bedingung ist erfüllt, wenn der eigene Wert gleich 20 ist.
`mytopic/data1 <= mytopic/data2`	Bedingung ist erfüllt, wenn der eigene Wert kleiner oder gleich dem Wert eines anderen Datenpunktes ist, welcher mit dem MQTT-Topic `mytopic/data2` definiert wurde. Gleichzeitig muss der andere Datenpunkt den eigenen Wert (vom MQTT-Topic `mytopic/data2` stammend) als Zahl (egal ob Ganzzahl oder Dezimalzahl) verarbeiten. Ist das nicht der Fall, ist diese Bedingung nicht erfüllt. Ebenfalls ist diese Bedingung nicht erfüllt, wenn kein Datenpunkt mit dem MQTT-Topic `mytopic/data2` gefunden wurde.
`mytopic/data1 = das ist ein text`	Diese Bedingung ist niemals erfüllt, da es sich hierbei um einen Vergleich zwischen einer Zahl und einem Text handelt. Würde der Datenpunkt den Inhalt nicht (wie vorgegeben) als Ganzzahl, sondern als Text interpretieren, könnte die Bedingung grundsätzlich erfüllt werden, was in diesem konkreten Beispiel jedoch nicht der Fall ist.
`3 = 3`	Diese Bedingung ist immer erfüllt und ergibt so keinen Sinn.
`abc = abc`	Diese Bedingung ist immer erfüllt, egal ob es sich bei `abc` um einen statischen Text oder um einen Datenpunkt bzw. um ein MQTT-Topic handelt.

Die Bedingungen der Alarme werden immer dann überprüft, wenn der zugehörige Datenpunkt einen neuen (oder auch exakt den gleichen) MQTT-Payload empfängt.