MQTT
Benutzerdokumentation

Änderungen

Version	Name	Datum	Geänderte Abschnitte	Änderungsgrund
3.3.3.7	wp	29.01.2019	Alle	Erzeugt
3.3.3.14	wp	17.11.2022	Service Configuration	Zusätzliches Payload-Beispiel
3.3.3.15	wp	17.11.2022	Allgemein Quality of Service	Ergänzt Hinzugefügt

Allgemein

MQTT stand einst für MQ Telemetry Transport und definiert heute einen OASIS- bzw. ISO Standard (ISO/IEC PRF 20922). Dahinter verbirgt sich eine leichtgewichtige Publish/Subscribe-Lösung, bei der Anwender Topics einrichten können, über die Clients (in der Rolle eines Publishers) Nachrichten bereitstellen und andere Clients (in der Rolle eines Subscribers) Nachrichten entnehmen. In Verbindung mit der Kommunikation via MQTT gibt es drei unterschiedliche Rollen:

- Publisher – Client der bestimmte Themen anbietet.
- Subscriber - Client der bestimmte Themen abonniert.
- Broker – Der für die Vermittlung der Themen zwischen Publisher und Subscriber sorgt.

Setup

MQTT ist ein Nuget-Paket, das den Elvis Server ergänzt. Das Paket fügt den MQTTDriver als Prozess-Port hinzu. Es handelt sich dabei um eine MQTT-Client-Implementierung. Für die Kommunikation wird ein aktiver MQTT-Broker vorausgesetzt.

Konfigurations-Eigenschaften

Eigenschaft	Datentyp	Beschreibung
Abfrageintervall beim Hochfahren	Int	Default: 200 (ms)
Certificate Store	String	Default: LocalMachine
Client Certificate Name	String	Der Name der Bedienseite, die das ElvisChart-Control enthält.
Host Name	String	Name/IP-Adresse des Brokers
Leseabfragen beim Hochfahren	Bool	Default: False
Lesen-Abstand (ms)	Int	Default: 0

Password	String	Optional
Port	Int	Ohne Secure üblicherweise 1883 Mit Secure überlicherweise 8883
Prefix		
Secure	Bool	Default: false
Service Configuration	Xml-Datei	Default: Keine Zuordnung.
User Name	String	Optional
Warten auf Lesebestätigung (ms)	Int	Default: 500
Warten auf Schreibbetätigung (ms)	Int	Default: 0
Startabfrage überspringen	Bool	Default: false
Treibertyp	Enum	MQTTDriver

Service Configuration

MQTT-Nutzdaten sind standardmäßig einfache Textwerte, die als Byte-Array übertragen werden. Mit Hilfe einer Konfigurations-Datei können Nutzdaten unterstützt werden, die eine Json-Struktur aufweisen. Die Konfigurations-Datei ist im xml-Format. Sie muss zunächst dem Server-Projekt hinzugefügt werden.

Das xml-Dokument enthält u. a. das Element MQTTDriverConfiguration mit den zwei Attributen PayloadPattern und ContentType. PayloadPattern definiert eine Vorlage für die Nutzdaten und ContentType legt das Format der Vorlage fest. Aktuell wird nur json unterstützt, für Textwerte wird keine Konfigurations-Datei benötigt.

Der PayloadPattern definiert, welche Eigenschaften in den Nutzdaten als Prozess-Adressen verwendet werden können. Die Werte solcher Eigenschaften sind durch das Muster {Property-Name} markiert. Hierzu das Beispiel einer Steuerdatei für die Nutzdaten eines 433 MHz-Controllers von Sonoff:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Configuration file for Elvis3 MQTTDriverConfiguration -->
<MQTTDriverConfiguration
  PayloadPattern=
  '{"RfReceived":{"Sync":5620,"Low":190,"High":530,"Data":"{Data}","RfKey":"None"}}'
  ContentType='json'
/>
```

Das PayloadPattern enthält das Property-Value-Paar Data:{Data}, welches hier als einziges Property für den Austausch von Prozess-Daten dient. Die zugehörige Projektierung der Prozess-Adresse für einen Datenpunkt ist dann:

Sonoff/RfReceived/Data -WI--.

Hierbei ist Sonoff das Topic (durch die Device-ID zu ersetzen) und RfReived/Data die Adressierung im Payload.

Eine Wert-Änderung des Datenpunktes ist somit mit einer Wert-Änderung der Eigenschaft Data im PayloadPattern verknüpft – sofern der Prozessanschluss des Datenpunktes mit dem MQTTDriver verbunden ist.

Zusätzlich wird noch eine Adressierung auf einen festen Anteil eines Wertes unterstützt. Wenn Sie dem Datenpunkt von oben die Prozess-Adresse

Sonoff/RfReceived/Data/4155 -WI—

zuordnen, wird der letzte Teil der Adresse dem geänderten Wert vorangestellt. Mit anderen Worten {Data} wird im PayloadPattern durch 4155C3 ersetzt, wenn C3 der zu versendende Wert ist. Beim Empfangen wird C3 entsprechend extrahiert.

Als weiteres Beispiel betrachten wir folgenden Payload:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Configuration file for Elvis3 MQTTDriverConfiguration -->
<MQTTDriverConfiguration
  PayloadPattern='{"tmp": {"value":"{ist}","units":
"C","is_valid":true},"target_t":{"enabled":true,"value":"{soll}","units":"C"},"temperature
_offset":0.0,"bat":"{bat}"}'
  ContentType='json'
/>
```

Für eine bessere Übersicht ist hier der json-Anteil extrahiert und entsprechend formatiert nochmals dargestellt:

```
{
        "tmp": {
                "value": "{ist}",
                "units": "C",
                "is_valid": true
        },
        "target_t": {
                "enabled": true,
                "value": "{soll}",
                "units": "C"
        },
        "temperature_offset": 0.0,
        "bat": "{bat}"
}
```

Die Eigenschaft mit dem Namen value kommt mehrfach vor. Durch die vollständige Angabe des jeweiligen Pfades in der Prozessadresse ist die Eindeutigkeit des Zugriffs aber immer gewährleistet. In dem Beispiel wird auf ist- und soll-Werte über folgende Prozessadressen zugegriffen:

Prozessadresse für den ist-Wert: shellies/shellytrv-8CF68110739D/status/tmp/value

Prozessadresse für den ist-Wert: shellies/shellytrv-8CF68110739D/status/target_t/value

shellies/shellytrv-8CF68110739D/status ist das Topic des verwendeten Gerätes.

Quality of Service (QoS)

QoS ist ein Schlüsselmerkmal des MQTT-Protokolls. QoS gibt dem Client die Möglichkeit, einen Service-Level zu wählen, der seiner Netzwerkzuverlässigkeit und Anwendungslogik entspricht. Da MQTT die erneute Übertragung von Nachrichten verwaltet und die Zustellung garantiert (selbst wenn der zugrunde liegende Transport nicht zuverlässig ist), erleichtert QoS die Kommunikation in unzuverlässigen Netzwerken erheblich.

MQTT definiert drei QoS-Level:

- At most once (1)
- At least once (2)
- Exactly once (3)

QoS 0 - höchstens einmal

Das minimale QoS-Niveau ist null. Dieses Servicelevel garantiert eine Best-Effort-Lieferung. Es gibt keine Liefergarantie. Der Empfänger bestätigt den Empfang der Nachricht nicht und die Nachricht wird vom Absender nicht gespeichert und erneut übertragen. QoS Level 0 wird oft als „Fire and Forget“ bezeichnet und bietet die gleiche Garantie wie das zugrunde liegende TCP-Protokoll.

QoS 1 - mindestens einmal

QoS Level 1 garantiert, dass eine Nachricht mindestens einmal beim Empfänger ankommt. Der Sender speichert die Nachricht, bis er vom Empfänger ein PUBACK-Paket (Publish Acknowledgement) erhält, das den Empfang der Nachricht bestätigt. Es ist möglich, dass eine Nachricht mehrmals gesendet oder zugestellt wird.

QoS 2 - genau einmal

QoS 2 ist die höchste Dienstebene in MQTT. Diese Ebene garantiert, dass jede Nachricht nur einmal von den beabsichtigten Empfängern empfangen wird. QoS 2 ist die sicherste und langsamste Dienstgütestufe. Die Garantie wird durch mindestens zwei Request/Response-Flüsse (einen vierteiligen Handshake) zwischen dem Sender und dem Empfänger bereitgestellt. Sender und Empfänger verwenden die Paketkennung der ursprünglichen PUBLISH-Nachricht, um die Zustellung der Nachricht zu koordinieren.

Festlegung des QoS-Levels

Der Subscriber bestimmt beim Abonnieren der Themen den gewünschten QoS-Level. Der Standard-Level ist 1, Kommandos erhalten automatisch den Level 3. Ein bestimmter, anzufordernder QoS-Level kann in der ProcessTypeInfo des Datenpunkttyps hinterlegt werden. Dazu wird das Patten

Q=1|2|3

verwendet. Beispielsweise kann ein QoS-Level durch den Eintrag Q=2 in ProcessTypeInfo festgelegt werden.

Beispiel

Das Beispielprojekt MQTTOnly2 befindet sich auf unserer Knowledge Base. In dem Server-Projekt sind die beiden Prozess-Ports Shelly und Sonoff projektiert. Bei letzterem ist der Eigenschaft „Service Configuration“ die Datei Sonoff.xml zugeordnet.

Das Projekt enthält zudem einen mobilen Client und ein Terminal-Projekt, mit denen Sie die Schaltvorgänge testen können.

Beachten Sie bitte: Das Beispiel funktioniert nur, wenn ein MQTT-Broker läuft und die „Host Name“- und „Port“-Eigenschaften den Broker adressieren. Ggf. sind u. a. auch Password und User Name einzustellen. Ein kostenfreier Broker steht in hier zur Verfügung: https://mosquitto.org/download/. Sie können auch den Broker des Elvis Viewer direct verwenden. Der Broker wird im Setting der App durch das Setzen des Flags „MQTT Server“ aktiviert.