MQTT Gateway - UDP Transformers

Die UDP-Transformer-Funktionalität ermöglicht die Änderung von Daten, die am MQTT Gateway per UDP eingehen und an MQTT weitergeleitet werden.

Mit einem Transformer können damit Daten, die vom Miniserver kommen, vor der Weiterleitung an MQTT geändert werden. Der Transformer entscheidet vollständig darüber, welche Daten schlussendlich an MQTT weitergegeben werden. Dabei kann der Transformer aufgrund von **einem** Eingangsdatensatz auch **mehrere** MQTT-Topics weitergeben.

Jeder Transformer ist konkret ein Script in beliebiger Programmiersprache (mit Shebang), das die Daten, die am MQTT Gateway per UDP eingegangen sind, übergeben bekommt, und dessen Output an MQTT weitergegeben wird.

Es gibt Transformer, die vom Plugin mitgeliefert werden, und man kann auch eigene Transformer anlegen:

Für Benutzer

Mitgelieferte und Benutzer-UDP-Transformer

Shipped: Das sind mitgelieferte Transformer, die im MQTT Gateway direkt verfügbar sind.

Custom: Das sind Beispiele von Benutzern. Diese musst du selbst aus dem Wiki herunterladen und ins Custom-Verzeichnis kopieren (siehe "Verzeichnisse der Transformer")

Verzeichnisse der Transformer

Typ	LoxBerry ab Version 3.0	MQTT Gateway bis Version 2.x
Shipped Transformer (mitgeliefert)	/opt/loxberry/bin/mqtt/transform/**shipped**/udpin	/opt/loxberry/data/plugins/mqttgateway/transform/**shipped**/udpin
Custom Transformer (für deine eigenen)	/opt/loxberry/bin/mqtt/transform/**custom**/udpin	/opt/loxberry/data/plugins/mqttgateway/transform/**custom**/udpin

Mitgelieferte Transformer

Folgende Transformer werden mit Loxberry mitgeliefert.

Custom/Shipped	Transformer-Name	Beschreibung	Link	Von
Shipped (V1.2.0)	php_textin_textout	Beispiel für Entwickler	Erklärung siehe unten	
Shipped (V1.2.0)	php_jsonin_jsonout	Beispiel für Entwickler	Erklärung siehe unten	
Shipped (V1.2.0)	php_jsonin_jsonarrayout	Beispiel für Entwickler	Erklärung siehe unten	

Shipped (V1.2.0)	shelly_rgb&w	Steuert Shelly RGBW-Geräte (z.B. RGBW2) direkt mit einem Loxone 0-100% (weiß) und RGB-Ausgang.	UDP Transformer - shelly_rgb&w	Christian Fenzl
Shipped (V2.0.0)	http2mqtt	Führt http/https-Requests aus und liefert Daten zurück.	UDP Transformer - http2mqtt (shipped)	Christian Fenzl
Shipped (V2.0.0)	moving_minmaxavg	Berechnet gleitende Mittelwerte, Minimum, Maximum, Summen	UDP Transformer - moving_minmaxavg (shipped)	Christian Fenzl
Shipped (V2.0.0)	sort	Sortiert übergebene Werte und gibt diese in gewisser Reihenfolge zurück	UDP Transformer - sort	Christian Fenzl

Eigene Transformer

Wenn du einen eigenen Transformer entwickelt hast, kannst du ihn hier veröffentlichen. Mach eine neue Zeile, und lege eine Seite mittels "Erstellen"-Button an mit einer kurzen Erklärung, wie man deinen Transformer benutzt.

Custom/Shipped	Transformer-Name	Beschreibung	Link	Von
Custom	php_execute	Führt PHP-Code aus, der direkt im VO-Befehl mitgegeben wird.	UDP Transformer - php_execute (custom)	Christian Fenzl
Custom	shelly_plus	Konvertieren eines einfachen UDP-Befehls in Shelly Plus MQTT JSON	UDP Transformer - shelly_plus (custom)	Daniel Kieslinger
Custom	zigbee_led	Konvertiert Lichtsteuerungssignale in einen MQTT JSON für Zigbee LEDs	UDP Transformer - zigbee_led (custom)	Andreas Ranalder

Einrichtung

- Du hast bereits einen Virtuellen Ausgang als UDP-Ausgang zum MQTT Gateway angelegt.

Die **normale** Syntax zum Senden im Virtuellen Ausgangsbefehl sieht so aus:

```
publish/retain <topic> <data>
```

z.B.

```
publish to/my/topic my_data

retain to/my/topic <v>
```

Ein **Transformer** wird hingegen folgendermaßen aufgerufen:

```
publish/retain <transformer> <topic> <data>
```

z.B.

```
publish shelly_rgb&w to/my/topic my_data

retain shelly_rgb&w to/my/topic <v>
```

Der `<transformer>` (im Beispiel `shellyrgb`) ist dabei der Name des Transformer-Scripts.

Eigenen Transformer entwickeln

UDP vs. Shell

Bitte denke beim Entwickeln und Testen daran: Der Aufruf des Scripts von der **Shell** (das ist, wie das Gateway dein Script mit Parametern aufruft) hat **andere Syntax** wie der Aufruf, den der Benutzer vom Miniserver aus per **UDP** an das Gateway sendet. Schaut euch die Syntax **genau** an! **Bash-Aufruf ist nicht gleich Benutzer-Aufruf!**

Die Ablage deiner Transformer –> Siehe oben unter "Verzeichnisse der Transformer".

- Das Verzeichnis der **custom**-Transformer wird von Updates nicht berührt, während das **shipped**-Verzeichnis durch Updates aktualisiert wird. Erstelle deine Transformer im **custom**-Verzeichnis!
- Unterordner werden unterstützt.
- Bei Transformern mit gleichem Namen wird immer jener bevorzugt, der im **custom**-Verzeichnis liegt.

Wie muss das Script heißen, wie wird es per UDP angesprochen

Scripte dürfen in genannten Ordnern und auch Unterordnern liegen. Das Script darf beliebig heißen und eine beliebige Dateierweiterung haben. Im Script muss der korrekte Shebang gesetzt sein.

Die Bezeichnung des Transformers per UDP ist:

- Der Dateiname OHNE Ordner und OHNE Dateierweiterung, UND
- alle Zeichen klein, UND
- Leerzeichen im Dateinamen werden durch Unterstrich (_) ersetzt.

Beispiele

Dateiname	Transformer-Name per UDP	Beispielaufruf im Virtuellen Ausgangs-Befehl
.../udpin/**shellyrgb**.pl	**shellyrgb**	publish **shellyrgb** send/to/my/topic <v>
.../udpin/Lumitech/**Lumitech RGB**.php	**lumitech_rgb**	publish **lumitech_rgb** send/to/my/topic <v>
.../udpin/**FHEM_SET**.sh	**fhem_set**	publish **fhem_send** send/to/my/topic <v>

Skills: Initialer Aufruf durch das Gateway

Das Gateway ruft **initial** das Transformer-Script **einmal** mit dem skills-Parameter auf:

Aufruf durch das Gateway: `shellyrgb.pl skills`

Das MQTT Gateway erwartet hier die zeilenweise Rückgabe von Informationen über die Kommunikation mit dem Transformer, in der Form `parameter=value`.

Keiner der skills-Parameter ist Pflicht - es wird dann immer der unten angegebene Standardwert verwendet. Auch wenn skills garnicht beantwortet wird, gelten die Standardwerte.

Folgende Parameter sind mit skills möglich:

Parameter	Standardwert	Mögliche Werte	Beschreibung
description	(leer)	Einzeiliger Text über den Transformer	Dies wird verwendet, um im Webinterface anzuzeigen, welche Transformer verfügbar sind und welche Funktion diese haben.
link	(leer)	https://.....	Link zu einer Erklärung des Transformers fürs Webinterface, z.B. ein Link ins LoxWiki.
input	text	text json	Die Übergabeparameter **an den** Transformer (json ist zu bevorzugen wegen etwaigem, standardisiertem Escaping) text: `topic#data` (Trennzeichen ist #) json: `{ "topic" : "data" }`
output	text	text json	Die **vom** Transformer erwartete Ausgabe an **STDOUT**. **text:** Mindestens eine, aber beliebig viele Zeilen Zeile 1: erstes/topic#data (Trennzeichen ist #, \n beendet den Datensatz) Zeile 2: zweites/topic#data **json:** Eine oder mehrere Topics im JSON-Format. Die Ausgabe muss nicht zwangsläufig einzeilig sein, der komplette Output wird ausgewertet. { "erstes/topic": "data", "zweites/topic" : "data" }, oder [{ "erstes/topic": "data", "zweites/topic" : "data" }] (Ausgabe als json Array: Die Reihenfolge bleibt beim Senden an MQTT erhalten!)

Parameter und Rückgabe

Das Format ist in der Skills-Tabelle angegeben.

Übergabe Gateway an Transformer

input=text: Aufruf durch das Gateway: `''%%shellyrgb.pl%%'' this/is/my/topic**#**here are`

`my data` *(Trennzeichen zwischen Topic und Daten ist das #-Zeichen)*

input=json: Aufruf durch das Gateway: ''%%shellyrgb.pl%%''`{ "this/is/my/topic" : "here are my data" }` *(beim Test aus der Bash: Anführungszeichen mit \" escapen!)*

Das MQTT-Gateway übergibt die Daten als Commandline-Parameter an das Transformer-Script mit der genannten Syntax.

Der skills-Parameter muss im Script zuerst berücksichtigt werden, und entsprechend geantwortet.

Ist der erste Parameter nicht `skills`, muss sich das Script darum kümmern, die Parameter zusammenzusetzen und entsprechend zu parsen (Splitten bei #, bzw. json-decoden).

Übergeben wird das `topic`, an das der Benutzer gesendet hat, sowie die Daten, die übermittelt wurden.

Rückgabe vom Transformer an das Gateway

output=text: Als Rückgabe erwartet vom Gateway

```
this/is/my/topic**#**here are my changed data
this/is/an/additional/topic**#**here are some more data
```
(optional mehrere Zeilen)

Bei text enthält jede Zeile ein Topic und einen Datensatz (mit #) getrennt.

output=json: Als Rückgabe erwartet das Gateway Json als "topic" : "value" Datensatz.

```
{
   "this/is/my/topic" : "here are my changed data",
   "this/is/an/additional/topic" : "here are some more data"
}
```
(optional mehrere Datensätze)

oder als Json-Array

```
**[**{
   "this/is/my/topic" : "here are my changed data",
   "this/is/an/additional/topic" : "here are some more data"
}**]
```
(optional mehrere Datensätze)

Bei json ist jeder Schlüssel der Name des Empfänger-Topics, und dessen Wert die Daten. Wird ein Json-*Array* geliefert, wird bei der Weiterleitung die Reihenfolge eingehalten. Allgemein:**

Die Rückgabe muss (aufgrund des übergebenen Topics) mit dem oder den vollständigen Topics erfolgen, an das die Daten an den MQTT-Broker weitergegeben werden.

Es ist also möglich, entweder das Topic zu belassen, und nur die Daten zu modifizieren. Es ist aber auch möglich, das Topic selbst anzupassen. Zudem ist es möglich, aus einem eingehenden Datensatz an mehrere MQTT-Topics verschiedene Daten zu senden.

Beachte!

- Das Script darf keine anderen Meldungen als die genannten ausgeben, deswegen jegliche andere Ausgabe an `>/dev/null` umleiten.
- Beim Input- und Output-Format `text` sind keine Zeilenumbrüche in den Daten möglich. Deswegen ist `json` zu bevorzugen.
- Wenn als Format `json` verwendet wird, und die zu übertragenen Daten sind selbst json, funktioniert das dennoch korrekt. Dein json-Parser kümmert sich um das korrekte Escaping.
- Wenn du Format json als Input verwendest, und dein Script von der Bash aus testest, dann musst du in der Commandline die Anführungszeichen im Json als \" escapen.
- Der Output des Transformers ersetzt vollständig die per UDP eingehenden Daten. Wenn der Transformer keine Daten zurückliefert, wird nichts an MQTT weitergeleitet.
- Die Laufzeit des Scripts verzögert die Laufzeit des MQTT Gateways insgesamt. Ein `sleep 1` im Transformerscript verzögert die gesamte Kommunikation um 1 Sekunde.
- Das MQTT Gateway erkennt das Hinzufügen und Entfernen von Transformerscripts automatisch nach etwa 5 Sekunden (kein Neustart des Gateways erforderlich), es erkennt jedoch keine Inhaltsänderung eines Scripts. Bei einer Änderung der Skills Zb von text auf json deswegen einmal das Script umbenennen, damit seine skills neu gelesen werden.
- Aktivitäten mit Transformern werden im Log protokolliert.

Beispiel-Outputs eines Transformers

skills-Aufruf

Der wird immer ausgeführt, wenn das MQTT-Gateway startet, oder (im laufenden Betrieb) wenn du einen Transformer erstellst oder umbenennst.

```
./php_textin_textout.php skills
description=Example of incoming text and outgoing text
link=https://www.loxwiki.eu/x/mQBABQ
input=text
output=text
```

Rückgabe bei output=text

Beispiel-Transformer transform/shipped/udpin/examples/php_textin_textout.php

```
./php_textin_textout.php my/topic#Meine Daten
my/topic#Meine Daten (10:03:46)
my/topic/time#10:03:46
my/topic/date#02.07.21
```

Das Gateway führt diesen Aufruf aus: `./php_textin_textout.php my/topic#Meine Daten`

Der Benutzer führt per UDP (vom Miniserver aus) diesen Aufruf aus:
`publish php_textin_textout my/topic Meine Daten`

Rückgabe bei output=json

Der JSON-Output kann in zwei Varianten zurückgegeben werden: Als reines Key/Value-Dataset, oder als Array mit Key/Value-Daten.

Die Rückgabe eines reinen Key/Value-Datasets ist einfacher, dabei werden die Daten alphabetisch nach Topic sortiert bei der Weitergabe an MQTT:

Beispiel-Transformer transform/shipped/udpin/examples/php_jsonin_jsonout.php

```
./php_jsonin_jsonout.php { \"my/topic\" : \"data\" }
{"my\/topic":"data
(10:07:04)","my\/topic\/time":"10:07:04","my\/topic\/date":"02.07.21"}
```

Um die Reihenfolge bei der Weitergabe an MQTT einzuhalten, kann das JSON-Dataset als Array zurückgegeben werden:

Beispiel-Transformer transform/shipped/udpin/examples/php_jsonin_jsonarrayout.php

```
./php_jsonin_jsonarrayout.php { \"my/topic\" : \"data\" }
[{"my\/topic":"data
(10:08:49)"},{"my\/topic\/time":"10:08:49"},{"my\/topic\/date":"02.07.21"}]
```

Das Gateway führt in beiden Fällen diesen Aufruf aus: `./php_jsonin_jsonout.php { "my/topic" : "Meine Daten" }`

Der Benutzer führt per UDP (vom Miniserver aus) diesen Aufruf aus:
`publish php_jsonin_jsonout my/topic Meine Daten`