

msg Gate Filter

<https://groups.google.com/forum/#!topic/node-red/f0Yv9VEmsDk>

Ein Gate Filter soll eine msg durchlassen, wenn eine/mehrere Bedingungen erfüllt sind. Es gibt viele Möglichkeiten dies zu realisieren.

mit Standard-nodes ohne zusätzliche Programmierung

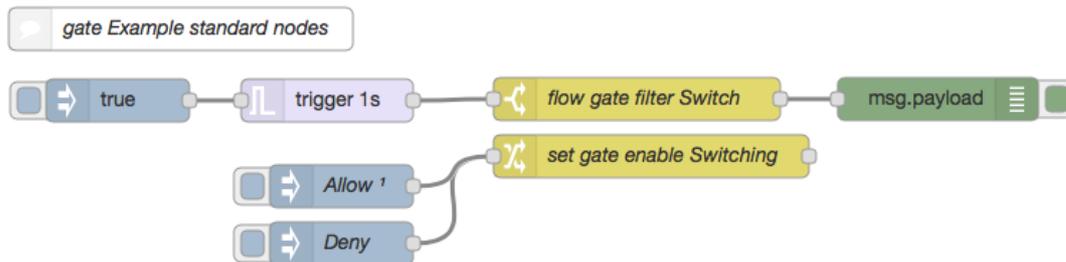


Bild 1: Gate Beispiel mit Standard nodes

Über den change-node wird eine flow Variable (gate.switch) entsprechend auf true/false gesetzt. Das switch-node sorgt dafür, dass die eingehende msg nur an den Ausgang weitergeleitet wird, wenn gate.switch true ist.

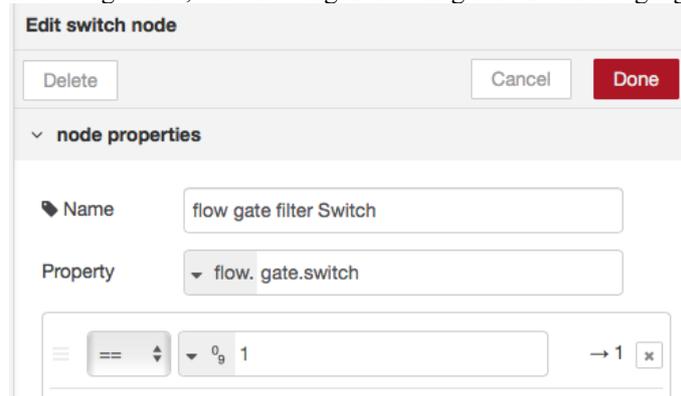


Bild 2: gate mit switch-node

Hinweis:

Die flow-Variablen werden beim flow-Start nicht initialisiert, was einen Fehler zur Folge hat. Abhilfe kann man schaffen, indem man beim flow-Start den inject-node mit dem gewünschten Startwert über die Option **Inject once after xx seconds** injiziert.

Vorteil dieses Verfahrens:

- Kommt i.d.R. ohne Programmierung mit 2 core-nodes aus.
- Gate setzen/sperrern vollständig getrennt vom Gate-node

Nachteil

- komplexere Bedingungen für die Gatesteuerung benötigen evtl. zusätzliche nodes

mit function-node

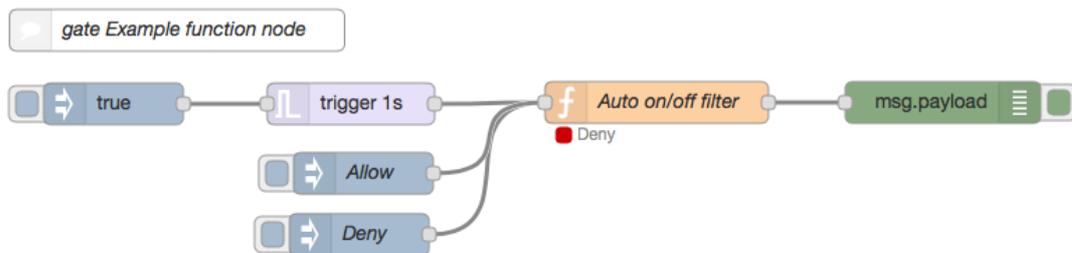


Bild 3: gate mit function-node

Die Version kommt mit einem node aus, benötigt aber entsprechende Programmierung im functions-node.

Edit function node

Delete Cancel **Done**

▼ **node properties**

Name

Auto on/off filter

✎ **Function**

```

1 //set default gateState
2 if (typeof context.get("gateState") === "undefined") context.set("gateState","0");
3
4 //if topic = "1" or "0" -> only gateState switch
5 if (msg.topic == "1" || msg.topic == "0") {
6     //save gateState
7     context.set("gateState",msg.topic);
8     return null;
9 }
10
11 let gateState = context.get("gateState");
12 let sFill = (gateState == "0") ? "red" : "green";
13 let sText = (gateState == "0") ? "Deny" : "Allow";
14
15 node.status({fill:sFill,shape:"dot",text:sText});
16
17 if (gateState == "1") {
18     return msg
19 } else {
20     return null;
21 }
22

```

Bild 4: gate Funktionalität im function-node

Prinzipiell arbeitet diese Version ähnlich der switch-node Version. Es vereinigt allerdings die komplette Funktionalität in einem functions-node und ist dadurch entsprechend flexibler und kommt mit einer node context Variablen aus.

Die Initialisierung der context Variablen erfolgt hier natürlich direkt im node. über node.status kann hier zudem der aktuelle gate-Status am node angezeigt werden.

Eine eingehende msg wird nur weitergeleitet, wenn die context Variable gateState gesetzt ist und msg.topic weder „1“ noch „0“ ist, was sicherstellen soll, dass die msg zum setzen/sperren des gate selbst nicht weitergeleitet wird.

Vorteil ist ganz klar, die höhere Flexibilität, da im function-node entsprechende Abhängigkeiten der gate-Steuerung programmiert werden können.

Nachteilig ist evtl. der Programmieraufwand.

mit contib nodes

<https://flows.nodered.org/node/node-red-contrib-traffic>

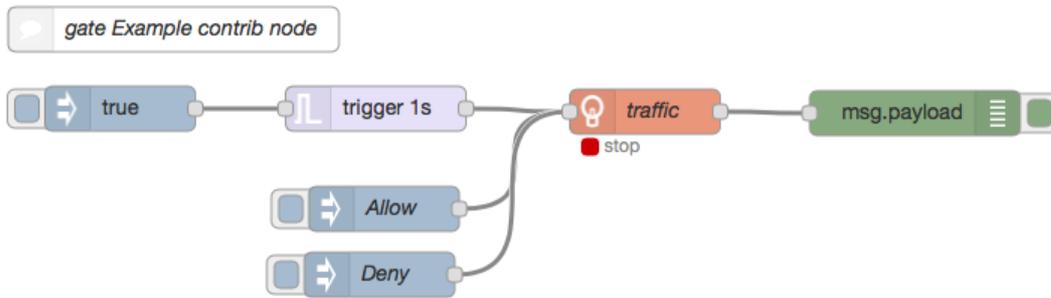


Bild 5: gate mit node-red-contrib-traffic

Das contrib node ist sehr sehr flexibel flexibel konfigurierbar. Damit sollten sich die allermeisten Anforderungen an ein gate verwirklichen lassen. Es arbeitet intern ebenfalls nur mit einer node context Variablen.

Delete
Cancel Done

▼ **node properties**

📁 Name

▶ Allow

☰ Property

▼ Regex

Ignore Case Negate

Resend

■ Stop

☰ Property

▼ Regex

Ignore Case Negate

Resend

Allow by default Store unsent

Bild 6: Konfiguration nore-red-contrib-traffic

node-red-contrib-timeframerlt (timeframe rate limit trigger)
 (siehe Fehler! Verweisquelle konnte nicht gefunden werden.)