第七讲
ThingsBoard（IV）
Lecture 7
ThingsBoard (IV)

明玉瑞 Yurui Ming

yrming@gmail.com

声明
Disclaimer

规则引擎
Rule Engine

- ThingsBoard里面的另外一个重要概念是规则引擎。规则引擎是一个高度可自定义和可配置的系统，用于复杂的事件处理。借助规则引擎，用户可以过滤、丰富和转换由IoT设备和相关资产发起的传入消息。用户还可以触发各种操作，例如，通知或与外部系统的通信。一言以蔽之，规则引擎是一个易于使用的框架，用于构建基于事件的工作流。

  Another important concept in ThingsBoard is the rule engine. Rule engine is a highly customizable and configurable system for complex event processing. With rule engine you are able to filter, enrich and transform incoming messages originated by IoT devices and related assets. You are also able to trigger various actions, for example, notifications or communication with external systems. To sum it up, rule engine is an easy to use framework for building event-based workflows.

规则引擎
Rule Engine

- 规则引擎有3个主要组件组成：
  - 消息—任何传入事件。它可以是来自设备的传入数据、设备生命周期事件、REST API事件、RPC请求等。
  - 规则节点—对传入消息执行的函数。有许多不同的节点类型可以过滤，转换或对传入消息执行某些操作。
  - 规则链—节点通过关系相互连接，因此来自规则节点的出站消息将发送到下一个连接的规则节点。
- There are 3 main components in rule engine:
  - Message - any incoming event. It can be an incoming data from devices, device life-cycle event, REST API event, RPC request, etc.
  - Rule Node - a function that is executed on an incoming message. There are many different Node types that can filter, transform or execute some action on incoming Message.
  - Rule Chain - nodes are connected with each other with relations, so the outbound message from rule node is sent to next connected rule nodes.

规则引擎
Rule Engine

- 规则引擎消息是一种可序列化的、不可变的数据结构，表示系统中的各种消息。例如：
  - 来自设备的传入遥测、属性更新或RPC调用；
  - 实体生命周期事件：创建、更新、删除、分配、未分配、属性更新；
  - 设备状态事件：已连接、已断开连接，活动、非活动等；
  - 其他系统事件。
- Rule Engine Message is a serializable, immutable data structure that represent various messages in the system. For example:
  - Incoming telemetry, attribute update or RPC call from device;
  - Entity life-cycle event: created, updated, deleted, assigned, unassigned, attributes updated;
  - Device status event: connected, disconnected, active, inactive, etc;
  - Other system events.

规则引擎
Rule Engine

- 规则引擎消息包含以下信息：
  - 消息ID：基于时间，通用唯一标识符；
  - 消息的发起人：设备、资产或其他实体标识符；
  - 消息类型：“发布遥测”或“不活动事件”等；
  - 消息的有效负载：具有实际消息有效负载的 JSON 正文；
  - 元数据：键值对的列表，其中包含有关消息的其他数据。
- Rule Engine Message contains the following information:
  - Message ID: time based, universally unique identifier;
  - Originator of the message: Device, Asset or other Entity identifier;
  - Type of the message: “Post telemetry” or “Inactivity Event”, etc;
  - Payload of the message: JSON body with actual message payload;
  - Metadata: List of key-value pairs with additional data about the message.

规则引擎
Rule Engine

- 规则节点是规则引擎的基本组件，它一次处理单个传入消息并生成一个或多个传出消息。规则节点是规则引擎的主逻辑单元。规则节点可以过滤、丰富、转换传入消息、执行操作或与外部系统通信。

  Rule Node is a basic component of Rule Engine that process single incoming message at a time and produce one or more outgoing messages. Rule Node is a main logical unit of the Rule Engine. Rule Node can filter, enrich, transform incoming messages, perform action or communicate with external systems.

- 规则节点可能与其他规则节点相关。每个关系都有关系类型，这是一个用于标识关系的逻辑含义的标签。当规则节点生成传出消息时，它始终指定用于将消息路由到下一个节点的关系类型。

  Rule Nodes may be related to other rule nodes. Each relation has relation type, a label used to identify logical meaning of the relation. When rule node produces the outgoing message it always specifies the relation type which is used to route message to next nodes.

规则引擎
Rule Engine

- 规则节点都根据其性质可分为以下类型：
  - 用于消息过滤和路由的过滤器节点;
  - 用于更新传入消息的元数据的扩充节点;
  - 用于更改传入的消息字段，如发起者，类型，有效负载，元数据的转换节点;
  - 根据传入的消息执行各种操作的操作节点;
  - 用于与外部系统交互的外部节点。
- Rule nodes are grouped into different types in correspondence with their nature:
  - Filter Nodes are used for message filtering and routing;
  - Enrichment Nodes are used to update meta-data of the incoming Message;
  - Transformation Nodes are used for changing incoming Message fields like Originator, Type, Payload, Metadata;
  - Action Nodes execute various actions based on incoming Message;
  - External Nodes are used to interact with external systems.

规则引擎
Rule Engine

- 典型的规则节点关系是“成功”和“失败”。表示逻辑操作的规则节点可以使用“正确”或“错误”。某些特定规则节点可能使用完全不同的关系类型，例如：“发布遥测数据”、“属性已更新”、“已创建实体”等。

  Typical rule node relations are “Success” and “Failure”. Rule nodes that represent logical operations may use “True” or “False”. Some specific rule nodes may use completely different relation types, for example: “Post Telemetry”, “Attributes Updated”, “Entity Created”, etc.

- 规则链是规则节点及其关系的逻辑组。租户管理员能够定义一个根规则链和多个其他规则链（可选）。根规则链处理所有传入消息，并可能将它们转发到其他规则链以进行其他处理。其他规则链也可能将消息转发到不同的规则链。

  Rule Chain is a logical group of rule nodes and their relations. Tenant administrator is able to define one Root Rule Chain and optionally multiple other rule chains. Root rule chain handles all incoming messages and may forward them to other rule chains for additional processing. Other rule chains may also forward messages to different rule chains.

规则引擎
Rule Engine

- 消息处理有三种可能的结果：成功、失败和超时。当处理链中的最后一个规则节点成功处理消息时，消息处理尝试将标记为“成功”。如果其中一个规则节点生成消息处理“失败”，并且没有规则节点来处理该失败，则消息处理尝试将标记为“失败”。当处理的总时间超过可配置的阈值时，消息处理尝试将标记为“超时”。

  There are three possible results of message processing: Success, Failure and Timeout. The message processing attempt is marked as “Success” when the last rule node in the processing chain successfully process the message. The message processing attempt is marked as “Failure” if one of the rule nodes produce “Failure” of message processing, and there is no rule nodes to handle that failure. The message processing attempt is marked as “Timeout” when overall time of processing exceed configurable threshold.

典型应用
Typical Use Cases

- 在保存到数据库之前对传入的遥测数据或属性进行验证和修改。
- 将遥测数据或属性从设备复制到相关资产进行聚合。例如，将来自多个设备的数据聚合到相关资产中。
- 根据定义的条件创建/更新/清除警报。
- 根据设备生命周期事件触发操作。 例如，根据设备在线/离线状态创建警报。
- 加载处理所需的附加数据。例如，在设备的客户或租户属性中定义的设备的负载温度阈值。
- 触发对外部系统的REST API调用。
- 发生复杂事件时发送电子邮件，并使用电子邮件模板中其他实体的属性。
- 在事件处理期间考虑用户偏好。
- 根据定义的条件进行 RPC 调用。
- 与 Kafka、Spark、AWS 服务等外部管道集成。

典型应用
Typical Use Cases

- Data validation and modification for incoming telemetry or attributes before saving to the database.
- Copy telemetry or attributes from devices to related assets so you can aggregate telemetry. For example data from multiple devices can be aggregated in related Asset.
- Create/Update/Clear alarms based on defined conditions.
- Trigger actions based on device life-cycle events. For example, create alerts if Device is Online/Offline.
- Load additional data required for processing. For example, load temperature threshold value for a device that is defined in Device’s Customer or Tenant attribute.
- Trigger REST API calls to external systems.
- Send emails when complex event occurs and use attributes of other entities inside Email Template.
- Take into account User preferences during event processing.
- Make RPC calls based on defined condition.
- Integrate with external pipelines like Kafka, Spark, AWS services, etc.

入门示例
Hello-World Example

- 在本教程中，假设设备正在使用DHT22传感器来收集温度并将数值推送到ThingsBoard，DHT22传感器可以测量-40°C至+80°C的温度。

  In this tutorial, let's assume the device is using DHT22 sensor to collect and push temperature to the ThingsBoard. DHT22 sensor can measure temperature from -40°C to +80°C.

- 我们将配置ThingsBoard规则引擎以将所有温度存储在-40至80°C范围内，并将所有其他读数记录到系统日志中。

  We will configure ThingsBoard Rule Engine to store all temperature within -40 to 80°C range and log all other readings to the system log.

- 在配置规则节点时，可以使用TBEL（ThingsBoard表达式语言）或JavaScript来开发用户定义的函数。我们建议使用TBEL，因为它在ThingsBoard中的执行比JS更有效率。

  When configure the rule node, one can use either TBEL (ThingsBoard expression language) or JavaScript to develop user defined functions. We recommend utilizing TBEL as it's execution in ThingsBoard is much more efficient compared to JS.

入门示例
Hello-World Example

- 在本教程中，假设设备正在使用DHT22传感器来收集温度并将数值推送到ThingsBoard，DHT22传感器可以测量-40°C至+80°C的温度。

  In this tutorial, let’s assume the device is using DHT22 sensor to collect and push temperature to the ThingsBoard. DHT22 sensor can measure temperature from -40°C to +80°C.

- 我们将配置ThingsBoard规则引擎以将所有温度存储在-40至80°C范围内，并将所有其他读数记录到系统日志中。

  We will configure ThingsBoard Rule Engine to store all temperature within -40 to 80°C range and log all other readings to the system log.

- 在配置规则节点时，可以使用TBEL（ThingsBoard表达式语言）或JavaScript来开发用户定义的函数。我们建议使用TBEL，因为它在ThingsBoard中的执行比JS更有效率。

  When configure the rule node, one can use either TBEL (ThingsBoard expression language) or JavaScript to develop user defined functions. We recommend utilizing TBEL as it’s execution in ThingsBoard is much more efficient compared to JS.

入门示例
Hello-World Example

- 要添加温度验证节点，请在 Thingsboard UI 中转到规则链部分并打开根规则链。

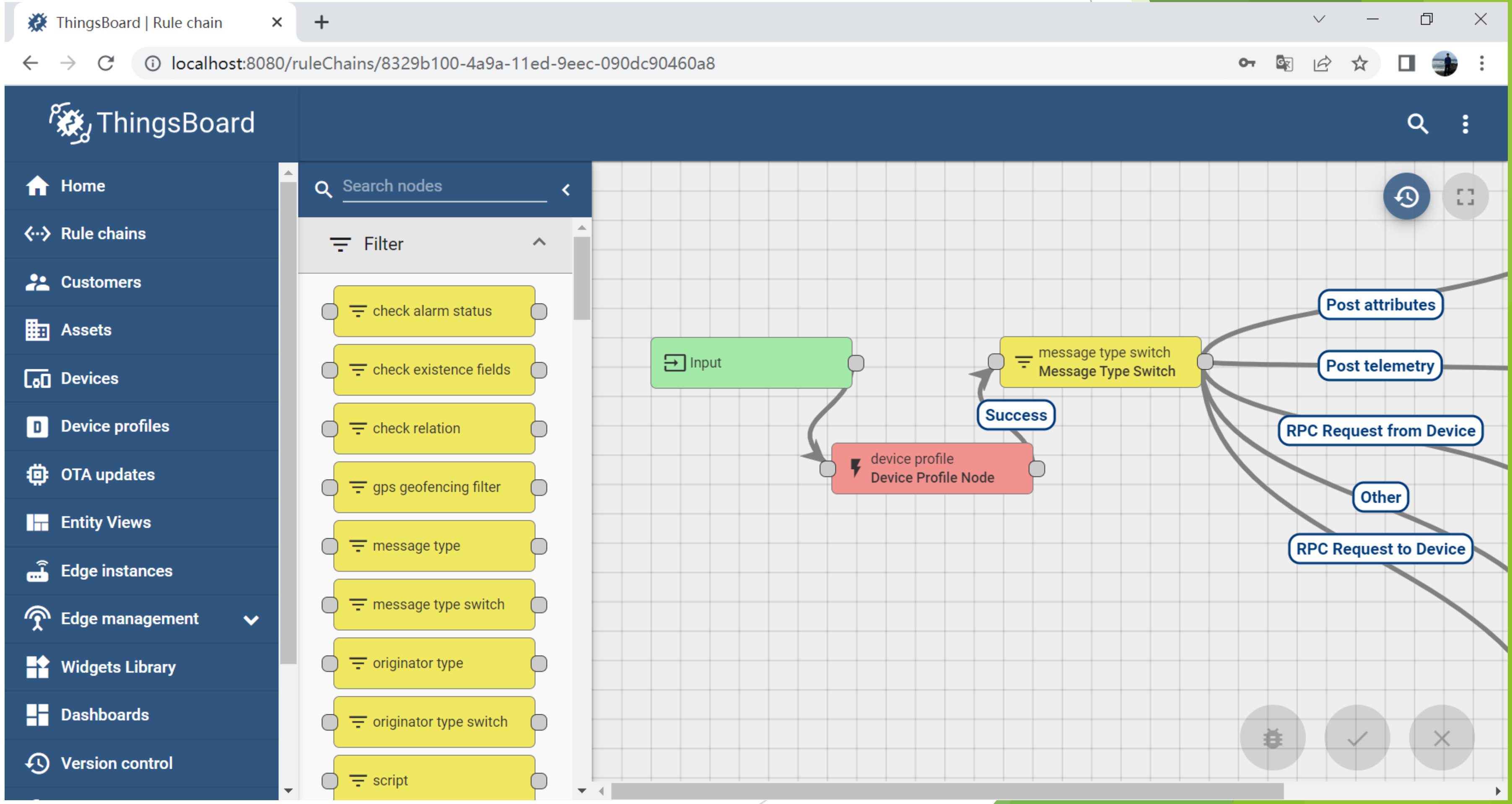
  To add temperature validation node, in Thingsboard UI go to Rule Chains section and open Root Rule Chain.

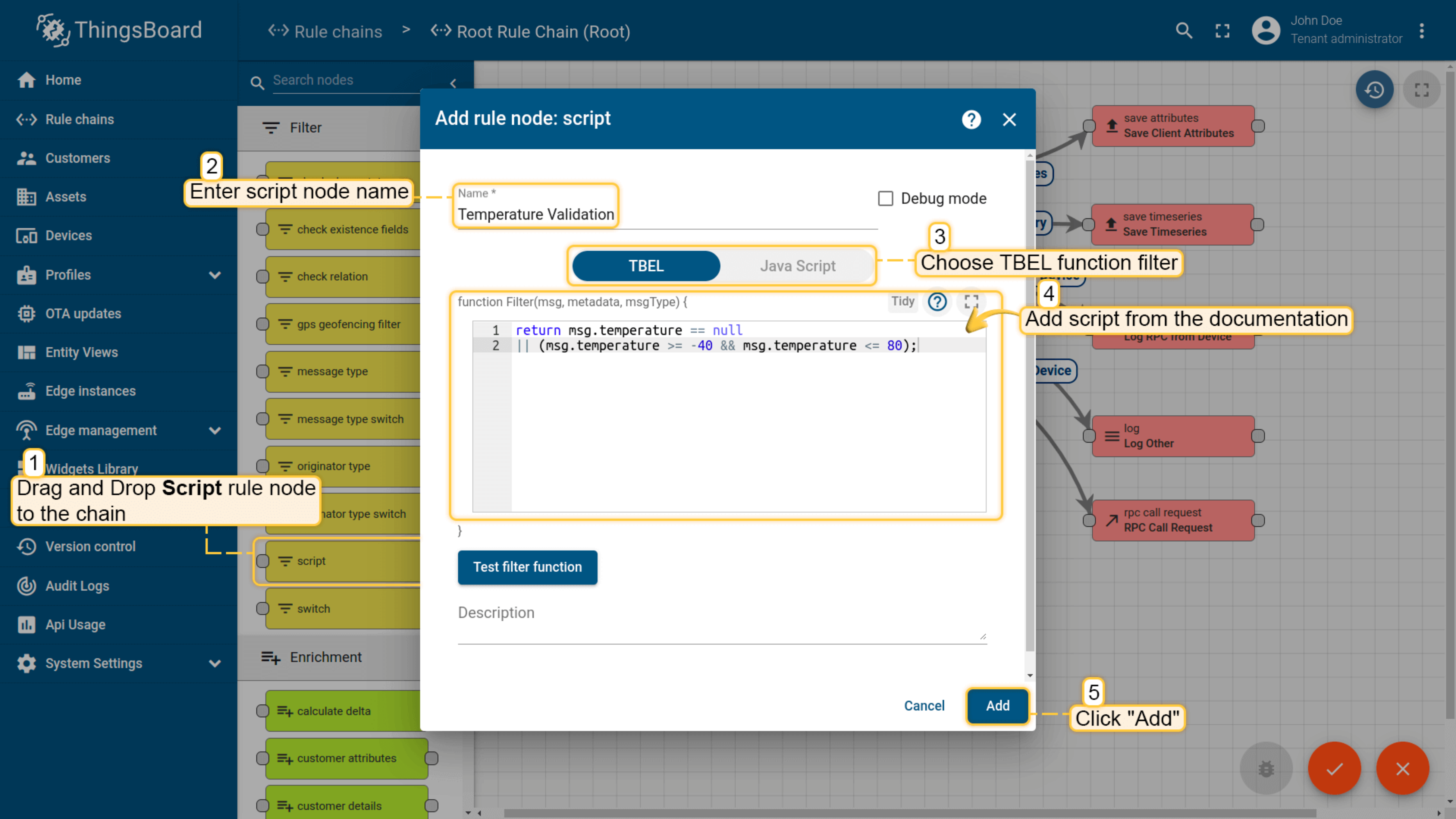
- 我们将使用此脚本进行数据验证：

  We will use this script for data validation:

  return msg.temperature == null

  || (msg.temperature >= -40 && msg.temperature <= 80);





1 Drag and Drop **Script** rule node to the chain

2 Enter script node name

3 Choose TBEL function filter

4 Add script from the documentation

5 Click "Add"

Add rule node: script

Name *: Temperature Validation

Debug mode

TBEL | Java Script

```
function Filter(msg, metadata, msgType) {
return msg.temperature == null
|| (msg.temperature >= -40 && msg.temperature <= 80);
}
```

Tidy

Test filter function

Description

Cancel | Add

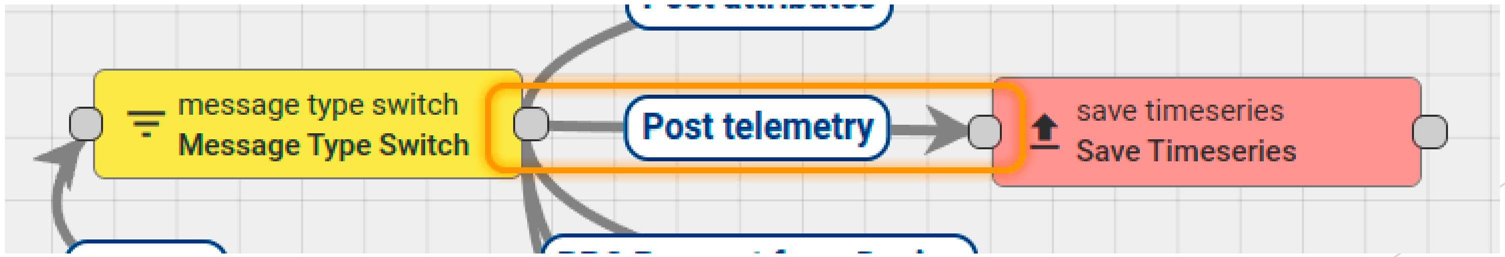
入门示例
Hello-World Example

- 如果未定义温度属性或温度有效，脚本将返回 True，否则将返回 False。如果脚本返回True，则传入消息将路由到与True关系连接的下一个节点。

  If temperature property not defined or temperature is valid, script will return True, otherwise it will return False. If script returns True, incoming message will be routed to the next nodes that are connected with True relation.

- 现在，我们希望所有遥测请求都通过此验证脚本。为此，我们需要删除消息类型交换节点和遥测数据保存节点之间的名称为“遥测后”的关系：

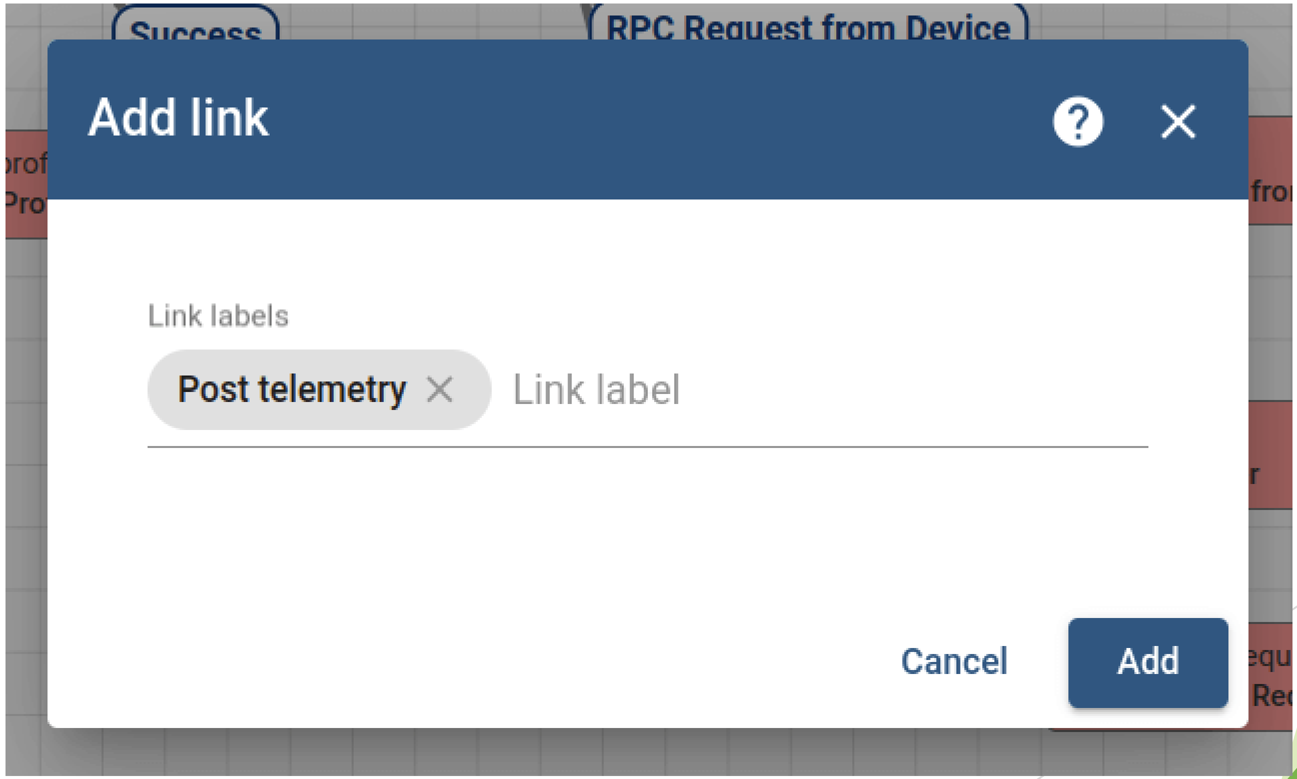
  Now we want that all telemetry requests pass through this validation script. For this we need to remove the existing Post Telemetry relation between Message Type Switch node and Save Telemetry node:



入门示例
Hello-World Example

- 接下来，使用遥测后关系将消息类型交换节点与脚本筛选器节点连接：

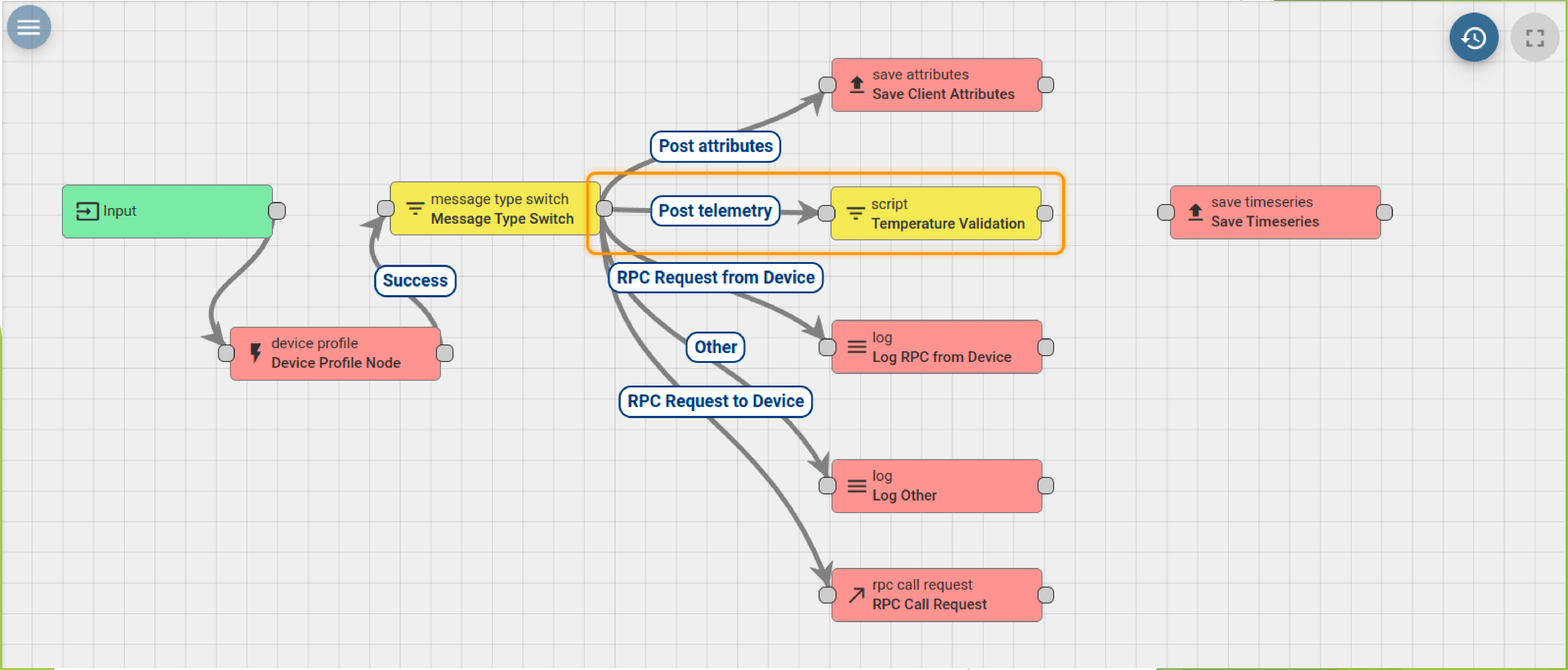
  Now connect Message Type Switch node with Script Filter node using Post Telemetry relation:


Add link

Link labels

Post telemetry ✕ Link label

入门示例 Hello-World Example



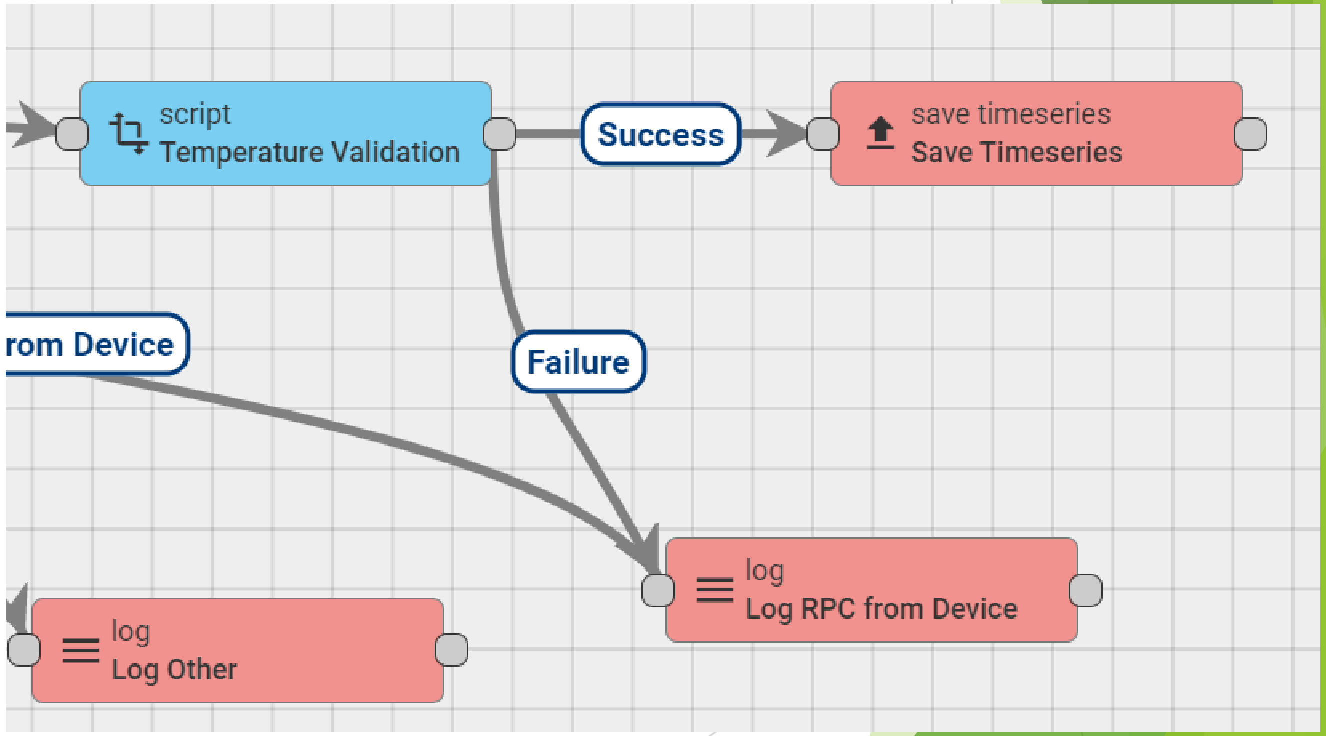
入门示例
Hello-World Example

- 接下来，我们需要使用成功关系将“脚本筛选器”节点与“保存遥测数据”节点连接起来。因此，所有有效的遥测数据将被保存。

  Next, we need to connect Script Filter node with Save Telemetry node using Success relation. So all valid telemetry will be saved.

- 同时，我们将使用失败关系将脚本过滤器节点与“记录其他”节点连接起来，以便所有无效的遥测都将记录在系统日志中：

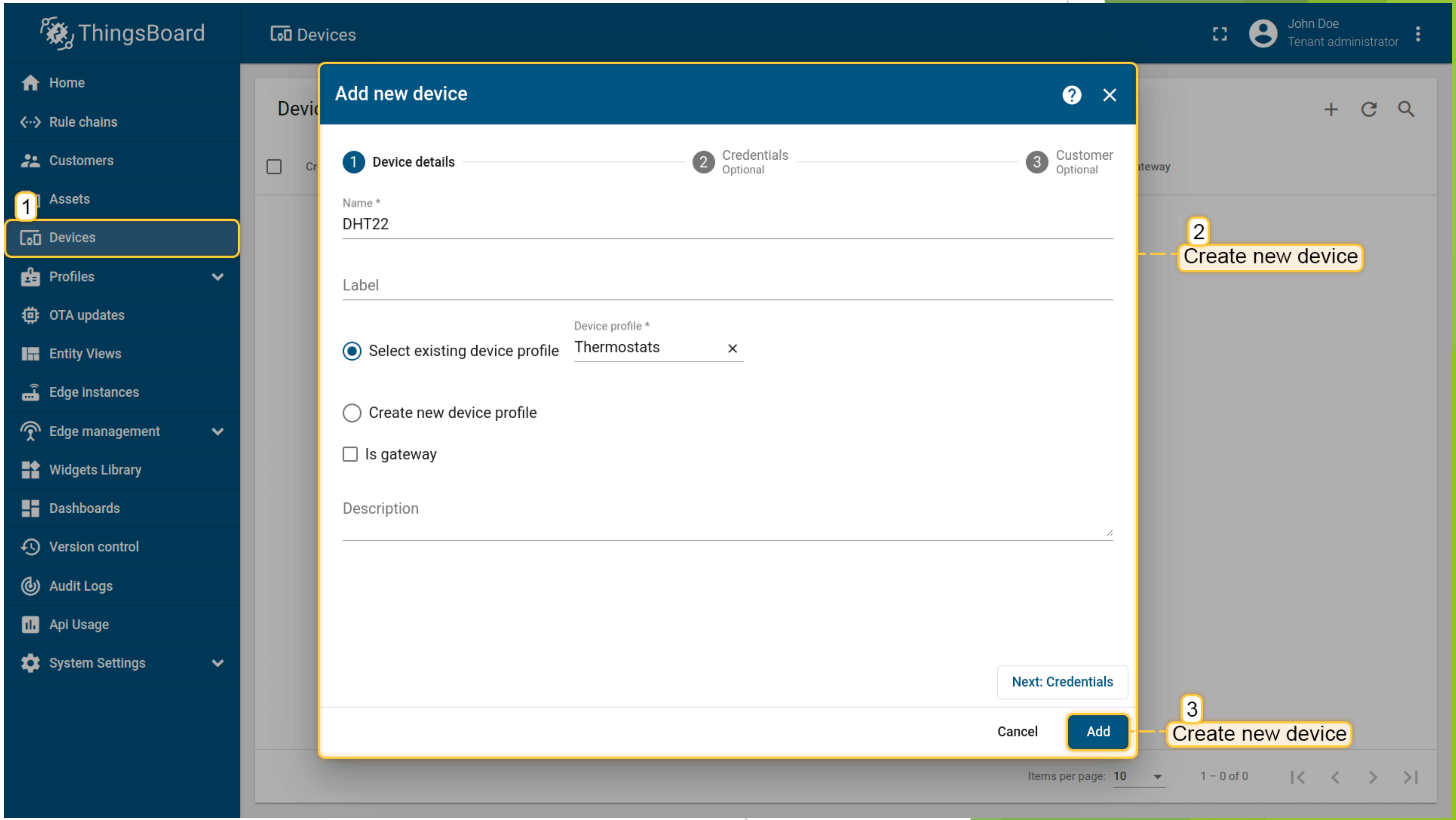
  Also, we will connect Script Filter node with Log Other node using Failure relation. So that all not valid telemetry will be logged in the system log:



入门示例 Hello-World Example

- 按保存按钮应用更改，然后创建设备并将遥测数据提交到事物板以便验证结果。转到“设备”部分并创建新设备：

  Press Save button to apply changes. For validating results we will need to create Device and submit telemetry to the Thingsboard. So go to Devices section and create new Device:



Name * DHT22

Label

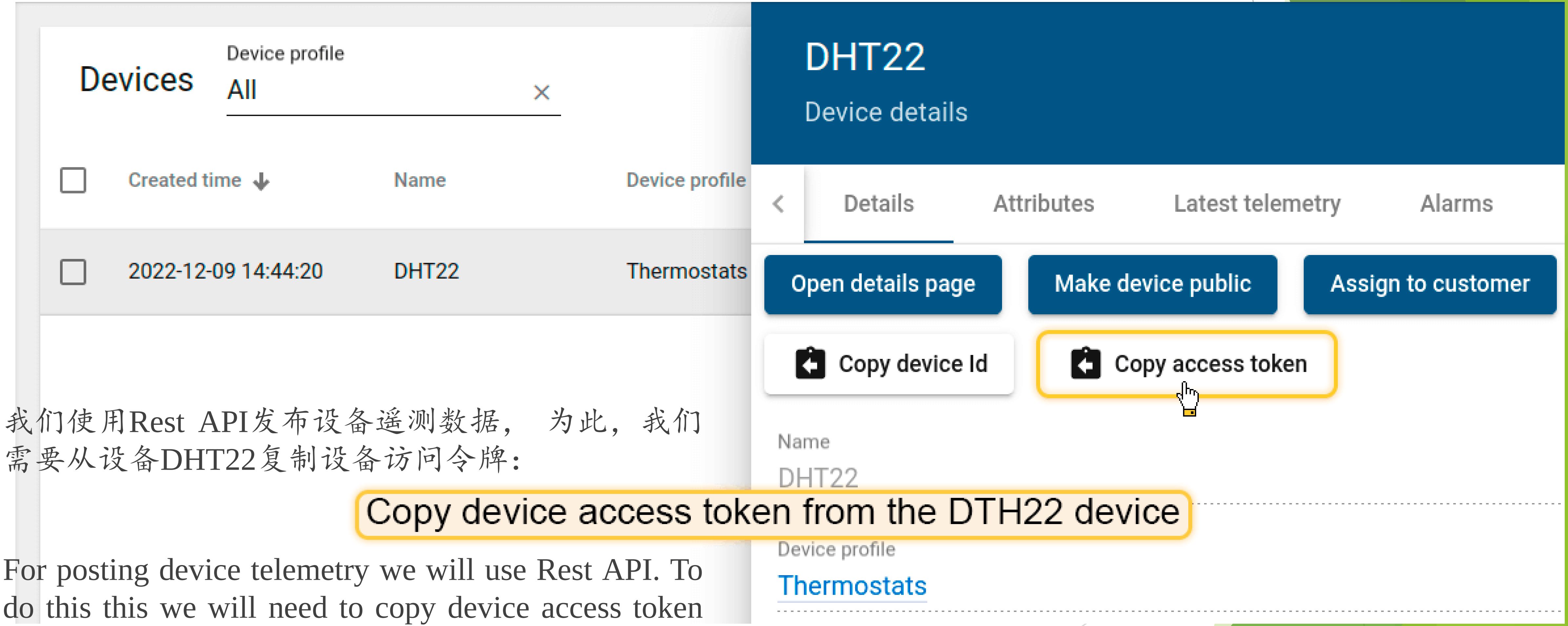
[✓ Select existing device profile] Device profile * Thermostats

Create new device profile

☐ Is gateway

Description

入门示例 Hello-World Example



Devices

Device profile: All

	Created time	Name	Device profile
☐	2022-12-09 14:44:20	DHT22	Thermostats

DHT22

Device details

Name: DHT22

Device profile: Thermostats

Copy device access token from the DTH22 device

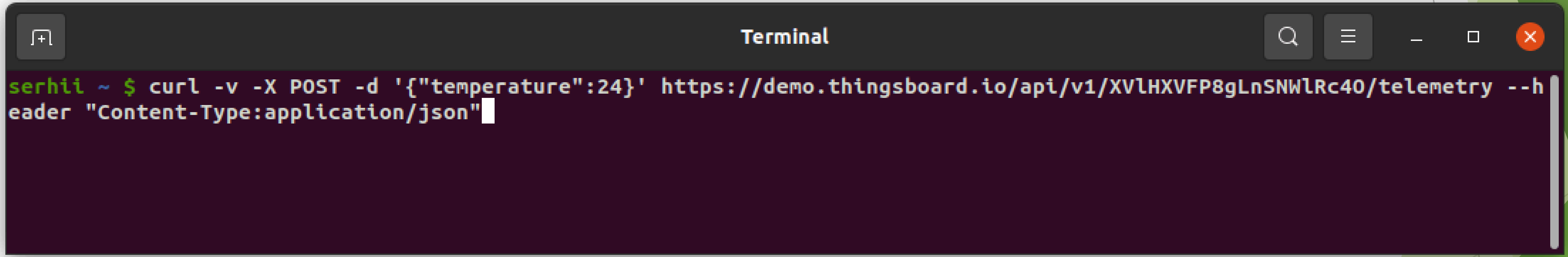
- 我们使用Rest API发布设备遥测数据， 为此，我们需要从设备DHT22复制设备访问令牌：

For posting device telemetry we will use Rest API. To do this this we will need to copy device access token from the device DHT22:

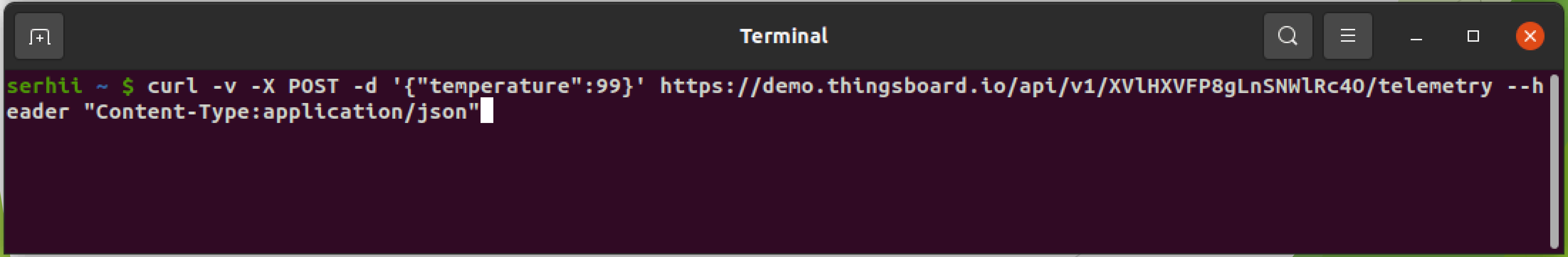
入门示例
Hello-World Example

- 使用终端分别将发送温度读数为24与99的消息。注意将$ACCESS_TOKEN替换为实际的设备令牌。

  Use terminal for will send a message with temperature readings equals to 24 and 99 respectively. Note to replace $ACCESS_TOKEN with actual device token.

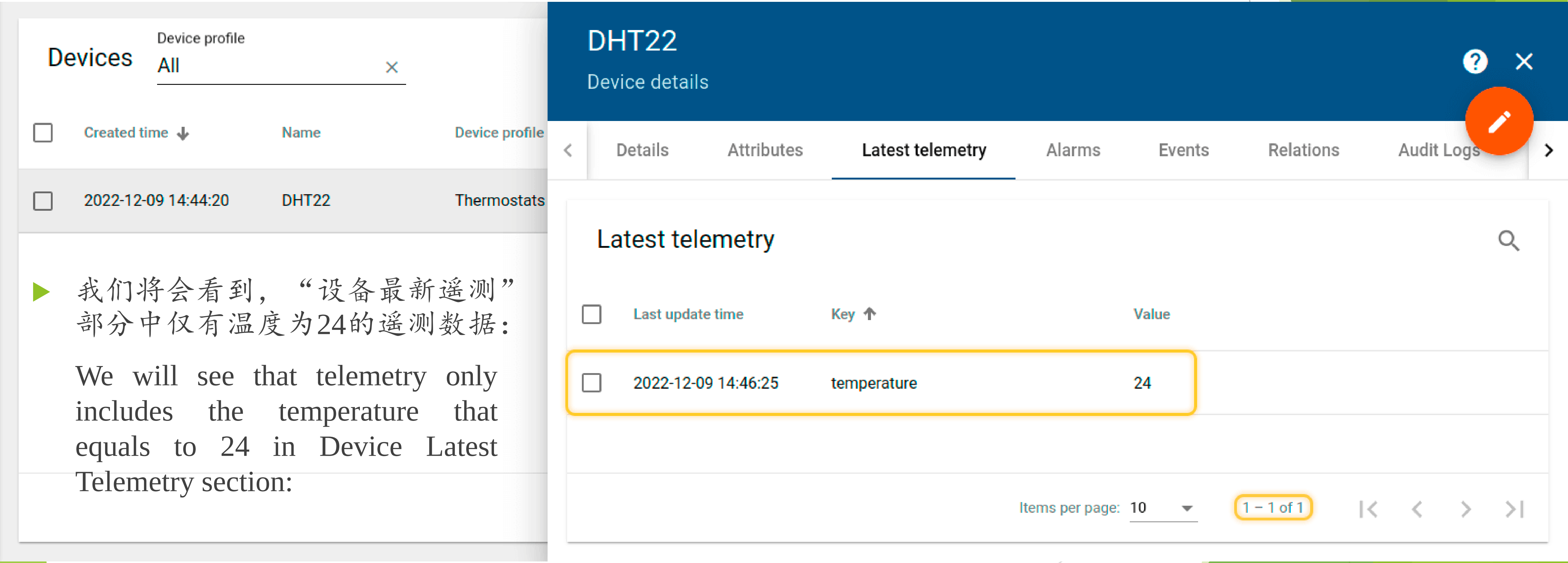


```
serhii ~ $ curl -v -X POST -d '{"temperature":24}' https://demo.thingsboard.io/api/v1/XVlHXVFP8gLnSNWlRc4O/telemetry --h
eader "Content-Type:application/json"
```



```
serhii ~ $ curl -v -X POST -d '{"temperature":99}' https://demo.thingsboard.io/api/v1/XVlHXVFP8gLnSNWlRc4O/telemetry --h
eader "Content-Type:application/json"
```

入门示例 Hello-World Example



Devices

Device profile: All

Created time	Name	Device profile
2022-12-09 14:44:20	DHT22	Thermostats

DHT22

Device details

Latest telemetry

Last update time	Key	Value
2022-12-09 14:46:25	temperature	24

Items per page: 10   1 – 1 of 1

- 我们将会看到，“设备最新遥测”部分中仅有温度为24的遥测数据：

  We will see that telemetry only includes the temperature that equals to 24 in Device Latest Telemetry section: