MuleSoft | Connect anything. Change everything.

HTTP Pub Sub Webhooks for Evented APIs

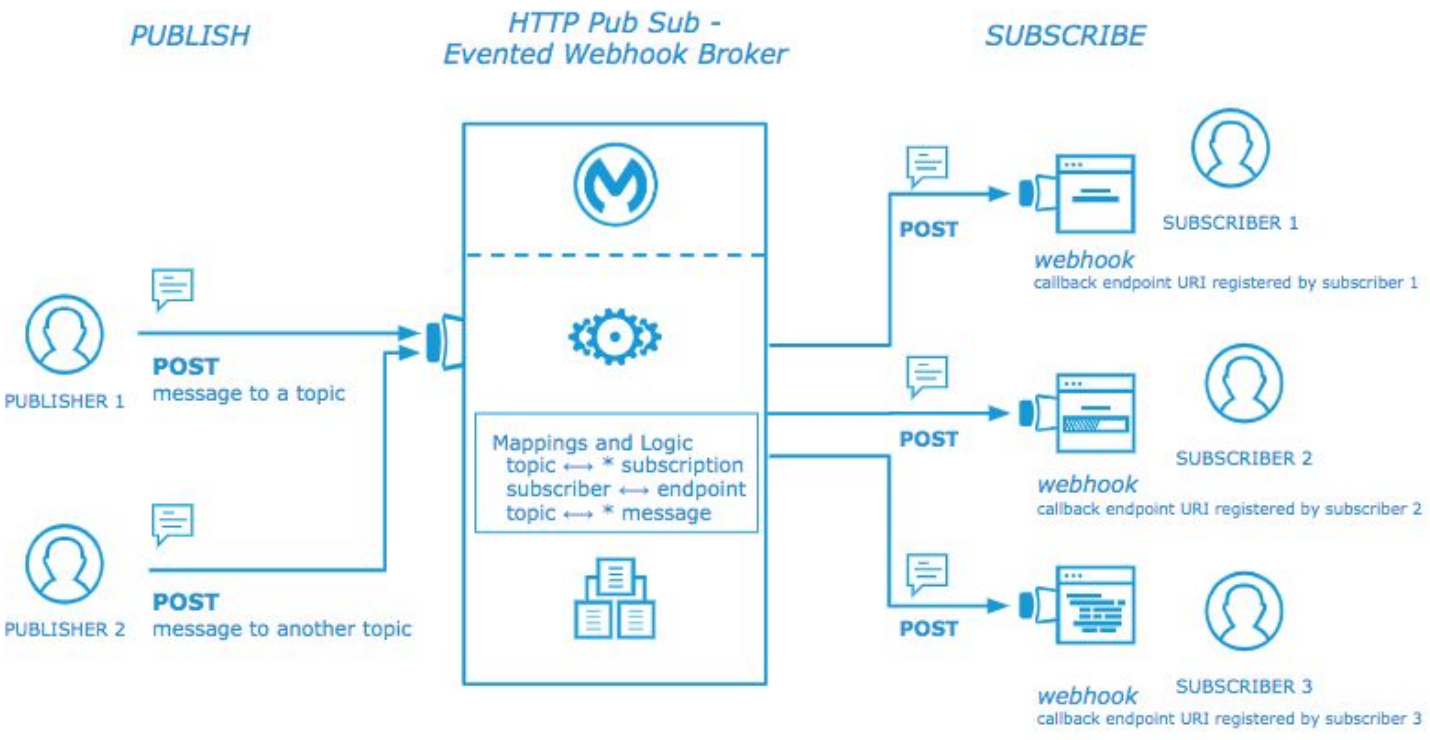
A Muley Point of View for Design and Deployment
nick.schott@mulesoft.com
January 2020

Why use Webhooks for Pub Sub Use Cases?

Evented APIs provide scalable pub sub messaging paradigms across a variety of messaging protocols. As standards evolve, many software development organizations are utilizing webhooks as a delivery mechanism for evented messaging. Akin to the value delivered by the prevalence of REST APIs in the last decade (representational state transfer based on methods already defined in the HTTP spec), this approach utilizes messaging patterns and protocols already supported by a wide array of development tools. Adherence to common standards becomes the only requirement to conform to this approach, subscribe to a topic, publish a message, and consume a message. Further, the ability to utilize a combination of synchronous and asynchronous aspects maintains the benefits of other async queueing protocols while facilitating development with existing standards. A considerable benefit of this approach is that webhooks provide an model for sending messages to subscribing clients on-demand via HTTP. This approach scales well over networks including the Internet, with common security models and HTTPS transport that can be utilized from any technology stack.

What does this look like?

Here is a representation of the services in play:



When should this approach be utilized?

Benefits of this approach:

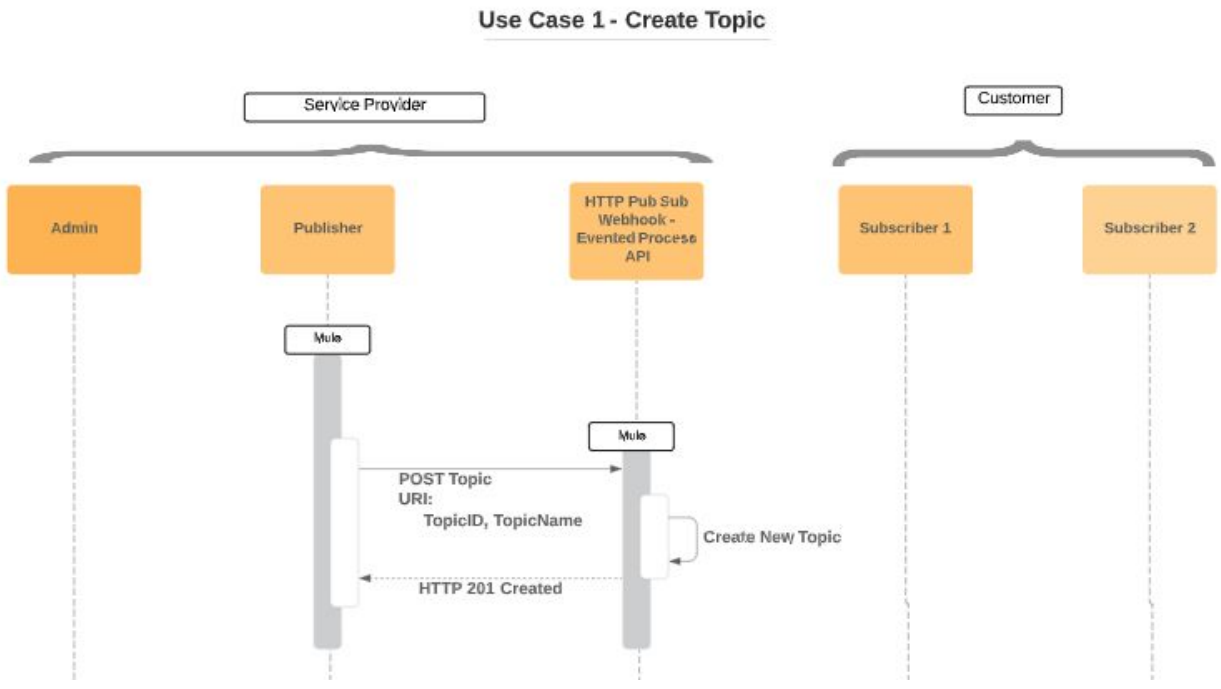
- Topic and queue admin via API or console
- Self-service subscription registration via API or console

- Publish messages to topics via API or console
- A considerable benefit of this approach is that HTTP webhooks provide an model for sending messages to subscribing clients on-demand over networks including the Internet with common security models and transport mechanism.
- Webhooks can be utilized from any technology stack that supports HTTP listeners - a common protocol for straightforward pub sub.

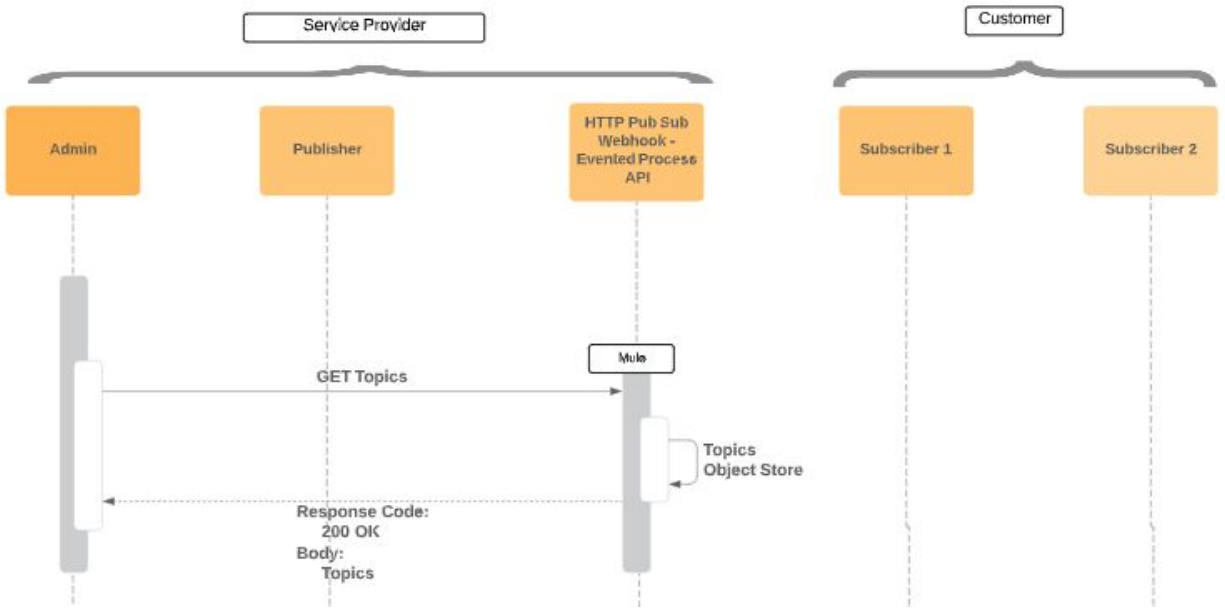
How does the data flow?

The below sequence diagrams represent the data flow for the following use cases:

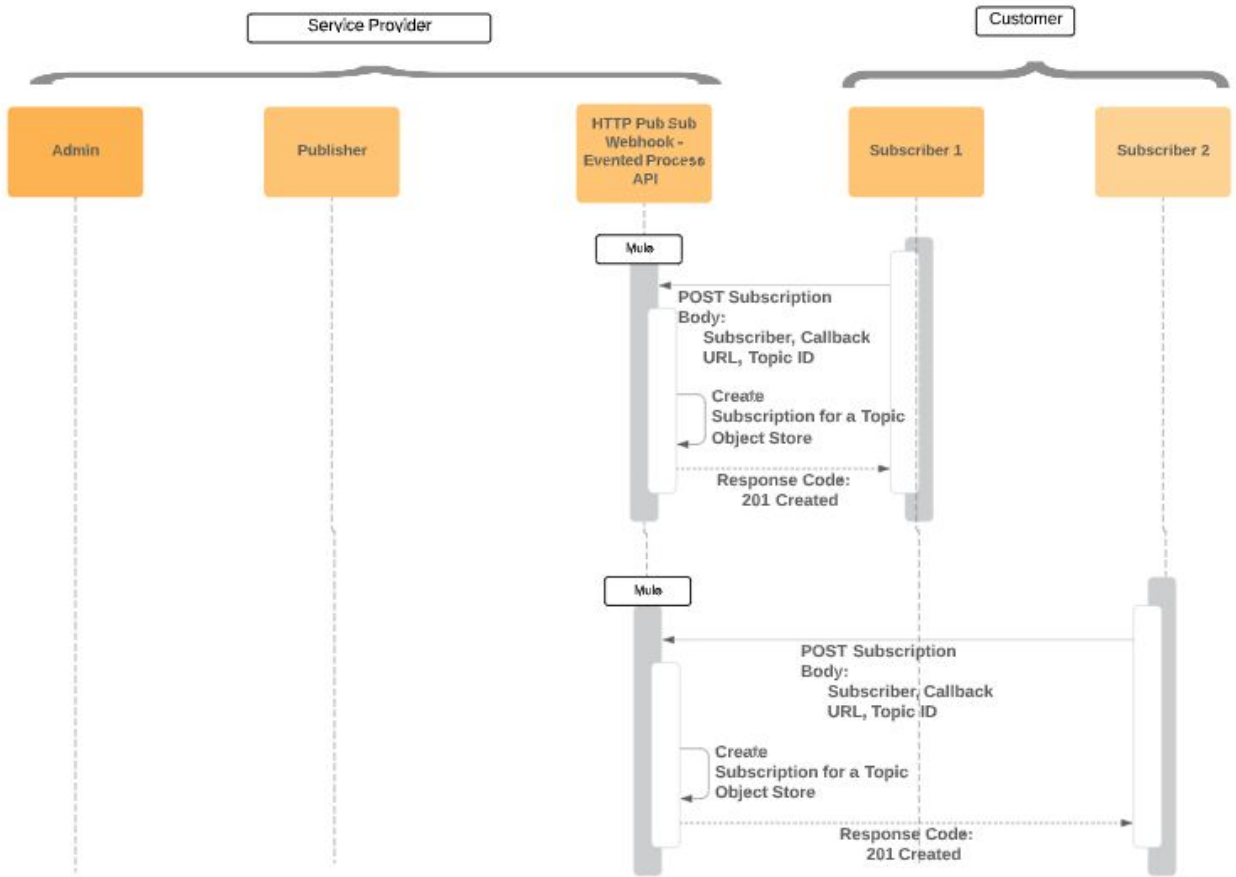
Use Case	Actors
1. Create topic	• Publisher; • HTTP Pub Sub Engine
2. List topics	• Publisher, subscriber, or admin; • HTTP Pub Sub Engine
3. Create subscription	• Subscriber; • HTTP Pub Sub Engine
4. List subscriptions	• Publisher, subscriber, or admin; • HTTP Pub Sub Engine
5. Publish message / receive mssages	• Publisher; • Subscriber; • HTTP Pub Sub Engine



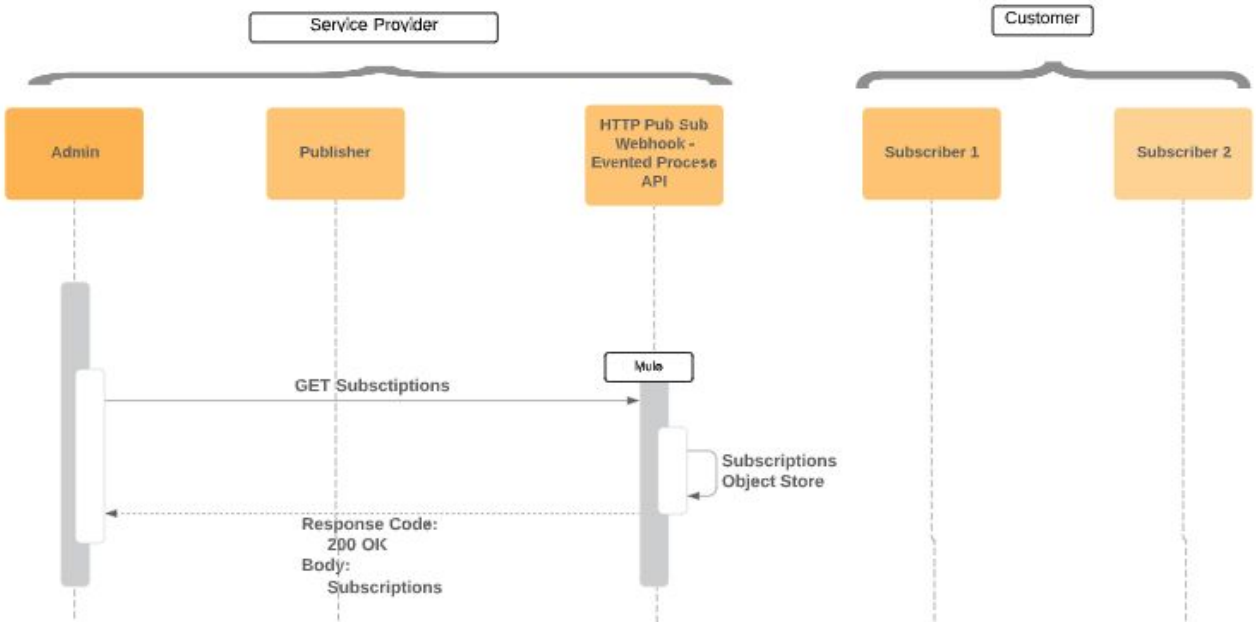
Use Case 2 - List Topics



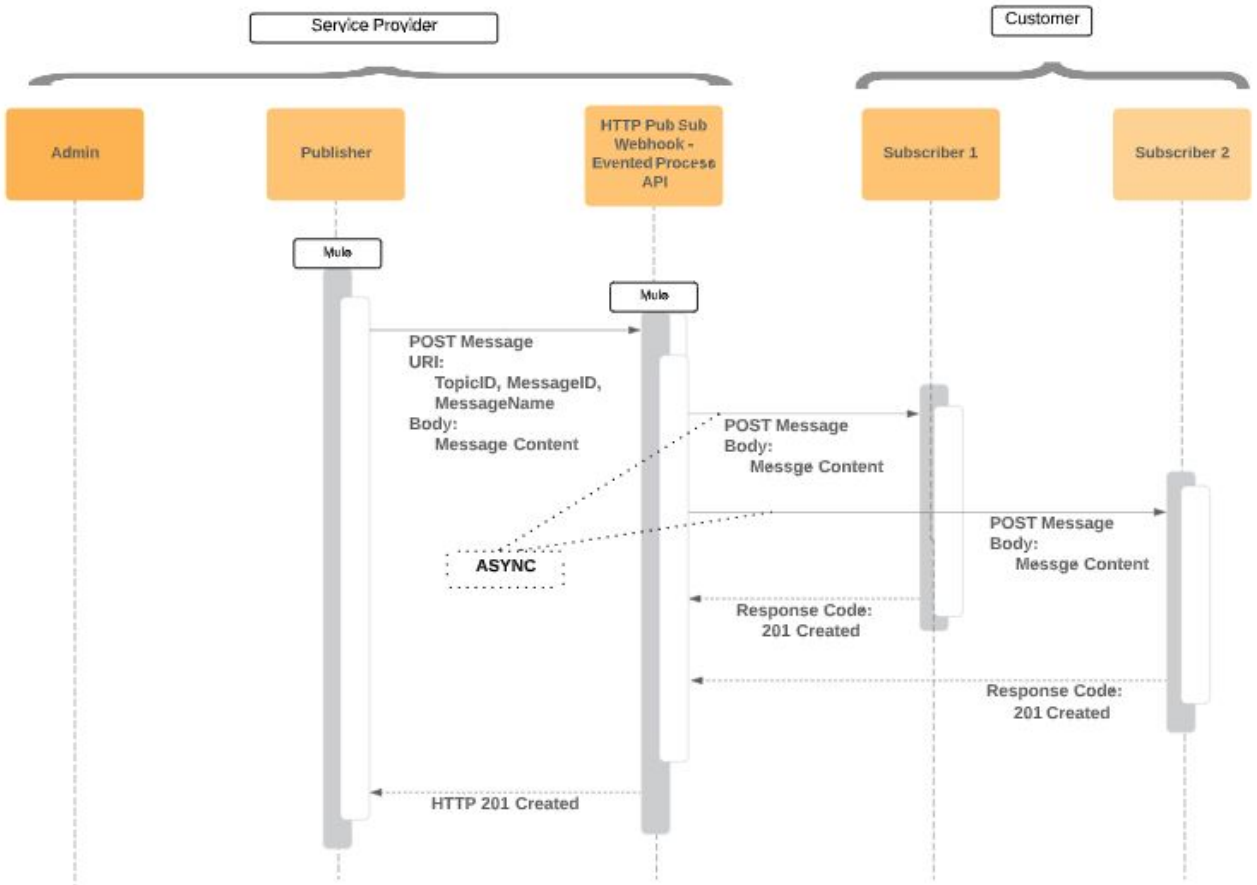
Use Case 3 - Create Subscription(s)



Use Case 4 - List Subscriptions



Use Case 5 - Publish Message(s)



What is the API specification?

These RAML (RESTful API Modeling Language) types provide a data model for HTTP Pub Sub use cases:

```
types:
  topicDetail:
    properties:
      topicName: string
      topicDescription: string
  subscriptionDetail:
    properties:
      topicName?: string
      callbackurl: string
      subscriptionId?: string
    example:
      topicName: MySampleTopic
      callbackurl: https://sample.mulesoft.com
      subscriptionId: demo
  message: object
```

A RESTful API spec is available to support the use cases documented above. Here is the summary level description of the spec:

API endpoints

/topic
GET DELETE

/topic/{topicname}

/topic/{topicname}/subscription
GET POST

/topic/{topicname}/message/{messagename}/{messagedescr}
POST

/topic/{topicname}/{topicdescr}
POST