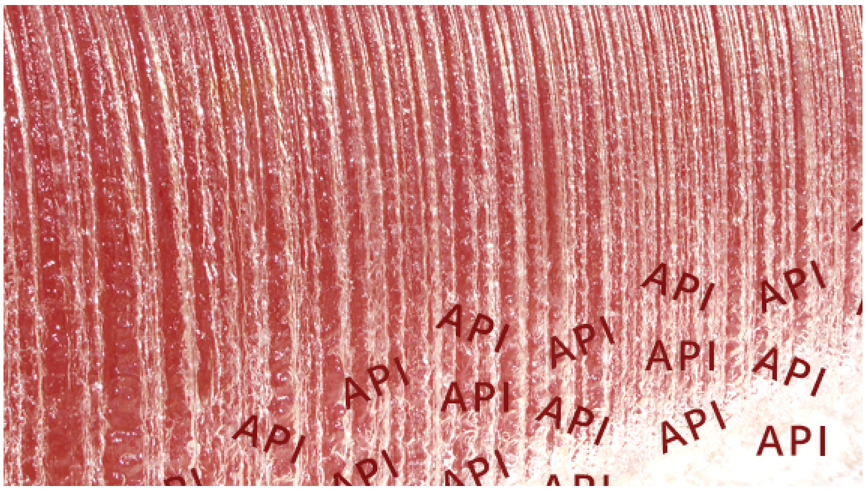
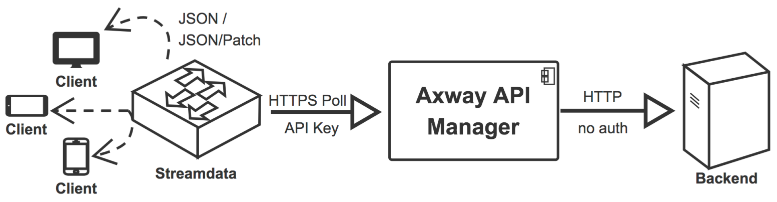
Build Streaming APIs with AMPLIFY API Management and Streamdata



By Uli Hitzel
July 12, 2018 @ 1:00am

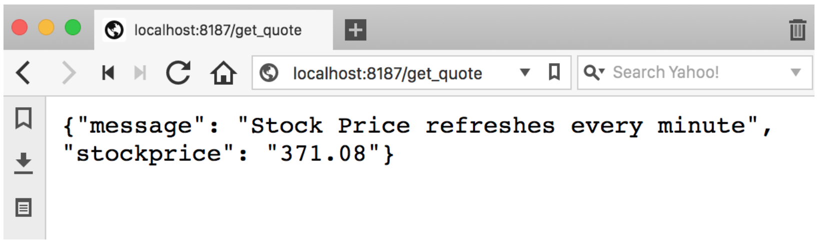
With streaming APIs, it's possible to push relevant updates to client devices from the server, which offers a couple of interesting benefits for large, distributed systems. In this post, I'll show you how to use a combination of Axway's API Management solution and **Streamdata** to securely virtualize and stream a JSON/REST API to end clients.



To do this exercise, we need the following:

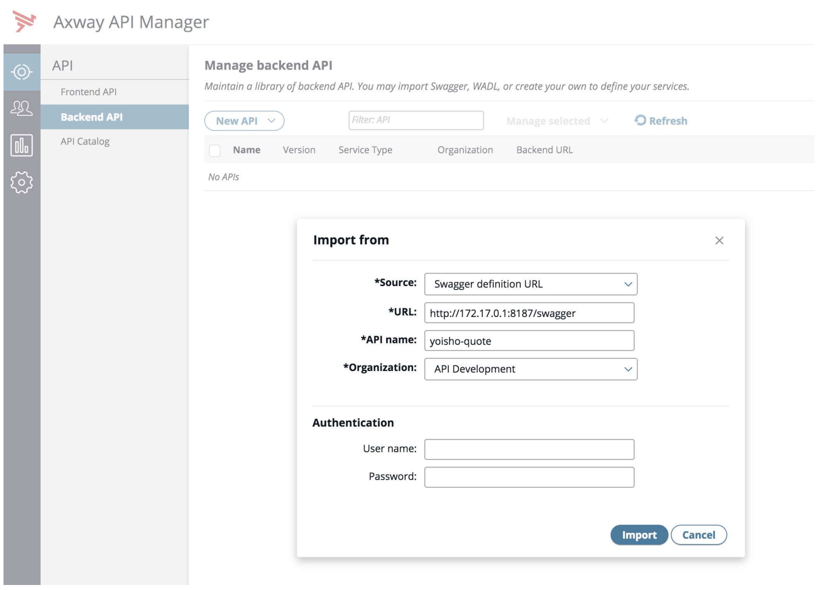
- Test installation of AMPLIFY API Manager & Gateway 7.5.3 [1]
- REST backend that gives us JSON data with frequent updates. Here we're using the Stock Quote container from the Yoisho project
- Free account at http://streamdata.io

Let's get started! You can choose to host the setup on a machine in the cloud. Here, I'm running all components (except, of course, Streamdata) on my laptop. After starting the Stock Quote service, we can inspect the JSON data in the browser:



```
{"message": "Stock Price refreshes every minute",
"stockprice": "371.08"}
```

Now, we're virtualizing the API in API Manager. To do this, we can import the Swagger using the URL that the Yoisho service exposes:



Manage backend API

Maintain a library of backend API. You may import Swagger, WADL, or create your own to define your services.

Name	Version	Service Type	Organization	Backend URL
No APIs				

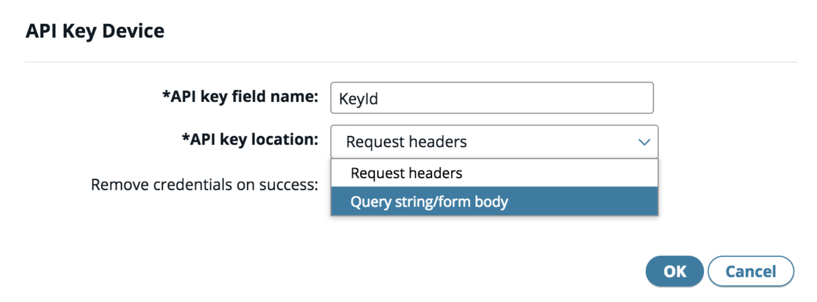
Import from

*Source:	Swagger definition URL
*URL:	http://172.17.0.1:8187/swagger
*API name:	yoisho-quote
*Organization:	API Development

Authentication

User name:	
Password:	

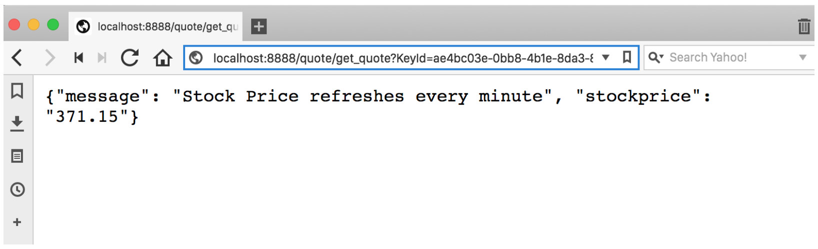
In this scenario, API Manager helps us to protect the backend with quota settings, preventing malicious content and only allowing authorized access. Specifically, we're keen to ensure that only Streamdata can access the endpoint. For this, we're definining API Keys as the authentication mechanism and specifying that the key has to be in the HTTP Query String:



API Key Device

*API key field name:	KeyId
*API key location:	Request headers
Remove credentials on success:	

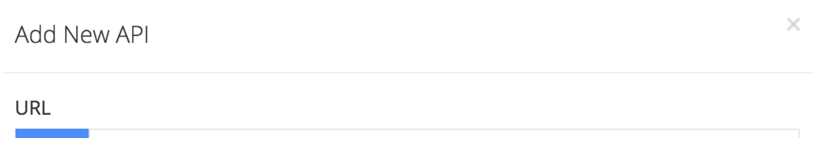
We now have a secured REST endpoint, protected with an API Key. This is the URL we need to specify in the streamdata interface. Let's test it in the browser to see if it works:

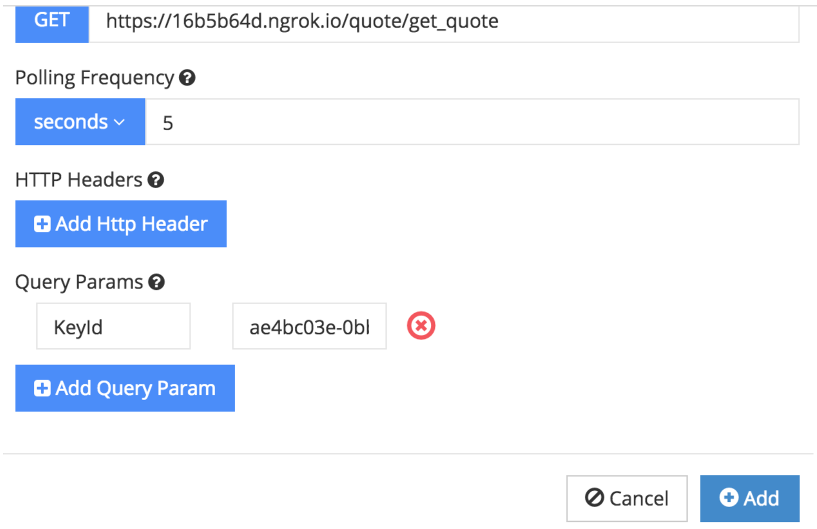


```
localhost:8888/quote/get_quote?KeyId=ae4bc03e-0bb8-4b1e-8da3-
```

```
{"message": "Stock Price refreshes every minute", "stockprice":
"371.15"}
```

In the **Streamdata admin tool**, we can now create a new stream API. Here, we're adding the URL of our endpoint and the API Key that the service needs to use in order to authenticate [2]:





GET https://16b5b64d.ngrok.io/quote/get_quote

Polling Frequency

seconds 5

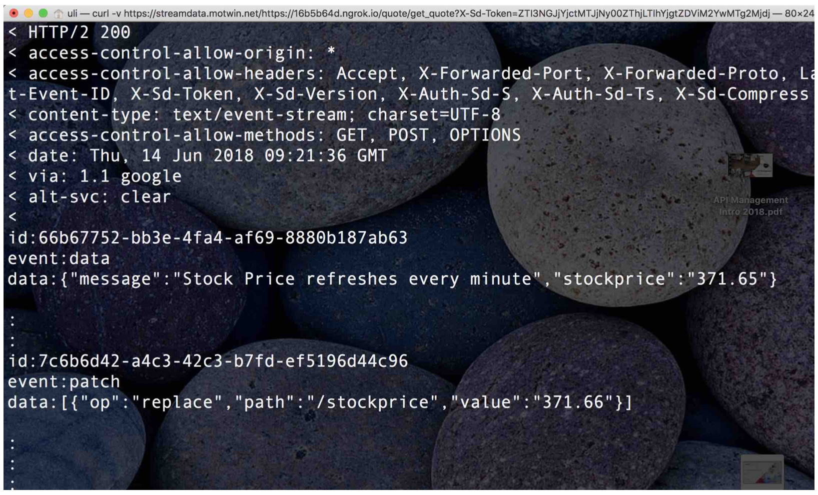
HTTP Headers

Query Params

KeyId ae4bc03e-0bl

And we're done! Now we can use cURL to trigger a client request and see the initial content and the updated stock quotes. Your command should look something like the following:

```
curl -v "https://streamdata.motwin.net/https://16b5b64d.ngrok.io/quot
```



```
uli — curl -v https://streamdata.motwin.net/https://16b5b64d.ngrok.io/quote/get_quote?X-Sd-Token=ZTI3NGJjYjctMTJjNy00ZThjLTlhYjgtZDViM2YwMTg2Mjdj — 80×24
< HTTP/2 200
< access-control-allow-origin: *
< access-control-allow-headers: Accept, X-Forwarded-Port, X-Forwarded-Proto, La
t-Event-ID, X-Sd-Token, X-Sd-Version, X-Auth-Sd-S, X-Auth-Sd-Ts, X-Sd-Compress
< content-type: text/event-stream; charset=UTF-8
< access-control-allow-methods: GET, POST, OPTIONS
< date: Thu, 14 Jun 2018 09:21:36 GMT
< via: 1.1 google
< alt-svc: clear
<
id:66b67752-bb3e-4fa4-af69-8880b187ab63
event:data
data:{"message":"Stock Price refreshes every minute","stockprice":"371.65"}

:
:
id:7c6b6d42-a4c3-42c3-b7fd-ef5196d44c96
event:patch
data:[{"op":"replace","path":"/stockprice","value":"371.66"}]

:
:
:
```

You will see the difference to a "classic" API request immediately – the connection does not terminate, which you can also notice from the `Content-Type: text/event-stream` header. Furthermore, while the initial response will return all data fields, the subsequent pushs will only return the ones that have changed (**JSON-Patch**) since the last poll from Streamdata to the backend.

I hope this helps you understand the concept of streaming APIS, the benefits and how this can be implemented using the Axway API Management solution.

[1] Important Note: If you're keen to try this yourself, please enable the `Content-Length` setting in API Manager as described in **this KB article**.