TRIDIUM

NS2022

ACCELERATING INNOVATION

The HTTP Client (Developers Edition)

Jason Woollard / Nick Dodd

April 4th 2022

TRIDIUM

Niagara HTTP Client

Allows the Niagara station to interact with HTTP web services and API's.

Data can be exchanged both **in** and **out** of Niagara.

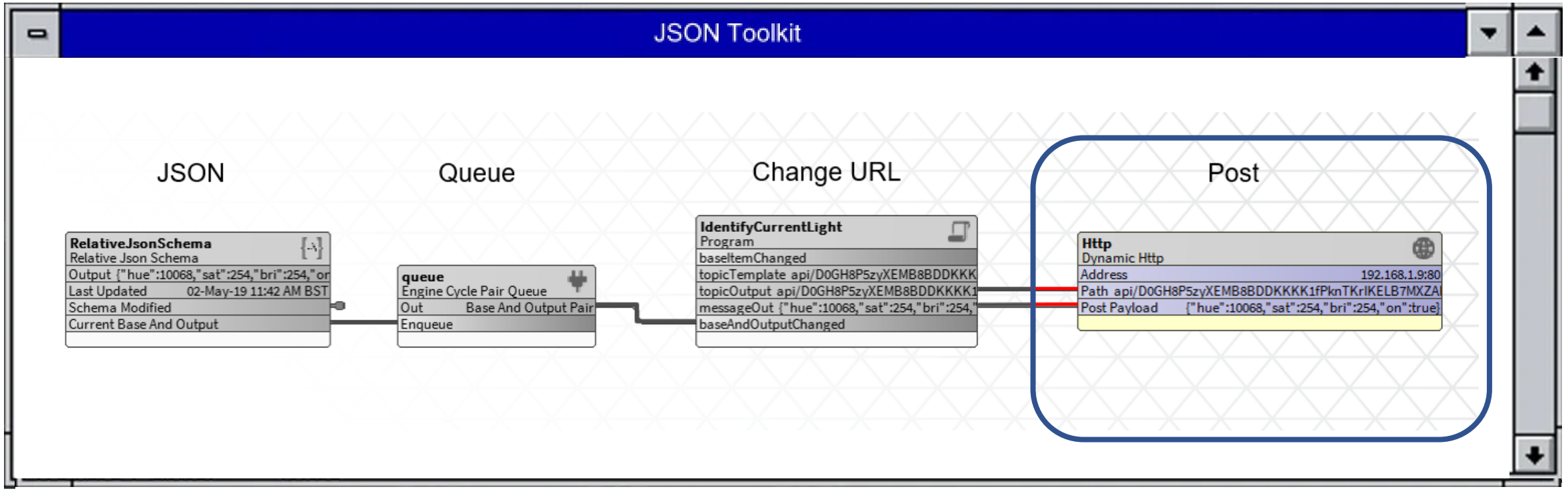
As many use cases as API's

- Bringing useful data **into** a building
  - Occupancy, Travel, Air Quality, Car Parks

- Exporting station data **out** to an external service / cloud
  - AWS IoT, Azure IoT, Google Cloud offer RESTful services
  - Messaging, Notifications

- Device Control
  - Integrate with IoT Devices / Gateways

Rewind.....Niagara Forum 2019



JSON Toolkit lighting control demo

Actually - repeated by us tomorrow @ 4.30

Next 2 years – where is the HTTP component?

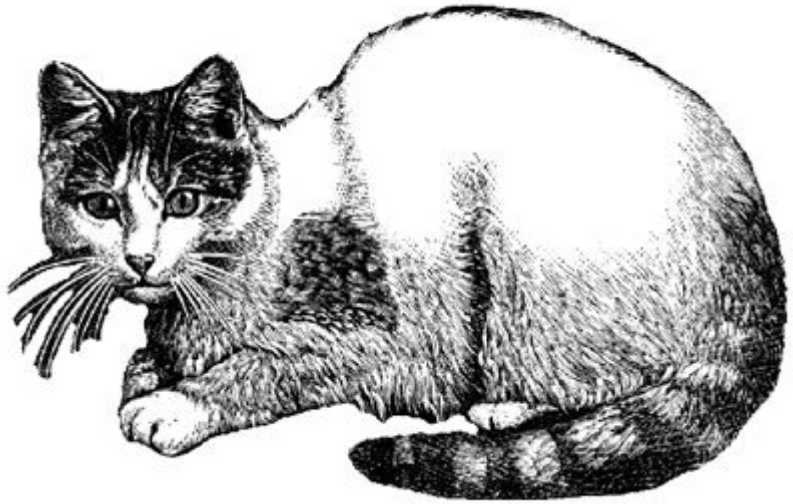
"Interested in 'Dynamic HTTP' component seen in JSON toolkit presentation….?"

"I recently saw a hint in one of your JSON Toolkit presentations and couldn't find the module which contains….."

"Is it possible to receive or send HTTP requests from our Niagara equipment"

"Could you advise whether there is a Rest API module"

https://http.cat/404

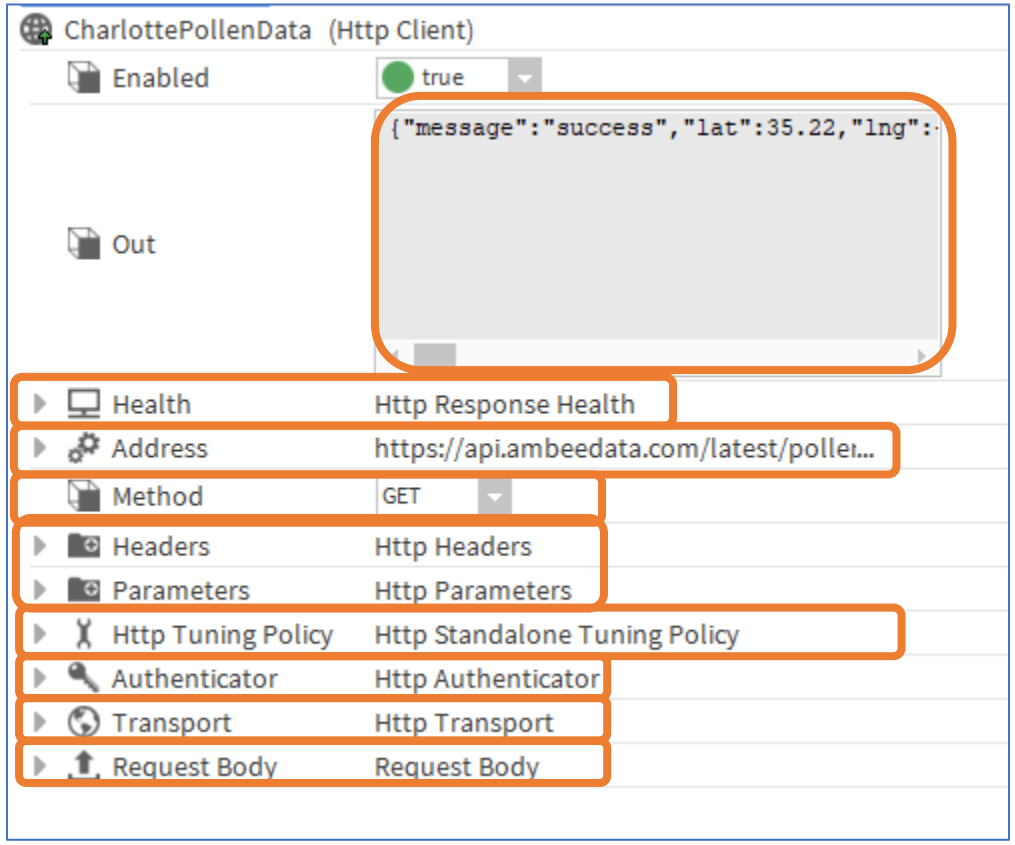


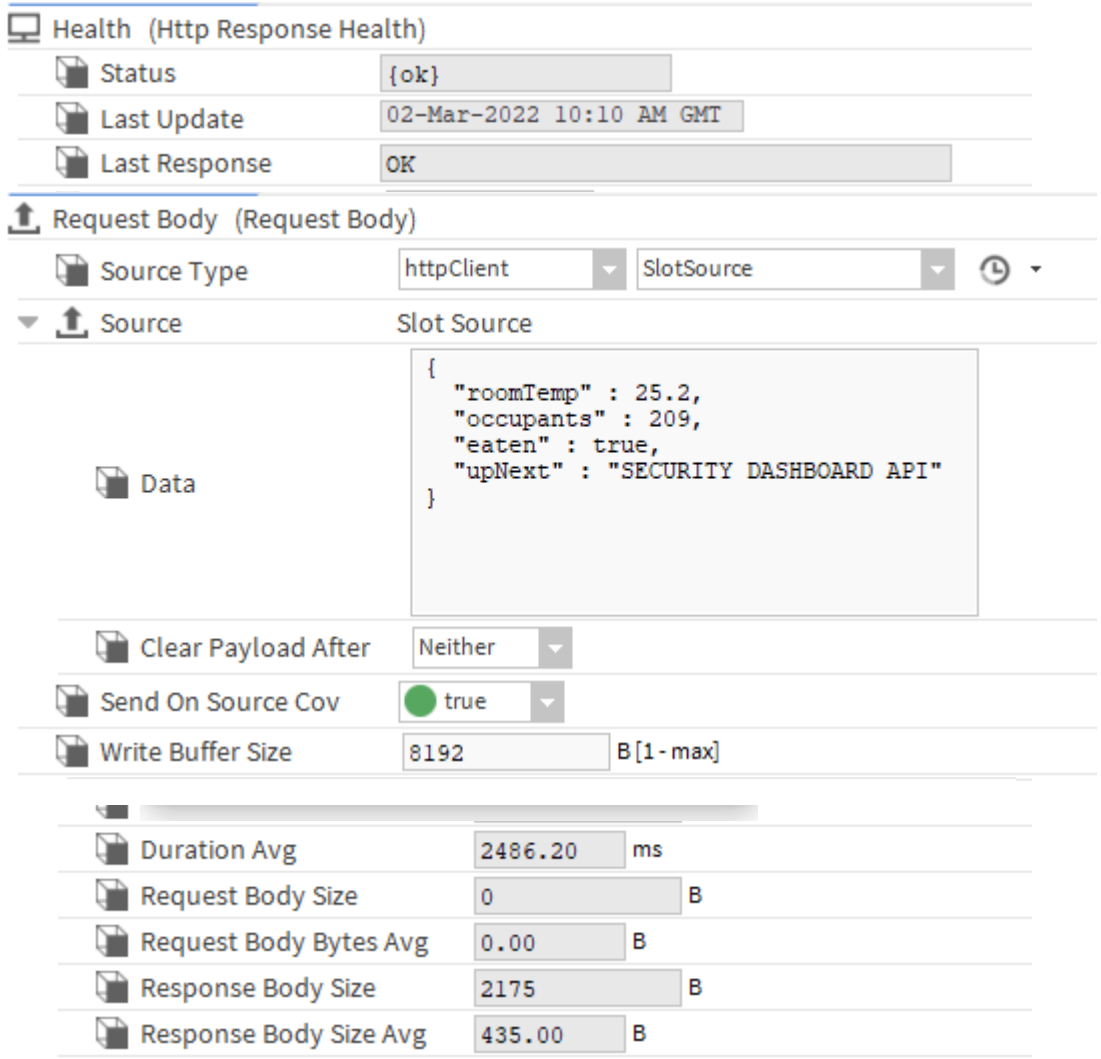
Essential

HTTP Client in a nutshell

TRIDIUM

Standalone Client

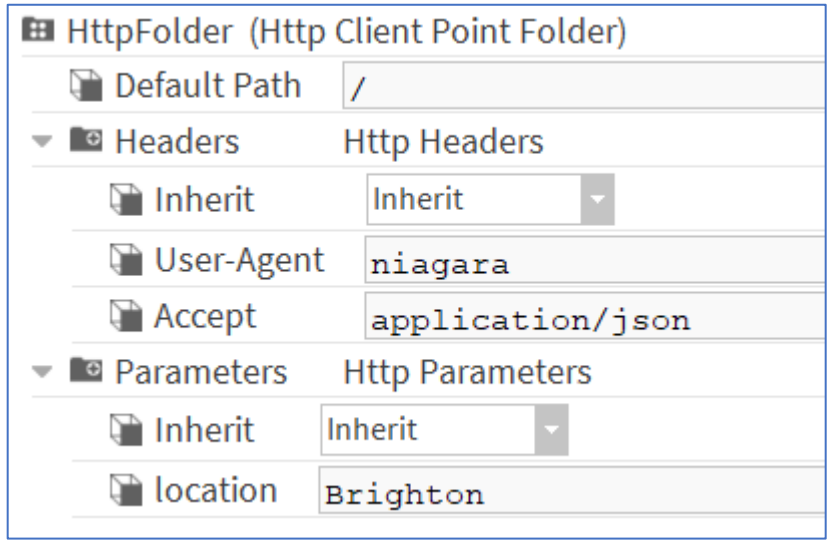




HTTP Client Driver

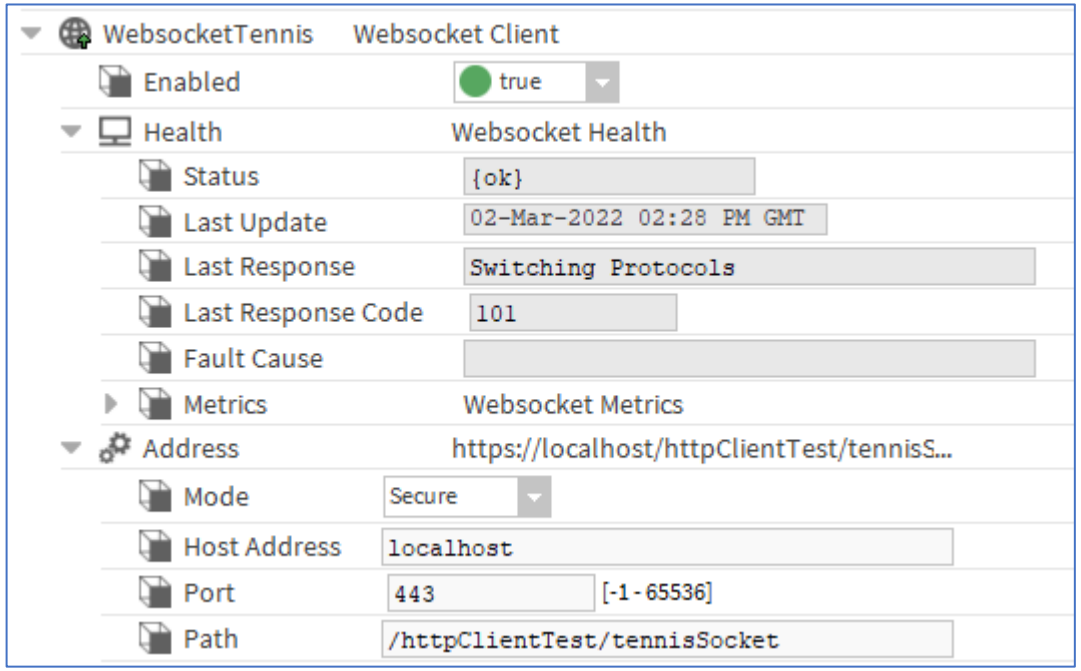
- HTTP Client Proxy Points *per resource*
  - Share auth and transport
  - Useful for polling multiple similar endpoints

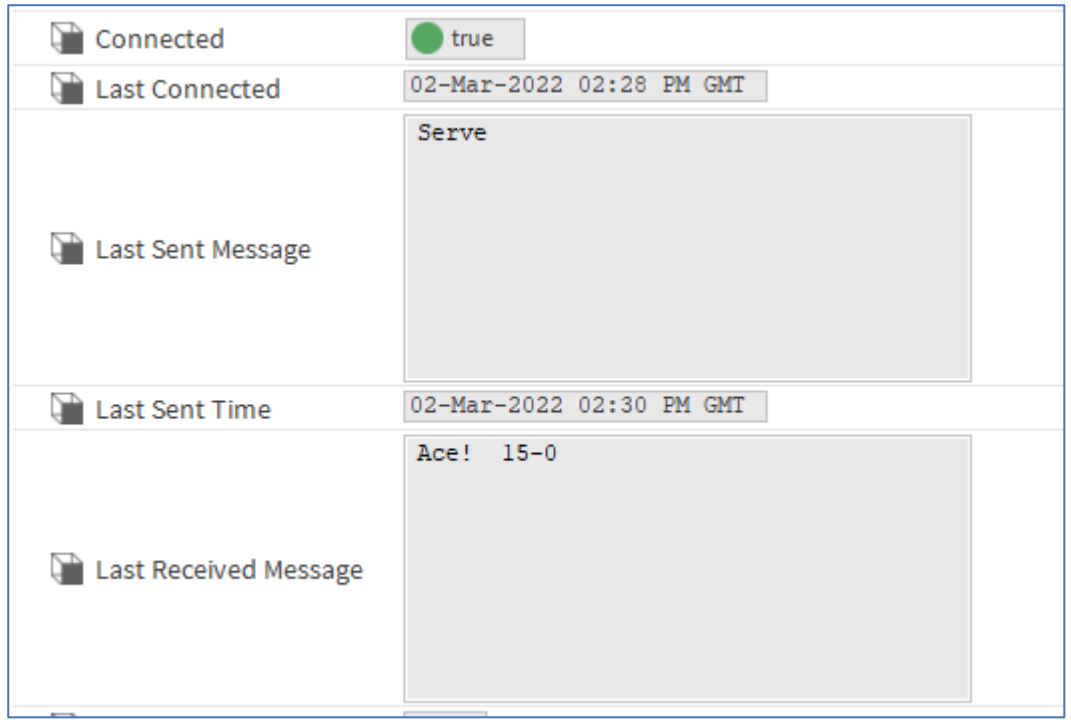
- Device Ping Address for service status (optional)
- Points Folder defines common Headers or Parameters



WebSocket Client

Persistent Connection

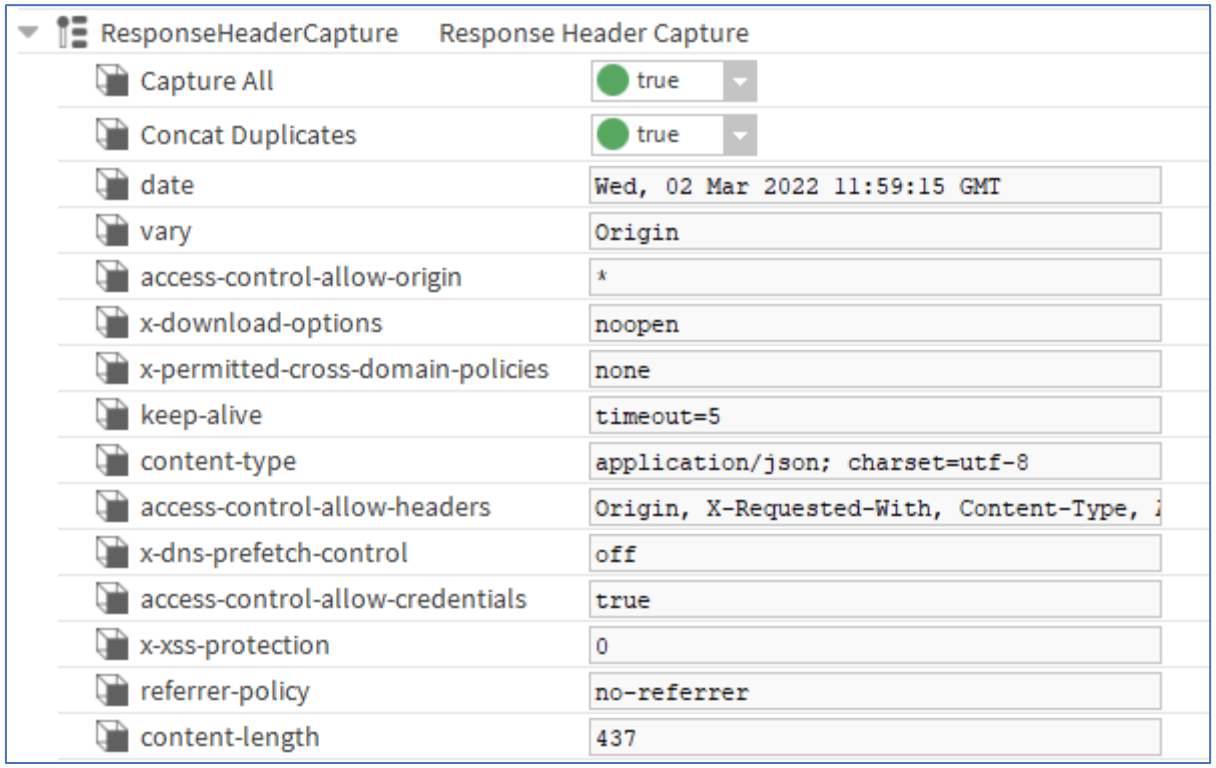


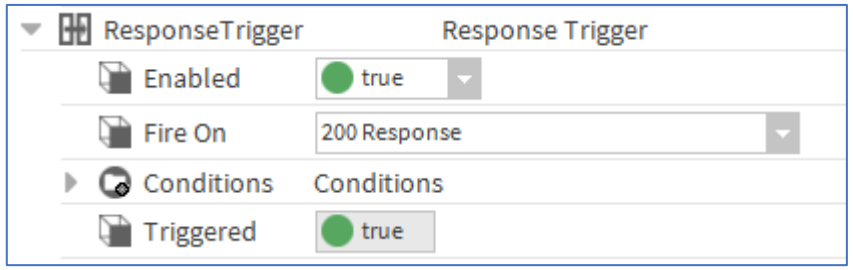


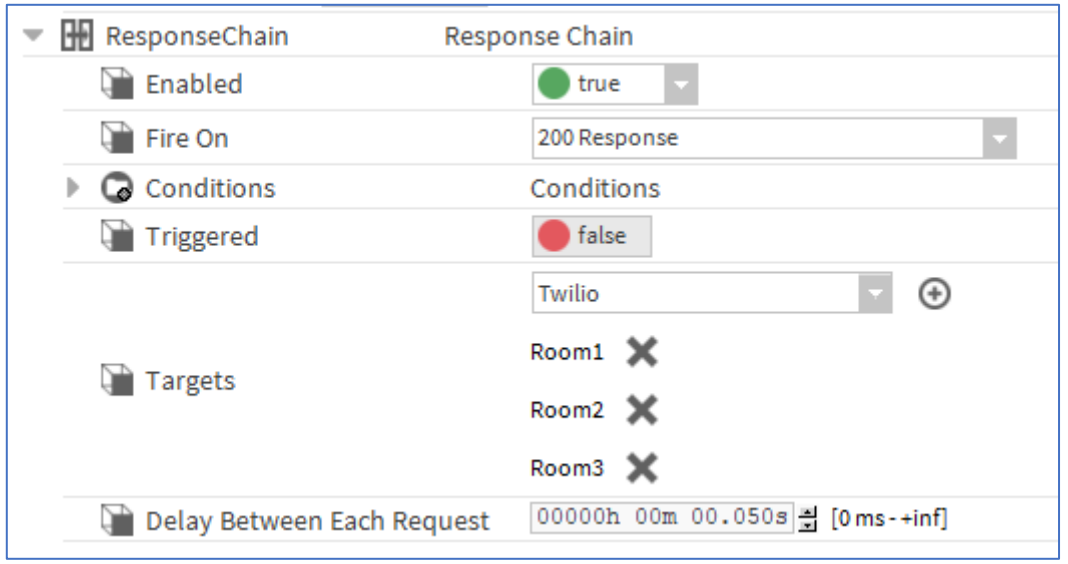
Full duplex comms with queues

Wiresheet Tools

Capture response headers/cookies



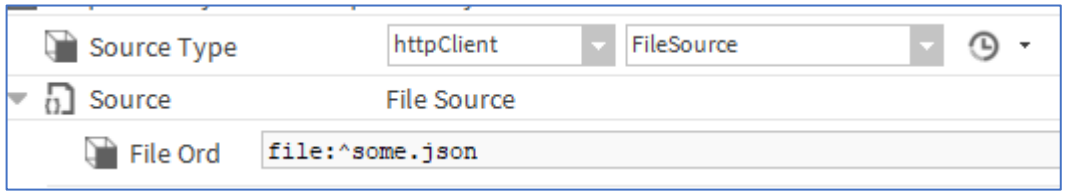




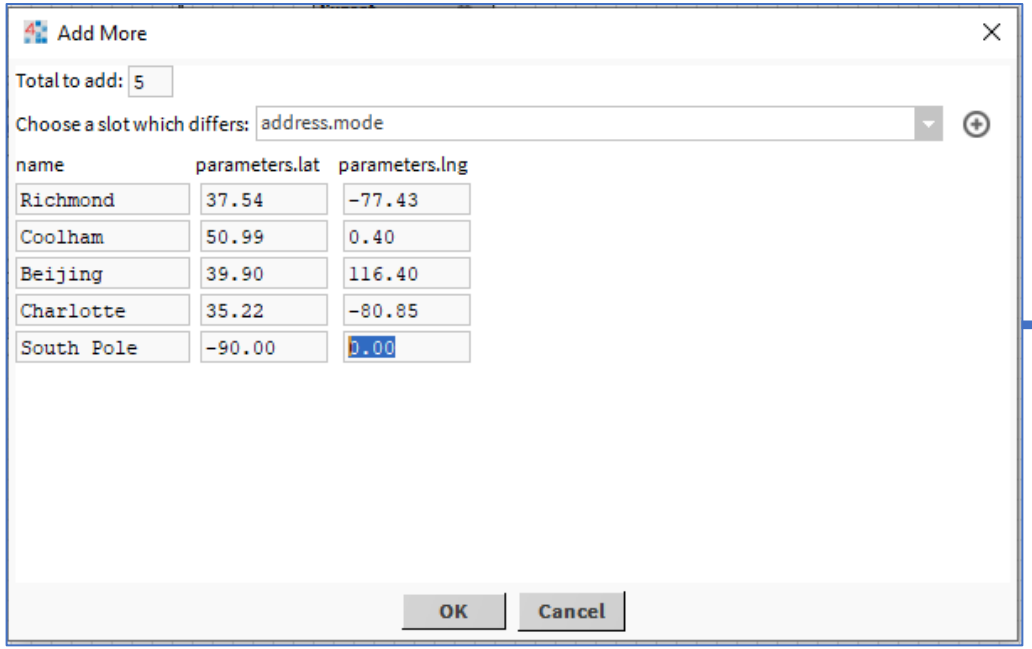
Trigger actions / other clients to send

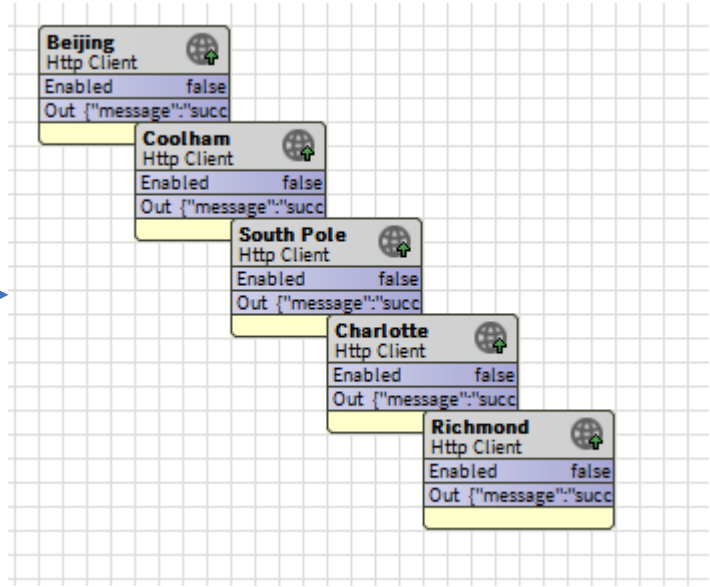
Other timesavers

- Use Files or Niagara Reports as payload



- Utility to duplicate clients with different settings

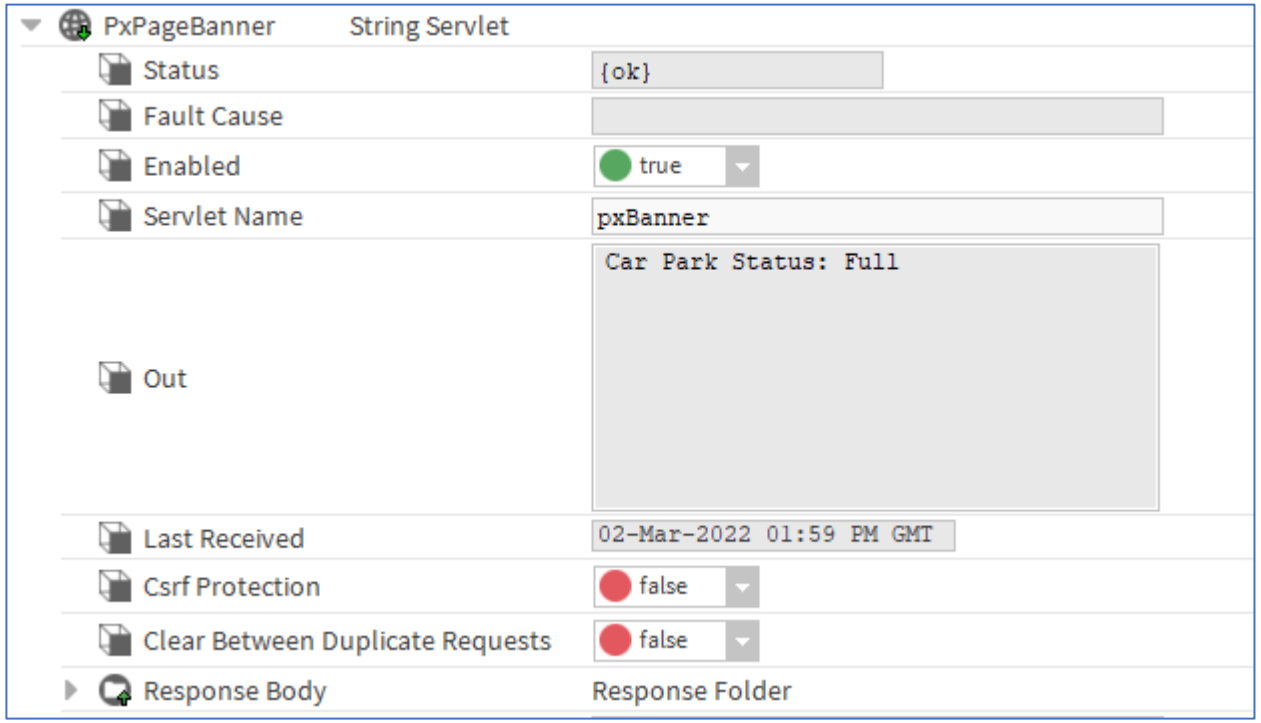


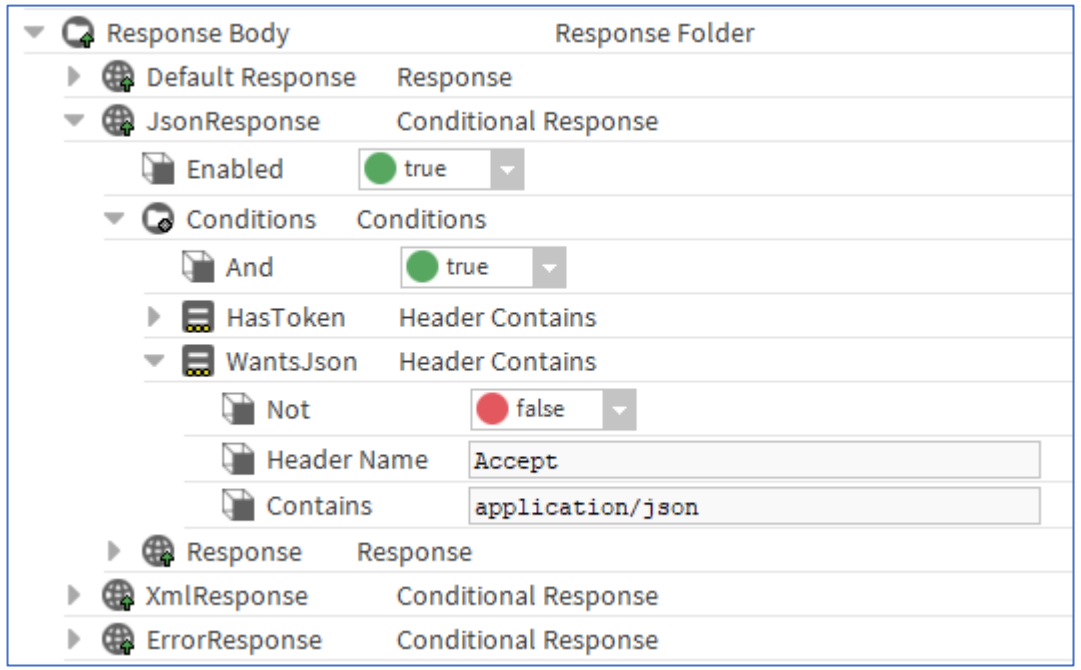


String Servlet

Send messages into the station

- trigger some logic
- inject into a px view
- append to a history





Setup conditional response based on headers / content

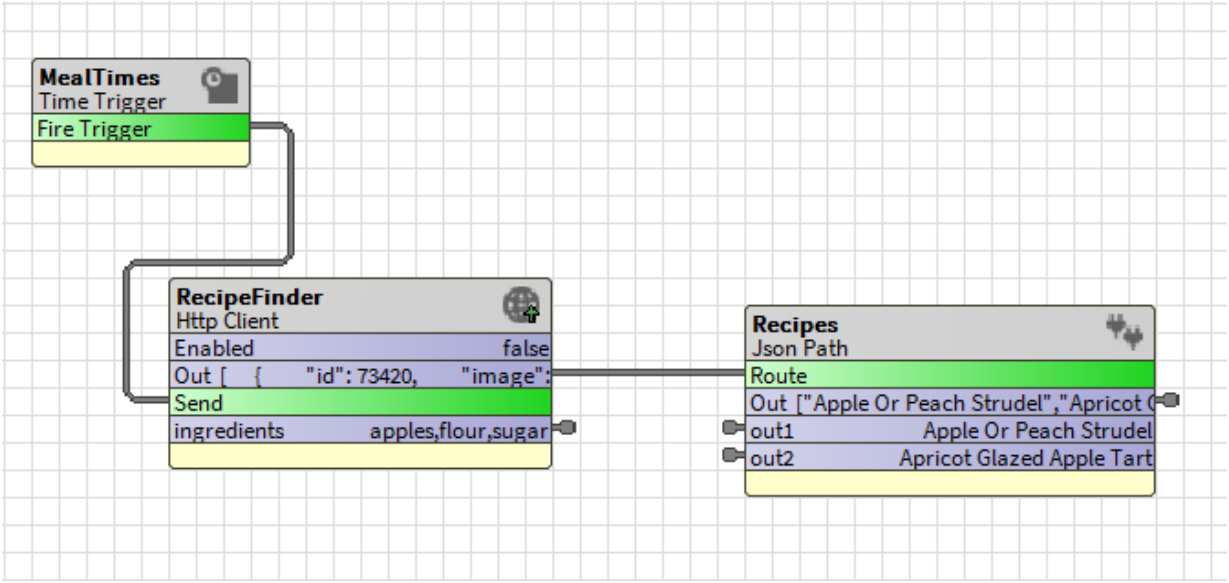
But hang on....why are you telling me all this?

https://http.cat/417

"I am at a developer day, I can do all this myself with a simple Java class?"

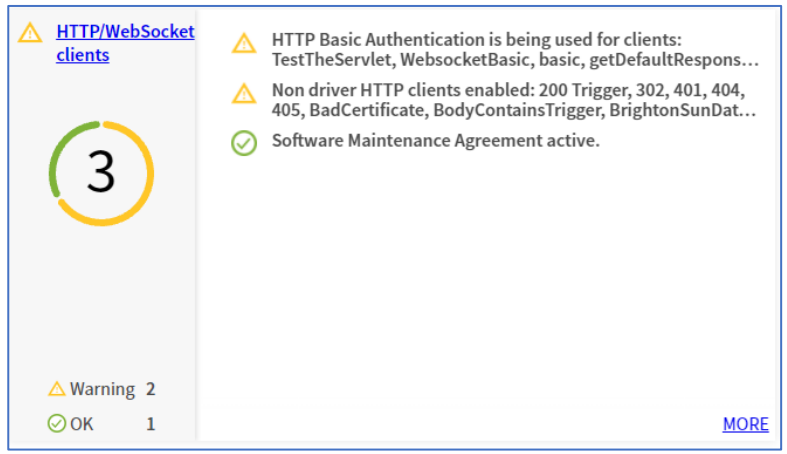
‘I want to save some time’

- Public API
  - Create working clients in a few lines of code
  - Forward compatibility
- Wiresheet Ready
  - Drag from the palette and go .....
- Ready to use driver
  - Attracts usual driver benefits .....
  - Poll Scheduler
  - Point Extensions
  - Consistent UI



‘I guess I need to do the Security thing’

- Certificate Management
  - Approval of SSL certificates in familiar tool
- Security Dashboard
  - Already integrated with warnings for various settings
- Security Logging
  - Dedicated audit log for all outgoing requests

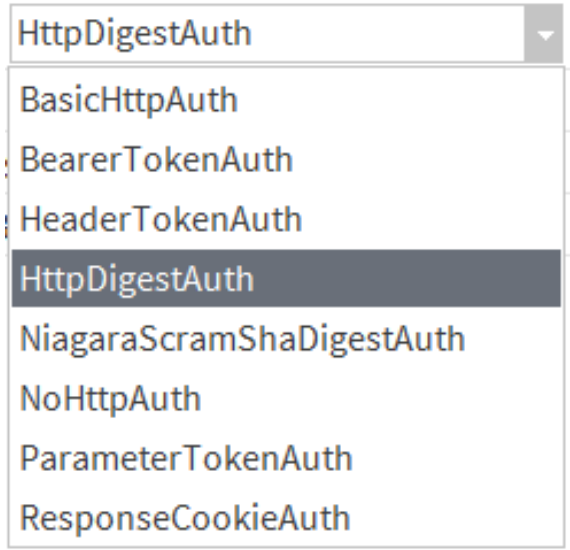


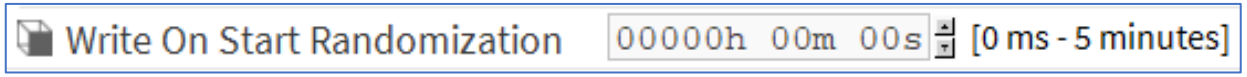
httpTesting/HttpClientRequestHistory

Timestamp	Operation	Target	Slot Name	Old Value	Value	User Name
24-Feb-22 12:12:55 PM GMT	GET	/api/CharlottePollenData	CharlottePollenData		https://api.ambeedata.com:443/latest/pollen/by-lat-lng	Dave Lister
24-Feb-22 12:12:53 PM GMT	GET	/api/BrightonSunData	BrightonSunData		https://api.sunrise-sunset.org:443/json	Dave Lister
24-Feb-22 12:12:45 PM GMT	GET	/Drivers/HttpClientNetwork/HttpClientDevi	Ping Address		https://localhost:443/ord/station:%7Cslot:/	system
24-Feb-22 12:12:32 PM GMT	POST	/httpClients/ReportSource	ReportSource		https://httpbin.org:443/post	system

‘Sometimes you need more than 1 class….’

- Authentication
  - Basic / Digest / Niagara / Token / Response Cookie
  - Extensible / pluggable authenticator model *
- Avoid bottlenecks
  - Randomization of send on startup
  - Don’t DDOS your endpoint!

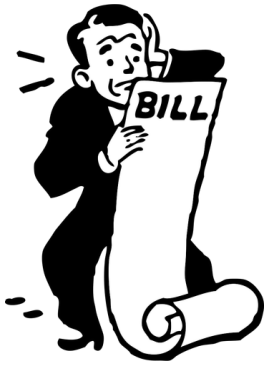




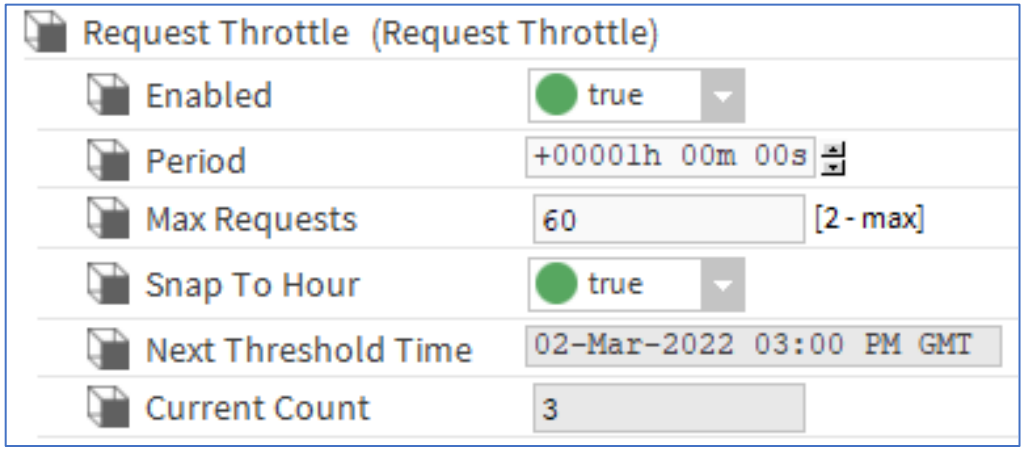
* not public API yet

‘Sometimes you need more than 1 class….’

- Request Throttling
  - Per Client + Global
  - Ensure you do not exceed API limits, avoiding unexpected bills.



- Concurrency challenges
  - Managed Thread pool
- Trigger and Response Chain
  - Trigger events or secondary requests.



Public API

Time to build something

TRIDIUM

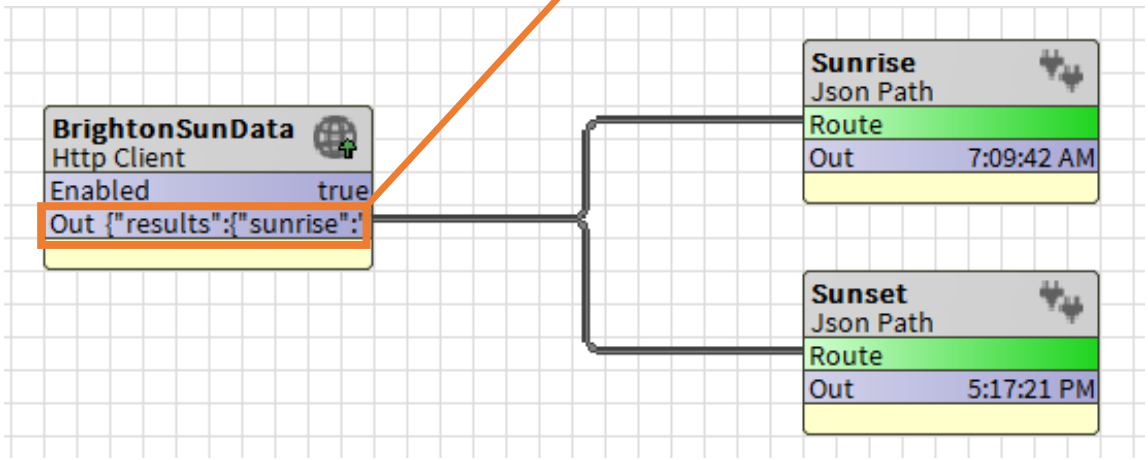
1. Building a simple client

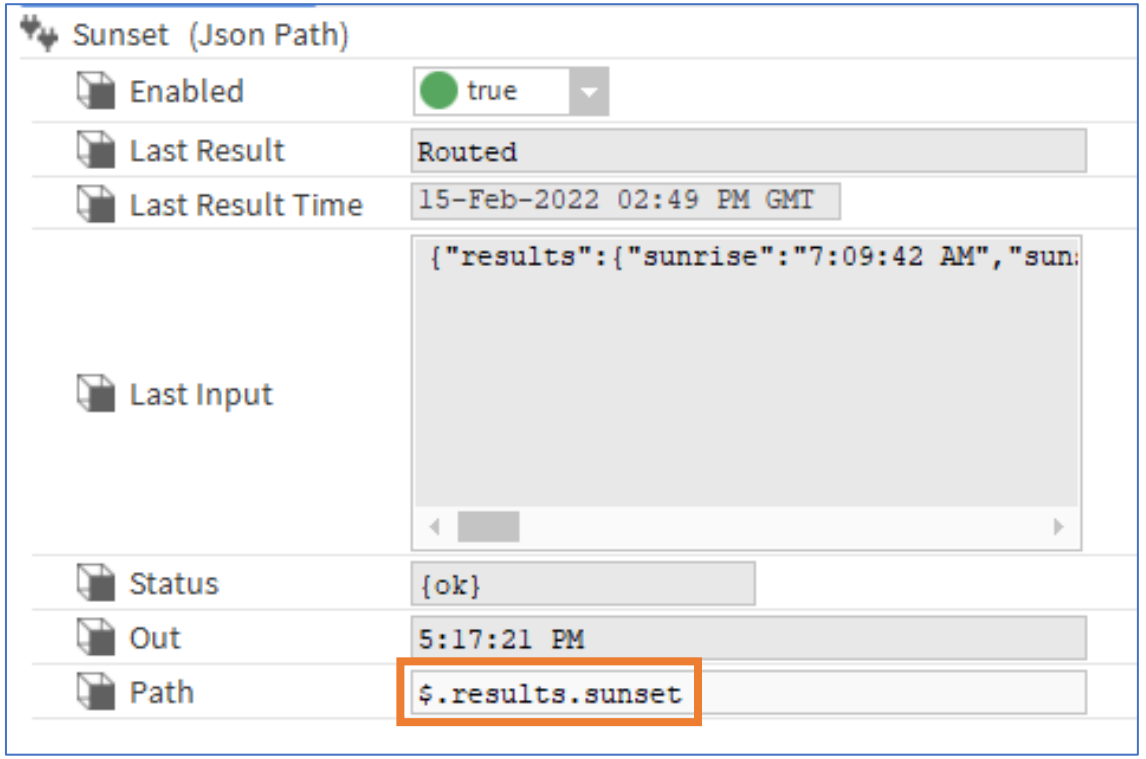
‘I want to bring the lights on at sunset’

```
IHttpClient client = HttpClientBuilder.instance()
    .withAddress("https://api.sunrise-sunset.org/json")
    .withParameter("lat", BDouble.make(50.822))
    .withParameter("lng", BDouble.make(0.137))
    .mountAtOrd("slot:/api", “BrightonSunData")
    .build();

client.send();
```

Simple JSON Toolkit integration


```
{
  "results": {
    "sunrise": "6:20:11 AM",
    "sunset": "6:20:11 PM",
    "solar_noon": ...
  }
},
```



2. Adding authentication

https://http.cat/401

https://http.cat/403

Adding authentication

‘My strategy needs to know the tree pollen index’

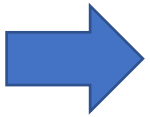
```
IHttpClient client = HttpClientBuilder.instance()
    .withAddress("https://api.ambeedata.com/latest/pollen/by-lat-lng")
    .withParameter("lat", BDouble.make(35.222))
    .withParameter("lng", BDouble.make(-80.847))
    .withHeaderTokenAuthenticator("x-api-key", "895e63e57a1……")
    .mountAtOrd("slot:/api", "CharlottePollenData")
    .build();
```

Reading the response – Synchronous Send

```
String responseBody = client.sendSync();

System.out.println("Received: " + client.getLastResponseCode() +
                                " : "  + responseBody);
```

Received: 200 :

{"message":"success","lat":35.22,"
lng":80.85,"data":[{"Count":{"gra
ss_pollen":0,"tree_pollen":157...



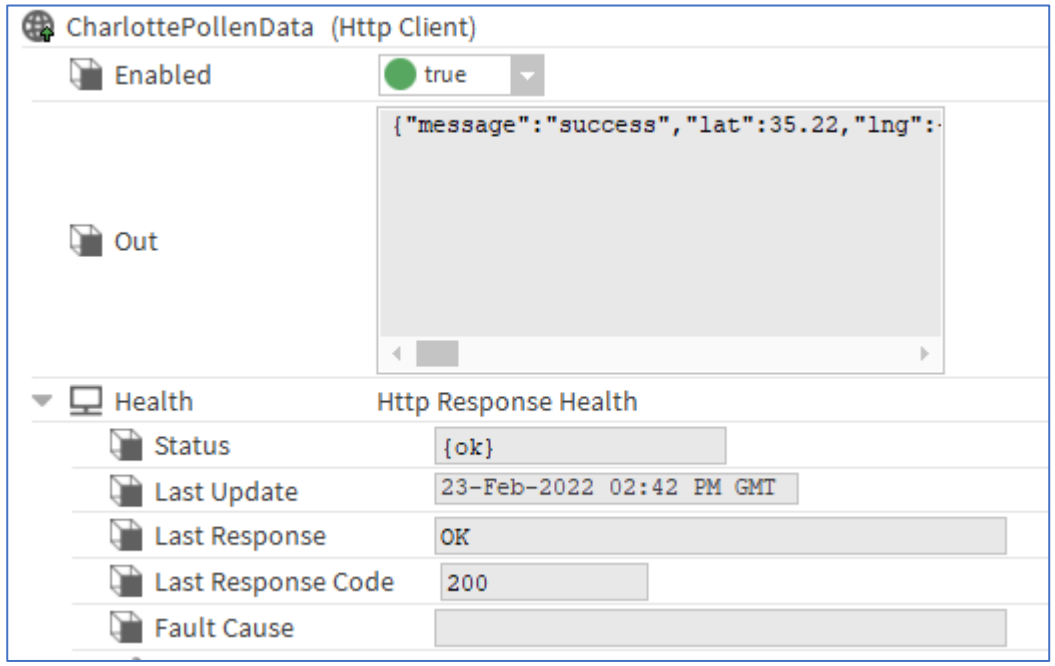

```
{ "message": "success",
  "lat": 35.22,
  "lng": 80.85,
  "data": [
 { "Count":
    {
      "grass_pollen": 0,
      "tree_pollen": 157,
      ...
```

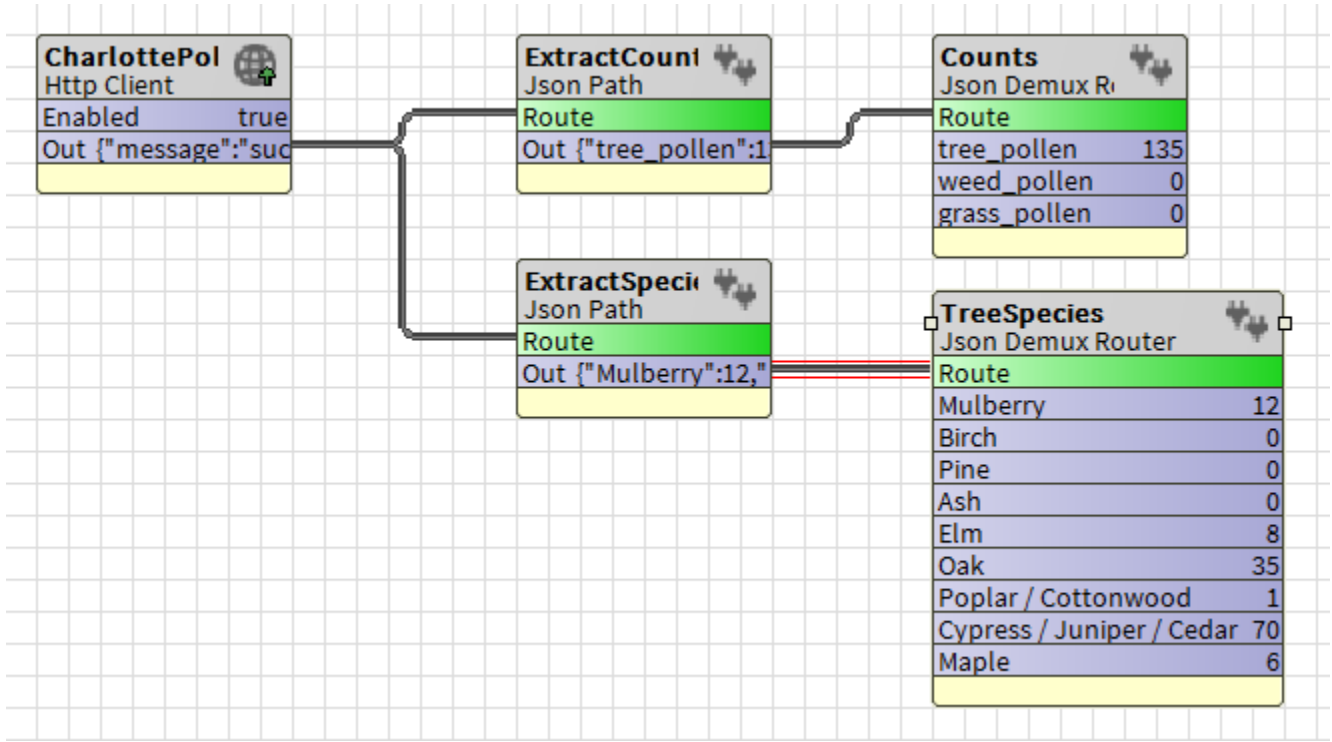
Reading the response – Asynchronous Send

```
Future<IHttpResponse> future = client.sendAsync();
IHttpResponse response = future.get(2, TimeUnit.SECONDS);

System.out.println(response.getResponseCode());
System.out.println(response.getBody());
System.out.println(response.getHeader("content-length"));

IHttpResponse responseObject = client.getLastResponse();
```

Received: 200 :
{"message":"success","lat":35.22,"lng":-
80.85,"data":[{"Count":{"grass_pollen":0,"tree_pollen":218..... Optional[437]

Polling for Pollen





(Tree pollen is Nick's kryptonite)

3. Adding POST Payload

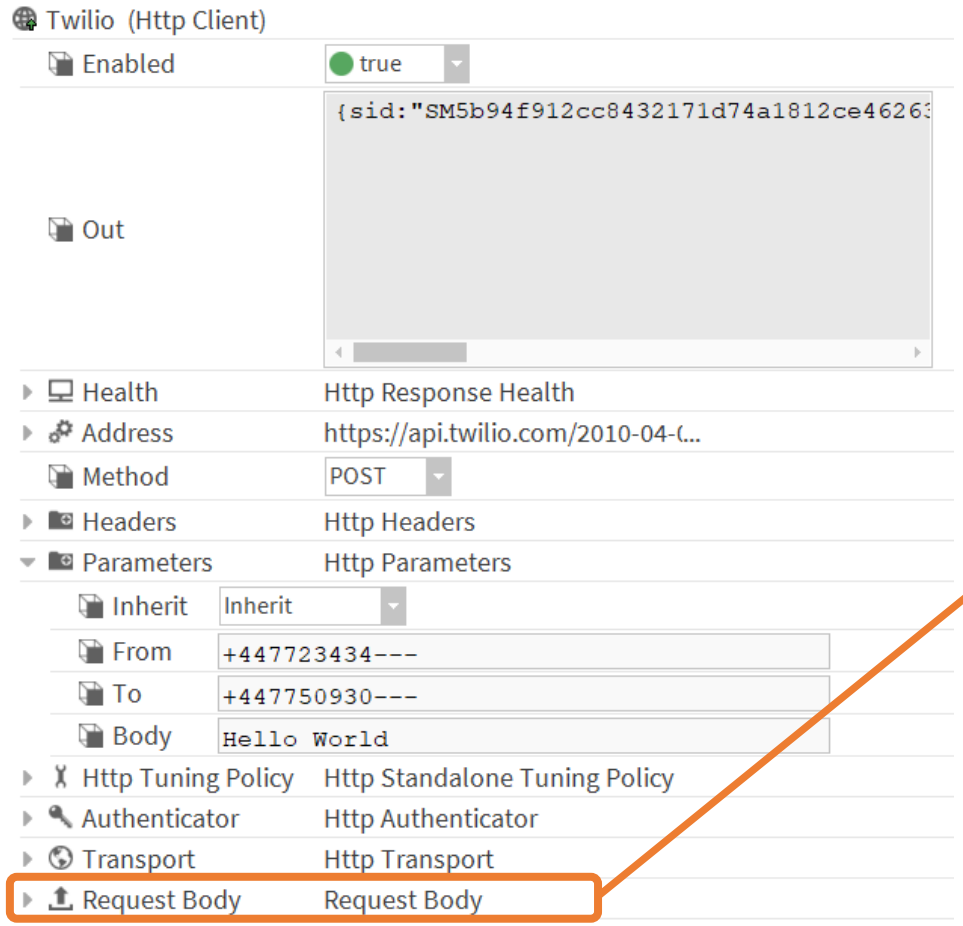
https://http.cat/413

Adding POST Payload

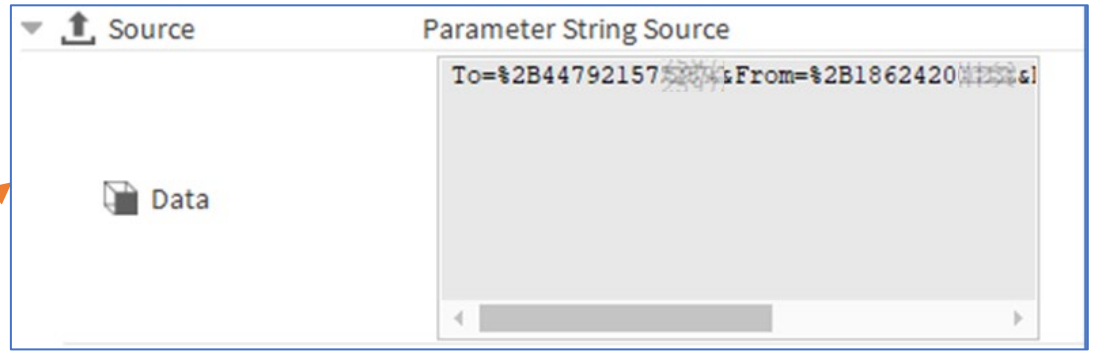
```
IHttpClient client = HttpClientBuilder.instance()
    .withAddress("https://api.twilio.com/2010-04-01/Accounts/AC57e...../Messages.json")
    .withMethod("POST")
    .withFormPayload()                                          // or withStringPayload("....")
    .withParameter("Body", BString.make("Hello Charlotte"))
    .withParameter("From", BString.make("+1862420......."))
    .withParameter("To", BString.make("+44792157......."))
    .withBasicAuthenticator("AC57e96db.....", "cc19b1f.....")
    .mountAtOrd("slot:/api", "TwilioSMS")
    .build();

client.send();
```

```
To=%2B44792157.....&From=%2B1862420.....&Body=Hello+Charlotte
```

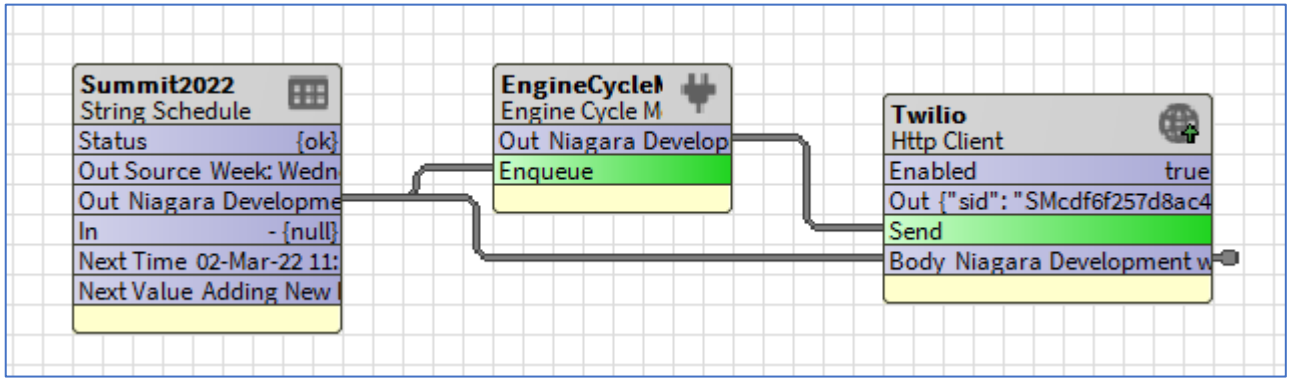
Sending the SMS

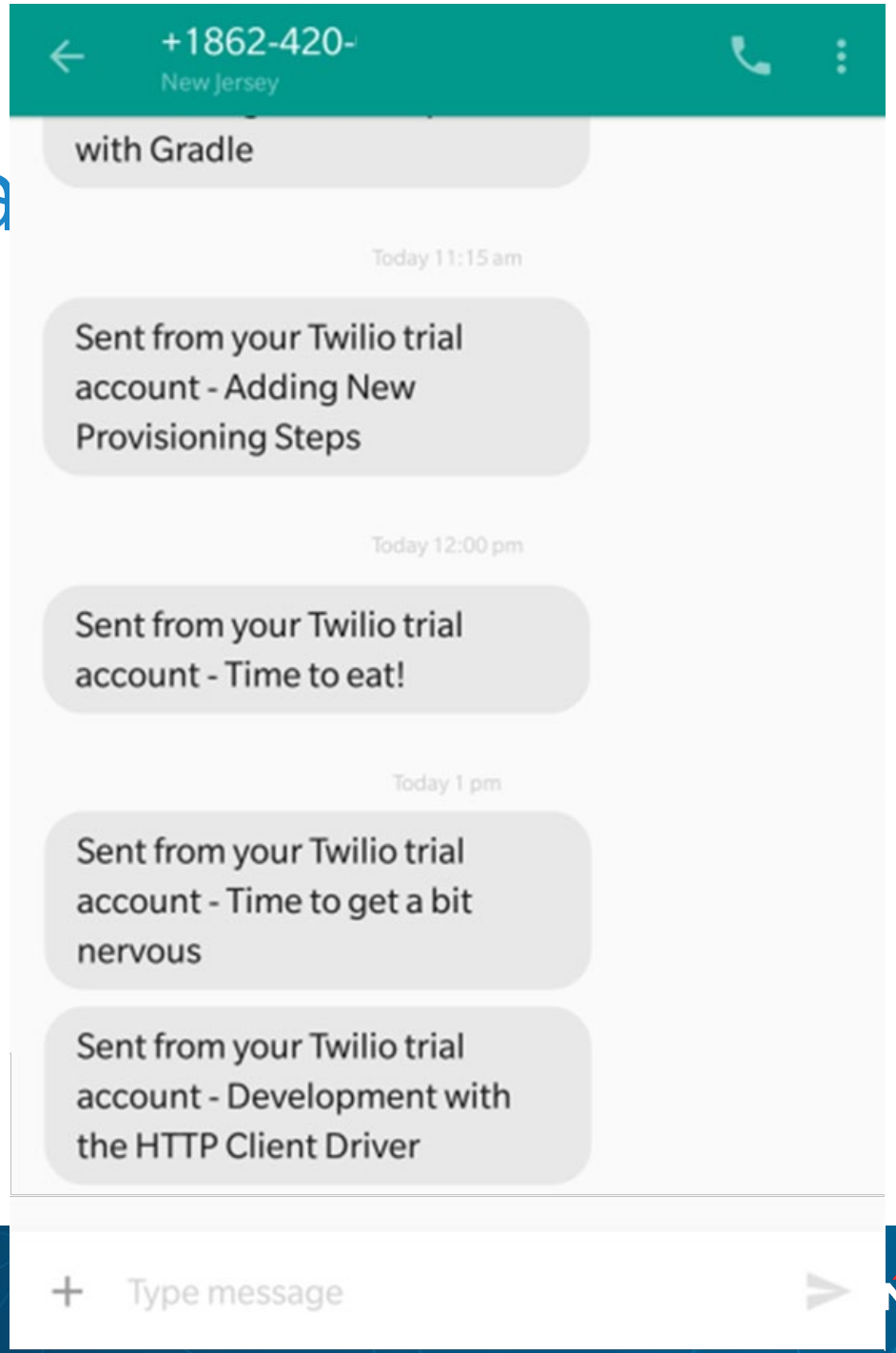


POST payload / Content-Type automatically generated from parameters



A slide with an agenda





302 Redirect, to more information

- Applications Track **Wednesday**!
  - 10.45 AM with **Curtis McKerlie**

- https://docs.niagara-community.com/bundle/HttpClientDriver

- **Bajadoc** for javax.baja.httpClient.HttpClientBuilder

Questions?

https://http.cat/429

Niagara Summit 2022

Monday, April 4 – Wednesday, April 6

Charlotte Convention Center & Westin

www.tridium.com/niagarasummit

TRIDIUM