

9970 Carroll Canyon Rd, San Diego, CA, 92131
Phone (858) 657-9800 Fax: (858) 657-9818 Toll Free 1800 324-5190
E-mail: sda_techsupport@teledyne.com Website: <http://www.teledyne-api.com>

22-002
18 April, 2022

REST API TUTORIAL FOR NUMAVIEW INSTRUMENTS

I. PURPOSE:

To provide examples of how the REST protocol can be used to collect data, change parameters, extract data logs, poll groups of parameter values, and trigger calibration functions.

II. TOOLS:

For GET commands, any web browser
For PUT commands, postman (www.postman.com) or other API platform

III. PARTS:

Computer with internet access and Numaview instrument both connected on the same network

IV. REST GET COMMANDS:

GET commands are commands used to read or “get” a value. These can be utilized using just a web browser and does not make any changes to the instrument. Note that most modern web browsers will download the response as a file with no file extension whereas older web browsers such as internet explorer will display the response within the browser. The file without an extension can have .txt added to the end in order to view the response in a text document.

1. TAGS

Tags are any parameter within the instrument that is used or measured. In the example below, we will use a concentration tag. **Note that tag names and command strings are case sensitive**

Using a web browser, we can view full details of the tag using the below command example:

http://10.20.22.10:8180/api/tag/CO_CONC

The response will be as shown below:

```
{
  "name": "CO_CONC", "type": "float", "value": "0.145923003554344", "properties": "{
    \"Precision\": 3, \"RawMin\": 0.0, \"RawMax\": 500000.0, \"EuMin\": -1000.0,
    \"EuMax\": 200000.0, \"EuTag\": \"SV_USER_UNITS\", \"Default\": 0.0, \"Name\":
    \"CO_CONC\", \"HmiLabel\": \"CO Concentration\", \"Description\": \"CO
    Concentration in PPM\", \"Group\": \"PRIGAS, LOG, TRIG, AOUTMAP, HIST, TRACK_ALL_UPDATES\",
    \"Units\": \"PPM\", \"IsValueValid\": true, \"IsReadOnly\": true, \"IsNetwork\": true,
    \"IsVisible\": true, \"IsNonVolatile\": false, \"IsDashboard\": false, \"CanDashboard\": true
  }"}
}
```

Here is a breakdown of the response string where commas separate each line. We won't go over every line in this tutorial but will cover some key ones:

`"name": "CO_CONC"` **This is the name of the tag**

`"type": "float"` **This is how the value is viewed; float refers to floating point which is a type of number where the decimal is not in a fixed position**

`"value": "0.145923003554344"` **This is the current value of the tag**

`"properties": {"Precision": 3` **This refers to the number of places after the decimal that the value is viewed on the instrument display**

```
\"RawMin\": 0.0
\"RawMax\": 500000.0
\"EuMin\": -1000.0
\"EuMax\": 200000.0
\"EuTag\": \"SV_USER_UNITS\"
\"Default\": 0.0
\"Name\": \"CO_CONC\"
\"HmiLabel\": \"CO Concentration\"
\"Description\": \"CO Concentration in PPM\"
```

`\"Group\": \"PRIGAS, LOG, TRIG, AOUTMAP, HIST, TRACK_ALL_UPDATES\"` **This tells us which groups of tags that the tag we're viewing reports in. We'll discuss groups further in the tutorial.**

```
\"Units\": \"PPM\"
\"IsValueValid\": true
```

`\"IsReadOnly\": true` **This tells us if this is a tag that we can change with a PUT command, or if it can only be changed by the software**

```
\"IsNetwork\": true
\"IsVisible\": true
\"IsNonVolatile\": false
\"IsDashboard\": false
\"CanDashboard\": true}"}
}
```

REST API TUTORIAL FOR NUMAVIEW INSTRUMENTS

22-002 Rev A (8529) 4/18/22

Page 2 of 10

PRINTED DOCUMENTS ARE UNCONTROLLED

2. TAG VALUES

When we know a tag name, and only want to get the value of the tag, we can use the below command which will generate the subsequent response:

http://10.20.22.10:8180/api/tag/CO_CONC/value

Response:

```
{"name":"CO_CONC","value":"0.496683984994888"}
```

3. TAGLIST

The taglist is a printout of all the tags that the instrument will respond to, and the properties of each tag as we saw with the tag command. Below is the command to generate a taglist, but the response string will be 100+ pages long if pasted into a word document so we will not include the entire taglist in the example.

Searching for a tag on the taglist can be difficult to find the correct tag you're looking for, so if you have any questions, please contact us at API-TECHSUPPORT@TELEDYNE.COM

<http://10.20.22.10:8180/api/taglist>

Response:

```
{"group":"","tags":[{"name":"NATIVE_APP_STATE","type":"string","value":"INITIALIZED","properties":{"Default":"","Name":"","HmiLabel":"","Description":"","Synchronization tag between native and managed","Group":"","Units":"","IsValueValid":true,"IsReadOnly":false,"IsNetwork":true,"IsVisible":true,"IsNonVolatile":false,"IsDashboard":false,"CanDashboard":false}},{name":"INSTRUMENT_MODE","type":"string","value":"SAMPLE","properties":{"Default":"","Name":"","HmiLabel":"","Description":"","Specifies the text that shows up for the instrument mode","Group":"","TRACK_ALL_CHANGES,LOG":"","Units":"","IsValueValid":true,"IsReadOnly":true,"IsNetwork":true,"IsVisible":true,"IsNonVolatile":false,"IsDashboard":false,"CanDashboard":false}},{name":"INSTRUMENT_TIME","type":"string","value":"02/17/2022 5:00:28 PM","properties":{"Default":"","Name":"","HmiLabel":"","Description":"","Current time on the instrument","Group":"","Units":"","IsValueValid":true,"IsReadOnly":false,"IsNetwork":true,"IsVisible":true,"IsNonVolatile":false,"IsDashboard":false,"CanDashboard":false}},{name":"DO_OUTPUT1","type":"bool","value":"False","properties":{"HmiValueMap":"","Default":false,"Name":"","DO_OUTPUT1":"","HmiLabel":"","Description":"","Physical Digital Output #1","Group":"","DOUT":"","Units":"","IsValueValid":true,"IsReadOnly":true,"IsNetwork":true,"IsVisible":true,"IsNonVolatile":false,"IsDashboard":false,"CanDashboard":false}},{name":"DO_OUTPUT1_MAP","type":"string","value":"Not Mapped","properties":{"Default":"","Name":"","DO_OUTPUT1_MAP":"","HmiLabel":"","Description":"","The tag mapped to digital
```

REST API TUTORIAL FOR NUMAVIEW INSTRUMENTS

22-002 Rev A (8529) 4/18/22

Page 3 of 10

PRINTED DOCUMENTS ARE UNCONTROLLED

4. TAGLIST GROUP VALUE LIST

In our first example, we saw that CO_CONC belongs to the following groups:

PRIGAS
LOG
TRIG
AOUTMAP
HIST
TRACK_ALL_UPDATES

Other parameters are also collected in these groups which makes it easier for data logging when the number of command strings that can be sent to the instrument are limited by the data logging program, but the response string does not have a limit. See below example.

<http://10.20.22.10:8180/api/valuelist/?group=HIST>

Response:

```
{"group":"HIST","values":[{"name":"O2_CONC","value":"10"}, {"name":"O2_STABILITY","value":"0"}, {"name":"CO2_CONC","value":"10"}, {"name":"CO2_STABILITY","value":"0"}, {"name":"CO_CONC","value":"-0.496628105640411"}, {"name":"CO_CONC_2","value":"50.9321937561035"}, {"name":"CO_STABILITY","value":"0.000235935774981044"}]}
```

5. DATA LOG LIST

The data log list command just shows a list of all the data logs that are setup on the instrument. See below example:

<http://10.20.22.10:8180/api/dataloglist>

Response:

```
{"logs":[{"name":"HIRES","description":"","active":true}]}
```

6. DATALOGS

Datalog data can be polled from the instrument in two different formats. This could either be extracted based on a defined page number, and number of records viewed on each page, or it can be extracted in between two different time periods.

Example 1

Defined number of records and pages. Page 1 would be the newest records, so if we ask to see page 2 with 5 records per page, this would generate a response starting from the 6th oldest record.

<http://10.20.22.10:8180/api/datalog/HIRES?page=2&recordperpage=5>

Response (NOTE: spaces added to the response for line clarity for the purposes of the tutorial):

Date & Time (Local), Date & Time (UTC), Auto Ref Ratio, Bench Temp, CO Concentration, CO Stability, Meas Detector, Oven Temp, PHT Drive., Ref 4096mV, Ref Detector, Ref Ground, Sample Flow, Sample Pressure, Wheel Temp

2/17/2022 5:40:00 PM, 2/18/2022 12:40:00 AM, 1.19251823425293,
47.98388671875, -0.496541202068329, 0.000250347424298525, 1922.47155761719,
45.972541809082, 2637.181640625, 4095.70874023437, 1637.97973632812, 0,
1760.56591796875, 28.6962261199951, 62.0275955200195

2/17/2022 5:39:00 PM, 2/18/2022 12:39:00 AM, 1.19251823425293,
47.9788818359375, -0.496563524007797, 0.000250347424298525, 1922.91455078125,
46.0130233764648, 2637.37158203125, 4095.77221679687, 1637.91650390625, 0,
1767.51098632812, 28.7013912200928, 61.975700378418

2/17/2022 5:38:00 PM, 2/18/2022 12:38:00 AM, 1.19251823425293,
47.9851303100586, -0.496568530797958, 0.00024774792836979, 1922.97778320312,
46.0400161743164, 2637.498046875, 4095.77221679687, 1638.54919433594, 0,
1782.30383300781, 28.6979465484619, 62.0370330810547

2/17/2022 5:37:00 PM, 2/18/2022 12:37:00 AM, 1.19251823425293,
47.9876327514648, -0.496535062789917, 0.00024774792836979, 1923.54736328125,
46.0142440795898, 2637.24487304687, 4095.70874023437, 1639.11877441406, 0,
1775.22009277344, 28.6983776092529, 61.9772567749023

2/17/2022 5:36:00 PM, 2/18/2022 12:36:00 AM, 1.19251823425293,
47.98388671875, -0.496587187051773, 0.000245207193074748, 1923.61059570312,
45.9541549682617, 2637.30810546875, 4095.77221679687, 1638.80236816406, 0,
1763.48278808594, 28.7000980377197, 62.0354537963867

Example 2

Defined time periods. In this example, we will ask for data between 2pm and 2:30pm on 02/17/2022. Note that we can omit the seconds field in the command as it will be assumed to be 00 if left out.

<http://10.20.22.10:8180/api/data/HIRES?t1=202202171700&t2=202202171705>

Response (NOTE: spaces added to the response for line clarity for the purposes of the tutorial):

Date & Time (Local), Date & Time (UTC), Auto Ref Ratio, Bench Temp, CO Concentration, CO Stability, Meas Detector, Oven Temp, PHT Drive., Ref 4096mV, Ref Detector, Ref Ground, Sample Flow, Sample Pressure, Wheel Temp

2/17/2022 5:00:00 PM, 2/18/2022 12:00:00 AM, 1.19251823425293,
47.98388671875, -0.496670335531235, 0.12853892147541, 1939.17663574219,
45.9921722412109, 2636.99169921875, 4095.70874023437, 1651.83728027344,
0.0632743835449219, 1774.87268066406, 28.6923522949219, 62.0354537963867

2/17/2022 5:01:00 PM, 2/18/2022 12:01:00 AM, 1.19251823425293,
47.98388671875, -0.49669149518013, 0.000218664499698207, 1939.30322265625,
46.0032119750977, 2637.05517578125, 4095.77221679687, 1652.02709960937, 0,
1758.96850585937, 28.6906280517578, 62.0055694580078

2/17/2022 5:02:00 PM, 2/18/2022 12:02:00 AM, 1.19251823425293,
47.9863891601563, -0.496633052825928, 0.000218664499698207, 1937.9111328125,
45.9983062744141, 2637.11840820312, 4095.64526367187, 1650.88818359375, 0,
1761.330078125, 28.6897678375244, 62.0212936401367

2/17/2022 5:03:00 PM, 2/18/2022 12:03:00 AM, 1.19251823425293,
47.9826354980469, -0.49664306640625, 0.000217841588892043, 1937.9111328125,
45.9394378662109, 2637.30810546875, 4095.70874023437, 1650.69836425781, 0,
1744.45336914062, 28.6906280517578, 62.0071487426758

2/17/2022 5:04:00 PM, 2/18/2022 12:04:00 AM, 1.19251823425293,
47.98388671875, -0.496636807918549, 0.000217841588892043, 1936.70886230469,
46.0007629394531, 2637.498046875, 4095.70874023437, 1649.8125, 0,
1777.025390625, 28.6936454772949, 62.0244445800781

2/17/2022 5:05:00 PM, 2/18/2022 12:05:00 AM, 1.19251823425293,
47.98388671875, -0.496623158454895, 0.000219163266592659, 1936.89868164062,
46.0154800415039, 2637.56127929687, 4095.77221679687, 1650.44519042969, 0,
1767.8583984375, 28.6919212341309, 61.9882736206055

V. REST PUT COMMANDS

Put commands are commands that are used to either write to a value or change a tag to a preset list of acceptable modes. Earlier in the tutorial, we spoke about tag parameters which include a read only value. Any tag with a read only value of false can be written to via the REST protocol. Using a PUT command requires software development tools, but we can use postman to test these functions.

Next, we will need to open postman.com which is a free, web based API platform. First, create a workspace, and try a get command to make sure it's working, also so that we can copy the response, because we will need this in order to fill in the body of our PUT command that we will push to the instrument.

1. PUT A VALUE TO A NUMERIC PARAMETER:

In this example we're going to change the target span concentration for range 2 of our CO analyzer.

Example:

First, we will use a GET command to see how the response body is formatted

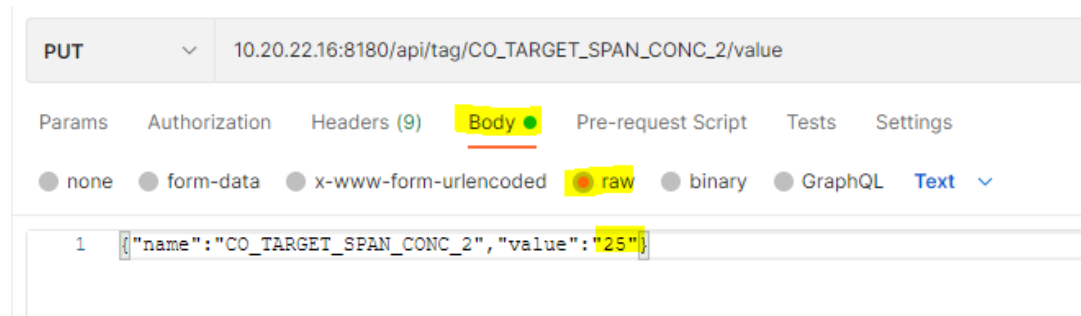
http://10.20.22.10:8180/api/tag/CO_TARGET_SPAN_CONC_2/value

Response:

```
{"name":"CO_TARGET_SPAN_CONC_2","value":"40"}
```

Now we can take the same response, and paste it into the body of the response for our PUT command. After pasting, we can change the "value" to another number and send the PUT command.

Example:



After sending the target concentration change command, we can use the GET function again to verify that the target concentration is now 25.

The screenshot shows a REST client interface. At the top, the method is 'GET' and the URL is '10.20.22.16:8180/api/tag/CO_TARGET_SPAN_CONC_2/value'. Below the URL bar are tabs for 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected, and the content type is set to 'Text'. The body contains a single line of JSON: `{ "name": "CO_TARGET_SPAN_CONC_2", "value": "25" }`. Below the body editor are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Body' tab is selected, and the view is set to 'Pretty'. The pretty-printed JSON is shown as:

```
1 { "name": "CO_TARGET_SPAN_CONC_2", "value": "25" }
2
```

2. PUT A COMMAND TO THE INSTRUMENT

PUT commands can also be used to put the instrument into diagnostic or calibration modes. In the below example, we will check the current instrument mode, and then put the instrument into an auto reference calibration mode.

Example to check the current mode:

10.20.22.16:8180/api/tag/INSTRUMENT_MODE/value

Response:

GET http://10.20.22.10:8180

http://10.20.22.10:8180/api/tag/INSTRUMENT_MODE/value

GET http://10.20.22.10:8180/api/tag/INSTRUMENT_MODE/value

Params Authorization Headers (7) **Body** Pre-request Script Tests

none form-data x-www-form-urlencoded **raw** binary Gr

1

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

```
1 {"name": "INSTRUMENT_MODE", "value": "SAMPLE"}
2
```

Next we will use a GET command to see the response body for the reset aref var which will trigger the instrument to run an aref if the value is set to true.

GET http://10.20.22.10:8180

http://10.20.22.10:8180/api/tag/RESET_AREF/value

GET http://10.20.22.10:8180/api/tag/RESET_AREF/value

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

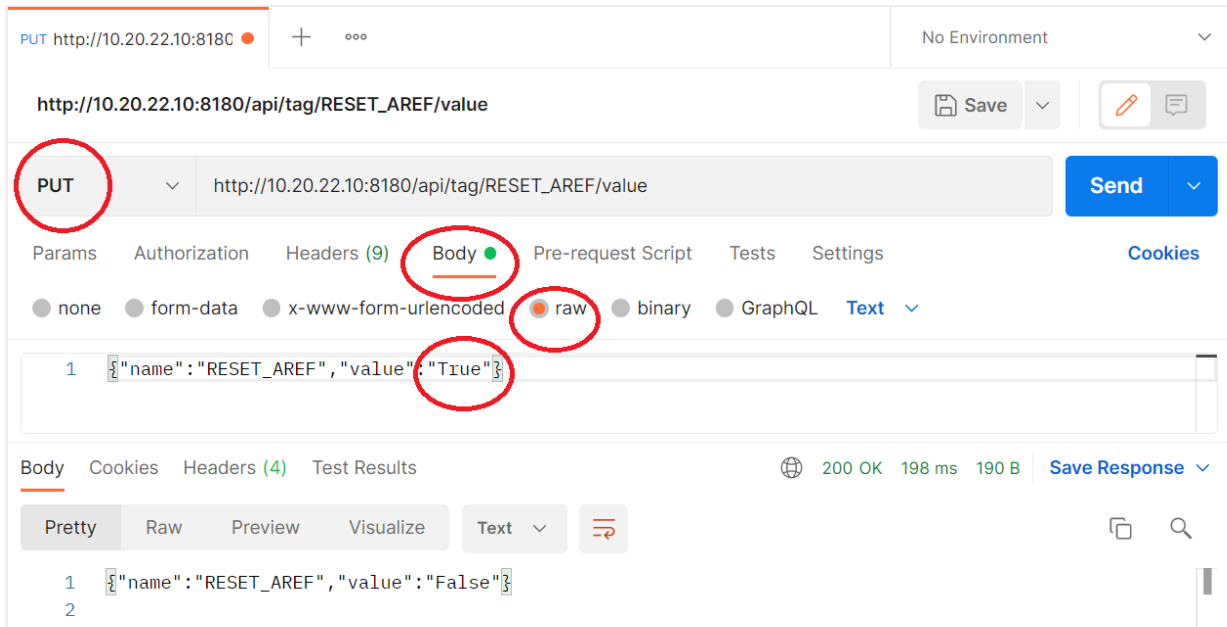
This request does not have a body

Body Cookies Headers (4) Test Results 200 OK 154 ms 190 B Save Response

Pretty Raw Preview Visualize Text

```
1 {"name": "RESET_AREF", "value": "False"}
2
```

Once we know the body response, we will then PUT the value to “True” which will now put the instrument into an aref mode.



Lastly, we will use the GET command again to verify the instrument mode is now showing that the instrument mode is in AUTO-REF

