00a Python Requests Core Auth Concepts

Any program sending REST web service calls to SystemLink must establish a web service connection, which requires the URL of the server and connection credentials to authenticate the connection request. Python sends native REST web service calls with the “requests” module, which establishes a connection that can send web service calls to any of the REST web service APIs in SystemLink.

Communicating with SystemLink using native REST web service calls requires the “requests” module

import requests

If the web service communication is happening without a web certificate from a certificate authority (using a self-signed certificate), then it is convenient to repress the certificate warnings, especially if you’re just prototyping code. You can do this with a quick invocation of the “urllib3” module.

import urllib3

urllib3.disable\_warnings(urllib3.exceptions.InsecureRequestWarning)

Opening a session from the requests module is bit like launching an internet browser application—you can use that session to send requests to any URL in the network.

session = requests.Session()

With each web service call issued by this session, you pass connection credentials, either as a user/password pair in the “auth” parameter or with an API key or Session key in the “headers” parameter

auth = (username, password)

headers = {'x-ni-api-key':api\_key, 'Content-Type':'application/json'}

You also pass the full URL to the server plus the REST web service route you want to invoke

url = "https://localhost:443/nitag/v2/tags-count"

For a GET web service route, there is no JSON body, so you can actually call those directly from a Chrome or Edge or Safari browser. For all others, the details of the request are provided in the JOSN body of the request

response = session.get(url, json={}, verify=False, headers=headers, auth=auth)

response = session.post(url, json=body, verify=False, headers=headers, auth=auth)

response = session.put(url, json=body, verify=False, headers=headers, auth=auth)

response = session.patch(url, json=body, verify=False, headers=headers, auth=auth)

response = session.delete(url, json=body, verify=False, headers=headers, auth=auth)

The response returned from your web service request will always have string representation that shows the response code (hopefully 200), and usually you can cast the response in dictionary/JSON form to inspect the response details

print(str(response))

print(str(response.json()))