

## NetSim Interfacing with Python v14.1

**Software:** NetSim v14.1 (64 bit), Python 3.11.1 or higher with NumPy and Matplotlib modules installed

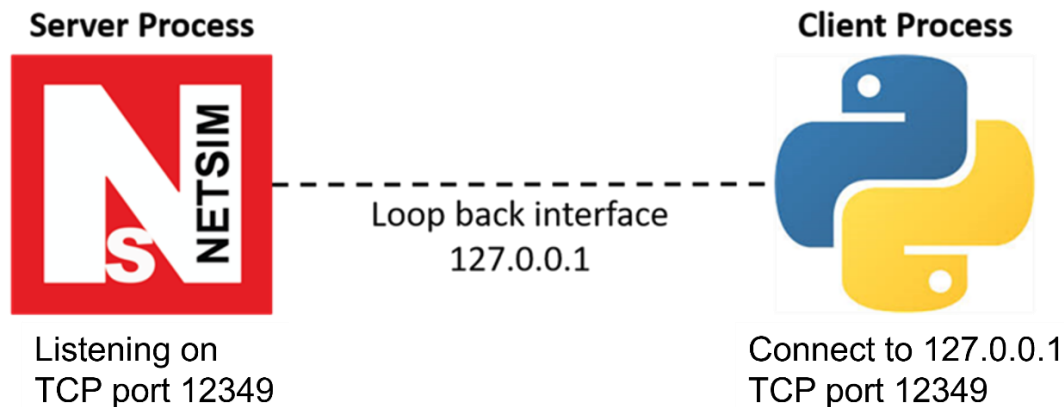
**Project Download Link:** [https://github.com/NetSim-TETCOS/NetSim\\_Python\\_Interface\\_DBR\\_v14.1/archive/refs/heads/main.zip](https://github.com/NetSim-TETCOS/NetSim_Python_Interface_DBR_v14.1/archive/refs/heads/main.zip)

Follow the instructions specified in the following link to download and setup the Project in NetSim:

<https://support.tetcos.com/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects>

### Introduction:

The provided workspace allows users to interface NetSim with Python scripts via a socket interface. This is achieved by establishing a socket connection between NetSim and a Python Script during runtime, which is used to send and receive data. Lot of work related to machine learning, artificial intelligence and specialized mathematical algorithms which can be used for networking research, can be carried out using this workspace.

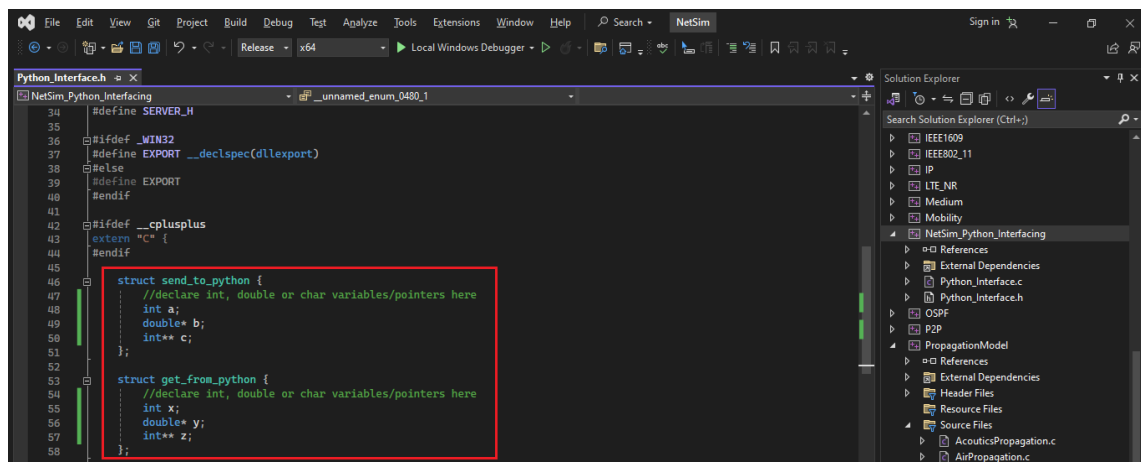


### NetSim-Python Socket Interface

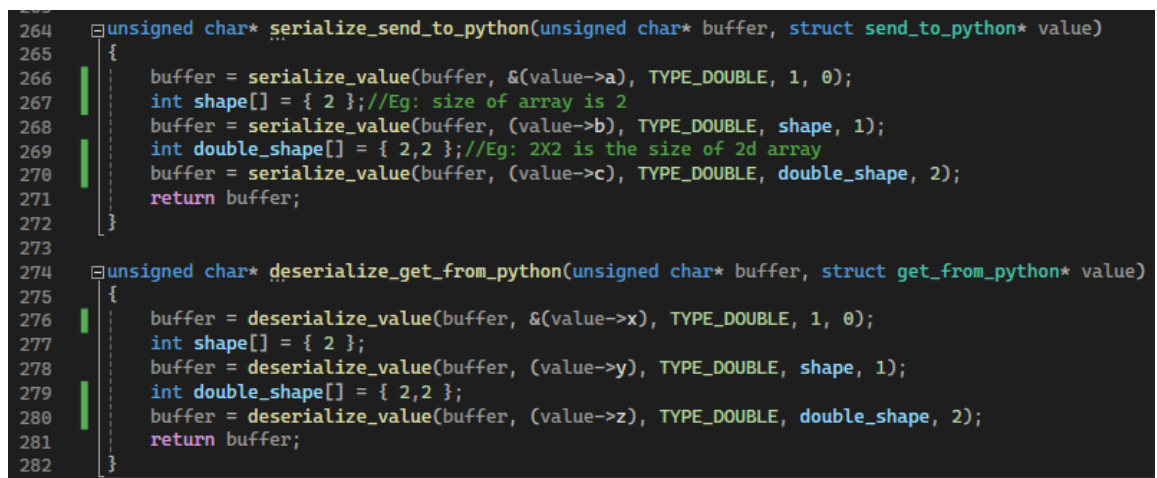
NetSim API's to interact with Python	Description
<code>init_python_interface_socket()</code>	This function initializes winsock library and setup the TCP listening socket.
<code>send_value_to_python(struct send_to_python* Var)</code>	This function is used to send data from NetSim to python.
<code>receive_value_from_python(struct get_from_python* Var):</code>	This function is used to receive data from python to NetSim.
<code>close_python_interface_socket()</code>	This function used for closing socket connection.

## Steps to Initialize variable for socket interface with Python:

- Open Source Code, under NetSim\_Python\_Interfacing Project, in the Python Interface.h file,
  - In **struct send\_to\_python** declare variables of specific data type to send to python.
  - In **struct get\_from\_python** declare variables of specific data type to receive from python.
  - Here users can declare the datatypes such as single variable int , 1d array of type double or 2d array of type int etc as shown in the image below.



- In the Python Interface.c file,
  - **serialize\_send\_to\_python(unsigned char\* buffer, struct send\_to\_python\* value)** function converts any datatype variables to bytes and sends to python.
  - **deserialize\_send\_to\_python(unsigned char\* buffer, struct send\_to\_python\* value)** function converts any datatype variables to bytes and sends back to netsim.
  - Users must initialize size of array as per requirements shown in figure below.



- Rebuild the NetSim\_Python\_Interfacing project.

### Example:

Select the section of code where you want to interface NetSim with Python.

Consider UWAN.c file,

1. Include the header and lib for calling the NetSim Python Interfacing API's.

- `#include "../NetSim_Python_Interfacing/Python_Interface.h"`
- `#pragma comment(lib,"NetSim_Python_Interfacing.lib")`

```
14  #include "main.h"
15  #include "UWAN.h"
16  #include "ErrorModel.h"
17  #include "NetSim_utility.h"
18  #include "../NetSim_Python_Interfacing/Python_Interface.h"
19
20  #pragma comment(lib,"NetSim_Python_Interfacing.lib")
```

2. Call the `init_python_interface_socket()` in the `int fn_NetSim_UWAN_Init()` function

```
25  _declspec(dllexport) int fn_NetSim_UWAN_Init()
26  {
27      init_python_interface_socket();
28      // return fn_NetSim_UWAN_Init_F();
29      return 0;
30  }
```

3. Call the `close_python_interface_socket()` in the `fn_NetSim_UWAN_Finish()` function

```
44
45  _declspec(dllexport) int fn_NetSim_UWAN_Finish()
46  {
47
48      close_python_interface_socket();
49      // return fn_NetSim_UWAN_Finish_F();
50      return 0;
51  }
```

4. Example of passing current time and packet Id to python.

```
296  int packetId = pstruEventDetails->pPacket->nPacketId;
297
298  struct send_to_python temp;
299
300  double arr_double[] = { ldEventTime / MILLISECOND, packetId };
301
302  temp.b = arr_double;
303
304  send_value_to_python(&temp);
```

5. In the `NetSim_Interface.py`, to extract the values from NetSim using the following code.

```
13  while a:
14      data = client_socket.recv(128)
15      if(data):
16          #extract values from netsim. Here mention the type specifiers such as I for int, d for double, c for char.
17          unpacked_data = struct.unpack(">dd", data) # here dd implies two double values are received.
18          print(f"Received values are",unpacked_data[0],unpacked_data[1])# Received values from NetSim
19
20      a= 5
21      b=10
22
23
24      packed_data = struct.pack(">dd", a,b)# pass the processed results back to NetSim
25
26      bytes = client_socket.send(packed_data)
27      else:
28          print("\nClosing connection")
29          a=0
```

6. To send values back to NetSim, For eg: a=5 and b =10,

```
13 while a:
14     data = client_socket.recv(128)
15     if(data):
16         #extract values from netsim. Here mention the type specifiers such as I for int, d for double,c for char.
17         unpacked_data = struct.unpack(">dd", data) # here dd implies two double values are received.
18
19         print(f"Received values are",unpacked_data[0],unpacked_data[1])# Received values from NetSim
20
21         a= 5
22         b=10
23
24         packed_data = struct.pack(">dd", a,b)# pass the processed results back to NetSim
25
26         bytes = client_socket.send(packed_data)
27     else:
28         print("\nClosing connection")
29         a=0
```

7. Print values in NetSim Console, in UWAN.c file use the following code

```
306 struct get_from_python received_temp;
307
308 double recv_arr[] = { 0.0,0.0};
309
310 received_temp.y = recv_arr;
311
312 receive_value_from_python(&received_temp);
313
314 fprintf(stderr, "\nReceived Values are %lf and %lf ", received_temp.y[0], received_temp.y[1]);
315
```

8. Now Rebuild UWAN project.  
9. Now open any scenario in UWAN and run.  
10. You will get the below message in NetSim console window.

```
C:\Users\PETER\Documents\NetSim>
BatteryModel.dll
libAloha.dll
libApplication.dll
libDSR.dll
libUWAN.dll
PropagationModel.dll
This message is normally shown if users link their own code to NetSim.
*****
Press any key to continue...
NetworkStack loaded from path- C:\Users\PETER\Documents\NetSim\Workspaces\DBR_IN_UWAN_v14_1\bin_x64/NetworkStack.dll

***
NetSim start
Network Stack loaded
Error in creating C:\Users\PETER\AppData\Local\Temp\NetSim\std_14.1\log directory. Error number 17
Initializing simulation
Config file reading complete
License re-validation complete
Protocol binaries loaded
Executing command --- DEL "C:\Users\PETER\AppData\Local\Temp\NetSim\std_14.1\*.pcap"
Could Not Find C:\Users\PETER\AppData\Local\Temp\NetSim\std_14.1\*.pcap
Emulation is disabled
Stack variables initialized
Could Not Find C:\Users\PETER\AppData\Local\Temp\NetSim\std_14.1\Plot_*
Metrics variables initialized

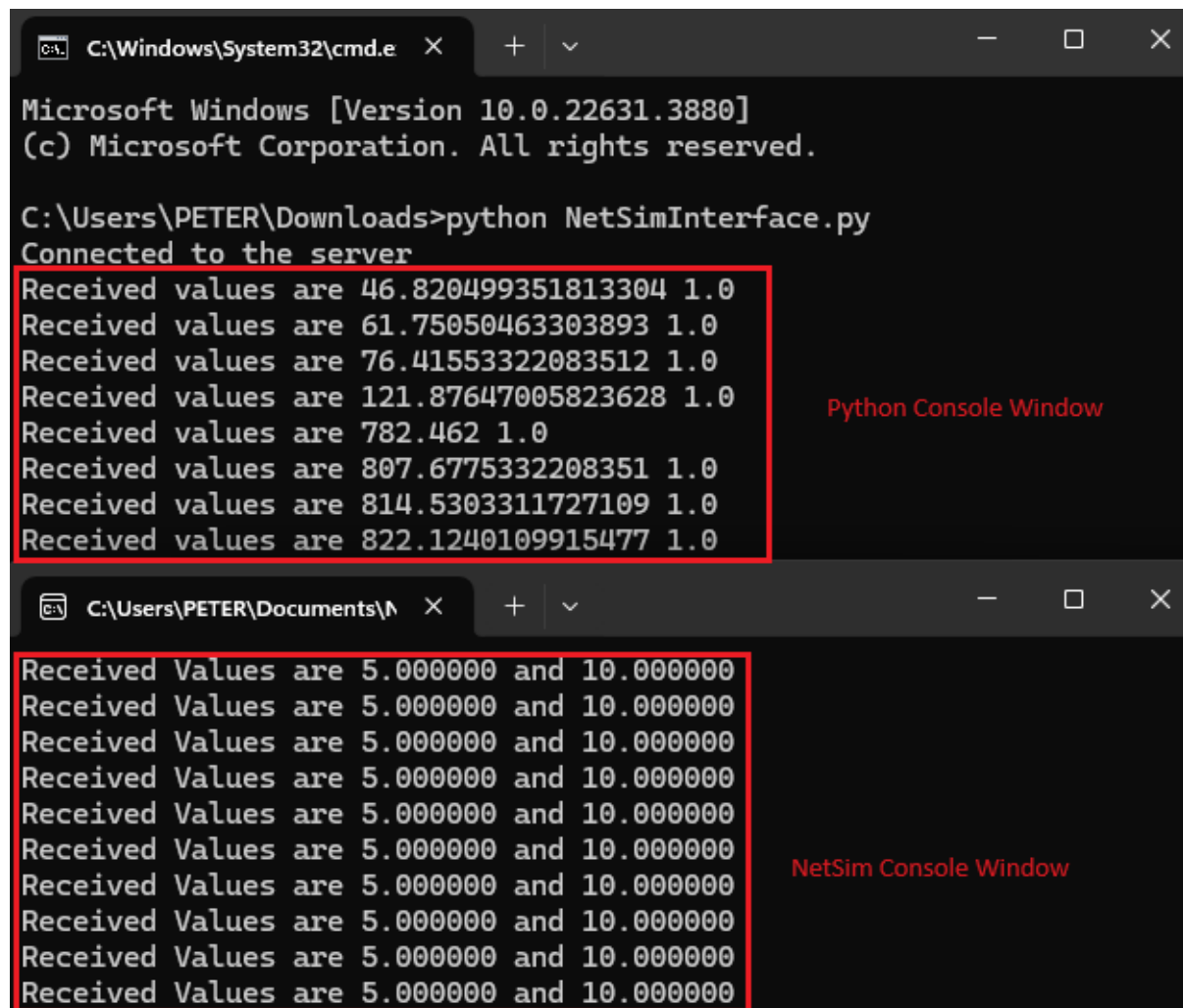
Initialising Winsock...Initialized.
Listening
```

11. Now run the NetSimInterface.py file in the command prompt from the saved location.

```
C:\Windows\System32\cmd.exe>
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PETER\Downloads>python NetSimInterface.py
```

12. Values sent from NetSim are printed on Python console, as well as values sent from Python are printed to NetSim console.



The image displays two screenshots of a Windows command prompt. The top screenshot shows the execution of a Python script named 'NetSimInterface.py' in the directory 'C:\Users\PETER\Downloads'. The script successfully connects to a server and outputs eight lines of received values, each consisting of a large floating-point number followed by '1.0'. These lines are highlighted with a red rectangular box. The bottom screenshot shows the output of the same script in a different directory, 'C:\Users\PETER\Documents\'. It displays ten lines of received values, each consisting of the string 'Received Values are' followed by '5.000000' and '10.000000' separated by 'and'. These lines are also highlighted with a red rectangular box. To the right of each set of highlighted text, the respective window is labeled: 'Python Console Window' for the top and 'NetSim Console Window' for the bottom.

```
C:\Windows\System32\cmd.e X + v
Microsoft Windows [Version 10.0.22631.3880]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PETER\Downloads>python NetSimInterface.py
Connected to the server
Received values are 46.820499351813304 1.0
Received values are 61.75050463303893 1.0
Received values are 76.41553322083512 1.0
Received values are 121.87647005823628 1.0
Received values are 782.462 1.0
Received values are 807.6775332208351 1.0
Received values are 814.5303311727109 1.0
Received values are 822.1240109915477 1.0

C:\Users\PETER\Documents\N X + v
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
Received Values are 5.000000 and 10.000000
```

Python Console Window

NetSim Console Window