

Follow these steps to integrate the Shared Data Access System in your [Python](#) code. All the examples were successfully tested in Windows and some linux distributions like Gentoo, Fedora and Red Hat.

Download extra packages

In case you don't already have a numeric Python package installed in your system (Numpy, numarray or Numeric) download [Numpy](#).

In order to plot and view data, you may also need more packages. For this, [Matplotlib](#) is recommended.

Instead, you can also install [Anaconda](#), which is a popular Python platform that includes these packages and many others.

Download the libraries

Download the following file into a folder of your system:

[sdas-1.1.tar.gz](#)

If you have Anaconda installed, you should install with *pip*. In Linux, you can run *pip* on a regular shell. In Windows you will need to run on the *Anaconda Prompt* shell: click *Start* and search or select *Anaconda Prompt* from the menu. Run this command in the same folder where the file *sdas-1.0.tar.gz* is located, or use the full path:

```
pip install sdas-1.1.tar.gz
```

If you don't have Anaconda (or for some reason *pip* doesn't work), you can unpack *sdas-1.1.tar.gz* and run:

```
python setup.py install
```

You may need to have root or administrator privileges.

Another option is to copy the *sdas* folder, inside the package, to the same folder where you have your code.

Get a connection to the sdas server

In a the python interpreter:

```
from sdas.core.client.SDASClient import SDASClient
from sdas.core.SDAStime import Date, Time, TimeStamp
host='baco.ipfn.ist.utl.pt'
port=8888
client = SDASClient(host, port)
```

Search events

```
found = client.searchDeclaredEventsByName('S')
```

```

found = client.searchDeclaredEventsByName('SHOT', 'pt')

found = client.searchDeclaredEventsByUniqueID('SHOT', 'pt')

found = client.searchDeclaredEventsByDescription('SHOT')

found = client.searchDeclaredEventsByDescription('SHOT', 'pt')

for item in found:
    print 'item', item
max = client.searchMaxEventNumber('0x0000')

min = client.searchMinEventNumber('0x0000')

```

Search parameters

```

parametersFound = client.searchParametersByName('DENS')

parametersFound = client.searchParametersByName('DENS', 'pt')

parametersFound = client.searchParametersByUniqueID('DENS')

for p in parametersFound:
    print p['descriptorUID']['uniqueID']

```

Search data

This function returns the parameters unique identifiers where the data isn't null for the selected event:

```

dataFound = client.searchDataByEvent('0x0000', 17898)
for d in dataFound:
    print d

```

Get data

NOTE: The unique identifiers are CASE-SENSITIVE

The returned data structure gives you information about:

- start time
- end time
- time of the event
- mime_type
- the parameter unique identifier

Import extra libraries

```

import numpy
import matplotlib.pyplot as plt

```

Data for only one parameter

```
dataStructArray=client.getData('POST.PROCESSED.DENSITY','0x0000', 17898)
dataStruct=dataStructArray[0]
density=dataStruct.getData()
tstart = dataStruct.getTStart()
tend = dataStruct.getTEnd()
```

Calculate the time between samples

```
tbs = (tend.getTimeInMicros() - tstart.getTimeInMicros())/len(density)
```

Get the events associated with this data

```
events = dataStruct.get('events')
```

The event time (I'm assuming the event I want is at the index 0, but I should check first...)

```
tevent = TimeStamp(tstamp=events[0].get('tstamp'))
```

The delay of the start time relative to the event time

```
delay = tstart.getTimeInMicros() - tevent.getTimeInMicros()
```

Finally create the time array

```
times = numpy.linspace(delay,delay+tbs*(len(density)-1),len(density))
```

And plot the data

```
plt.plot(times, density)
```

If the plot doesn't show, you may need to run.

```
plt.show()
```

TimeStamps

To maintain compatibility with the Java and C++ libraries, the month in the SDAS TimeStamps start at 0 (January=0). You can convert to a Python datetime object with the method `getAsDateTime()`. For example:

```
tstart = dataStruct.getTStart().getAsDateTime()
```

You can also convert from datetime to TimeStamp with `fromDateTime()`. For example:

```
startdate = datetime.datetime(2026, 8, 31, 16, 19)
tstart = TimeStamp()
tstart.fromDateTime(startdate)
```