

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

Test the java connection

Before starting to use the system check if JAVA is well configured in MatLab, type in the matlab console:

```
version -java
```

If you get an answer like: Java 1.5.0_04 with Sun Microsystems Inc. Java HotSpot(TM) Server VM, then you're ready to start.

If your version of MatLab has an older version of Java, then you will need to install a newer version and set the MATLAB_JAVA environment variable, as described next.


Set the MATLAB_JAVA value

You should use the latest version (at least 1.5) of the Sun [Java](#). If your copy of MatLab has a more recent version of Java you can skip this step.

First you have to find out where is your *java home* located. To avoid errors, download and run this [utility](#). The value returned by the utility is the **MATLAB_JAVA** value.

NOTE: Each time you change system properties, you have to restart MatLab

Now you have to set the **MATLAB_JAVA** as a system variable:

 Open the windows *System Properties* (right - click on My Computer or go to the Control Panel)
Select the tab *Advanced*

Click on *Environment Variables...*

In the system variables click *New...*

The Variable name is: **MATLAB_JAVA**

In the value field (supposing you have the *java_home* in the libraries in C:\Program Files\java\jdk1.5.0_04\jre) enter the following value: C:\Program Files\java\jdk1.5.0_04\jre

 Supposing you have the *java_home* in /opt/jdk1.5.0_04/jre/ :

```
export MATLAB_JAVA="/opt/jdk1.5.0_04/jre/"
```

Download the libraries

if you have a recent version of Matlab that includes a XML-RPC library, download the following library into a folder of your system:

[SDAS Client](#)

If you are not sure if your version supports it, you can run on the Matlab console:

```
javaclasspath
```

This will give you a list of all java libraries being loaded at start. If there are no references to *apache_xmlrpc_client.jar*, *apache_xmlrpc_common.jar* and *apache_xmlrpc_server.jar*, then you need to download the following files instead:

[Apache XML-RPC](#)

[Apache Jakarta Commons](#)

[SDAS Core Libraries](#)

Set the classpath

Add all of the downloaded libraries to your system *classpath*.

Use the matlab **static path**. More information at the [matlab documentation site](#).

If your are unable to set the **static path** on your computer you can use the [dynamic path](#).

Get a connection to the sdas server

```
import org.sdas.core.client.*;
client = SDASClient('baco.ipfn.tecnico.ulisboa.pt', 8890);
```

If you get an error check the classpath.

Search events

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

found = client.searchDeclaredEventsByName('S');

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

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

found = client.searchDeclaredEventsByDescription('SHOT');

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

for i=1:1:size(found)
    found(i)
end

max = client.searchMaxEventNumber('0x0000')

min = client.searchMinEventNumber('0x0000')
```

Search events in a time window

NOTE: You can construct time with a resolution of picoseconds, just add to the example values for millis, micros, nanos and picos

NOTE 2: Date constructors have the months index to 0 (January is 0 and December is 11)

Search events in December 2005:

```
date_start = Date(2005, 11, 1);
date_end = Date(2005, 11, 31);
tstart = TimeStamp(date_start);
tend = TimeStamp(date_end);
eventsFound = client.searchEventsByEventTimeWindow(tstart, tend);
for i = 1:1:size(eventsFound)
    eventsFound(i)
end
```

Search events in the 22 December 2005 between 5pm and 6pm:

```
date_start = Date(2005, 11, 22);
date_end = Date(2005, 11, 22);
time_start = Time(17, 0, 0);
time_end = Time(18, 0, 0);
tstart = TimeStamp(date_start, time_start);
tend = TimeStamp(date_end, time_end);
eventsFound = client.searchEventsByEventTimeWindow(tstart, tend);
for i = 1:1:size(eventsFound)
    eventsFound(i)
end
```

Search parameters

```
parametersFound = client.searchParametersByName('DENS');

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

parametersFound = client.searchParametersByUniqueID('DENS');

parametersFound = client.searchParametersByDescription('current');

parametersFound = client.searchParametersByDescription('corrente', 'pt');

for i = 1:1:size(parametersFound)
    parametersFound(i)
end
```

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 i = 1:1:size(dataFound)
    dataFound (i)
end
```

Get data

NOTE: The unique identifiers are CASE-SENSITIVE

Data for only one parameter

```
dataStruct=client.getData('POST.PROCESSED.DENSITY','0x0000', 17898)
dataStruct=dataStruct(1);
```

Data for several parameters in the same event

```
dataStruct=client.getMultipleData({'POST.PROCESSED.DENSITY',
'POST.PROCESSED.IPLASMA'}, '0x0000', 17898)
dataStructDens=dataStruct(1,1);
dataStructIP=dataStruct(2,1);
dens=dataStructDens.getData();
ip=dataStructIP.getData();
```

Data for several parameters in different events

```
dataStruct=client.getMultipleData({'POST.PROCESSED.DENSITY',
'POST.PROCESSED.IPLASMA'}, {'0x0000','0x0000'}, [17898,17899])
dataStructDens=dataStruct(1,1);
dataStructIP=dataStruct(2,1);
dens=dataStructDens.getData();
ip=dataStructIP.getData();
```

Data for the same parameter in different events

```
dataStruct=client.getMultipleData('POST.PROCESSED.DENSITY',{'0x0000','0x0000'},
[17898,17899])
dataStructDens=dataStruct(1,1);
dataStructIP=dataStruct(2,1);
dens=dataStructDens.getData();
ip=dataStructIP.getData();
```

Data for the same parameter in different event numbers

```
dataStruct=client.getMultipleData('POST.PROCESSED.DENSITY', '0x0000',
[17898,17899])
```

This data structure gives you information about:

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

The following example shows how to calculate and plot the density at ISTTOK:

1. The data

```
dataStruct=client.getMultipleData({'CENTRAL.OS9_ADC_VME_I8.IF0CS',
'CENTRAL.OS9_ADC_VME_I8.IF0SN'}, '0x0000', 11244);
cosine = dataStruct(1,1).getData;
isine = dataStruct(2,1).getData;
```

2. Calculate the phase

```
len = length(cosine)
cosavg = cosine - max(movavg(cosine, 10, 10))/2 - min(movavg(cosine, 10,
10))/2;
sinavg = isine - max(movavg(isine, 10, 10))/2 - min(movavg(isine, 10,
10))/2;
for j = 1:len
    phase(j) = atan2(double(sinavg(j)), double(cosavg(j)));
end
```

3. Unwrap

```
unwrapped = unwrap(phase);
```

4. The start time

```
tstart = dataStruct(1,1).getTStart
```

5. The end time

```
tend = dataStruct(1,1).getTEnd
```

6. The time between samples is:

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

7. The events

```
events = dataStruct(1,1).getEvents;
```

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

```
tevent = events(1,1).getTimeStamp
```

9. The delay of the start time relative to the event time

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

10. Create the time array

```
times = delay:tbs:delay+tbs*(len-1);
```

11. Normalize the data

```
dens = -7e17 * unwrapped;  
thold = dens(len-100:len);  
density = dens-mean(thold);
```

11. Plot the chart

```
plot(times, density)
```