If you are using a version prior to the 3.8 update, follow these steps to integrate the Shared Data Access System in your Octave code. All the examples were successfully tested in MS Windows and Linux distributions like Fedora, Gentoo and Debian.

Installation of Java support

Although no issues with OpenJDK have been reportd, it is recommended to use the last version (at least 1.5) of the Oracle Java Development Kit (JDK).

Linux

Many Linux distros offer java-octave on their repositories. Since the exact name of the package varies and in some cases it comes bundled with other packages, you should refer to the documentation of the distro you are using. If you weren't able to install the Java support from a repository, please follow these instructions.

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 **JAVA_JRE** value.
Now you have to set **JAVA_HOME** as a system variable:

Supposing you have the **JAVA_JRE** in `/opt/jdk1.6.0_18/`jre:

```
export JAVA_HOME="/opt/jdk1.6.0_18/"
```

Download and install the Octave JAVA support. Packages installation instructions can be found on this Octave documentation page.

Windows

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 **JAVA_JRE** value.
Now you have to set **JAVA_HOME** 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: **JAVA_HOME**
In the value field (supposing the value of **JAVA_JRE** is `C:\java\jdk1.6.0_18\`jre) enter the following value: `C:/java/jdk1.6.0_18` (notice that it uses / instead of \).

Attention: Try avoiding paths with spaces. If you already have Java installed on your machine, one solution is to create a symbolic link to your installation and set the variable to it. The build architecture of Octave and Java must match (32 bits with 32 bits, 64 bits with 64 bits).

After this is done, follow the instructions on the SourceForge download page, to load the Java support package. More information can be found here.

Download the libraries

Download the following libraries into a folder of your system:

Apache XML-RPC
Apache Jakarta Commons

Since the server at baco computer uses an older version, if you are planing on accessing it server, you should download this version of SDAS Core Libraries and Client:

SDAS Core Libraries
SDAS Client

If you are using a different server, download these instead:

SDAS Core Libraries
SDAS Client

Add the SDAS libraries to the octave classpath

These example assume that you have saved all the jar files in the folder /home/user/sdas/

```
javaaddpath("/home/user/sdas/SDAS.jar")
javaaddpath("/home/user/sdas/SDASClient.jar")
javaaddpath("/home/user/sdas/commons-codec-1.3.jar")
javaaddpath("/home/user/sdas/xmlrpc-2.0.jar")
```

Beware that on Windows you must use / instead of the regular \. If you have saved all the jar files in the folder C:\sdas\, for example:

```
javaaddpath("C:/sdas/SDAS.jar")
javaaddpath("C:/sdas/SDASClient.jar")
javaaddpath("C:/sdas/commons-codec-1.3.jar")
javaaddpath("C:/sdas/xmlrpc-2.0.jar")
```

Get a connection to the sdas server

```
client = java_new("org.sdas.core.client.SDASClient",
"baco.ipfn.ist.utl.pt", 8888);
```

Search events

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

found = client.searchDeclaredEventsByName('S');

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

found = client.searchDeclaredEventsByUniqueID('SHOT', 'en');
```

```
found = client.searchDeclaredEventsByDescription('SHOT');

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

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

max = client.searchMaxEventNumber('0x0000')

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

Search events in a time window

NOTE: You can construct time with a resolution of picosseconds, 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 = java_new("org.sdas.core.time.Date", 2005, 11, 1);
date_end =  java_new("org.sdas.core.time.Date", 2005, 11, 31);
tstart =  java_new("org.sdas.core.time.TimeStamp", date_start);
tend =  java_new("org.sdas.core.time.TimeStamp", date_end);
eventsFound = client.searchEventsByEventTimeWindow(tstart, tend);
for i = 1:1:size(eventsFound)
    eventsFound(i).toString
end
```

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

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

Search parameters

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

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

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

parametersFound = client.searchParametersByDescription('current');


for i = 1:1:size(parametersFound)
    parametersFound(i).toString
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

NOTE2: Some java types like float and long are not automatically recognised by octave. A transformation must be manually performed using the intValue and doubleValue methods

The returned data structure gives you information about:

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

Data for only one parameter

```
dataStructArray=client.getData('POST.PROCESSED.DENSITY','0x0000', 17898)
dataStruct=dataStructArray(1);
dens=dataStruct.getData;
for i=1:length(dens)
density(i)=dens(i).doubleValue;
end
tstart = dataStruct.getTStart;
tend = dataStruct.getTEnd;
```

Calculate the time between samples

```
tbs= (tend.getTimeInMicros.intValue -
tstart.getTimeInMicros.intValue)/length(density);
```

Get the events associated with this data

```
events = dataStruct.getEvents;
```

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

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

The delay of the start time relative to the event time

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

Finally create the time array

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

And plot the data

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

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])
```

More example code is available on the Octave Code page.