Follow these steps to integrate the Shared Data Access System in your IDL code.
All the examples were successfully tested in MS Windows and some linux distributions like Gentoo, Fedora and Red Hat.

Download the libraries

Download the following libraries into a folder of your system:

Apache XML-RPC
Apache Jakarta Commons
SDAS Core Libraries
SDAS Client

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

SDAS Core Libraries
SDAS Client

Set the classpath

Add all of the previous libraries to your system classpath:

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: **CLASSPATH**
In the value field (supposing you saved the libraries in *c:\sdas*) enter the following value:

```
.;C:\sdas\xmlrpc-2.0.jar;C:\sdas\commons-codec-1.3.jar;C:\sdas\SDAS.jar;C:\sdas\
```

Supposing you saved the libraries in */home/yourname/sdas/* do:

```
export SDAS_LIBS=/home/yourname/sdas
export CLASSPATH="$CLASSPATH:.:$SDAS_LIBS/xmlrpc-2.0.jar:$SDAS_LIBS/
SDAS.jar:$SDAS_LIBS/commons-codec-1.3.jar:$SDAS_LIBS/SDASClient.jar"
```

Set the IDLJAVAB_LIB_LOCATION value

First you have to find out where is your java home located. To avoid errors, download and run this utility
Knowing your java home, the value of the **IDLJAVAB_LIB_LOCATION** will be:

```
java_home\lib\i386\client\
or
java_home\bin\client\
```

Example:

```
C:\Program Files\java\jdk1.5.0_04\jre\lib\i386\client
```

```
java_home/lib/i386/client/
```

Example:

```
/opt/jdk1.5.0_04/jre/lib/i386/client/
```

Now you have to set the **IDLJAVAB_LIB_LOCATION** 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: **IDLJAVAB_LIB_LOCATION**
In the value field (supposing you have the java_home in the libraries in `C:\Program Files\java\jdk1.5.0_04\jre` and the client directory in `lib\i386\client`) enter the following value:

```
C:\Program Files\java\jdk1.5.0_04\jre\lib\i386\client
```

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

```
export IDLJAVAB_LIB_LOCATION="/opt/jdk1.5.0_04/jre/lib/i386/client/"
```

Test the java connection

NOTE: Each time you change system properties, you have to restart IDL
Before starting to use the system check if IDL can use JAVA

```
oJSession = OBJ_NEW('IDLjavaObject$IDLJAVABRIDGESESSION')
oJVersion = oJSession -> GetVersionObject()
PRINT, "Java version=", oJVersion -> GetJavaVersion()
```

If you get an answer like: `Java version=1.5.0_04`, then you're ready to start. Otherwise check all the previous steps.

Get a connection to the SDAS server

```
oJSDASConnection=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_CLIENT_SDASCLIENT',
'org.sdas.core.client.SDASClient', 'baco.ipfn.ist.utl.pt', 8888)
```

If you get an error check the classpath.

Search events

NOTE: Only the first result is printed! Change the idl code to print more results (to get the number of results, check the SIZE of the returned value)

```
eventsDescFound=oJSDASConnection -> searchDeclaredEventsByName('SHOT')
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection -> searchDeclaredEventsByName('SHOT',
'pt')
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection -> searchDeclaredEventsByUniqueID('0x0')
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection ->
searchDeclaredEventsByDescription('start')
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection->searchDeclaredEventsByDescription('descarga',
'pt')
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection->searchEventsByEventNumber(12020)
PRINT, eventsDescFound[0] -> toString()

eventsDescFound=oJSDASConnection->searchMaxEventNumber('0x0000')
PRINT, eventsDescFound

eventsDescFound=oJSDASConnection->searchMinEventNumber('0x0000')
PRINT, eventsDescFound
```

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=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_DATE',
'org.sdas.core.time.Date',2005, 11, 1)
date_end=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_DATE',
'org.sdas.core.time.Date', 2005, 11, 31)
tstart=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIMESTAMP',
'org.sdas.core.time.TimeStamp', date_start)
tend=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIMESTAMP',
'org.sdas.core.time.TimeStamp', date_end)
eventsFound=oJSDASConnection->searchEventsByEventTimeWindow(tstart, tend)
size = SIZE(eventsFound)
n=size[1]
PRINT, "N=", n
FOR i = 0L, n-1 DO PRINT, eventsFound[i]->toString()
```

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

```
date_start=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_DATE',
'org.sdas.core.time.Date',2005, 11, 22)
date_end=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_DATE',
'org.sdas.core.time.Date', 2005, 11, 22)
time_start=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIME',
'org.sdas.core.time.Time', 17, 0, 0)
time_end=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIME',
'org.sdas.core.time.Time', 18, 0, 0)
tstart=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIMESTAMP',
'org.sdas.core.time.TimeStamp', date_start, time_start)
tend=OBJ_NEW('IDLjavaObject$ORG_SDAS_CORE_TIME_TIMESTAMP',
'org.sdas.core.time.TimeStamp', date_end, time_end)
eventsFound = oJSDASConnection->searchEventsByEventTimeWindow(tstart,
tend)
size = SIZE(eventsFound)
n=size[1]
PRINT, "N=", n
FOR i = 0L, n-1 DO PRINT, eventsFound[i]->toString()
```

Search parameters

NOTE: Only the first result is printed! Change the idl code to print more results (to get the number of results, check the SIZE of the returned value)

```
parametersFound=oJSDASConnection -> searchParametersByName('DENS')
PRINT, parametersFound[0] -> toString()

parametersFound=oJSDASConnection -> searchParametersByName('DEN', 'pt')
PRINT, parametersFound[0] -> toString()

parametersFound=oJSDASConnection -> searchParametersByUniqueID('DENSITY')
PRINT, parametersFound[0] -> toString()
```

```
parametersFound=oJSDASConnection ->
searchParametersByDescription('current')
PRINT, parametersFound[0] -> toString()

parametersFound=oJSDASConnectionsearchParametersByDescription('corrente',
'pt')
PRINT, parametersFound[0] -> toString()
```

Search data

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

```
dataFound = oJSDASConnection -> searchDataByEvent('0x0000', 12050)
size = SIZE(dataFound)
n=size[1]
PRINT, "N=", n
FOR i = 0L, n-1 DO PRINT, dataFound[i]
```

Get data

If data exists this method will return an array with one or more data structures (in the case data was acquired with different acquisition frequencies) .

NOTE: The unique identifiers are CASE-SENSITIVE Data for only one parameter.

```
dataStruct=oJSDASConnection->getData('POST.PROCESSED.DENSITY','0x0000',
17899)
dataStruct=dataStruct[0]
```

Data for several parameters in the same event

```
dataStruct=oJSDASConnection->getMultipleData(['POST.PROCESSED.DENSITY',
'POST.PROCESSED.IPLASMA'],'0x0000', 17899)
dataStructDens = dataStruct[0]
dataStructDens = dataStructDens[0]
density = dataStructDens->GetData()

dataStructIP = dataStruct[1]
dataStructIP = dataStructIP[0]
ip = dataStructIP->GetData()
```

Data for several parameters in different events

```
dataStruct=oJSDASConnection->getMultipleData(['POST.PROCESSED.DENSITY',
'POST.PROCESSED.IPLASMA'],['0x0000','0x0000'], [17898, 17899])
dataStructDens = dataStruct[0]
dataStructDens = dataStructDens[0]
density = dataStructDens->GetData()
```

```
dataStructIP = dataStruct[1]
dataStructIP = dataStructIP[0]
ip = dataStructIP->GetData()
```

Data for the same parameter in different events

```
dataStruct=oJSDASConnection->getMultipleData('POST.PROCESSED.DENSITY',
['0x0000','0x0000'], [17898, 17899])
dataStructDens17898 = dataStruct[0]
dataStructDens17898 = dataStructDens17898[0]
density17898 = dataStructDens17898->GetData()

dataStructDens17899 = dataStruct[1]
dataStructDens17899 = dataStructDens17899[0]
dens17899 = dataStructDens17899->GetData()
```

Data for the same parameter in different event numbers

```
dataStruct=oJSDASConnection->getMultipleData('POST.PROCESSED.DENSITY',
'0x0000', [17898, 17899])
dataStructDens17898 = dataStruct[0]
dataStructDens17898 = dataStructDens17898[0]
density17898 = dataStructDens17898->GetData()

dataStructDens17899 = dataStruct[1]
dataStructDens17899 = dataStructDens17899[0]
dens17899 = dataStructDens17899->GetData()
```

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 plot data for the IIGBT parameter:

1. The data

```
y = dataStruct -> getData()
```

2. The number of points

```
points = SIZE(y)
npoints = points[1]
```

3. The start time

```
tstart= dataStruct -> getTStart()
```

4. The end time

```
tend= dataStruct -> getTEnd()
```

4. The time between samples is:

```
tbs= (tend->getTimeInMicros() - tstart->getTimeInMicros())/npoints
```

5. The events

```
events = dataStruct -> getEvents()
```

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

```
tevent = events[0] -> getTimeStamp()
```

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

```
tini = tstart->getTimeInMicros() - tevent->getTimeInMicros()
```

8. Create the time array

```
x = FLTARR(npoints)
FOR i = 0L, npoints-1 DO x[i] = ti + i*tbs
```

9. Plot the chart

```
PLOT, x, y
```