Follow these steps to integrate the Shared Data Access System (SDAS) in your Mathematica code.
This tutorial uses the *J/Link* package, which is included in the standard installation of Mathematica from version 4.1.
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"
```

Test the java connection

First you will need to load the *J/Link* package:

```
Needs["JLink`"]
```

Check the version of java that Mathematica is using. You can do this with the fuction
`ShowJavaConsole[]`
The SDAS libraries requires java 1.5 or higher. If this is what your version of Mathematica uses, you can simply call

```
ReinstallJava[];
```

If not, you can tell it to use another java runtime. In Linux, it could be something like this, for example:

```
ReinstallJava[CommandLine->"/opt/jdk1.5.0_06/jre/bin/java"];
```

In Windows, it could be:

```
ReinstallJava[CommandLine->"C:\\Program
Files\\java\\jdk1.5.0_06\\jre\\java.exe"];
```

Get a connection to the sdas server

```
client=JavaNew["org.sdas.core.client.SDASClient",
"baco.ipfn.ist.utl.pt",8888];
```

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"];

found=client@searchDeclaredEventsByDescription["SHOT"];

found=client@searchDeclaredEventsByDescription["SHOT","pt"];

Do[Print[found[[i]]@toString[]], {i, Length[found]}];

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:

```
datestart=JavaNew["org.sdas.core.time.Date", 2005, 11, 1];
dateend=JavaNew["org.sdas.core.time.Date", 2005, 11, 31];
tstart=JavaNew["org.sdas.core.time.TimeStamp", datestart];
tend=JavaNew["org.sdas.core.time.TimeStamp", dateend];

eventsFound=client@searchEventsByEventTimeWindow[tstart,tend];

Do[Print[eventsFound[[i]]@toString[]], {i, Length[eventsFound]}];
```

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

```
datestart=JavaNew["org.sdas.core.time.Date", 2005, 11, 22];
dateend=JavaNew["org.sdas.core.time.Date", 2005, 11, 22];
timestart=JavaNew["org.sdas.core.time.Time", 17, 0,0];
timeend=JavaNew["org.sdas.core.time.Time", 18, 0,0];
tstart=JavaNew["org.sdas.core.time.TimeStamp", datestart, timestart];
tend=JavaNew["org.sdas.core.time.TimeStamp", dateend, timeend];

eventsFound=client@searchEventsByEventTimeWindow[tstart,tend];

Do[Print[eventsFound[[i]]@toString[]], {i, Length[eventsFound]}];
```

Search parameters

```
parametersFound = client@searchParametersByName["DENS"];

parametersFound = client@searchParametersByName["DENS", "pt"];

parametersFound = client@searchParametersByUniqueID["DENS"];

parametersFound = client@searchParametersByDescription["current"];

parametersFound = client@searchParametersByDescription["corrente", "pt"];

Do[Print[parametersFound[[i]]@toString[]], {i, Length[parametersFound]}];
```

Search data

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

```
dataFound=client@searchDataByEvent["0x0000",12050];
Do[Print[dataFound[[i]]], {i, Length[dataFound]}];
```

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

```
dataFound = client@searchDataByEvent["0x0000", 17898];
```

Data for several parameters in the same event

```
dataStruct=client@getMultipleData[{"POST.PROCESSED.DENSITY",
"POST.PROCESSED.IPLASMA"}, "0x0000", 17898]
```

Data for several parameters in different events

```
dataStruct=client@getMultipleData[{"POST.PROCESSED.DENSITY",
"POST.PROCESSED.IPLASMA"}, {"0x0000","0x0000"}, {17898, 17899}]
```

Data for the same parameter in different events

```
dataStruct=client@getMultipleData["POST.PROCESSED.DENSITY",
{"0x0000","0x0000"}, {17898, 17899}]
```

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

This is an example that shows how to extract the data from the structure:

```
dataStruct=client@getMultipleData[{"POST.PROCESSED.DENSITY",
"POST.PROCESSED.IPLASMA"},"0x0000", 17898];
cosine = dataStruct[[1]]
cosine = cosine[[1]]@getData[];
isine = dataStruct[[2]];
isine = isine[[1]]@getData[];
```

The time stamps for the starting time, end time and event time can be obtained like this:

```
dataStruct=dataStruct[[1]];
tstart=dataStruct[[1]]@getTStart[];
tend=dataStruct[[1]]@getTEnd[];
events=dataStruct[[1]]@getEvents[];
tevent=events[[1]]@getTimeStamp[];
```

To convert a time stamp to microseconds:

```
tstart@getTimeInMicros[]
```