# Troubleshooting Query Database Problems

## 1. Check that the IDL JVM is running.

```
IDL> str1 = OBJ_NEW('IDLjavaObject$java.lang.String', 'java.lang.String', 'Hello World')
IDL> help,str1
STR1            OBJREF    = <ObjHeapVar1(IDLJAVAOBJECT$JAVA.LANG.STRING)>
```

Errors at this step indicate that the JVM is missing or that a mismatch exists between the bit size, 32-bit IDL requires 32-bit Java and 64-bit IDL requires 64-bit Java. The IDL bit size is indicated when IDL starts up, for example:
<br>64-bit: IDL Version 8.1 (linux x86_64 <u>m64</u>). (c) 2011, ITT Visual Information Solutions
<br>32-bit: IDL Version 8.1 (linux x86 <u>m32</u>). (c) 2011, ITT Visual Information Solutions

If Java is not installed, have it installed. If the bit size is mismatched, replace the JVM with the correct bit size or use set the environmental variable IDLJAVAB_LIB_LOCATION to point to the correct JVM.

## 2. Check the CLASSPATH that IDL uses for Java.

```
IDL> print,getenv('CLASSPATH')
/Users/home/IDLWorkspace/IdlDbInterface/idldb.jar:.:
/Applications/exelis/idl82/resource/bridges/export/java/javaidlb.jar
```

If the CLASSPATH does not contain the idldb.jar file, then the variable is not being set properly. Recheck the installation steps.

## 3. Check the IDL path to ensure that the necessary IDL (".pro") files are present

```
IDL> print,\!path
```

If the necessary Query Database IDL (".pro") files are not in the path, recheck the installation step for adding the path directory or add the path manually.

## 4. Check the JVM properties, including bit size

```
IDL> pjava_printprops
% Compiled module: PJAVA_PRINTPROPS.
java.version: 1.6.0_33
java.vendor: Apple Inc.
java.class.path:
/Applications/exelis/idl82/idlde/idlde.darwin.x86_64.app/Contents/MacOS/..
/../../plugins/org.eclipse.equinox.launcher_1.0.201.R35x_v20090715.jar
java home: /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home
java.vm.name: Java HotSpot(TM) 64-Bit Server VM
java.vm.version: 20.8-b03-424
java.vm.vendor: Apple Inc.
java.net.preferIPv4Stack: true
user home: /Users/home
user dir: /Applications/exelis/idl82/idlde/idlde.darwin.x86_64.app/Contents/MacOS
Free Memory:                35493264
Max Memory:                129957888
Total Memory:               85000192
```

## 5. Check the socket connection from IDL

```
IDL> socket = OBJ_NEW('IDLjavaObject$lasp_idltester.SocketConnectionTest',
'lasp_idltester.SocketConnectionTest')
IDL> socket->socketTest,'sorce-db', 4100
% Successful connection: sorce-db 4100
```

A failure indicates the socket is not reachable (similar to a ping failure).

## 6. Testing the Java side

***Note the following feature is not available for either Query Database 2.0 or 2.1, but may be added in later releases.***

Outside of IDL (i.e., Unix command line) with the idldb.jar defined in the operating system CLASSPATH variable:

```
java lasp_idltester.IdlDbTester <connect_file> <cmd_file>
```

where <connect_file> is a text file containing the following information as follows:

- user = <login_name>
- password = <password>
- url = <database url>
- driver = <jdbc driver> (optional, needed only if neither Sybase nor Oracle)
- SQLINITSTRING = <parameters> (optional connection properties>
- <cmd_file> a text file of sql to be executed, one per line

```
For example, with
<connect_file> = sorce.connect
user     = testuser
password = <password>
url = jdbc:sybase:Tds:sorce-db:4100/SORCE
driver = com.sybase.jdbc3.jdbc.SybDriver
SQLINITSTRING = set quoted_identifier off
<cmd_file> = cmd.sql
select getdate()
then
java lasp_idltester.IdlDbTester sorce.connect cmd.sql
returns the following:
working directory = /Users/home/misc
home directory = /Users/home
The number of args is 2
The lasp_dbexchange program is at version 2
with a date of 2011-09-07
args[0] = lasp.connect
args[1] = cmd.sql
command: select getdate()
connection properties:
server = null
database = null
instance = null
url = jdbc:sybase:Tds:sorce-db:4100/SORCE
driver = com.sybase.jdbc3.jdbc.SybDriver
-- retrieved 1 row.
columns:
1       \[B array
rows:
2012-07-26 15:33:42.253
```

## 7. Increase the size of the JVM

The JVM size might need to be increased as the default size is small. A size too small causes memory and connection errors. Increasing the JVM size uses the same files that were required to set the java.net.preferIPv4Stack variable in the fifth step of installing. Here are the details. The .idljavabrc file is for command line IDL and the idlde.ini is for the IDL workbench.

It is recommended that the .idljavabrc be copied to prevent loss when the IDL version is updated.

1. Copy the .idlejavabrc file to the desired location.
2. Add the following lines to the copied file after the "JVM start options" section. (Note that Option2 below might be Option3, Option4, etc. depending on the number of already existing options) In the suggestion below, Option2 sets the minimum size; Option 3 the maximum.

```
JVM Option2 = \-Xms40M
JVM Option3 = \-Xmx512M
```

3. In the idlstartup.pro file add the following line where <path> is the path to the copied file:

**Linux / Unix / Mac**

```
SETENV, 'IDLJAVAB_CONFIG=<path>/.idljavabrc'
```

**Windows**

```
SETENV, 'IDLJAVAB_CONFIG=<path>\.idljavabrc'
```

A copy of the idlde.ini file cannot be used; therefore, after modification, a backup copy of idlde.ini should be made to preserve values whenever the IDL version is updated. Also, modifying the idlde.ini file might require admin access; if so, contact IT to modify the file or have them grant you access to the file.

Add the following lines to the idlde.ini file after "-vmargs". The first line sets the minimum size; the second the maximum.

```
\-Xms40M
\-Xmx512M
```

# 8. Connection Problems between Linux and Oracle

***Note that Query Database from version 2.0 AUTOMATICALLY fixes this problem; and that it WILL NOT be present with version 2.0 or higher unless the user has made a custom alteration to the settings***

Intermittent connection problems might occur when trying to connect to a database from a Linux machine with the Oracle JDBC driver. On Linux systems, the Oracle thin client uses the /dev/random file for security purposes (i.e., password encryption). Unfortunately, this file relies on user activity to generate outputs. If there is not enough activity, the output will stall. To avoid this, the /dev/urandom file should be used although it's less secure. The following needs to be added when Java is started:
-Djava.security.egd=file:///dev/urandom.

Here is the process. The .idljavabrc file is for command line IDL and the idlde.ini is for the IDL workbench.

It is recommended that the .idljavabrc be copied to prevent loss when the IDL version is updated.

1. Copy the .idljavabrc file to the desired location
2. Add the following line to the copied after the "JVM start options" sections (Note that Option1 below might be Option2, Option3, etc. depending on the number of already existing options)

```
JVM Option1 = \-Djava.security.egd=[file:///dev/urandom]
```

3. In the idlstartup.pro file, add the following line where <path> is the path to the copied file:

**Linux / Unix / Mac**

```
SETENV, 'IDLJAVAB_CONFIG=<path>/.idljavabrc'
```

**Windows**

```
SETENV, 'IDLJAVAB_CONFIG=<path>\.idljavabrc'
```

A copy of the idlde.ini file cannot be used; therefore, after modification, a backup copy of idlde.ini should be made to preserve values whenever the IDL version is updated. Also, modifying the idlde.ini file might require admin access; if so, contact IT to modify the file or have them grant you access to the file.

Add the following lines to the idlde.ini file after "-vmargs". The first line sets the minimum size; the second the maximum.

```
\-Djava.security.egd=[file:///dev/urandom]
```

<!-- More information can be found at

http://asanga-pradeep.blogspot.de/2012/12/javasqlsqlrecoverableexception-io-error.htm

http://www.usn-it.de/index.php/2009/02/20/oracle-11g-jdbc-driver-hangs-blocked-by-devrandom-entropy-pool-empty/
-->