

**Progress DataDirect® OpenAccess™ SDK 7.0**  
**Debugging an IP**

June 2011

The logo consists of an orange rounded rectangle containing the text "PROGRESS" in a light orange, sans-serif font, and "DATADIRECT" in a white, bold, sans-serif font below it.

**PROGRESS**  
**DATADIRECT**

<b>DEBUGGING A C/C++ IP .....</b>	<b>3</b>
DEBUGGING ON WINDOWS .....	3
<i>Attaching to a Running Service .....</i>	<i>3</i>
<i>Launching an OpenAccess™ SDK Service from the Debugger.....</i>	<i>3</i>
<i>Known Issue When Debugging on Windows.....</i>	<i>4</i>
DEBUGGING ON UNIX.....	4
<i>Attach to an OpenAccess™ SDK Server with the Debugger .....</i>	<i>5</i>
Example Debug Session with dbx on Solaris: .....	5
Example Debug Session with dbx on AIX .....	6
Example Debug Session with GNU gdb on Linux or with HP gdb on HP-UX.....	8
<i>Starting an OpenAccess™ SDK Server from the Debugger .....</i>	<i>9</i>
Debugging steps .....	9
Example Debug Session with dbx on Solaris: .....	9
Example Debug Session with dbx on AIX .....	10
<i>Other Tips .....</i>	<i>12</i>
Viewing Debug Information Sent to stdout .....	12
STOPPING AN OPENACCESS™ SDK SERVER BEING DEBUGGED.....	13
THE OASOA –DEBUG FLAG .....	13
OPENACCESS SDK SERVICE EXIT CODES.....	14
<i>Examples of Common Startup Problems .....</i>	<i>14</i>
<b>DEBUGGING A .NET IP .....</b>	<b>15</b>
<b>DEBUGGING A JAVA IP.....</b>	<b>15</b>
OVERVIEW OF USING ECLIPSE .....	15
DETAILED PROCEDURE FOR USING ECLIPSE .....	16
<i>Loading/Compiling an IP thru Eclipse.....</i>	<i>16</i>
<i>Debugging Using Eclipse .....</i>	<i>23</i>
<b>TIP: USE WINDOWS AUTHENTICATION.....</b>	<b>25</b>

---

## Debugging a C/C++ IP

### Debugging on Windows

There are two methods to debug an IP on Windows:

- Attach to a running OpenAccess SDK service
- Launch the OpenAccess SDK service process from the debugger

The simplest method, which can be used for most of the cases, is to attach to a running OpenAccess SDK service. You must launch the OpenAccess SDK service from the debugger if you want to debug OAIP\_init. This function is called during server initialization.

For the Local Client *for* ODBC, you can attach to the process that will be loading the ODBC driver or you can run the client application directly from the debugger.

On Windows, a connection from a client to a server that is being debugged may fail. Simply retry and the next connection will succeed.

### Attaching to a Running Service

Assuming that you have stopped all services except the one you want to debug, attach to the process oaso.exe from the debugger.

1. Stop the OpenAccessSDK700\_C Service.
2. Verify that ServiceIPModule is set to the IP module being debugged.
3. Build the IP module in Debug mode.
4. Start the service from the OpenAccess SDK Management console.
5. From the debugger, attach to the process oaso.exe.
6. Set a function break at OAIP\_connectW/ OAIP\_connect, or OAIP\_execute.
7. Connect from an OpenAccess SDK client. Your break point at OAIP\_connect is activated.
8. Execute a SQL statement: Breakpoint at OAIP\_execute is activated

### Launching an OpenAccess™ SDK Service from the Debugger

1. Stop the OpenAccessSDK700\_C Service.
2. Verify that ServiceIPModule is set to the IP module being debugged.
3. Build the IP module in Debug mode.
4. Update the project settings to run the service that loads the IP module. Use the following Debugging settings:
  - Command: C:\Program Files\DataDirect\oaserver070\bin\oaso.exe
  - Command Arguments: -debug -n OpenAccessSDK700\_C -d "C:\Program
5. Start the service using **Debug / Start**. Breakpoint OAIP\_init will be activated. Notice that the IP module is loaded.
6. Connect from a client. A breakpoint OAIP\_connectW/ OAIP\_connect is activated.
7. Execute a SQL statement: A breakpoint at OAIP\_execute is activated.

## Known Issue When Debugging on Windows

As documented in article Q173260 “PRB: Synchronization Failure When Debugging” in the Microsoft Knowledge Base, there are problems on Windows with synchronization while an application is running in a debug environment. Read the whole article at:

<http://support.microsoft.com/kb/173260/>

Due to this defect, connections to the OpenAccess SDK Server may fail with the following error in the server log file:

```
Wed Nov 28 13:21:36 2007:sched.swschd.3581.Internal error, fatal server
error detected.
Wed Nov 28 13:21:36 2007:Thread
Pooler.THRDP_StartTask.2243().4.Internal error.
Wed Nov 28 13:21:36 2007:Thread
Pooler.THRDP_AssignTask2Thread.1971().4.Internal error.
Wed Nov 28 13:21:36 2007:Thread
Pooler.THRDP_Thread_Start.3189().4.Internal error.
Wed Nov 28 13:21:36 2007:State
Synchroniser.STASYNC_WaitForStateChange.511().13.Internal error.
Timeout occurred on a wait operation.
```

To work around this type of problem while debugging a C/C++/NET IP with the OpenAccess SDK Server on Windows:

- Start oasoa.exe from the debugger with the `-debug` flag. When started with this flag enabled, oasoa.exe puts a `Sleep(0)` before each `PulseEvent()` or `SetEvent()` as suggested by Microsoft in their article Q173260 “PRB: Synchronization Failure When Debugging” at <http://support.microsoft.com/kb/173260/>
- Set `ServiceInternalTimeOut` to a value between 2000 to 5000 ms. The default value for this configuration parameter is 60000 ms. By using a lower value than 60000 ms, oasoa.exe recovers more quickly from any Windows Synchronization failure.
- Use the OpenAccess SDK trace function `tm_trace()`. Avoid using `OutputDebugString()`.
- Set a function breakpoint at the IP entry point as `OAIP_init`, `OAIP_connectW`, `OAIP_execute`,... Avoid stopping execution of oasoa.exe using “Break All”.

## Debugging on UNIX

There are two methods to debug an IP on UNIX –

1. Attach to a running OpenAccess SDK service
2. Launch the OpenAccess SDK service process from the debugger

The simplest is to attach to a running OpenAccess SDK service and can be used for most cases. You must launch the OpenAccess SDK service from the debugger if you need to debug into `OAIP_init`. This function is called during server initialization.

## Attach to an OpenAccess™ SDK Server with the Debugger

This method can be used for debugging most IP code.

Debugging steps:

1. Compile and link the IP module for debugging (-g). The setenvd.sh (or .csh) files that are supplied with the SDK set the correct flags. Make sure to disable any optimization.
2. Start the server, for example, using the ServiceStart oacla command.
3. Determine pid of the server -- for example, use grep or pgrep.
4. Attach with dbx to this server.
5. Set a breakpoint in the IP code.
6. Continue execution of the server.
7. Use odbcsql, for example, to connect to the server and execute a SQL-statement to activate your breakpoint.

### Example Debug Session with dbx on Solaris:

```
# start the server
$ /home/car1/oaserver700/admin/oacla.sh -l
DataDirect OpenAccess SDK Manager Version 7.0.0.0.0184
(c) Copyright 1995-2011 Progress Software Corporation, All rights
reserved
oacla> ServiceStart OpenAccessSDK700_C
oacla> exit

# determinate pid of the server
$ ps -ef | grep oasoa
   carl  5190      1  0 17:15:27 ?          0:01
/export/home/car1/oaserver700/bin/oasoa -n OpenAccessSDK700_C -d
/export/home/car
   carl  5225  4260  0 17:16:39 pts/2    0:00 grep oasoa

# attach with dbx to this server

$ dbx - 5190
Reading oasoa
Reading ld.so.1
...
Reading oadamipmem.so
Attached to process 5190 with 4 LWPs
t@1 (l@1) stopped in _libc_poll at 0xf9e9ae00
0xf9e9ae00: _libc_poll+0x0004: ta      8

# set breakpoint in the ip-code to debug
(dbx) stop in OAIP_connectW
(2) stop in OAIP_connectW
(dbx) stop in OAIP_execute
(3) stop in OAIP_execute

#continue execution of the server
(dbx) cont

a client connects...
```

```
t@7 (l@7) stopped in OAIP_connectW at line 686 in file "mem_drv.c"
 686      char      sFunctionName[]="OAIP_connectW";
```

...

*the client executes a query*

```
@7 (l@7) stopped in OAIP_execute at line 917 in file "mem_drv.c"
 917      char      sFunctionName[]="OAIP_execute";
```

...

### Example Debug Session with dbx on AIX

```
# start the server
$ /home/carl/oaserver700/admin/oacla.sh -l

DataDirect OpenAccess SDK Manager Version 7.0.00.0184
(c) 1995-2011. Progress Software Corporation. All Rights Reserved.

oacla> ServiceStart OpenAccessSDK700_C
oacla> exit

# determinate pid of the server
$ ps -fu carl | grep oasoa
   carl 413906      1   0 14:57:30      -   0:00
/home/carl/oaserver700/bin/oasoa
-n OpenAccessSDK700_C -d /home/carl/oaserver700/cfg/oadm.ini
   carl 430186 364778   1 14:58:45 pts/1   0:00 grep oasoa

# attach with AIX dbx to this server

$ dbx -a 413906
Waiting to attach to process 413906 ...
Successfully attached to oasoa.
warning: Directory containing oasoa could not be determined.
Apply 'use' command to initialize source path.

Type 'help' for help.
reading symbolic information ...
stopped in _event_sleep at 0xd005dec4 ($t4)
0xd005dec4 (_event_sleep+0xa8) 80410014      lwz    r2,0x14(r1)

# set breakpoint in the ip-code to debug
(dbx) stop in OAIP_connectW
[1] stop in OAIP_connectW
(dbx) stop in OAIP_execute
[2] stop in OAIP_execute

#continue execution of the server
(dbx) cont

a client connects...

[1] stopped in OAIP_connectW at line 710 in file "" ($t8)
could not read "src/mem_drv.c"

(dbx) use /home/carl/oaserver700/ip/oac/memory
(dbx) list
```

```

710     char          sFunctionName []="OAIP_connectW";
711     MEM_ENV_DA    *pEnvDA = (MEM_ENV_DA *)henv;
712     MEM_CONN_DA   *pConnDA;
713     OAWCHAR       sExpectedDbName []=OAL_WTEXT("memory");
714     OAWCHAR       sExpectedUserName []=OAL_WTEXT("pooh");
715     OAWCHAR       sExpectedPassword []=OAL_WTEXT("bear");
716     OAWCHAR       wsBuf[MEM_WSBUF_LEN];
717     int           iLen;
718     int           iUserAuthentication = 1;
719     int           iRetCode;
(dbx) where
OAIP_connectW(dam_hdbc = 0x20e19eb8, henv = 0x20ad7e68, pMemTree =
0x20b19f48, s
DatasourceName = 0x20e1a54e, sUserName = 0x20e1a0ca, sPassword =
0x20e53308, sCurrentCatalog = 0x20e1a650, sIPProperties = 0x20e1ab54,
sIPCProperties = 0x2
0e1fdde, phdbc = 0x20e1b63c), line 710 in "mem_drv.c"
sqldrv_connect() at 0xd5518f1c
OADS_connect() at 0xd5439850
SWANDB_Logon() at 0x101ca0d4
SWANLOGON_Logon() at 0x101ae39c
SWANSRVC_ExecLogon() at 0x1019a9c0
SWANSRVSSP_ExecChain() at 0x10205694
SWANSRVSSPDispatcher() at 0x10209c5c
SWANCONN_DoRPC() at 0x10021f50
SWANCONN_DoRPC() at 0x10021f50
SWANSCHED_DoRPCCB() at 0x10018fc4
SESMGR_Rpc_ServerSession_Callback() at 0x10156134
SESMGR_Rpc_ServerSession_DoRpc() at 0x10153974
SESMGR_DoRPC_137_126() at 0x101692ac
SWANSCHED_DoRPC() at 0x10014b88
ThreadMain() at 0x10216a2c

#dump unicode string sUserName in hex ( wchar_t is 2 bytes on AIX 32bit)
(dbx) &sUserName[0] /8x
0x20e1a0ca:  0070 006f 006f 0068 0000 0000 0000 0000

#continue execution of the server
(dbx) cont

...
the client executes a query

[2] stopped in OAIP_execute at line 921 in file
"/home/carl/oaserver700/ip/oac/m
mory/src/mem_drv.c" ($t8)
  921     char          sFunctionName []="OAIP_execute";

```

## Example Debug Session with GNU gdb on Linux or with HP gdb on HP-UX

```
# start the server
$ /home/car1/oaserver700/admin/oacla.sh -l

DataDirect OpenAccess SDK Manager Version 7.0.00.0184
(c) 1995-2011. Progress Software Corporation. All Rights Reserved.

oacla> ServiceStart OpenAccessSDK700_C
oacla> exit

# determinate pid of the server
$ ps -fu car1 | grep oasoa
carl    28656      1  0 20:02 ?          00:00:00
/home/car1/oaserver700/bin/oasoa -n OpenAccessSDK700_C -d
/home/car1/oaserver700/cfg/oa
dm.ini
carl    28681 26786  0 20:03 pts/2    00:00:00 grep oasoa

# attach with gdb to this server
$ gdb -p 28656
GNU gdb Red Hat Linux (6.3.0.0-0.30.1rh)
Copyright 2004 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and
you
are welcome to change it and/or distribute copies of it under certain
conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for
details.
This GDB was configured as "i386-redhat-linux-gnu".
Attaching to process 28656
Reading symbols from /home/car1/oaserver700/bin/oasoa...done.
...
Reading symbols from /home/car1/oaserver700/ip/bin/liboadsdam.so...done.
Loaded symbols for /home/car1/oaserver700/ip/bin/liboadsdam.so
Reading symbols from /home/car1/oaserver700/ip/bin/oadamipmem.so...done.
Loaded symbols for /home/car1/oaserver700/ip/bin/oadamipmem.so
0x004f26cd in poll () from /lib/tls/libc.so.6
# set breakpoint in the ip-code to debug

(gdb) break OAIP_connectW
Breakpoint 1 at 0x5e3c8e: file src/mem_drv.c, line 710.
(gdb) break OAIP_execute
Breakpoint 2 at 0x5e4543: file src/mem_drv.c, line 921.
#continue execution of the server
(gdb) cont
Continuing.

a client connect...

[Thread 52853680 (LWP 29079) exited]
[New Thread 52853680 (LWP 29320)]
[Switching to Thread 52853680 (LWP 29320)]
Breakpoint 1, OAIP_connectW (dam_hdbc=0x9fde38c, henv=0x9f96974,
pMemTree=0x9fa4ad0,sDatasourceName=0x9fdee28,  sUserName=0x9fde5a0,
```

```

sPassword=0x9f992f0, sCurrentCatalog=0x9fdf02c, sIPProperties=0x9fdf634,
sIPCustomProperties=0x9fe52a4,
    phdbc=0x9fe091c) at src/mem_drv.c:710
710      char      sFunctionName[]="OAIP_connectW";

#dump unicode string sUserName in hex ( wchar_t is 4 bytes on Linux)
(gdb) x/8w sUserName
0x9fde5a0: 0x00000070    0x0000006f    0x0000006f    0x00000068
0x9fde5b0: 0x00000000    0x00000000    0x00000000    0x00000000

#continue execution of the server
(gdb) cont
Continuing.

the client execute a query

Breakpoint 2, OAIP_execute (hdbc=0x9fa4b24, hstmt=0x9fa36cc,
iStmtType=8, hSearchCol=0x0, piNumResRows=0x32673cc)
at src/mem_drv.c:921
921      char      sFunctionName[]="OAIP_execute";
(gdb)

```

### Starting an OpenAccess™ SDK Server from the Debugger

This method of debugging has to be used when the OAIP\_init() IP function that is called during the OpenAccess SDK server startup must be debugged.

#### Before debugging

- Compile and link the IP module in debug (-g flag). The setenvd.sh (or .csh) scripts that are supplied with the SDK set the correct flags. Make sure to disable optimization (-O flag).
- Make sure that the ServiceIPModule configuration attribute contains your IP module.
- Check the server is not running

#### Debugging steps

1. Set the library path environment variable to *install\_dir/bin*. The name of the environment variable is LD\_LIBRARY\_PATH on Solaris and Linux, LIBPATH on AIX, SHLIB\_PATH on HP-UX.
2. Export the environment variable OASDK\_DEBUG\_NOFORK.
3. Export OASDK\_DEBUG\_NOFORK=1.
4. Start the debugger with the OpenAccess SDK server image *install\_dir/bin/oaso*.
5. Set a breakpoint in OAIP\_init().
6. Run the OpenAccess SDK service with the following options  
-debug -n <ServiceName> -d <configfile>

#### Example Debug Session with dbx on Solaris:

```

# export the needed environment variables
$ export LD_LIBRARY_PATH=/home/car1/oaserver700/bin
$ export OASDK_DEBUG_NOFORK=1

```

```

# start dbx with OpenAccess SDK server image
$ dbx /home/carl/oaserver700/bin/oasoa
Reading oasoa
Reading ld.so.1
....

# preload your IP module you want to debug
(dbx) loadobject -load /home/carl/oaserver700/ip/bin/oadamipmem.so
Reading oadamipmem.so
Loaded loadobject: /home/carl/oaserver700/ip/bin/oadamipmem.so

# set breakpoint in OAIP_init()
(dbx) stop in OAIP_init
(2) stop in OAIP_init

# start OpenAccess SDK service

(dbx) run -dbx -n OpenAccessSDK700_C -d
/home/carl/oaserver700/cfg/oadm.ini
Running: oasoa -dbx -n OpenAccessSDK700_C -d
/home/carl/oaserver700/cfg/oadm.ini
(process id 4726)
Reading en_US.ISO8859-1.so.2
Reading liboadsdam.so
Reading libw.so.1
t@1 (l@1) stopped in OAIP_init at line 246 in file "mem_drv.c"
    246      char          sUnicode[]="0";
(dbx) next
...

```

### Example Debug Session with dbx on AIX

```

# export the needed env variables
$ export LD_LIBRARY_PATH=/home/carl/oaserver700/bin
$ export OASDK_DEBUG_NOFORK=1

#start dbx with openAccess SDK service image
# using -I to passing the location of the IP-sources to debug.
$dbx -I /home/carl/oaserver700/ip/oac/memory
/home/carl/oaserver700/bin/oasoa

# set breakpoint in dlopen()
(dbx) stop in dlopen
[1] stop in dlopen

# start OpenAccess SDK server and stop whne dlopen() is called to load
the IP module
(dbx) run -debug -n OpenAccessSDK700_C -d
/home/carl/oaserver700/cfg/oadm.ini
[1] stopped in dlopen at 0xd02ac4c0 ($t1)
0xd02ac4c0 (dlopen)    be1ffdc      stmw    r23,-36(r1)
(dbx) where
dlopen(0x2ff21be4, 0x4) at 0xd02ac4c0
SHLLoad() at 0x1003ddf0
SWANDBSRVC_Init() at 0x101cba34
SWANSRVC_Init() at 0x101a4554

```

```

RealMain() at 0x1000134c
main() at 0x10000518
# load of liboadsdam.so, continue
(dbx) cont
[1] stopped in dlopen at 0xd02ac4c0 ($t1)
0xd02ac4c0 (dlopen)    beelffdc      stmw    r23,-36(r1)
(dbx) where
dlopen(0x2ff21480, 0x4) at 0xd02ac4c0
LoadLibrary() at 0xd5441af0
ipRegisterInterface() at 0xd558799c
dam_init_ip() at 0xd558a518
sqldrv_init() at 0xd551b89c
OADS_init() at 0xd543b034
SWANDBSRVC_Init() at 0x101cc54c
SWANSRVC_Init() at 0x101a4554
RealMain() at 0x1000134c
main() at 0x10000518
(dbx) 0x2ff21480/s
0x2ff21480: "/home/carl/oaserver700/ip/bin/oadamipmem.so"
# loading the IP-module
# set breakpoint in dlsym() and OpenAccess should call dlsym() for the
symbol "OAIP_init"
(dbx) stop in dlsym
[3] stop in dlsym
(dbx) status
[1] stop in dlopen
[3] stop in dlsym
(dbx) cont
[3] stopped in dlsym at 0xd02ac2d4 ($t1)
0xd02ac2d4 (dlsym)    93elfffc      stw    r31,-4(r1)
(dbx) where
dlsym(0x3, 0x20b81f9c) at 0xd02ac2d4
GetProcAddress() at 0xd5441a80
ipRegisterInterface() at 0xd5587abc
dam_init_ip() at 0xd558a518
sqldrv_init() at 0xd551b89c
OADS_init() at 0xd543b034
SWANDBSRVC_Init() at 0x101cc54c
SWANSRVC_Init() at 0x101a4554
RealMain() at 0x1000134c
main() at 0x10000518
(dbx) 0x20b81f9c /s
0x20b81f9c: "OAIP_init"
# delete breakpoint in dlsym() and dlopen()
# set breakpoint in OAIP_init()
(dbx) status
[1] stop in dlopen
[3] stop in dlsym
(dbx) delete 1
(dbx) delete 3
(dbx) stop in OAIP_init
[4] stop in OAIP_init
(dbx) cont
[4] stopped in OAIP_init at line 247 in file
"/home/carl/oaserver700/ip/oac/memory/src/mem_drv.c" ($t1)
  247      char          sUnicode[]="0";
(dbx) next

```

## Other Tips

### Viewing Debug Information Sent to stdout

If the service is started from the console, the debug information should be written to the tty. The service uses stdout of its parent process.

Where your printf-messages are written when the service is started in a “normal” way depends on:

- When your services are started up through the oaagent from a remote admin client, you see them in *install\_dir/logging/startOAAgent<pid>.log*
- When your services are started up using the start-scripts in the *install\_dir/admin*, you see them in *install\_dir/logging/startOAService<pid>.log*
- When your services are started from the *install\_dir /admin/oacla*, you see them on the tty.

To run the service in foreground, turn off the fork using “export OASDK\_DEBUG\_NOFORK=1”.

You can use an analogous procedure as described in “Starting an OpenAccess™ SDK Server from the Debuggers” without the debugger as follows:

```
# export the needed env variables
$ export LD_LIBRARY_PATH=/home/carl/oaserver700/bin
$ export OASDK_DEBUG_NOFORK=1

# start there OpenAccess SDK server image with the -debug flag,
servicename, and
# Configfile
$ /home/carl/oaserver700/bin/oasoa -debug -n OpenAccessSDK700_C -d
/home/carl/oaserver700/cfg/oadm.ini
```

Avoid the use of printf() to write trace messages from an IP. Instead, use the OpenAccess SDK trace function tm\_trace().

## Stopping an OpenAccess™ SDK Server Being Debugged

Before stopping the server, always disable all breakpoints. You can then use the OpenAccess SDK Manager (Management Console or Command Line) to stop the server:

1. From the OpenAccess SDK Management Console:
  - Select the server to stop, and select Refresh .The server should become active.
  - Select the server to stop, and select Stop.
2. From the OpenAccess SDK Command Line (oaso.exe or oaso.sh), use the oacla command ServiceStop.

### For example:

```
DataDirect OpenAccess SDK Manager Version .7.0.0.0184
- Copyright 1995-2011 Progress Software Corporation, All rights reserved
oacla> ActivateLocalConfig
oacla> ServiceList
-----
Name                               Host           Status         Description
-----
OpenAccessSDK700_Agent             belg-carl7     active         [OpenAccess SDK 7.0]
Agent service
OpenAccessSDK700_C                 belg-carl7     active         [OpenAccess SDK 7.0]
Service for C/C++
OpenAccessSDK700_Java              belg-carl7     inactive      [OpenAccess SDK 7.0]
Service for Java
OpenAccessSDK700_NET               belg-carl7     inactive      [OpenAccess SDK 7.0]
Service for .NET
OpenAccessSDK700_C_SQL             belg-carl7     inactive      [OpenAccess SDK 7.0]
Service for C/C++
OpenAccessSDK700_Java_SQL          belg-carl7     inactive      [OpenAccess SDK 7.0]
Service for Java
oacla> ServiceStop OpenAccessSDK700_C
oacla>
```

### The oaso -debug flag

In a typical service startup, either bin/oaso or bin\oaso.exe is started up by bin/oastrtr or bin\oastrtr.exe. The oastrtr executable exports the environment variables set in ServiceEnvironmentVariable before starting bin/oaso or bin\oaso.exe.

When bin/oaso or bin\oaso.exe is started directly from the command line or from the debugger, the service runs with just the environment variables set within the user's environment. However, the use of the -debug flag forces oaso to read the ServiceEnvironmentVariable service attributes and set them for the process.

On Windows, the -debug flag also enables the PRB Q173260 workaround as explained in <http://support.microsoft.com/kb/173260/>

## OpenAccess SDK Service Exit Codes

The following table lists the exit codes displayed when a service (oasoa) is started from the debugger or from the command line.

### Code Reason

10	oasoa is started without the required command arguments
11	The <code>-servicename</code> or <code>-n</code> option is missing the servicename argument
12	The <code>-connectmodel</code> or <code>-m</code> option is missing an argument
13	The <code>-sessionid</code> option is missing an argument
14	The <code>-connectinfo</code> option is missing an argument
15	The <code>-datamodel</code> or <code>-d</code> option is missing an argument
16	The <code>-msgfile</code> or <code>-g</code> option missing an argument
17	Missing the required servicename
18	Invalid value for connection model passed
	...
21	Cannot open the message file
22	Cannot open the configuration file
23	Configuration file has incorrect version
24	Servicename name was not found in configuration file
25	ServiceMessageFile attribute was not set in configuration file
	...
91	Internal error, memory allocation failed.
95	Internal error, fatal server error detected.

### Examples of Common Startup Problems

```
$ dbx /home/carl/oaserver700/bin/oasoa
Type 'help' for help.
reading symbolic information ...warning: no source compiled with -g

# without argument
(dbx) run
execution completed (exit code 10)

# without required servicename argument
(dbx) run -debug
execution completed (exit code 17)

# without required configuration file argument
(dbx) run -debug -n OpenAccessSDK700_C
execution completed (exit code 22)

# with incorrect servicename
(dbx) run -debug -n noneExistingService -d
/home/carl/oaserver700/cfg/oadm.ini
execution completed (exit code 24)

# with incorrect configuration file
(dbx) run -debug -n OpenAccessSDK700_C -d /WrongDirectory/cfg/oadm.ini
execution completed (exit code 22)
```

---

## Debugging a .NET IP

You must use Visual Studio 2008 or higher to debug a .NET IP.

1. Build the IP module in Debug mode. Set break point in ipConnect.
2. Start the OpenAccessSDK700\_NET Service from OpenAccess SDK Manager.
3. Select **Debug / processes**. Select **oasoa.exe** and click **Attach**.
4. From an OpenAccess SDK client, connect to your data source.  
The process should break in ipConnect.

You must restart the OpenAccess SDK service OpenAccessSDK700\_NET after making any code changes.

---

## Debugging a Java IP

This section provides a summary of using Eclipse to compile and debug a Java IP and a detailed procedure for creating an IP project in Eclipse and then debugging it.

### Overview of Using Eclipse

This section summarizes the procedure for debugging using Eclipse. Refer to the *Detailed Procedure for Using Eclipse* section for a step by step procedure.

1. Setup project in Eclipse to build IP.
  - a. Create SDK700 project. We will refer to the project location as *java\_build\_sdk700*.
  - b. Select the Import option to import the IP files and oasql.jar.
    - I. Select the directory as *install\_dir\ip*  
(for example: C:\Program Files\DataDirect\oaserver700\ip).
    - II. Expand the directory tree and select the oajava and oajava\ip folders. This selects the oajava\oasql.jar and oajava\ip\\*.java files. This will create the oajava folder under {*java\_build\_sdk700*}
    - III. Select Project properties and modify settings in Java Build Path. In the Libraries tab, Select **Add external JARs** and select *java\_build\_sdk700\oajava\oasql.jar*
    - IV. Complete the build.
2. Configure a Service to run JVM to allow remote debugging.
  - a. Open the OpenAccess SDK Management Console and select **IP Parameters** under the OpenAccessSDK700\_Java service.
  - b. Create an attribute ServiceJVMOptions with the following values:  
-Xrunjdwp:transport=dt\_socket,address=9003,server=y,suspend=n -Xdebug
3. Open the OpenAccess SDK Management Console and select **IP Parameters** under the OpenAccessSDK700\_Java service. Select the attribute “ServiceJVMClassPath” (if it doesn’t exist, create one) and add the current workspace directory (*java\_build\_sdk700\SDK700*) along with *install\_dir\ip\oajava\oasql.jar* and any additional elements required by your IP. Remove the *install\_dir\ip* entry.

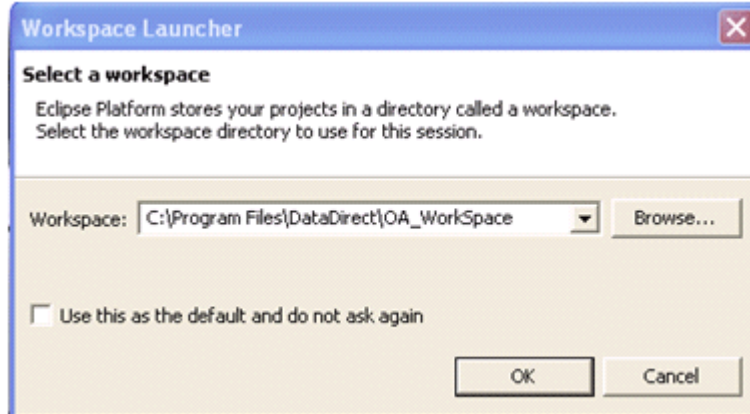
4. Save the settings and Restart/Start OpenAccessSDK700\_Java Service.
5. Connect from an OpenAccess SDK client and verify that the data source is working correctly.
6. Configure Eclipse to Debug.
  - Select Debug / Debug As... Select Remote Java Application.
  - Create new Debug profile called SDK700.
  - Set Host:localhost and Port:9003. (This Port number comes from the address value in step 2b.
7. Set a breakpoint in ipExecute or anywhere in the code where you need to debug.
8. Connect from the Client and Run query. Eclipse should allow you to debug the IP. After recompiling the code, you must restart the service in order to load the newly compiled class file.

## Detailed Procedure for Using Eclipse

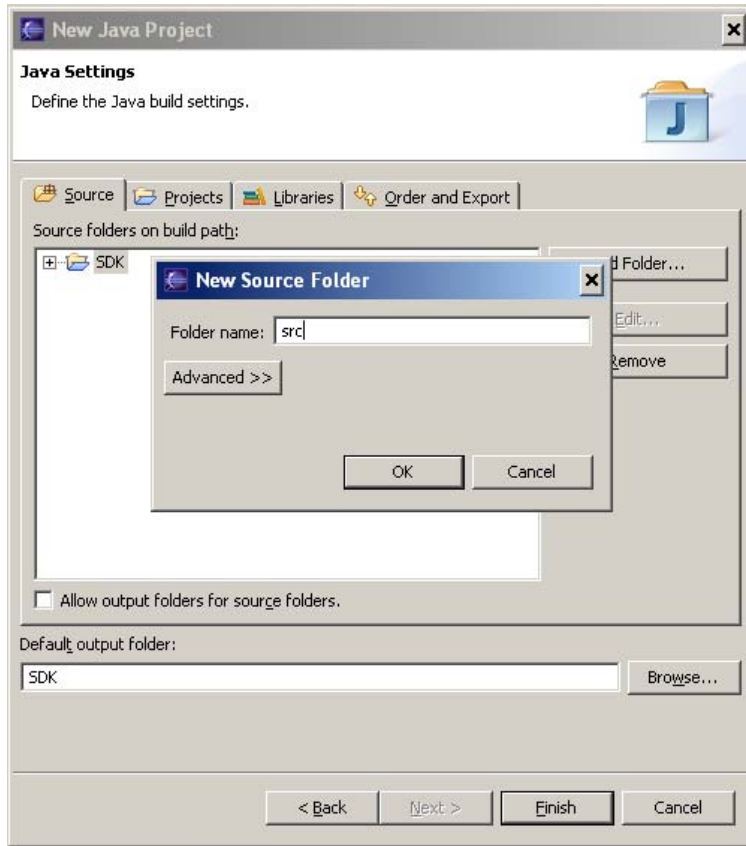
### Loading/Compiling an IP thru Eclipse

This section provides a procedure for working with Eclipse 3.0.

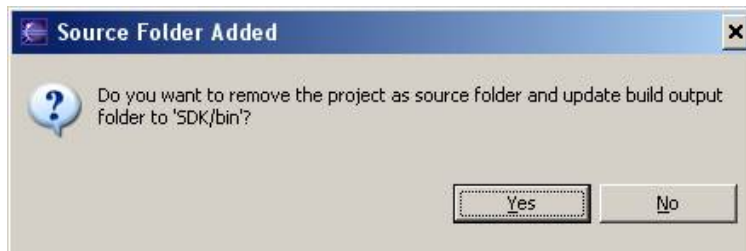
1. Launch Eclipse.
2. Create / Select Workspace.  
Create a new project by selecting **File / New / Project** and choosing **Java Project**. Name the project SDK. Use the setting as shown in the following figure and then click **Next**.



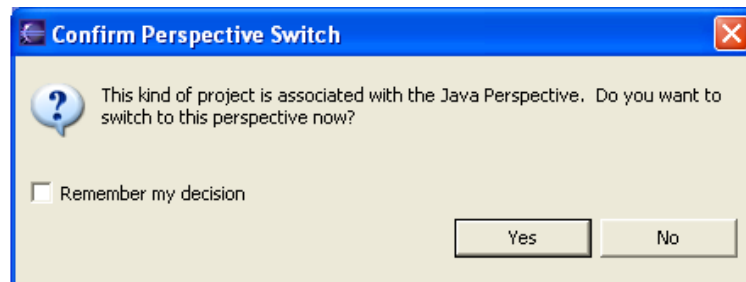
3. Select **SDK** and then select **Add Folder**. Type `src` under Folder Name and then click **OK**.



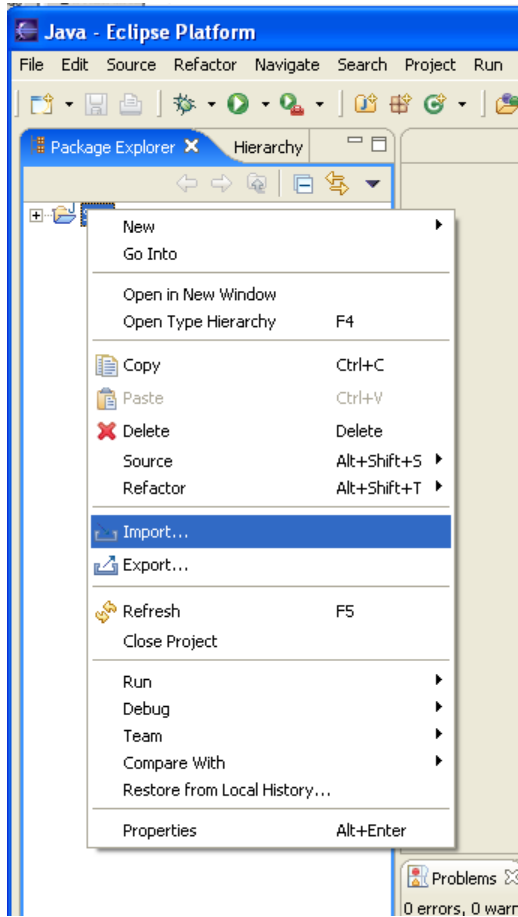
4. The Source Folder Added dialog box appears. Select **No**.



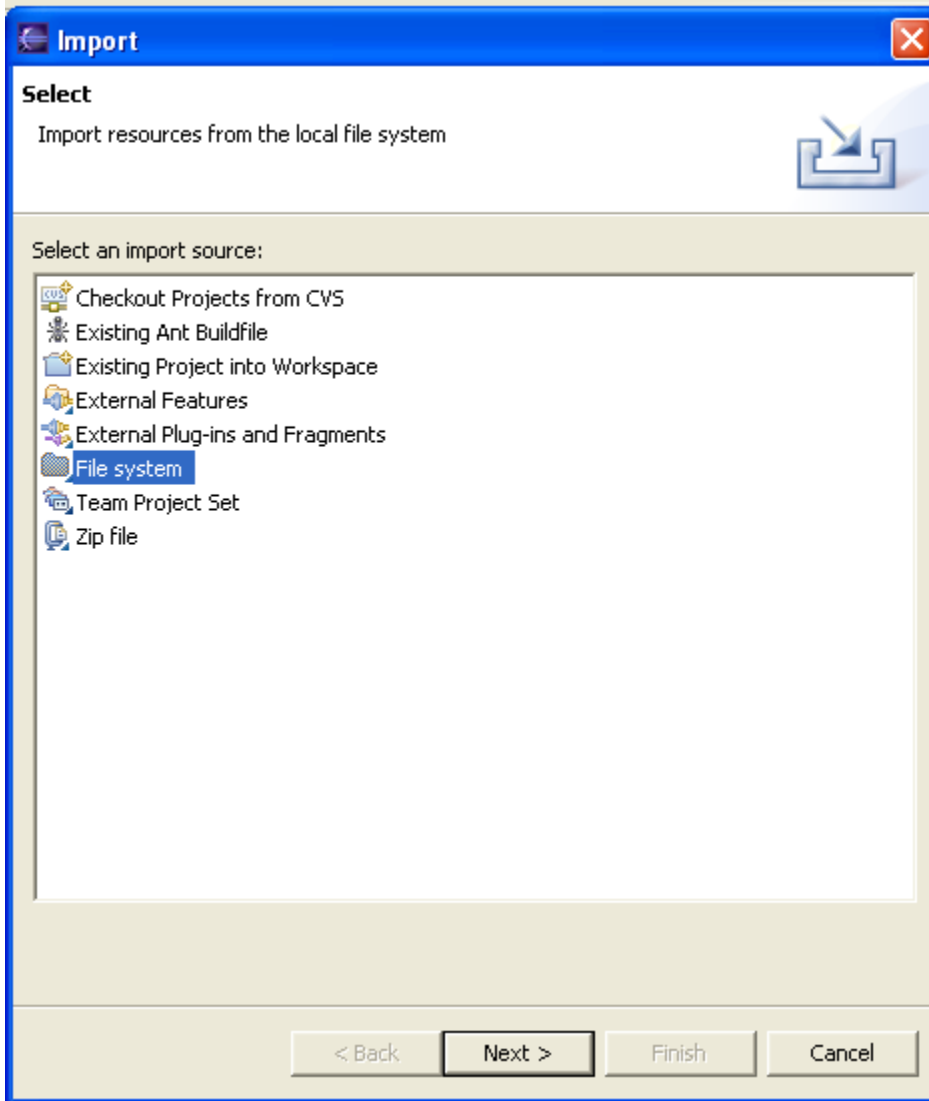
5. Click **Finish**. A confirmation window appears. Click **Yes**.



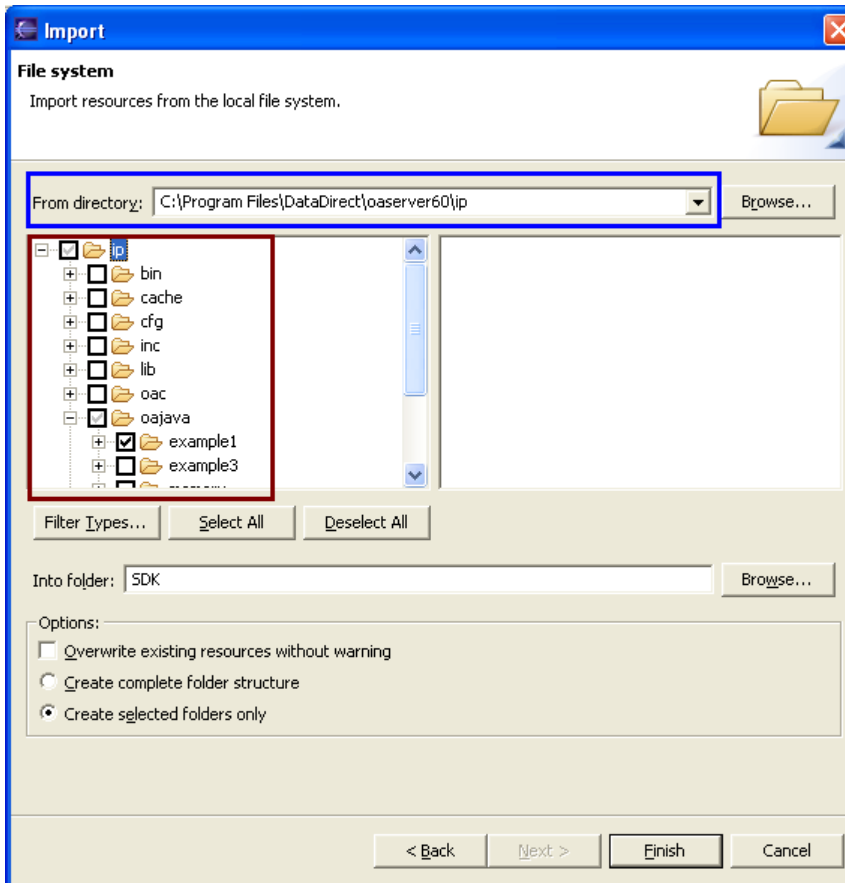
6. Go to the Eclipse window. Under Package Explorer Tab, right-click **SDK**. Select **Import**.



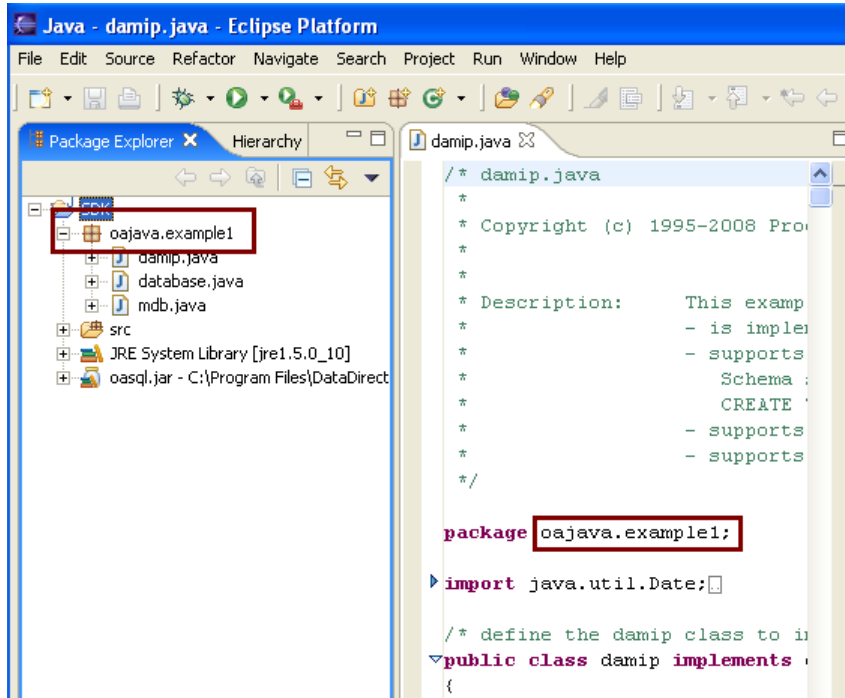
7. Select **File system** and click **Next**.



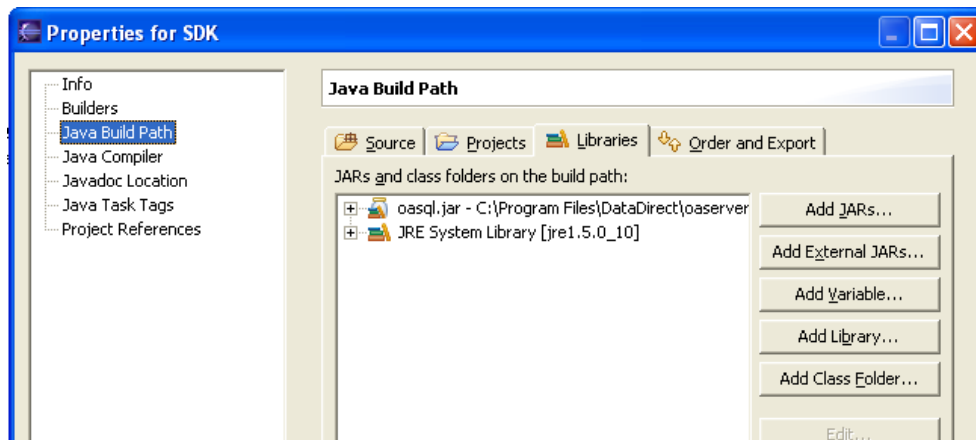
8. Under “From Directory”, browse to C:\Program Files\DataDirect\oaserver700\ip and select the example to run. Then, click **Finish**.



- Make sure that the name under SDK tree matches with the package name in the java file, for example, “oajava.example1”.



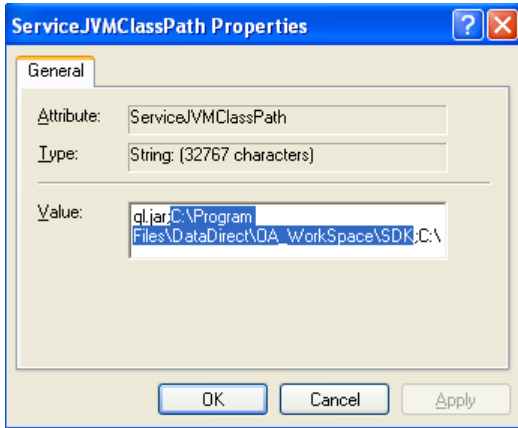
Select **Project- / Properties** and select **Java Build Path**. Click the **Libraries** tab and click **Add External JARs**. Choose *install\_dir\oaserver\ip\oajava\oasql.jar*.



The preceding steps create an Eclipse project called SDK. Now you need to configure the OpenAccess SDK Service to use the IP classes that will be compiled using this Eclipse project.

- Open the OpenAccess SDK Management Console and select **IP Parameters** under OpenAccessSDK700\_Java service. Select the ServiceJVMClassPath attribute and add the current workspace folder (C:\Program Files\DataDirect\OA\_WorkSpace\SDK ) as the first element. Remove the *install\_dir\ip* entry to ensure that the IP code is picked up from the Eclipse workspace. The ServiceJVMClassPath attribute should have

*install\_dir*\ip\oajava\oasql.jar, c:\Program File\DataDirect\OA\_WorkSpace\SDK, and any others elements required by your IP.



If the preceding attribute does not exist, then create one.

11. Select the ServiceJVMOptions attribute. If the attribute does not exist, then create it and set its value to:  
 “-Xrunjwp:transport=dt\_socket,address=9003,server=y,suspend=n -Xdebug”.

IP Parameters 7 attribute(s)			
Attribute	Type	Value	
ServiceIPPath	String	C:\Program Files\DataDirect\oaserver60\ip\bin	
ServiceIPSqlEngineModule	String	oadsdam.dll	
ServiceIPConfigFile	String	oasql.ini	
ServiceIPModule	String	oadamipjava.dll	
ServiceJVMLocation	String	C:\Program Files\Java\jdk1.5.0_10\jre\bin\server	
ServiceJVMClassPath	String	C:\Program Files\DataDirect\oaserver60\ip\oajava\oasql...	
ServiceJVMOptions	String	-Xrunjwp:transport=dt_socket address=9003,server=y...	

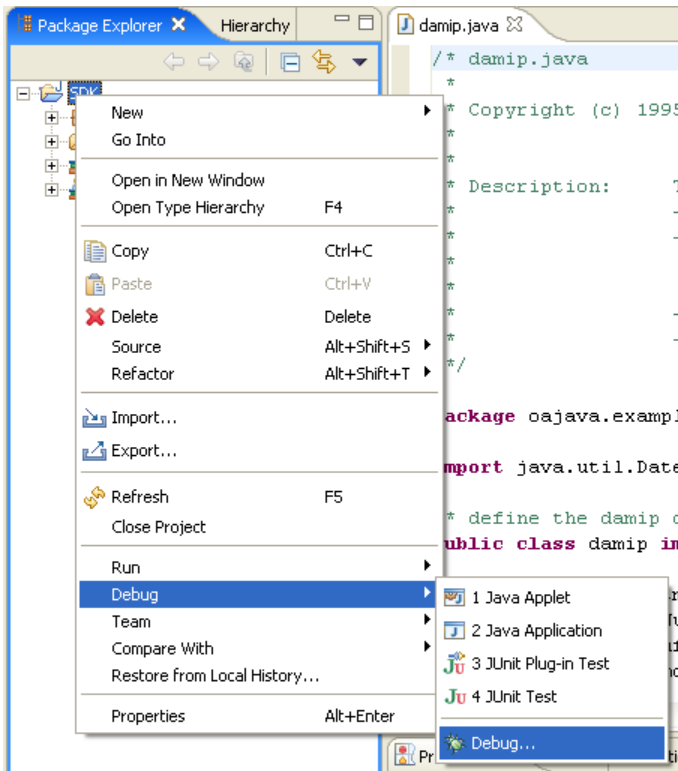
Record the port number (9003 in this example). You will need it to connect from Eclipse during the remote debug sessions.

12. Save the settings of the OpenAccess Management Console and start the OpenAccessSDK700\_Java service.
13. From an OpenAccess SDK client, connect to this service and the data source configured under it for your IP. This step is to make sure the Java environment is set up correctly.

**NOTE: Every time you create a new project or compile the project, you must restart the OpenAccessSDK700\_Java service to load the new class.**

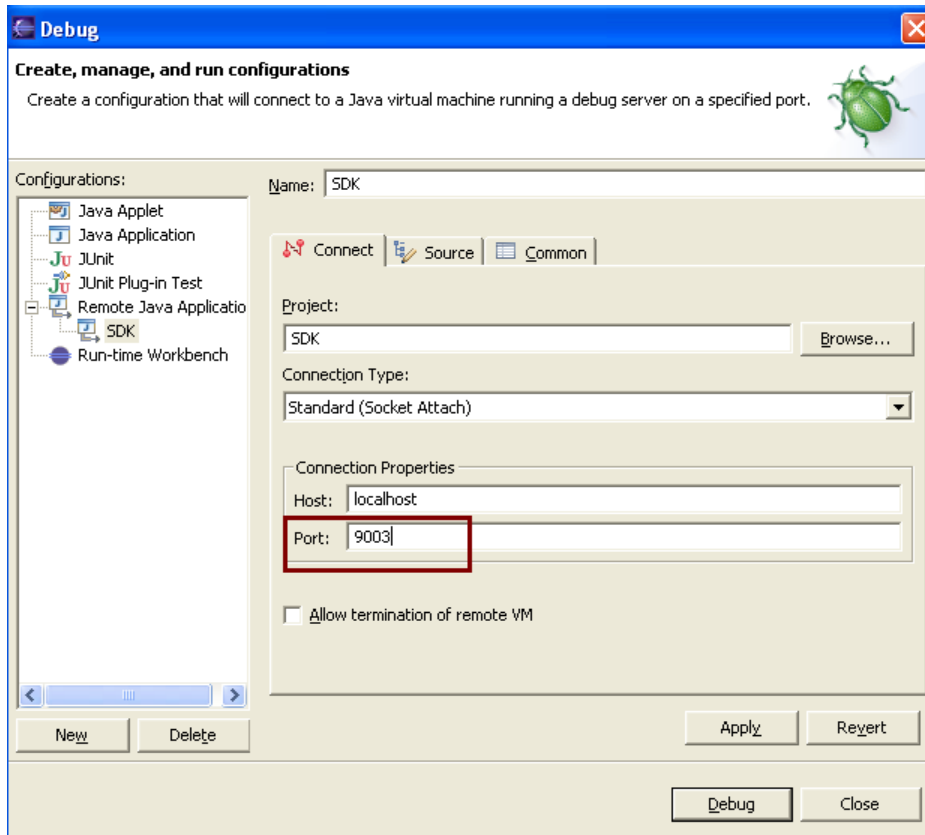
## Debugging Using Eclipse

1. Move to an Eclipse window. Right-click the SDK under Package Explorer and select **Debug-/Debug**.



2. Under Configurations, select **Remote Java Application** and click **New**.

3. Change the port number in the new window to the one recorded in step 13 and click **Debug**.



4. Set breakpoints in the IP code.
5. From an OpenAccess SDK client, connect to the data source this IP code is tied to. Eclipse should break at the breakpoints.

**NOTE: Every time you create a new project or compile the project, you must restart the OpenAccessSDK700\_Java service to load the new class.**

---

## Tip: Use Windows Authentication

Configure the OpenAccess SDK Agent on Windows for single sign-on and the administrator clients will not ask for a username and password.

To do this, add Kerberos or `integrated_nt` to `ServiceAdminAuthMethods` attribute and add the users authorized to connect to this Agent to the `ServiceAdministrator` attribute.

### Example:

```
oacla> ServiceAttributeAdd OpenAccessSDK700_Agent
ServiceAdminAuthMethods integrated_nt
```

```
oacla> ServiceInfo OpenAccessSDK700_Agent
```

```
-----
Configuration information for OpenAccessSDK700_Agent
-----
```

```
Administration Security
-----
```

```
ServiceAdminAuthMethods[0]: OSLogon(UID,PWD)
ServiceAdminAuthMethods[1]: integrated_nt
ServiceAdministrator[0]: EMEA\carl
```

Notice that the preceding configuration includes two entries for `ServiceAdminAuthMethods`. The administrator clients use `integrated_nt` for as the authentication method, because `authMethod integrated_nt` is more secure than `authMethod OSLogon(UID,PWD)`

In addition, in this configuration, only the user 'EMEA\carl' is allowed access to the Agent, because this is the only user defined in the `ServiceAdministrator` attribute.

Copyright © 2011. Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved. Actional, Apama, Artix, Business Empowerment, DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect Technologies, DataDirect XML Converters, DataDirect XQuery, DataXtend, Dynamic Routing Architecture, EdgeXtend, Empowerment Center, Fathom, Fuse Mediation Router, Fuse Message Broker, Fuse Services Framework, IntelliStream, IONA, Making Software Work Together, Mindreef, ObjectStore, OpenEdge, Orbix, PeerDirect, POSSENET, Powered by Progress, PowerTier, Progress, Progress DataXtend, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress Software Developers Network, Progress Sonic, ProVision, PS Select, Savvion, SequeLink, Shadow, SOAPscope, SOAPStation, Sonic, Sonic ESB, SonicMQ, Sonic Orchestration Server, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software. Our Technology-Experience the Connection are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Business Making Progress, Cache-Forward, DataDirect Spy, DataDirect SupportLink, Fuse, FuseSource, Future Proof, GVAC, High Performance Integration, ObjectStore Inspector, ObjectStore Performance Expert, OpenAccess, Orbacus, Pantero, POSSE, ProDataSet, Progress Control Tower, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress RPM, Progress Software Business Making Progress, PSE Pro, SectorAlliance, SeeThinkAct, Shadow z/Services, Shadow z/Direct, Shadow z/Events, Shadow z/Presentation, Shadow Studio, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Sonic Business Integration Suite, Sonic Process Manager, Sonic Collaboration Server, Sonic Continuous Availability Architecture, Sonic Database Service, Sonic Workbench, Sonic XML Server, The Brains Behind BAM, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Any other trademarks contained herein are the property of their respective owners.

#### Third Party Acknowledgments:

Progress OpenAccess v7.0 incorporates OpenLDAP (v.2.2.6) library. Such technology is subject to the following terms and conditions: The OpenLDAP Public License Version 2.2, 1 March 2000 Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The name "OpenLDAP" must not be used to endorse or promote products derived from this Software without prior written permission of the OpenLDAP Foundation. 4. Products derived from this Software may not be called "OpenLDAP" nor may "OpenLDAP" appear in their names without prior written permission of the OpenLDAP Foundation. 5. Due credit should be given to the OpenLDAP Project (<http://www.openldap.org/>). 6. The OpenLDAP Foundation may revise this license from time to time. Each revision is distinguished by a version number. You may use the Software under terms of this license revision or under the terms of any subsequent the license. THIS SOFTWARE IS PROVIDED BY THE OPENLDAP FOUNDATION AND CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENLDAP FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. OpenLDAP is a trademark of the OpenLDAP Foundation. Copyright 1999-2000, The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved. Permission to copy and distributed verbatim copies of this document is granted.

Progress OpenAccess v7.0 incorporates OpenSSL toolkit version 1.0.0. Such technology is subject to the following terms and conditions: LICENSE ISSUES ===== The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact [openssl-core@openssl.org](mailto:openssl-core@openssl.org). OpenSSL License ----- Copyright (c) 1998-2011 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (<http://www.openssl.org/>)" 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact [openssl-core@openssl.org](mailto:openssl-core@openssl.org). 5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. 6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>)" THIS SOFTWARE IS PROVIDED BY THE OPENSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OPENSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,

OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

----- This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). - Original SSLeay License -----  
 ----- Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved. This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL. This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com). Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :-). 4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)" THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

Progress OpenAccess v7.0 incorporates ICU version 4.2.1. Such technology is subject to the following terms and conditions: ICU License - ICU 1.8.1 and later COPYRIGHT AND PERMISSION NOTICE Copyright (c) 1995-2011 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder. All trademarks and registered trademarks mentioned herein are the property of their respective owners.