ObjectQ Sample Code

This is code which has been developed in concert with the ObjectQ (formerly DSAP) Programmer's Guide. It provides examples in the use of ObjectQ but often makes compromises so that the code is relatively self-contained (e.g., simple user interface, no database interaction, and canned server responses).
Users of this code should be able to quickly adapt it to their needs and then augment what is here for operational systems.

Building the executables

Here is a simple make file that can be used to build the executables:

```
SHELL = /bin/bash
CC = g++

CFLAGS = -I../../include

#DMQLIB = -L/opt/bmq/lib -ldmq
DMQLIB = -L../../lib -lfdmq
MQSLIB = -L/opt/mqm/lib64 -lmqm
#MQSLIB = -L../../lib -lfmqm
BSAFELIB = -L../../lib -lfbsafe
OQLIB = -L../../lib -lcp

LIBRARIES = $(OQLIB) $(DMQLIB) $(MQSLIB) $(BSAFELIB)

CSOURCEFILES = client.c demomgr.c instance.c request.c utility.c
SSOURCEFILES = server.c demoagt.c demomgr.c instance.c subscr.c utility.c
COBJECTFILES = $(CSOURCEFILES:.c=.o)
SOBJECTFILES = $(SSOURCEFILES:.c=.o)

DATAFILES = clientc.dat clients.dat servercp.dat serversp.dat serverss.dat
DEFFILES = Demo.adf Demo.cdf Demo.edf Demo.sdf Demo_Sales.snr Demo_Sales1.snr
tux.config

all : client server

client : $(COBJECTFILES)
        $(CC) -o $@ $(COBJECTFILES) $(CFLAGS) $(LDFLAGS) $(LIBRARIES)

server : $(SOBJECTFILES)
        $(CC) -o $@ $(SOBJECTFILES) $(CFLAGS) $(LDFLAGS) $(LIBRARIES)

clean :
        $(RM) core *.o nohup.out

clobber : clean
        $(RM) client server
```

Using this makefile will build two executables:

```
[sample]$ make -f sample.mk
g++ -I../../include   -c -o client.o client.c
g++ -I../../include   -c -o demomgr.o demomgr.c
g++ -I../../include   -c -o instance.o instance.c
g++ -I../../include   -c -o request.o request.c
g++ -I../../include   -c -o utility.o utility.c
g++ -o client client.o demomgr.o instance.o request.o utility.o -I../../include
-L../../lib/linux -lcp -L../../lib -lfdmq -L/opt/mqm/lib64 -lmqm -L../../lib
```

```
-lfbsafe
g++ -I../../include   -c -o server.o server.c
g++ -I../../include   -c -o demoagt.o demoagt.c
g++ -I../../include   -c -o subscr.o subscr.c
g++ -o server server.o demoagt.o demomgr.o instance.o subscr.o utility.o
-I../../include  -L../../lib/linux -lcp -L../../lib -lfdmq -L/opt/mqm/lib64
-lmqm -L../../lib -lfbsafe
```

Setting up the environment

For WebSphere MQ, you may need to export an environment variable to specify the name of the queue manager:

```
export MQS_QUEUE_MANAGER=APOLLO
```

For BEA messageQ, you will need three environment variables:

```
export BEADIR=/opt/bea
export DMQ_BUS_ID=1
export DMQ_GROUP_ID=100
```

The sample code will look for its MIB files (Demo.* and DSAP.*) in the local directory, but you can change this via another environment variable:

```
export DEFINITION_FILES=$(HOME)/files
```

You will also need to create queues as specified in the .snr file (Demo_Sales.snr).

Running the sample code

The client

The ObjectQ CLIENT DEMONSTRATION can be run in interactive mode by entering

```
client
```

or in non-interactive mode by entering

```
client inputFileName
```

where *inputFileName* includes the path and the file name.

When an input file is used, it has the following format:

```
traceLevel
reqNum  synchFlag  notifyFlag  timeoutValue auditFlag
. . .
reqNum  synchFlag  notifyFlag  timeoutValue auditFlag
0
```

where

- traceLevel is the tracing level (-1, 0, 1, 2, 3, 4, or 5)
- reqNum is the request number (1 thru 14)
- synchFlag is S for synchronous or A for asynchronous
- notifyFlag is P for partial or C for complete
- timeoutValue is the timeout value in seconds (>= 0)

- auditFlag is Y for audit on or N for audit off

A uniform random number generator is used.
A sample file looks like the following:

```
4
1 S P 0 Y
1 S C 0 Y
1 A P 0 Y
1 A C 0 Y
2 S P 0 Y
2 S C 0 Y
2 A P 0 Y
2 A C 0 Y
0
```

The server

The ObjectQ SERVER DEMONSTRATION can be run in interactive mode by entering

```
server
```

or in non-interactive mode by entering

```
server inputFileName
```

where *inputFileName* includes the path and the file name.

When an input file is used, it has the following format:

```
traceLevel  serverType serverMode locationTag numAgents auditFlag
successFract
minMsgPerEnv  maxMsgPerEnv
minMessages  maxMessages
```

where

- traceLevel is the tracing level (-1, 0, 1, 2, 3, 4, or 5)
- serverType is S for Simple or C for Complex (allows forwarding & hybrid)
- serverMode is P for Primary (receives requests from clients) or S for Secondary (receives requests from primary servers)
- locationTag is either ! for no location tag or a string
- numAgents is the number of agents (>= 1)
- auditFlag is Y for audit on or N for audit off
- successFract is the fraction of successful messages when get for class is used (between 0 and 1)
- minMsgPerEnv is the minimum number of messages per envelope when get for tree is used (>= 1)
- maxMsgPerEnv is the maximum number of messages per envelope when get for tree is used (>= minMsgPerEnv)
- minMessages is the minimum number of accounts when get for tree is used (>= 0)
- maxMessages is the maximum number of accounts when get for tree is used (>= minMessages)

A uniform random number generator is used.

A sample file looks like the following:

```
4 S P ! 2 Y
 .5
```

```
3 5
5 10
```

The code

The server code is somewhat "contrived" for the purpose of demonstrating the use of ObjectQ in a number of ways:

- Simple, primary server responding to a client request [filter number = 1, account status = A]
- Complex, primary server which forwards a client request to a secondary server [filter number = 1, account status = C]
- Complex, primary server which satisfies part of the client request and sends a request to a secondary server for the rest of the request; complete response is sent back to the client [filter number = 2]
- Secondary server which receives a request from a primary server and sends the response to the original client [filter number = 1, account status = C]
- Secondary server which receives a request from a primary server and sends the response back to the primary server [filter number = 2]

The server and all of the agents in a specific server process are configured one of three ways:

1. Simple & primary
2. Complex & primary
3. Simple & secondary

Simple servers cannot send to other servers (i.e., cannot participate in forwarding or hybrid requests) while complex servers can.

Primary servers receive requests from clients while secondary servers receive requests from primary servers

Assumptions and constraints include:

- All secondary servers are simple
- Minimal error checking for bad requests
- Canned responses (to canned requests)

Running the client and server

Invoke the server first in one window, and then the client in another window, as described earlier.

A transcript of two simple scenarios is shown below. If the run is successful, both log files (cpLOG*) will contain a single line indicating the version of ObjectQ that the executables were built against:

```
(0919) 08:53: Version 6.0.2.c
```

Both client and server will also generate their own output – OUTFILE.* - examples of which, corresponding to these scenarios, are shown below the transcript.

Client	Server
	[sample]$ ./server ********** * * * WELCOME TO THE DSAP SERVER DEMONSTRATION * * * ********** Enter the trace level (-1, 0, 1, 2, 3, 4, or 5): 4 Enter the server type: S or s - Simple (requests stop here) C or c - Complex (allows forwarding & hybrid requests) : s Enter the server mode: P or p - Primary (receives requests from clients) S or s - Secondary (receives requests from other servers) : p Enter the service location tag name: (or <CR> if none) : Enter the number of agents (> 0): 1 Auditing: Y or y - Yes (auditing on) N or n - No (auditing off) : y
[sample]$ ./client ********** * * * WELCOME TO THE DSAP CLIENT DEMONSTRATION * * ********** Enter the trace level (-1, 0, 1, 2, 3, 4, or 5): 4 ========== Enter the request or scenario type from the following: 0 No more 1 Get for class (Allows response to succeed or fail)	

```
 2 Get for tree (Allows one or more
envelopes)
 3 Forwarding get for tree (Servers
must be properly configured!)
 4 Hybrid get for tree (Servers must be
properly configured!)
 5 Set for class
 6 Set for tree
 7 Create for class
 8 Create for tree (Multiple create
messages in one request envelope)
 9 Delete for class
10 Delete for tree
11 Action for class (payBill)
12 Action for tree (numOrders)
13 Subscription
14 Conversation
: 1

Enter the synch flag:
S or s - Synchronous
A or a - Asynchronous
: s

Enter the notify flag:
P or p - Partial
C or c - Complete
: p

Enter the timeout value in seconds (>=
0): 30

Auditing:
Y or y - Yes (auditing on)
N or n - No (auditing off)
: y

***** BUILD REQUEST

***** SEND REQUEST
```

```
=======================================

***** RECEIVE REQUEST

Processing get request for class

Do you want this request to:
F Fail
S Succeed
: S

***** BUILD RESPONSE for class
Demo.Account

***** SEND RESPONSE
```

```
***** RECEIVE COMPLETE(D) RESPONSE

=======================================

Enter the request or scenario type from
```

```
the following:
 0 No more
 1 Get for class (Allows response to
succeed or fail)
 2 Get for tree (Allows one or more
envelopes)
 3 Forwarding get for tree (Servers
must be properly configured!)
 4 Hybrid get for tree (Servers must be
properly configured!)
 5 Set for class
 6 Set for tree
 7 Create for class
 8 Create for tree (Multiple create
messages in one request envelope)
 9 Delete for class
10 Delete for tree
11 Action for class (payBill)
12 Action for tree (numOrders)
13 Subscription
14 Conversation
: 2

Enter the synch flag:
S or s - Synchronous
A or a - Asynchronous
: s

Enter the notify flag:
P or p - Partial
C or c - Complete
: c

Enter the timeout value in seconds (>=
0): 30

Auditing:
Y or y - Yes (auditing on)
N or n - No (auditing off)
: y

***** BUILD REQUEST

***** SEND REQUEST
```

```
=======================================

***** RECEIVE REQUEST

Processing get request for tree

***** BUILD RESPONSE for tree 1

Enter the number of messages per
envelope (>=1): 3
Enter the number of accounts (>=1): 2

***** SEND RESPONSE - ENVELOPE 1
^C
[sample]$
```

```
***** RECEIVE COMPLETE(D) RESPONSE

=======================================

Enter the request or scenario type from
```

```
the following:
 0 No more
 1 Get for class (Allows response to
succeed or fail)
 2 Get for tree (Allows one or more
envelopes)
 3 Forwarding get for tree (Servers
must be properly configured!)
 4 Hybrid get for tree (Servers must be
properly configured!)
 5 Set for class
 6 Set for tree
 7 Create for class
 8 Create for tree (Multiple create
messages in one request envelope)
 9 Delete for class
10 Delete for tree
11 Action for class (payBill)
12 Action for tree (numOrders)
13 Subscription
14 Conversation
: 0
[sample]$
```

Server output

```
(09/19/16) 08:53:38: DSAP SERVER DEMONSTRATION
(09/19/16) 08:53:38: INTERACTIVE MODE
(09/19/16) 08:53:38: Definition files directory: ./
(09/19/16) 08:53:40: Trace level = 4
(09/19/16) 08:53:45: Server type = simple
(09/19/16) 08:53:48: Server mode = primary
(09/19/16) 08:53:48: Service name = Demo.Sales
(09/19/16) 08:53:48: Vendor name = 2
(09/19/16) 08:53:50: Location tag = <>
(09/19/16) 08:53:52: Number of agents = 1
(09/19/16) 08:53:54: Audit flag = 1
(09/19/16) 07:54:18: ****************************************
(09/19/16) 07:54:18: SUCCESSFUL RECEIPT OF REQUEST
(09/19/16) 07:54:18: ****************************************
(09/19/16) 07:54:18: ***** RECEIVE REQUEST
(09/19/16) 07:54:18: Instance:
(09/19/16) 07:54:18:   class name: Demo.Account
(09/19/16) 07:54:18:   instance ID: A101
(09/19/16) 07:54:18:   attribute ID list:
(09/19/16) 07:54:18:     Demo.status, CHAR
(09/19/16) 07:54:18:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:18:       Demo.streetNumber, STRING
(09/19/16) 07:54:18:       Demo.streetName, STRING
(09/19/16) 07:54:18:       Demo.city, STRING
(09/19/16) 07:54:18:       Demo.state, STRING
(09/19/16) 07:54:18:       Demo.zipCode, STRING
(09/19/16) 07:54:18:   containment tree: 0
(09/19/16) 07:54:18: ----- CLASS -----
(09/19/16) 07:54:24: ***** BUILD RESPONSE for class Demo.Account
(09/19/16) 07:54:24: Response:
(09/19/16) 07:54:24:   attribute list:
(09/19/16) 07:54:24:     Demo.status, CHAR, Value: A
(09/19/16) 07:54:24:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:24:       Demo.streetNumber, STRING, Value: 101
(09/19/16) 07:54:24:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:24:       Demo.city, STRING, Value: Anytown
```

```
(09/19/16) 07:54:24:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:24:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:24:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:24:   link ID: 1
(09/19/16) 07:54:24:   instance ID: A101
(09/19/16) 07:54:24:   class name: Demo.Account
(09/19/16) 07:54:24: ***** SEND RESPONSE
(09/19/16) 07:54:41: ****************************************
(09/19/16) 07:54:41: SUCCESSFUL RECEIPT OF REQUEST
(09/19/16) 07:54:41: ****************************************
(09/19/16) 07:54:41: ***** RECEIVE REQUEST
(09/19/16) 07:54:41: Instance:
(09/19/16) 07:54:41:   class name: Demo.Customer
(09/19/16) 07:54:41:   attribute ID list:
(09/19/16) 07:54:41:     Demo.name, STRING
(09/19/16) 07:54:41:     Demo.phoneNo, STRING
(09/19/16) 07:54:41:     Demo.accountID, STRING
(09/19/16) 07:54:41:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:41:       Demo.streetNumber, STRING
(09/19/16) 07:54:41:       Demo.streetName, STRING
(09/19/16) 07:54:41:       Demo.city, STRING
(09/19/16) 07:54:41:       Demo.state, STRING
(09/19/16) 07:54:41:       Demo.zipCode, STRING
(09/19/16) 07:54:41:   instance filter number: 1
(09/19/16) 07:54:41:     attribute list:
(09/19/16) 07:54:41:       Demo.customerID, STRING, Value: C102
(09/19/16) 07:54:41:       Demo.Account.Demo.status, CHAR, Value: A
(09/19/16) 07:54:41:   containment tree: 1
(09/19/16) 07:54:41: ----- TREE -----
(09/19/16) 07:54:41: ----- NON-HYBRID -----
(09/19/16) 07:54:41: ----- SIMPLE PRIMARY SERVER -----
(09/19/16) 07:54:41: ***** BUILD RESPONSE for tree 1
(09/19/16) 07:54:48: Number of messages per envelope= 3
(09/19/16) 07:54:48: Response for Envelope 1 and Customer:
(09/19/16) 07:54:48:   attribute list:
(09/19/16) 07:54:48:     Demo.name, STRING, Value: ACME Widgets
(09/19/16) 07:54:48:     Demo.phoneNo, STRING, Value: (800)555-1212
(09/19/16) 07:54:48:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:48:   link ID: 1
(09/19/16) 07:54:48:   instance ID: C102
(09/19/16) 07:54:48:   class name: Demo.Customer
(09/19/16) 07:54:51: Number of accounts: 2
(09/19/16) 07:54:51: Response for Envelope 1 and Account 1:
(09/19/16) 07:54:51:   attribute list:
(09/19/16) 07:54:51:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:51:       Demo.streetNumber, STRING, Value: 100
(09/19/16) 07:54:51:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:51:       Demo.city, STRING, Value: Anytown
(09/19/16) 07:54:51:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:51:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:51:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:51:   link ID: 2
(09/19/16) 07:54:51:   instance ID: A101
(09/19/16) 07:54:51:   class name: Demo.Account
(09/19/16) 07:54:51: Response for Envelope 1 and Account 2:
(09/19/16) 07:54:51:   attribute list:
(09/19/16) 07:54:51:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:51:       Demo.streetNumber, STRING, Value: 101
(09/19/16) 07:54:51:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:51:       Demo.city, STRING, Value: Anytown
(09/19/16) 07:54:51:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:51:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:51:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:51:   link ID: 3
```

```
(09/19/16) 07:54:51:   instance ID: A102
(09/19/16) 07:54:51:   class name: Demo.Account
(09/19/16) 07:54:51: ***** SEND RESPONSE - ENVELOPE 1
```

Client output

```
(09/19/16) 08:54:02: DSAP CLIENT DEMONSTRATION
(09/19/16) 08:54:02: INTERACTIVE MODE
(09/19/16) 08:54:02: Definition files directory: ./
(09/19/16) 08:54:04: Trace level = 4
(09/19/16) 08:54:04: Service name = Demo.Sales
(09/19/16) 08:54:04: Vendor name = 2
(09/19/16) 08:54:09: ****************************************
(09/19/16) 08:54:09: Get for class (Allows response to succeed or fail)
(09/19/16) 08:54:09: ****************************************
(09/19/16) 08:54:10: Synch flag = s
(09/19/16) 08:54:13: Notify flag = c
(09/19/16) 08:54:17: Timeout value = 30
(09/19/16) 08:54:18: Audit flag = 1
(09/19/16) 08:54:18: ***** BUILD REQUEST
(09/19/16) 08:54:18: Request:
(09/19/16) 08:54:18:   class name: Demo.Account
(09/19/16) 08:54:18:   attribute list:
(09/19/16) 08:54:18:     Demo.status, CHAR
(09/19/16) 08:54:18:     Demo.billingLocation, COMPLEX:
(09/19/16) 08:54:18:       Demo.streetNumber, STRING
(09/19/16) 08:54:18:       Demo.streetName, STRING
(09/19/16) 08:54:18:       Demo.city, STRING
(09/19/16) 08:54:18:       Demo.state, STRING
(09/19/16) 08:54:18:       Demo.zipCode, STRING
(09/19/16) 08:54:18:   instance ID: A101
(09/19/16) 08:54:18:   tree: 0
(09/19/16) 08:54:18: ***** SEND REQUEST
(09/19/16) 08:54:18: Demo_SalesMgr: get(instanceId) <Demo.Account>
(09/19/16) 07:54:24: Demo_SalesMgr: processGetResp called
(09/19/16) 07:54:24: Requestor: synchNotifyComplete called
(09/19/16) 07:54:24: ***** RECEIVE COMPLETE(D) RESPONSE
(09/19/16) 07:54:24: Requestor: processResp1 successful return
(09/19/16) 07:54:24: Requestor: processResp1: number of instances = 1
(09/19/16) 07:54:24: Instance:
(09/19/16) 07:54:24:   class name: Demo.Account
(09/19/16) 07:54:24:   instance ID: A101
(09/19/16) 07:54:24:   attribute list:
(09/19/16) 07:54:24:     Demo.status, CHAR, Value: A
(09/19/16) 07:54:24:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:24:       Demo.streetNumber, STRING, Value: 101
(09/19/16) 07:54:24:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:24:       Demo.city, STRING, Value: Anytown
(09/19/16) 07:54:24:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:24:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:24:   link ID: 1
(09/19/16) 07:54:24:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:24: Audit data for entry (hop) 1 of 2:
(09/19/16) 07:54:24:   Send data:
(09/19/16) 07:54:24:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:24:     Pid: 15762
(09/19/16) 07:54:24:     User ID: 5007
(09/19/16) 07:54:24:     Message size: 1318
(09/19/16) 07:54:24:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:24:     Service name: Demo.Sales
(09/19/16) 07:54:24:     Queue: queue manager: TESTQ queue:
(09/19/16) 07:54:24:   Receive data:
(09/19/16) 07:54:24:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:24:     Pid: 15758
```

```
(09/19/16) 07:54:24:     User ID: 5007
(09/19/16) 07:54:24:     Message size: 520
(09/19/16) 07:54:24:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:24:     Service name: Demo.Sales
(09/19/16) 07:54:24:     Queue: TESTQ
(09/19/16) 07:54:24: Audit data for entry (hop) 2 of 2:
(09/19/16) 07:54:24:   Send data:
(09/19/16) 07:54:24:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:24:     Pid: 15758
(09/19/16) 07:54:24:     User ID: 5007
(09/19/16) 07:54:24:     Message size: 734
(09/19/16) 07:54:24:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:24:     Service name: Demo.Sales
(09/19/16) 07:54:24:     Queue: queue manager: 57C993A120005102 queue: APOLLO
(09/19/16) 07:54:24:   Receive data:
(09/19/16) 07:54:24:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:24:     Pid: 15762
(09/19/16) 07:54:24:     User ID: 5007
(09/19/16) 07:54:24:     Message size: 916
(09/19/16) 07:54:24:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:24:     Service name: Demo.Sales
(09/19/16) 07:54:24:     Queue: 57C993A120005102
(09/19/16) 07:54:29: ****************************************
(09/19/16) 07:54:29: Get for tree (Allows one or more envelopes)
(09/19/16) 07:54:29: ****************************************
(09/19/16) 07:54:32: Synch flag = s
(09/19/16) 07:54:34: Notify flag = c
(09/19/16) 07:54:38: Timeout value = 30
(09/19/16) 07:54:41: Audit flag = 1
(09/19/16) 07:54:41: ***** BUILD REQUEST
(09/19/16) 07:54:41: Request:
(09/19/16) 07:54:41:   class name: Demo.Customer
(09/19/16) 07:54:41:   attribute list:
(09/19/16) 07:54:41:     Demo.name, STRING
(09/19/16) 07:54:41:     Demo.phoneNo, STRING
(09/19/16) 07:54:41:     Demo.accountID, STRING
(09/19/16) 07:54:41:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:41:       Demo.streetNumber, STRING
(09/19/16) 07:54:41:       Demo.streetName, STRING
(09/19/16) 07:54:41:       Demo.city, STRING
(09/19/16) 07:54:41:       Demo.state, STRING
(09/19/16) 07:54:41:       Demo.zipCode, STRING
(09/19/16) 07:54:41:   instance filter: 1
(09/19/16) 07:54:41:   instance filter attribute list:
(09/19/16) 07:54:41:     Demo.customerID, STRING, Value: C102
(09/19/16) 07:54:41:     Demo.Account.Demo.status, CHAR, Value: A
(09/19/16) 07:54:41:   tree: 1
(09/19/16) 07:54:41: ***** SEND REQUEST
(09/19/16) 07:54:41: Demo_SalesMgr: get(instance filter) <Demo.Customer>
(09/19/16) 07:54:51: Demo_SalesMgr: processGetResp called
(09/19/16) 07:54:51: Requestor: synchNotifyComplete called
(09/19/16) 07:54:51: ***** RECEIVE COMPLETE(D) RESPONSE
(09/19/16) 07:54:51: Requestor: processResp1 successful return
(09/19/16) 07:54:51: Requestor: processResp1: number of instances = 3
(09/19/16) 07:54:51: Instance:
(09/19/16) 07:54:51:   class name: Demo.Customer
(09/19/16) 07:54:51:   instance ID: C102
(09/19/16) 07:54:51:   attribute list:
(09/19/16) 07:54:51:     Demo.name, STRING, Value: ACME Widgets
(09/19/16) 07:54:51:     Demo.phoneNo, STRING, Value: (800)555-1212
(09/19/16) 07:54:51:   link ID: 1
(09/19/16) 07:54:51:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:51: Instance:
(09/19/16) 07:54:51:   class name: Demo.Account
```

```
(09/19/16) 07:54:51:   instance ID: A101
(09/19/16) 07:54:51:   attribute list:
(09/19/16) 07:54:51:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:51:       Demo.streetNumber, STRING, Value: 100
(09/19/16) 07:54:51:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:51:       Demo.city, STRING, Value: Anytown
(09/19/16) 07:54:51:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:51:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:51:   link ID: 2
(09/19/16) 07:54:51:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:51: Instance:
(09/19/16) 07:54:51:   class name: Demo.Account
(09/19/16) 07:54:51:   instance ID: A102
(09/19/16) 07:54:51:   attribute list:
(09/19/16) 07:54:51:     Demo.billingLocation, COMPLEX:
(09/19/16) 07:54:51:       Demo.streetNumber, STRING, Value: 101
(09/19/16) 07:54:51:       Demo.streetName, STRING, Value: Demo Drive
(09/19/16) 07:54:51:       Demo.city, STRING, Value: Anytown
(09/19/16) 07:54:51:       Demo.state, STRING, Value: NJ
(09/19/16) 07:54:51:       Demo.zipCode, STRING, Value: 12345
(09/19/16) 07:54:51:   link ID: 3
(09/19/16) 07:54:51:   return code: DSAP.MSUCCESS, Success
(09/19/16) 07:54:51: Audit data for entry (hop) 1 of 2:
(09/19/16) 07:54:51:   Send data:
(09/19/16) 07:54:51:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:51:     Pid: 15762
(09/19/16) 07:54:51:     User ID: 5007
(09/19/16) 07:54:51:     Message size: 1338
(09/19/16) 07:54:51:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:51:     Service name: Demo.Sales
(09/19/16) 07:54:51:     Queue: queue manager: TESTQ queue:
(09/19/16) 07:54:51:   Receive data:
(09/19/16) 07:54:51:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:51:     Pid: 15758
(09/19/16) 07:54:51:     User ID: 5007
(09/19/16) 07:54:51:     Message size: 544
(09/19/16) 07:54:51:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:51:     Service name: Demo.Sales
(09/19/16) 07:54:51:     Queue: TESTQ
(09/19/16) 07:54:51: Audit data for entry (hop) 2 of 2:
(09/19/16) 07:54:51:   Send data:
(09/19/16) 07:54:51:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:51:     Pid: 15758
(09/19/16) 07:54:51:     User ID: 5007
(09/19/16) 07:54:51:     Message size: 1094
(09/19/16) 07:54:51:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:51:     Service name: Demo.Sales
(09/19/16) 07:54:51:     Queue: queue manager: 57C993A120005102 queue: APOLLO
(09/19/16) 07:54:51:   Receive data:
(09/19/16) 07:54:51:     DateTime: 09/19/16, 07:54
(09/19/16) 07:54:51:     Pid: 15762
(09/19/16) 07:54:51:     User ID: 5007
(09/19/16) 07:54:51:     Message size: 1276
(09/19/16) 07:54:51:     Host name: apollo.idi-middleware.com
(09/19/16) 07:54:51:     Service name: Demo.Sales
(09/19/16) 07:54:51:     Queue: 57C993A120005102
```