z390 Service Oriented Architecture User Guide v1.4.03

Table of Contents

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

1. Introductions

The z390 Portable Mainframe Assembler open source project supports low level Service Oriented Architecture (SOA) type application generation and execution. The z390 macro TCPIO invoking svc x'7C' supports TCP/IP sockets messaging between client and server components. Any z390 assembler program can open up to 10 different client ports and 10 different server ports. The server ports can support up to a total of 20 concurrent connections from other client programs. z390 client and server programs can also interact with any other clients or server programs written in any language supporting compatible messaging via TCP/IP sockets. Both a COBOL and a z390 assembler version of the same client are now included. The COBOL demo was successfully run on Windows using Micro Focus COBOL calling z390 assembler services also running on Windows.

z390 includes an SOA directory with SOA generation macro library and a demo application which can be generated and executed as either a classic statically linked application or an SOA generated client server application using TCP/IP sockets.

The SOA demo application consists of a z390 mainframe assembler main program DEMOMAIN.MLC and an equivalent COBOL version DEMOMAIN.CBL which calls two subroutines DEMOSUB1.MLC and DEMOSUB2.MLC. DEMOSUB1 calculates the sum of two numbers with up to 34 digits in scientific notation using extended decimal floating point. DEMOSUB2 calculates the sum of two 32 bit integers.

To assemble, statically link, and execute the demo application run soa\demo\demostd.bat. To generate, build, and execute the demo SOA application with the client and server running on the same processor, run soa\demo\demosoa.bat. The demo SOA application client and server can also be run using two or more separate processors on a TCP/IP network. The application generator supports both assembler and COBOL clients. Multiple copies of the demo client can be concurrently run using the same demo server. The network can be a local cable or wireless network with optional connections to other TCP/IP networks over the Internet using Virtual Private Network (VPN) connections. See Appendix V for statistics showing the timings from each of these network configurations. The conclusion from timings is that the average overhead for TCP/IP messaging is about 2-3 milli-seconds after initialization. This overhead is negligible for services which also perform database I/O. The advantage of using the SOA architecture is that services are more easily shared across diverse applications and user networks and maintenance is simpler since server code does not need to be statically linked into client application code.

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

2. z390 TCP/IP Sockets Support

The TCPIO macro operations OPEN, CLOSE, SEND, and RECEIVE support messaging between client and server programs using TCP/IP sockets. Messages of any length can be exchanges between any client and server programs on a TCP/IP network. The server program must first open a socket port. Then up to 20 concurrent clients can open connections to any server port. The same program can have up to 10 server ports and 10 client ports open. The macro commands are as follows:

1. **TCPIO OPEN,PORT=port,HOST=host**

   The port can be any standard port number less than 1024 or any private port number above 1024. The port number can be numeric constant, symbolic absolute value, or can be specified in (register). The only requirement is that port numbers not conflict with other port numbers being used on the same network with the same processor hosts. If the HOST= keyword parameter is coded specifying a specific host IP address such as 162.692.1.3 or * for the current processor, then a client port will be opened with a connection to the server port on the indicated processor. If a connection to the specified server port cannot be made, then a return code of 12 will be set. If the HOST= parameter is omitted, then a server port will be opened on the current processor which can handle up to 20 concurrent connections from client ports on the network.

2. **TCPIO CLOSE,PORT=port**

   Close the specified port. Note the same processor cannot open a client and sever port with the same number so there is no need to indicate which type it is. When a server port is closed, all associated port and connection threads are also terminated. All ports are automatically closed at program termination if not closed explicitly.

3. **TCPIO SEND,PORT=port,MSG=addr,LMSG=length**

   Send the message with specified address and length to the specified port. The message starting address can be RX type label or can be specified as (register). The message length can be absolute value or (register). If the send fails, a return code of 12 will be set.

4. **TCPIO RECEIVE[,NOWAIT],PORT=port,MSG=addr,LMSG=max-length [,CONN=id]**

Receive a message from the specified port starting at address with length up to max-length. The message starting address can be RX type label or can be specified as (register). The message maximum length can be absolute value or (register). If the

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

optional second positional parameter NOWAIT is specified, a return code of 4 will be returned if no message is ready otherwise the RECEIVE will wait until at least 1 byte of the message is available. If the port is a server port then the optional keyword parameter CONN=id may be specified indicating a specific connection that was previously returned by prior TCPIO RECEIVE in register 2. If a CONN value of -1 is specified or the parameter is omitted, the next message from any connection will be returned along with the connection id in register 2. At least 1 byte will be returned on a successful RECEIVE with return code 0 along with the number of bytes returned in register 1. Up to the max-length bytes may be returned. If more than one message arrives prior to RECEIVE, it is up to the user to determine where one message ends and next message starts. For example the demo uses a 4 byte message length prefix to determine each message length. Another option for client server applications passing ASCII text could be the ending line feed character (hex x'0A'). The TCPIO service itself is not sensitive to any special characters and all byte values are allowed anywhere in messages. Note more than one RECEIVE operation may be required to retrieve an entire logical message since the TCP/IP network may not transfer the entire message in one packet on the network and a portion of the logical message may be ready whereas the next RECEIVE may have to wait for the next part of the message.

If any TCPIO operation fails for any reason on the client or server, a non-zero return code is returned. For examples of how to use this new macro service, see the SOA generated client and server message managers (soa\demo\democmgr.prn and soa\demo\demosmgr.prn). If the TRACE and CON options are specified on the execution of the demo batch command soa\demo\DEMOSOA.BAT, then the generated client and server message managers will display trace of all instructions including additional trace information on each TCPIO svc plus memory snap dump of each message sent and received. The option TRACET can be specified for just a log of all TCPIO events including connections and disconnects plus any errors. Note there are two separate logs for the client and server when running the demo SOA application. The server must be closed normally to see the generated log file. The utility DEMOSTOP.MLC can be assembled, linked, and executed to shut down the demo server by running soa\demo\demostop. The shut down server function is triggered by sending a message from the client with -1 length in the first 4 bytes of message. All other messages in the demo have 4 byte length of message in the first 4 bytes so the server and client can read variable length messages up to the size of the maximum message buffer length calculated during the SOA application generation. The client and server generate code to issue multiple TCPIO receives to retrieve an entire variable length message whenever only a partial message is received which can occur when the TCP/IP network is busy or under stress.

The TCPIO server port support includes multiple threads to support concurrent connections. There is one thread for each open server port which waits for new connections on the server port, starts new connection thread, and then returns to wait for another connection. Each connection thread waits for any current available messages to be read from that connection input buffer by the main TPCIO

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

user thread. When all messages have been retrieved, then the connection thread issues a connection client socket read for first byte of the next input message. The connection thread will wait for the next message to arrive in the input buffer or for a disconnect. If a disconnect occurs, the connection thread is cancelled. If a pending RECEIVE is waiting on the connection which disconnected, a return code of 12 is returned. If the read of first byte is successful the thread returns to wait for the main TCPIO user thread to retrieve the full or partial message available in the connection input buffer. Since partial messages can arrive from multiple connections in any sequence, the server must be sure to retrieve a complete message from a specific connection prior to returning to non specific RECEIVE for the next message.

See the z390 SOA Client Server Overview slides available on www.z390.org for diagram and sample code for single client server plus diagrams for single server with multiple clients and a diagram with multiple servers and multiple clients.

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

3. z390 SOA application generation support

The z390 SOA application generation macro SOAGEN can be used to generate customized client and server message managers plus stubs for each service called by the client application. In addition the SOAGEN macro generates two batch commands to build the SOA application and to execute the client server SOA application. The SOAGEN macro uses the z390 PUNCH extension operands DSNAME= and FORMAT to generate 3 or more source MLC files plus the two BAT files in one macro expansion execution. The keyword parameters on the SOAGEN macro call are as follows:

1. **MAIN= - name of main client program. If specified, an assembly and link of the main program with the generated service call stubs will be generated.**
2. **CTYPE=MLC/CBL define language type for client. COBOL clients generate IBM standard EZASOKET calls to TCP/IP to connect to services.**
3. **CLIENT= - name of the generated client message manager called from stubs.**
4. **SERVER= - name of the generated service message manager which loads and calls services based on service request messages.**
5. **HOST= - IP address of server host processor or * for local processor**
6. **PORT= - port # for this application (must be greater than 1023)**
7. **SERVICES= - one or more sublists defining the name of each called service and the length of each parameter being passed to service. If the length is negative, that indicates the parameter is read only and the updated parameter will not be returned in response message.**
8. **MACDIR= – directory containing the SOAGEN macros**
9. **GENDIR= – directory to contain the generated source files and command files**
10. **GENBLD= - name of the generated build command file**
11. **GENRUN= - name of the generated run command file**

The client message manager is generated using a call to the SOACMGR macro with the required parameters from the SOAGEN macro call. The client message manager performs the following functions when called from a client source call stub:

1. **On first call, dynamically allocate the required message buffer based on the maximum service message required.**
2. **On first call open a TCP/IP socket connection to the server message manager using the port and IP address specified.**
3. **Build a send message with message length, time stamp, service name, and all the parameters required by the service. Note the service can only access the parameters passed with the length specified. If a service needs to access additional parameters such as control blocks in memory, they need to all be passed to the service.**
4. **Send the message from client to server message manager using specified port**

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

5. **Wait for response from the server with matching time stamp and service name plus updated parameters, and return code from service. Note the generated client message manager has logic to issue more than 1 RECEIVE to fetch the entire variable length response message if necessary.**
6. **Move the returned updated parameters to the original calling list addresses.**
7. **exit to calling stub which exits to the calling client main application program.**

The server message manager is generated using a call to the SOASMGR macro with the required parameters from the SOAGEN macro call. The server message manager performs the following functions:

1. **Opens server socket for specified port which starts thread which listens for new connections from clients and starts new connection threads as required. The server port and listens for incoming messages from clients.**
2. **Issues receive on the server port to receive all or part of a message from a client on an open connection. The server logic fetches at least 4 bytes and uses the message length in the first 4 bytes to determine how many bytes must be read to complete the variable length message which may require additional receive commands.**
3. **Look up the service name specified. If service not found, an error is generated on server log and server returns to get next message.**
4. **Build call parameter address less pointing to the parameters in the received message. Note this implies that all updates by the service will be made to parameter areas in the message buffer.**
5. **Load the service on the first call and save entry address.**
6. **Call the service to update parameters in the message buffer.**
7. **Store the service return code in the message buffer.**
8. **Build return message truncated to just the updated parameters as indicated by positive lengths in the SOAGEN SERVICES parameter.**
9. **Send response message back to client message manager using same connection as request message.**
10. **Return to wait for next service request message from any client connection.**
11. **If client disconnect occurs, the disconnect is logged and server returns to wait for next request message from any other client connections.**

Stubs for each service name called by the client application are generated using calls to the macro SOASTUB. The functions performed by the generated stubs are:

1. **On first call load the client message manager and save address.**
2. **Call the client message manager passing the name of the service and the calling parameter list.**
3. **Upon return, exit to caller with return code.**

If the GENBLD parameter specifies a name, then the SOAGEN macro will generate a batch command file which assembles each of the above source programs to create an executable SOA type client server application. If the GENRUN parameter

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

specifies a name, then the SOAGEN macro will generate a batch command to start the named SOA server message manager on the same processor, and then run the client application. If the HOST parameter specifies a different processor, the generated server message manager will need to be copied to that processor and started prior to running the client application.

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

4. Demo SOA application

z390 SOA support includes a demo SOA application in the directory SOA\DEMO consisting of a main program DEMOMAIN.MLC which calls two subroutines DEMOSUB1.MLC and DEMOSUB2.MLC. The first subroutine calculates the sum of two numbers using extended decimal floating point, and the second subroutine adds two integer numbers. As a base line, the demo application can be assembled, statically linked, and executed using the command soa\demo\demostd.bat. See appendix II for the DEMOMAIN.LOG listing showing the results of static calls to each of the two subroutines 5 times including elapsed time measurements.

The same demo application can be generated as an SOA client server application using the command soa\demo\demosoa.bat. See III and IV for DEMOMAIN.LOG and DEMOSMGR .LOG showing the demo client and server logs. Note the difference in calculated end to end service response times and the fact that WTO messages from the services appear on the DEMOSMGR log instead of the DEMOMAIN log. Response time comparisons for the demo application when run on same processor, two processors on local LAN, and two processors on wireless LAN are summarized in Appendix V.

The soa\demo\demosoa.bat command executes the user defined SOAGEN macro call defined in soa\demo\demosoa.mlc:

```
 SOAGEN MAIN=DEMOMAIN,          MAIN CLIENT APPLICATION PGM   X
      CLIENT=DEMOCMGR,          SOA CLIENT MSG MGR NAME       X
      SERVER=DEMOSMGR,          SOA SERVER MSG MGR NAME       X
      HOST=*,  (192.168.1.3)    HOST SERVER NAME (*=LOCAL)    X
      PORT=3900,                HOST SERVER PORT              X
      SERVICES=((DEMOSUB1,-45,-45,45), SERVICES WITH PARM LENX
      (DEMOSUB2,-4,-4,4)),      (NOTE -LENGTH FOR READ ONLY)  X
      MACDIR=D:\WORK\Z390\SOA\MACLIB, SOA GEN MACRO DIRECTORYX
      GENDIR=D:\WORK\Z390\SOA\DEMO, DIRECTORY FOR SOA APPL   X
      GENBLD=DEMOBLD,           GENERATED BUILD BAT FILE      X
      GENRUN=DEMORUN            GENERATED RUN BAT FILE
END
```

The above SOA application generation macro call generates server message manager DEMOSMGR to run on the same host as client using HOST=*. To generate the same application to run server on a specific host, change the HOST= parameter to specify the IP address of the desired server. The IP address of any windows PC can be viewed by type IPCONFIG from the command prompt. The same client application generated with a specific host and port, can be run on the same processor or any processor on the same network as the server including networks connected via VPN connections over the Internet.

The above SOAGEN macro call generates the following source files in the soa\demo directory using z390 PUNCH extended operands DSNAME= and FORMAT to control PUNCH output files:

1. **DEMOCMGR.MLC – source macro call to SOACMGR to generate SOA client message manager for the demo application.**

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

2. **DEMOSMGR.MLC – source macro call to SOASMGR to generate SOA server message manager for the demo application.**
3. **SOA_STUB_DEMOSUB1.MLC - source macro call to SOASTUB to generate SOA stub for DEMOSUB1 service call.**
4. **SOA_STUB_DEMOSUB2.MLC – source macro call to SOASTUB to generate SOA stub for DEMOSUB2 service call.**
5. **DEMOBLD.BAT – generated command file to build the SOA demo application.**
6. **DEMORUN.BAT – generated command file to start the DEMOSMGR server on the same processor and then run the DEMOMAIN client application.**

The DEMOBLD.BAT command file assembles and links the following executable client server programs:

1. **DEMOCMGR.390 – client message manager**
2. **DEMOSMGR.390 – server message manager**
3. **DEMOMAIN.390 – main application program statically linked with the two stubs assembled from above stub source code.**
4. **DEMOSUB1.390 – service loaded and executed by DEMOSMGR when requested via message from DEMOCMGR.**
5. **DEMOSUB2.390 – service loaded and executed by DEMOSMGR when requested via message from DEMOCMGR.**

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

4. References:

- **SOA**
  - **IBM - Migrating to SOA**
  - **JavaWorld definition of XML based SOA architecture**
  - **Microsoft SOA**
  - **Software Developers Introduction to SOA**
  - **Sun SOA and Web Services**
- **TCP/IP**
  - **TCP/IP Transmission Control Protocol RFP**
  - **Sockets Tutorial**
  - **J2SE ServerSocket Class**
  - **J2SE Socket Class**
- **Host IP Addressing**
  - **J2SE InetAddress Class**
  - **IP Addressing RFC**
- **Socket Ports**
  - **Choosing a Socket Port**
  - **Registered Ports**
  - **Register a Port**

For latest z390 downloads and additional information visit www.z390.org

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

5. Appendix

Appendix I: Demo application source code

```
*********************************************************************
* Copyright 2006 Automated Software Tools Corporation               *
* This source code is part of z390 assembler/emulator package       *
* The z390 package is distributed under GNU general public license  *
* Author - Don Higgins                                              *
* Date   - 12/26/06                                                 *
*********************************************************************
* 12/31/06 RPI 523 CODE DEMOMAIN CALLING DEMOSUB1/DEMOSUB2
* 01/08/07 RPI 523 ADD TIMING IN MICRO-SEC AND PERFORM TWICE
*********************************************************************
* 1.  DEMOMAIN CALLS DEMOSUB1 TO CALCULATE AND DISPLAY SUM OF 2
*     DISPLAY SCIENTIFIC NOTATION VALUES USING CFD AND CTD MACROS.
*     FOR CONVERSION TO/FROM LD EXTENDED DFP FORMAT FOR ADDITION
* 2.  DEMOMAIN CALLS DEMOSUB2 TO CALCULATE AND DISPLAY SUM OF 2
*     INTEGER VALUES.
* 3.  RUN SOA\DEMO\DEMOSTD.BAT TO ASSEMBLE, STATICALLY LINK, AND
*     EXECUTE DEMO APPLICATION AS STANDARD LOCAL SINGLE PROCESS.
* 4.  RUN SOA\DEMO\DEMOSOA.BAT TO GENERATED, ASSEMBLE, LINK, AND
*     EXECUTE DEMO APPLICAITON USING SOA CLIENT SERVER TO ALLOW
*     RUNNING THE TWO CALLED SUBROUTINES AS SERVICES RUNNING ON
*     SEPARATE PROCESS ON SAME OR ANY TCP/IP CONNECTED PLATFORM.
*********************************************************************
         COPY  ASMMSP
DEMOMAIN SUBENTRY
         WTO   'DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION'
       ZAP   COUNT,=P'5'
       DO WHILE=(SP,COUNT,P,=P'0')
          BAL   R12,START_TIME
          CALL  DEMOSUB1,(DFP1,DFP2,DFP3),VL
          MVC   DSUM1,DFP3
          WTO   MF=(E,WTOMSG1)
          BAL   R12,STOP_TIME
          IF    (CLC,DFP3,NE,DFP4)
                WTO 'DEMOMAIN DEMOSUB1 DFP SUM INVALID - ABORTING'
                ABEND 111
          ENDIF
          BAL   R12,START_TIME
          CALL  DEMOSUB2,(INT1,INT2,INT3),VL
          L     R0,INT3
          CVD   R0,PWORK
          MVC   DSUM2,MASK2
          ED    DSUM2,PWORK+4
          WTO   MF=(E,WTOMSG2)
          BAL   R12,STOP_TIME
          IF    (CLC,INT3,NE,INT4)
                WTO 'DEMOMAIN DEMOSUB2 INT SUM INVALID - ABORTING'
                ABEND 111
          ENDIF
          SP   COUNT,=P'1'
       ENDDO
          WTO  'DEMOMAIN ENDED OK'
          SUBEXIT
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

```
START_TIME DS 0H
         TIME  NS,NS_START
         BR    R12
STOP_TIME DS 0H  SHOW SRERVICE TIME IN MICRO-SECONDS
         TIME  NS,NS_STOP
         LG    R1,NS_STOP
         SG    R1,NS_START
         DSG   R0,=FD'1000'
         CVD   R1,PWORK
         MVC   DMICS,MICS_MASK
         ED    DMICS,PWORK+3
         WTO   MF=(E,SHOW_MSG)
         BR    R12
NS_START DC    D'0' START TOD IN NANO-SECONDS
NS_STOP  DC    D'0' END   TOD IN NANO-SECONDS
SHOW_MSG DC    AL2(SHOW_END-*,0),C'SERVICE TIME IN MIRCO-SEC ='
DMICS    DC    C' ZZZ,ZZZ,ZZZ'
SHOW_END EQU   *
MICS_MASK DC   X'40202020',C',',X'202020',C',',X'202120'
         LTORG
COUNT    DC   PL4'0'
WTOMSG1  DC   AL2(WTOEND1-*,0),C'DEMOMAIN DEMOSUB1 DFP SUM='
DSUM1    DC   CL45' '
WTOEND1  EQU  *
WTOMSG2  DC   AL2(WTOEND2-*,0),C'DEMOMAIN DEMOSUB2 INT SUM='
DSUM2    DC   C' Z,ZZZ,ZZZ'
WTOEND2  EQU  *
MASK2    DC   C' ',X'20',C',',X'202020',C',',X'202120'
PWORK    DC   PL8'0'
DFP1     DC   CL45'1.1'
DFP2     DC   CL45'2.2'
DFP3     DC   CL45' '
DFP4     DC   CL45'3.3'  VERIFY SUM VALUE
INT1     DC   F'1'
INT2     DC   F'2'
INT3     DC   F'0'
INT4     DC   F'3'
         EQUREGS
         END
*********************************************************************
* Copyright 2006 Automated Software Tools Corporation               *
* This source code is part of z390 assembler/emulator package       *
* The z390 package is distributed under GNU general public license  *
* Author - Don Higgins                                              *
* Date   - 12/26/06                                                 *
*********************************************************************
* 12/31/06 RPI 523 CODE EXAMPLE APPLICATION DEMOSUB1 ROUTINE
*********************************************************************
* CALC SCIENTIFIC DISPLAY PARM1 + PARM2 = PARM3 USING EXTENDED DFP
*********************************************************************

DEMOSUB1 SUBENTRY
         LM    R3,R5,0(R1       LOAD 3 PARM ADDRESSES
         WTO   'DEMOSUB1 ENTERED'
         CFD   CFD_LD,IN=(R3),OUT=0  F0,R2 = LD PARM1
         CFD   CFD_LD,IN=(R4),OUT=1  F1,R3 = LD PARM2
         AXTR  0,0,1                 F0,R2 = LD PARM1 + LD PARM2
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

```
         CTD   CTD_LD,IN=0,OUT=DSUB  DISPLAY VALUE OF SUB
         WTO   MF=(E,WTOMSG)    DISPLAY PARM3 = LD PARM1 + LD PARM2
         MVC   0(L'DSUB,R5),DSUB     UPDATE PARM3 FROM DSUB
         WTO   'DEMOSUB1 EXITING'
         SUBEXIT
WTOMSG   DC   AL2(WTOEND-*,0),C'DEMOSUB1 SUM='
DSUB     DC   CL45' '
WTOEND   EQU  *
         EQUREGS
         END

*********************************************************************
* Copyright 2006 Automated Software Tools Corporation               *
* This source code is part of z390 assembler/emulator package       *
* The z390 package is distributed under GNU general public license  *
* Author - Don Higgins                                              *
* Date   - 12/26/06                                                 *
*********************************************************************
* 12/31/06 RPI 523 CODE EXAMPLE APPLICATION CALLED DEMOSUB2 ROUTINE
*********************************************************************
* CALC INT PARM1 + INT PARM2 = INT PARM3
*********************************************************************
DEMOSUB2 SUBENTRY

         LM    R3,R5,0(R1)      GET 3 PARM ADDRESSES
         WTO   'DEMOSUB2 ENTERED'
         L     R0,0(R3)         LOAD  INT PARM1
         A     R0,0(R4)         ADD   INT PARM2
         ST    R0,0(R5)         STORE INT PARM3
         CVD   R0,PWORK
         MVC   DSUM,MASK
         ED    DSUM,PWORK+4
         WTO   MF=(E,WTOMSG)    DISPLAY PARM3 = INT PARM1 + INT PARM2
         WTO   'DEMOSUB2 EXITING'
         SUBEXIT
WTOMSG   DC   AL2(WTOEND-*,0),C'DEMOSUB2 SUM='
DSUM     DC   C' Z,ZZZ,ZZ9'
WTOEND   EQU  *
MASK     DC   C' ',X'20',C',',X'202020',C',',X'202120'
PWORK    DC   PL8'0'
         EQUREGS
         END
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

Appendix II: Demo application execution log for statically linked base line

```
EZ390I V1.3.02 Current Date 03/05/07 Time 09:18:28
EZ390I Copyright 2006 Automated Software Tools Corporation
EZ390I z390 is licensed under GNU General Public License
EZ390I program = DEMOMAIN
EZ390I options =
DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       3,009
DEMOSUB2 ENTERED
DEMOSUB2 SUM=         3
DEMOSUB2 EXITING
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       1,224
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       3,717
DEMOSUB2 ENTERED
DEMOSUB2 SUM=         3
DEMOSUB2 EXITING
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       1,683
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       3,412
DEMOSUB2 ENTERED
DEMOSUB2 SUM=         3
DEMOSUB2 EXITING
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       1,340
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       8,581
DEMOSUB2 ENTERED
DEMOSUB2 SUM=         3
DEMOSUB2 EXITING
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

DEMOMAIN DEMOSUB2 INT SUM= 3
SERVICE TIME IN MIRCO-SEC = 869
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC = 9,103
DEMOSUB2 ENTERED
DEMOSUB2 SUM= 3
DEMOSUB2 EXITING
DEMOMAIN DEMOSUB2 INT SUM= 3
SERVICE TIME IN MIRCO-SEC = 786
DEMOMAIN ENDED OK
EZ390I Stats total instructions = 584
EZ390I Stats current date 03/05/07 time 09:18:28
EZ390I Stats total seconds = 0
EZ390I Stats instructions/sec = 6212
EZ390I total errors = 0
EZ390I return code(DEMOMAIN)= 0

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

Appendix III: Demo SOA client application execution log

```
EZ390I V1.3.02 Current Date 03/05/07 Time 09:26:04
EZ390I Copyright 2006 Automated Software Tools Corporation
EZ390I z390 is licensed under GNU General Public License
EZ390I program = DEMOMAIN
EZ390I options =
DEMOMAIN SERVICE ORIENTED ARCHITECTURE APPLICATION
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =     117,273
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       7,243
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       7,037
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       4,827
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =      15,476
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       5,733
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       7,031
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       3,563
DEMOMAIN DEMOSUB1 DFP SUM=3.3
SERVICE TIME IN MIRCO-SEC =       3,883
DEMOMAIN DEMOSUB2 INT SUM=         3
SERVICE TIME IN MIRCO-SEC =       3,918
DEMOMAIN ENDED OK
EZ390I Stats total instructions    = 1409
EZ390I Stats current date 03/05/07 time 09:26:04
EZ390I Stats total seconds         = 0
EZ390I Stats instructions/sec      = 6906
EZ390I total errors         = 0
EZ390I return code(DEMOMAIN)= 0
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

Appendix IV: Demo SOA server application execution log

```
EZ390I V1.3.02 Current Date 03/05/07 Time 09:26:00
EZ390I Copyright 2006 Automated Software Tools Corporation
EZ390I z390 is licensed under GNU General Public License
EZ390I program = DEMOSMGR
EZ390I options =
DEMOSMGR STARTED
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
SERVICE TIME IN MIRCO-SEC =       2,881
DEMOSUB2 ENTERED
DEMOSUB2 SUM=        3
DEMOSUB2 EXITING
SERVICE TIME IN MIRCO-SEC =       2,521
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
SERVICE TIME IN MIRCO-SEC =       3,489
DEMOSUB2 ENTERED
DEMOSUB2 SUM=        3
DEMOSUB2 EXITING
SERVICE TIME IN MIRCO-SEC =       1,885
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
SERVICE TIME IN MIRCO-SEC =      12,218
DEMOSUB2 ENTERED
DEMOSUB2 SUM=        3
DEMOSUB2 EXITING
SERVICE TIME IN MIRCO-SEC =       1,757
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
SERVICE TIME IN MIRCO-SEC =       4,042
DEMOSUB2 ENTERED
DEMOSUB2 SUM=        3
DEMOSUB2 EXITING
SERVICE TIME IN MIRCO-SEC =       1,191
DEMOSUB1 ENTERED
DEMOSUB1 SUM=3.3
DEMOSUB1 EXITING
SERVICE TIME IN MIRCO-SEC =       1,726
DEMOSUB2 ENTERED
DEMOSUB2 SUM=        3
```

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

DEMOSUB2 EXITING
SERVICE TIME IN MIRCO-SEC = 1,184
TCPIO disconnect during read for conn=0 port=3900
DEMOSMGR TERMINATING AT USER REQUEST
EZ390I Stats total instructions = 1208
EZ390I Stats current date 03/05/07 time 09:26:06
EZ390I Stats total seconds = 6
EZ390I Stats instructions/sec = 197
EZ390I total errors = 0
EZ390I return code(DEMOSMGR)= 0

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License

Appendix V: z390 Demo SOA Application Timing Statistics

Z390 Service Oriented Architecture Demo Timings in milli-seconds as of v1.3.02

Event	Static Link	Single CPU	LAN	Wireless LAN
Initialize server	0	120	120	120
Initialize connection	0	40	40	40
DEMOSUB1 service time	2	2	2	2
DEMOSUB1 total time	2.2	5	5	5
DEMOSUB2 service time	1.5	1.5	1.5	1.5
DEMOSUB2 total time	1.7	3	5	5

Conclusions:

1. **Static linking will always be faster, but makes sharing services more difficult.**
2. **For SOA TCP/IP messaging the average overhead is about 2-3 milli-seconds.**
3. **The major advantage of SOA is that services can be easily shared and maintained.**

Notes:

1. **Initialize server time occurs on first call to service after startup and includes loading services.**
2. **Initialization connection time only occurs on first call after starting client to establish connection.**
3. **Service time is the time to execute service on the server as measured by DEMOSMGR.MLC**
4. **Total time is end to end average time excluding first as measured by DEMOMAIN..MLC**
5. **Single CPU timings are for both client and server running on same 3.0 GHZ Dell PC**
6. **100 MB LAN timing is for client on one PC and server on anther PC on same 100 MB LAN**
7. **Wireless LAN timing is for client on PC Laptop connected to LAN server via wireless router.**
8. **No changes were made to programs generated by SOA\DEMO\DEMOSOA.BAT.**
9. **It may be necessary to add IP address of client to server security system like Norton NIS.**
10. **There is no security provided with z390 SOA applications current so they should only be run on private secure networks or via VPN connections over the Internet.**

Copyright 2008 Automated Software Tools Corporation
z390 Licensed under General Public License