z/TPF TCP/IP SOCKET Driver Users Guide

1.0 Introduction

The socket driver consists of multiple DLMs that issue TCP/IP API calls to send and receive TCP data. There are many variations to the driver functional message to start different types of servers and clients. Depending on the functional message entered the driver can start TCP servers and clients, UDP servers and clients, servers and clients sending and receiving data using various API calls, sockets that send and receive out-of-band (OOB) data, sockets that run in blocking and non-blocking mode, etc. For more information on how to use and run the driver see ***Section 2.0 - Syntax Information***.

2.0 Syntax Information

I. Syntax Diagram:

Most of the functional messages are arranged in the way shown below.
Note: There are socket functional messages that do not follow this format

```
>>--|-ZTEST-----|--SOCK--|--socket-----------|-->
                         |--protocol---------|-->
                         |--type-------------|-->
                         |-------------------|-->
                            |---ip address--|-->
                            |---port no.----|-->
                            |---msgs--------|-->
                            |---msg_size----|-->
                            |---resends-----|-->
```

Where:

PARAMETER	DESCRIPTION
socket	Specifies whether the socket is a server socket or a client socket. For a server socket, specify *server*. For a client socket, specify *client*.
protocol	Specifies whether the socket is the UDP or TCP protocol. For a UDP socket, specify *datagram*. For a TCP socket specify *stream*. Note: The *nstream* and *ndatag* can also be used to invoke certain drivers (see below)
type	Specifies the type of socket API and/or socket options that will be used. Available types are: NOAOR AOR TPFTOTPF SELECT AORAOR AORSENDTO SERRC SETSOCKOPT BLOCK NONBLOCK OOB THREADNOAOR THREADAOR THREAD CREM SWEEP NOCONN NOBIND **Note: Not all types can be used with every *socket* and every *protocol* (see below)**
ip address	This field is only valid if the socket is a CLIENT. It specifies the IP

	address of the remote server you wish to communicate with.
port no.	This field is only valid if the socket is a CLIENT. It specifies the port number of the remote server you wish to communicate with.
msgs	This field is only valid if the socket is a CLIENT. It specifies the number of messages you wish to send to the server.
msg_size	This field is only valid if the socket is a CLIENT. It specifies the size of the message you wish to send to the server.
resends	This field is only valid if the socket is a CLIENT and the protocol is DATAGRAM. It specifies the number of times you want the UDP client to attempt to resend the message without getting a response before it considers the connection dead.
function	This field is only valid when connecting to servers with the nstream or ndatagr protocol specified. It specifies the type of function you want the driver to perform: PING-PONG, PINGPONG1, PINGPONG2: Echo TCP messages back and forth SEND-ONLY, SENDONLY1, SENDONLY2: Client just continually sends while server just does reads READ-ONLY,READONLY1, READONLY2: Server just continually sends while client just does reads

******* ZTEST SOCK HELP/? can be used to display an on-line help *******

I. Sample Invocations

The next section is split into servers that can be started and clients that can be started using the socket driver. Each application (server or client) has documented the associated server or client that it can establish connections with as well as a short description of the processing that is involved with the application. The servers also have associated with them the port that the server is bound to so when starting the client to that server, you can specify the correct port.

Server Applications

ZTEST SOCK SERVER STREAM NOAOR - *segment QXYD* ***(PORT 5003)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5003. It will then issue a listen() and an accept() waiting for a client connection. When the client connects, the application issues a recvfrom() reading the data and immediately issues sendto() to echo the data back to the client. When all the client messages have been read and echoed back, the server will issue another accept waiting for the next connection. Other APIs issued: getsockname(), gethostname(), poll(), close(), shutdown()

CLIENTS: *ZTEST SOCK CLIENT STREAM BLOCK*

ZTEST SOCK SERVER STREAM AOR - *segment QXYE* ***(PORT 5004)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5004. It will then issue a listen() and an accept() waiting for a client connection. When

the client connects, the application issues an activate_on_receipt() activating program QXYK (which is contained in its own DLM - QXYK). QXYE goes on to issue another accept() waiting for another connection request. QXYK reads in the clients messages with with read() and echoes the data back to the client with a write(). Other APIs that may be issued: getsockname(), gethostname(), close(), shutdown(), getpeername().
CLIENTS: *ZTEST SOCK CLIENT STREAM BLOCK*

ZTEST SOCK SERVER DATAGRAM NOAOR - *segment QXYB* ***(PORT 5001)***
This application creates a UDP socket binding to IP address IPADDR_ANY and port 5001. It will then issue a recvfrom() waiting for remote application data and when data is received it will echo the data back with a sendto(). It will continue issuing recvfrom() and sendto() infinitely. Other APIs that may be issued: getsockname(), setsockopt(), close().
CLIENTS: *ZTEST SOCK CLIENT DATAGRAM BLOCK*
ZTEST SOCK CLIENT DATAGRAM NOCONN
ZTEST SOCK CLIENT DATAGRAM NOBIND

ZTEST SOCK SERVER DATAGRAM AOR - *segment QXYC* ***(PORT 5002)***
This application creates a UDP socket binding to IP address IPADDR_ANY and port 5002. It will then issue an activate_on_receipt() which will activate segment QXYL (which is contained in its own DLM - QXYL). QXYL issues a recvfrom() and echoes back the data with a sendto(). QXYL will then issue another AOR() activating itself when data arrives. Other APIs that may be issued: getsockname(), setsockopt(), shutdown(), close().
CLIENTS: *ZTEST SOCK CLIENT DATAGRAM BLOCK*
ZTEST SOCK CLIENT DATAGRAM NOCONN
ZTEST SOCK CLIENT DATAGRAM NOBIND

ZTEST SOCK SERVER STREAM TPFTOTPF - *segment QXYF* ***(PORT will be assigned by TPF)***
This application creates a TCP socket binding to IP address IPADDR_ANY and to a port of 0, meaning TPF will assign an available port. The application then issues a listen() and accept() to wait for client connect requests. When a client connects, the application issues an activate_on_receipt() and when data is available segment QXYM is activated. QXYF loops back to issue another accept() to wait for another connection. QXYM issues read() and echoes back each message the client sends with a write(). When all the messages are read, it closes the socket. Other APIs that may be issued: shutdown(), getpeername(), getsockname(), gethostname(). Note: when connecting a client to this server, you have to connect to the port TPF assigned to the server.
CLIENTS: *ZTEST SOCK CLIENT STREAM CREM*

ZTEST SOCK SERVER DATAGRAM SELECT - *segment QXYI* ***(PORT 5005)***
This application creates a UDP socket binding it to IP address IPADDR_ANY and port 5005. The application will then go into a loop doing a select() for read. When data arrives, the select() gets posted and the application will issue a recvfrom() and echo the data back with a sendto(). This is an infinite loop with it issuing another select(). Other

APIs that may be issued: close(),setsockopt(), getsockname().
CLIENTS: *ZTEST SOCK CLIENT DATAGRAM BLOCK*
ZTEST SOCK CLIENT DATAGRAM NOCONN
ZTEST SOCK CLIENT DATAGRAM NOBIND

ZTEST SOCK SERVER STREAM AORAOR - *segment QXYJ* ***(PORT 5006)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5006. The server will do a listen() and go into an infinite accept() loop. When a client connects the server will do an activate_on_receipt() and when data is available to read, QXYL will be activated. QXYL will issue read() and echo back the data with a send() until all the clients messages are processed. Other APIs that may be issued: close(), shutdown(), getsockname(), gethostname().
CLIENTS: *ZTEST SOCK CLIENT STREAM BLOCK*

ZTEST SOCK SERVER STREAM AORSENDTO - *segment QXYQ* ***(PORT 5007)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5007. The server will do a listen() and go into an infinite accept() loop. When a client connects the server will do an activate_on_receipt() and when data is available to read, QXYL will be activated. QXYL will issue recvfrom() and echo back the data with a sendto() until all the clients messages are processed. Other APIs that may be issued: close(), shutdown(), getsockname(), gethostname().
CLIENTS: *ZTEST SOCK CLIENT STREAM BLOCK*

ZTEST SOCK SERVER STREAM SERRC - *segment QXYR* ***(PORT will be assigned TPF)***

This application creates a TCP socket binding to IP address IPADDR_ANY and to a port of 0, meaning TPF will assign an available port. The application then issues a listen() and and accept() to wait for client connect requests. When a client connects, the application issues an activate_on_receipt(), which will activate segment QXYM when data becomes available to read. QXYM will issue a read() and echo the data back with a write() until all the messages are processed. As QXYM is processing the client QXYR continues doing accepts. When the fifth client connects, QXYR will issue an OPR dump and exit. Other APIs that may be issued: close(), shutdown(), gethostname(), getsockname(), getpeername(), poll().
CLIENTS: *ZTEST SOCK CLIENT STREAM CREM*

ZTEST SOCK SERVER STREAM SETSOCKOPT - *segment QXYT* ***(PORT 5009)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5009. The application will then issue numerous setsockopt() and getsockopt() API calls to set values for that socket and to display them. The server will then issue a listen() and an accept(), waiting for a client connection. When the connection request is received and the connection is established, the application issues more setsockopt() and getsockopt() APIs on the new socket block. It will then read the data sent by the remote client

application and echo it back using read() and send() APIs. Other APIs that may be issued: close(), shutdown(), gethostname(), getpeername(), getsockname(), gethostid().
CLIENTS: *ZTEST SOCK CLIENT STREAM BLOCK*

ZTEST SOCK SERVER STREAM NONBLOCK - *segment QXYU* ***(PORT 5010)***
This application creates a TCP socket binding to IP address IPADDR_ANY and port 5010. The application will the issue a listen() and then do an ioctl() to change the socket to run in non-blocking mode. The server then issues an accept() and goes into a select() loop to verify if a connection request was received. When a connection is established, we CREM to segment QXZX to exchange the data and QXYU will issue another accept waiting for connections. Using, select() for reads and select() for writes, QXZX can verify if data is available for read/write and process the data accordingly. Other APIs that may be issued: close(), shutdown(), setsockopt(), getsockopt().
CLIENTS: *ZTEST SOCK CLIENT STREAM NONBLOCK*

ZTEST SOCK SERVER DATAGRAM NONBLOCK - *segment QXYW* ***(PORT 5011)***

This application creates a UDP socket binding to IP address IPADDR_ANY and port 5011. The application will then do an ioctl() to change the socket to run in non-blocking mode. The server then issues a recvfrom() waiting for data from the client. When data is received, the server will echo the data back using the sendto() API. Other APIs that may be issue close(), setsockopt(), getsockopt(), getsockname().

CLIENTS: *ZTEST SOCK CLIENT DATAGRAM NONBLOCK*

ZTEST SOCK SERVER STREAM OOB - *segment QXZA* ***(PORT 5012)***

This application creates a TCP socket binding to IP address IPADDR_ANY and port 5012. The application will then do a listen() and accept(), waiting for connection requests. The server will process both OOB data and regular data using the select() API to determine which type of data is on the queue and doing a recv() to read the data. Whatever is sent to the server will be echoed back to the client using the send() API. Other APIs that may be issued: setsockopt(), ioctl(), close().

CLIENTS: *ZTEST SOCK CLIENT STREAM OOB*

ZTEST SOCK IP-xxx.xxx.xxx.xxx PORT-yyyyy STYPE-vvv SPROT-zzz AOR-Y/N SELECT-Y/N - *segment QXZI*

Where: xxx.xxx.xxx.xxx is the local IP address to bind to
yyyyy is the server port number to bind to
vvv is the type of socket to create (UDP/TCP)
zzz is the protocol to use (UDP/TCP)

The server application creates the type of socket specified by the user and issues the functions specified by the user (ie. AOR, SELECT, etc.). Depending on what type of server you create is the way the application will process the data. The function specified by the client also determines certain server behavior, for example specifying AOR-Y will cause the server to actually use activate_on_receipt_of_TCP_message() instead of activate_on_receipt() with a PONGPONG1 or SENDONLY1 client.

CLIENTS: *ZTEST SOCK CLIENT NSTREAM BLOCK <ip> <port> <# msg> <size> <func>*
ZTEST SOCK CLIENT NSTREAM NONBLOCK <ip> <port> <# msg> <size> <func>
ZTEST SOCK CLIENT NDATAGR BLOCK <ip> <port> <# msg> <size> <func>
ZTEST SOCK CLIENT NDATAGR NONBLOCK <ip> <port> <# msg> <size> <func>

ZTEST SOCK FCS-xx [PORT-yyyy] [IP-x.x.x.x]

The FCS option invokes function oriented tests in QXZI. Some of these tests require the use of a remote application to act as a client or server, which the test will prompt for.

FCS-	*Tested API or condition*
1	*accept*
2	*activate_on_receipt*
2a	*activate_on_accept*
3	*bind*
4	*close*
5	*connect*
6	*gethostbyaddr*
7	*gethostbyname*
8	*gethostid*
9	*gethostname*
10	*getpeername*
10a	*getsockname*
11	*getsockopt*
12	*ioctl*
13	*listen*
13a	*poll*
14	*read*
15	*recv*
16	*recvfrom*
17	*select*
18	*send*
19	*sendto*
20	*setsockopt*
21	*shutdown*
22	*socket*
23	*write*
24	*writev*
25	*sendmsg*
26	*recvmsg*
101	*UACC*
102	*ESOCINACT*
106	*Client connects then closes*
107	*APAR PJ29020*
108-114	*poll*
210	*UDP connect*

Client Applications

ZTEST SOCK CLIENT DATAGRAM BLOCK ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < NO. RESEND >*** *- segment QXYG*

This application creates a UDP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024-5000). The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to resend a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the read() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), getsockname(), select()

SERVERS: ZTEST SOCK SERVER DATAGRAM NOAOR ***(port 5001)***
ZTEST SOCK SERVER DATAGRAM AOR ***(port 5002)***
ZTEST SOCK SERVER DATAGRAM SELECT ***(port 5005)***

ZTEST SOCK CLIENT STREAM BLOCK ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE >*** *- segment QXYH*

This application creates a TCP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. The message size sent is also specified by the user. Once all of the messages have been sent and have been echoed back by the server and read using the read() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), getsockname(), select()

SERVERS: ZTEST SOCK SERVER STREAM NOAOR ***(port 5003)***
ZTEST SOCK SERVER STREAM AOR ***(port 5004)***
ZTEST SOCK SERVER STREAM AORAOR ***(port 5006)***
ZTEST SOCK SERVER STREAM AORSENDTO ***(port 5007)***

ZTEST SOCK CLIENT STREAM CREM ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < # CLIENTS >*** *- segment QXYN*

This application initially enters QXYA (the socket driver parser) and issues CREMC to QXYN for the number of clients specified on input to the driver. The application creates a TCP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). Each client (created by CREMC) will issue a connect to the server and begin sending the messages specified by the user on the functional message using the write() API. The message size sent is also specified by the user. Once all of the messages have been sent and have been echoed back by the server and read using the read() API, we will issue close() to clean up

the connection. Other APIs that may be issued: hutdown().
SERVERS: ZTEST SOCK SERVER STREAM TPFTOTPF ***(port assigned by TPF)***

ZTEST SOCK CLIENT DATAGRAM SWEEP ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < NO. RESEND >*** *- segment QXYG*

This application creates a UDP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to resend a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the read() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), getsockname(), pool()
SERVERS: ZTEST SOCK SERVER DATAGRAM NOAOR ***(port 5001)***
ZTEST SOCK SERVER DATAGRAM AOR ***(port 5002)***
ZTEST SOCK SERVER DATAGRAM SELECT ***(port 5005)***

ZTEST SOCK CLIENT DATAGRAM NOSWEEP ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < NO. RESEND >*** *- segment QXY9*

This application creates a UDP socket with TPF_NOSWEEP on and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to re - send a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the read() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), getsockname(), poll()
SERVERS: ZTEST SOCK SERVER DATAGRAM NOAOR ***(port 5001)***
ZTEST SOCK SERVER DATAGRAM AOR ***(port 5002)***
ZTEST SOCK SERVER DATAGRAM SELECT ***(port 5005)***

ZTEST SOCK CLIENT STREAM NONBLOCK ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE>*** *- segment QXYV*

This application creates a TCP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 -5000). The client will issue an ioctl() to put the socket in non-blocking mode. The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. The message size sent is also specified by the user. Once all of the messages have been sent and have been echoed

back by the server and read using the recv() API, we will issue close() to clean up the connection. Other APIs that may be issued:setsockopt(), shutdown(), close(), select()
SERVERS: ZTEST SOCK SERVER STREAM NOBLOCK ***(port 5010)***

ZTEST SOCK CLIENT DATAGRAM NONBLOCK ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE> < # RESENDS >*** *- segment QXYX*

This application creates a UDP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will issue an ioctl() to put the socket in non-blocking mode. The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() and sendto() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to resend a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the recvfrom() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), shutdown(), close(), select()
SERVERS: ZTEST SOCK SERVER DATAGRAM NOBLOCK ***(port 5011)***

ZTEST SOCK CLIENT DATAGRAM NOCONN ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < NO. RESEND >*** *- segment QXYY*

This application creates a UDP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will not issue a connect to the server, which causes the communication to be connectionless. The client will begin sending the messages specified by the user on the functional message using the sendto() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to resend a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the recvfrom() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), getsockname(), select()

SERVERS: ZTEST SOCK SERVER DATAGRAM NOAOR ***(port 5001)***
ZTEST SOCK SERVER DATAGRAM AOR ***(port 5002)***
ZTEST SOCK SERVER DATAGRAM SELECT ***(port 5005)***

ZTEST SOCK CLIENT DATAGRAM NOBIND ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE > < NO. RESEND >*** *- segment QXYZ*

This application creates a UDP socket. A bind() is not issued causing the socket to be bound under the covers when sending the first message. The client will not issue a connect to the server, which causes the communication to be connectionless. The client will begin sending the messages specified by the user on the functional message using the sendto() API. The message size sent is also specified by the user. The number of resends on input to the application is the number of times to resend a lost message before breaking the connection. Once all of the messages have been sent and have been echoed back by the server and read using the recvfrom() API, we will issue close() to clean up the connection. Other APIs that may be issued:setsockopt(), getsockname(), select(), poll().

SERVERS: ZTEST SOCK SERVER DATAGRAM NOAOR ***(port 5001)***
ZTEST SOCK SERVER DATAGRAM AOR ***(port 5002)***
ZTEST SOCK SERVER DATAGRAM SELECT ***(port 5005)***

ZTEST SOCK CLIENT STREAM OOB ***< IP ADDR > < PORT # > < # MSGS > < MSG SIZE >***
- segment QXZB

This application creates a TCP socket and issues a bind(), which will bind the socket to the system's default IP address and the next client port in the port number range (1024 - 5000). The client will issue a connect to the server and begin sending the messages specified by the user on the functional message using the send() API. These messages will contain OOB data that the server will have to parse out. The message size sent is also specified by the user. Once all of the messages have been sent and have been echoed back by the server and read using the recv() API, we will issue close() to clean up the connection. Other APIs that may be issued: setsockopt(), select()
SERVERS: ZTEST SOCK SERVER STREAM OOB ***(port 5012)***

ZTEST SOCK CLIENT NSTREAM BLOCK ***<ip> <port> < # msg > < msg size > <function >***
ZTEST SOCK CLIENT NSTREAM NONBLOCK ***<ip> <port> < # msg > < msg size > <function >***
ZTEST SOCK CLIENT NDATAGR BLOCK ***<ip> <port> < # msg > < msg size > <function >***
ZTEST SOCK CLIENT NDATAGR NONBLOCK ***<ip> <port> < # msg > < msg size > <function >***
SEGMENT - QXY7

These clients connect to many different types of servers. Based on the client driver being NSTREAM or NDATAGR will determine if socket is TCP or UDP, respectively. Then there is the BLOCK or NONBLOCK format, which determines if the client will run in blocking or non-blocking mode. The parameters are:

<ip> - IP address of server to connect to
<port> - Port of server to connect to
<#msg> - Number of messages to send to server
<msg size> - Size of messages to send to server
<function> - Can be one of the following:

SEND-ONLY - Client just sends messages to server; does not read any messages in from server
SENDONLY1 - Client sends TCP formatted messages, server uses either tpf_read_TCP_message or activate_on_receipt_of_TCP_message
SENDONLY2 - Client sends TCP format 2 messages, server uses either tpf_read_TCP_message2 or activate_on_receipt_of_TCP_message2
READ-ONLY - Client just reads messages from server, does not send any messages to the server
READONLY1 - Client just reads formatted messages from the server using tpf_read_TCP_message, does not send any messages to the server
READONLY2 - Client just reads format 2 messages from the server using tpf_read_TCP_message2, does not send any messages to the server
PING-PONG - Server and Client ping-pong messages between them.
PINGPONG1 - Server and Client ping-pong messages between them, using TCP formatted messages.
PINGPONG2 - Server and Client ping-pong messages between them, using TCP format 2 messages.