

Socket, Socket,
Who has the Socket
WAVV 2004

Tony Thigpen
Tony@VSE2PDF.COM

What is EZA?

- EZA is the IBM product prefix for TCP/IP on MVS
- MVS has three major programming interfaces to TCP/IP
 - BSD/C Sockets
 - EZASMI (Assembler Macro)
 - EZASOKET (HLL API)
 - REXX

MVS EZA BSD/C Sockets

- Based on “Berkeley” standards
 - Open Group Technical Standards for Networking Services
 - <http://www.opengroup.org/onlinepubs/009619199/>
- Example:
 - `int recv(int, char *, int, int);`
 - `result = recv(socket,&buffer,length,flags);`

EZASMI

- Assembler Macro Interface
 - OS/390 SecureWay Communications Server
IP Application Programming Interface Guide
Version 2 Release 8
Document Number SC31-8516-03
- Example:
 - EZASMI Type=Recv,S=socket,
Buf=,Nbyte=,Flags=,Errorno=,Retcode

EZASOKET

- High Level Language API
 - (Same document as EZASMI)
- Example:
 - CALL 'EZASOKET' USING
SOC-FUNCTION S FLAGS NBYTE BUF
ERRNO RETCODE

REXX

- REXX API
 - (Same document as EZASMI)
- Example:
 - `Socket('RECV',s,maxlength,flags)`

EZA and VSE

- BSD/C
 - Implemented by LE/VSE C Runtime
- REXX
 - Implemented by REXX/VSE
- EZASMI and EZASOCKET
 - Implemented on VSE 2.5 by IBM
 - Implemented on VSE 2.1 and higher by BSI for use on their TCP/IP stack

Relationships

- BSD/C calls are the basic building block
- Other interfaces just enable other languages to communicate to the BSD/C calls.
- REXX, EZASMI, and EZASOKET all have calls that are subsets of the available BSD/C calls.

Why Use the EZA Interface?

- Portability
 - HLL (call 'EZASOCKET')
 - ASM (EZASMI macro)
 - REXX (s=SOCKET('Open',...))
- Non-portability
 - HLL (EXEC TCP ...)
 - ASM (SOCKET macro)
 - REXX (s=SOCKET('TCP','OPEN'))

Why Use the EZA Interface?

- For simple open/send/receive/close functions, the CSI Interface is easier to code, but it does require a pre-translate step for the API.
- And the CSI API is TCP/IP Version specific.
 - Going to TCP/IP 1.4 required relinking of all phases using the API.

Why Use the EZA Interface?

- Each CSI open or close performs many TCP/IP functions.
 - For programs that perform multiple opens, this overhead can not be eliminated.
- Each EZA call performs only the function being used.
 - For programs performing multiple opens, the overhead is greatly reduced.

Why Use the EZA Interface?

- Some capabilities of TCP/IP can not be used when using the CSI Interface
 - Simultaneous Reads and Writes
 - Giving and Taking of open communication links (can be done, but not documented)
 - “Look Ahead” or “PEEK” processing
 - IBM could not program NJE over TCP/IP without first implementing EZASMI in VSE

Support Routines

- EZACIC04 EBCDIC-to-ASCII
- EZACIC05 ASCII-to-EBCDIC
- EZACIC06 SELECT bit stream setup
- EZACIC08 HOST field processor

Types of Programs

- Client
 - Connects to a Server
- Iterative Server
 - All processing is self-contained
- Concurrent Server
 - A Listener that spawns a Child when connected
- Child
 - A “partial” server to handle sends/receives

Concurrent Server and Child

- Why?
 - Iterative Server has deficiencies
 - 1 to 1 only
 - processing is tied up while handling the sends and receives
 - Additional Clients can not get a connection

EZA Client Program Flow

- INITAPI (EZASMI only)
- SOCKET
- CONNECT
- SEND/RECV loop
- SHUTDOWN
- CLOSE
- TERMAPI (EZASMI only)

EZA Client Program Flow

- INITAPI (EZASMI only)
 - Loads interface programs into GETVIS
 - Allocates storage
 - Initializes default control information
 - Verifies that the TCP/IP stack is available
 - The EZASOKET interface performs this function behind the scenes

EZA Client Program Flow

- SOCKET
 - Assigns a socket number (Binary half-word)
 - Allocates socket specific storage
 - Informs caller of socket number
- CONNECT
 - Establishes a communications session with the requested server

EZA Client Program Flow

- WRITE, SEND, or SENDTO
 - Transmits data
- READ, RECV, or RECVFROM
 - Receives Data

EZA Client Program Flow

- SHUTDOWN
 - Informs stack to close down communications once all buffers are transmitted
- CLOSE
 - Releases socket specific storage acquired by the SOCKET call
- TERMAPI (EZASMI only)
 - Releases all storage acquired by the INITAPI call
 - The EZASOCKET interface performs this function behind the scenes

EZA Iterative Server Flow

- INITAPI (EZASMI only)
- SOCKET
- BIND
- LISTEN
- ACCEPT loop
 - SEND/RECV loop
 - SHUTDOWN
 - CLOSE
- SHUTDOWN
- CLOSE
- TERMAPI (EZASMI only)

EZA Iterative Server Flow

- INITAPI
 - Same as Client Program
- SOCKET
 - Same as Client Program
- BIND
 - Informs interface as to what local port to use
- LISTEN
 - Informs the stack that the program wants any data destined for the local port specified by the BIND

EZA Iterative Server Flow

- ACCEPT Loop
 - Informs the stack that the program is ready to receive data
 - When data is received, a new socket area is allocated and the program is informed of this new socket number on which the communication is to occur.
 - The original socket number is NOT used. It remains available for more ACCEPT calls

EZA Iterative Server Flow

- SEND/RECV
 - Transfers data (on the NEW socket)
- SHUTDOWN
 - Informs stack to close down communications once all buffers are transmitted (on the NEW socket)
- CLOSE
 - Releases socket specific storage acquired by the ACCEPT call for the new socket

EZA Iterative Server Flow

- **ACCEPT Loop**
 - Accepts continue to be performed against the original socket. Anytime data is available, SEND/RECV loops are performed
- **SHUTDOWN**
 - Informs the stack that the program no longer wishes to receive data on a specific port
- **CLOSE**
 - Releases socket specific storage acquired by the original SOCKET call

EZA Iterative Server Flow

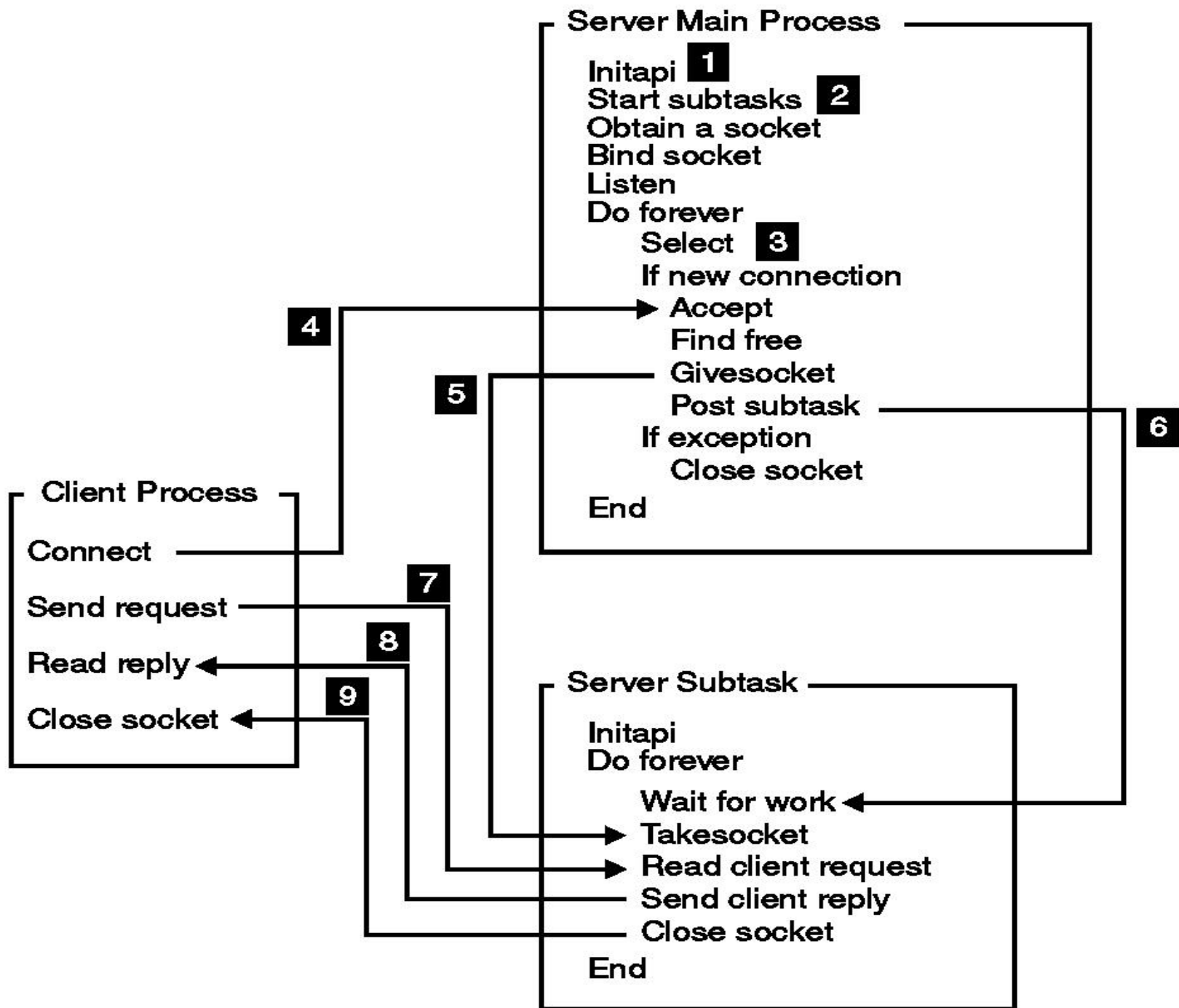
- TERMAPI (EZASMI only)
 - Releases all storage acquired by the INITAPI call

EZA Concurrent Server

- The original server continues to perform ACCEPT calls, but instead of handling any SEND/RECV calls, it transfers the socket to another program.
- This allows the original program to quickly handle many requests without being slowed by data transfers

EZA Concurrent Server

- Used to service multiple clients simultaneously
- Depends on multiple tasks
 - Main Server
 - Client Subtasks
- Connections are passed using
 - GIVESOCKET
 - TAKESOCKET



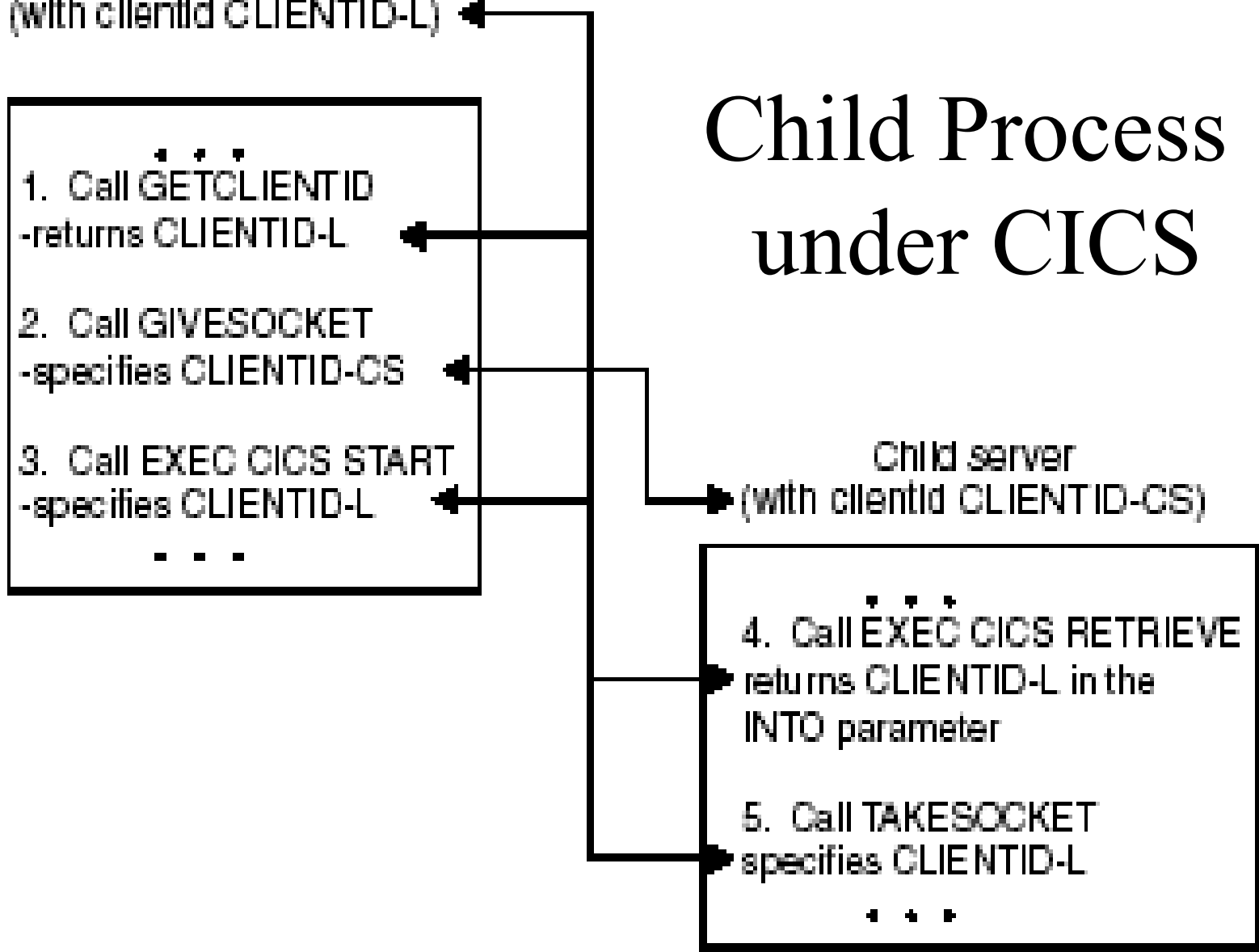
Listener
(with clientid CLIENTID-L)

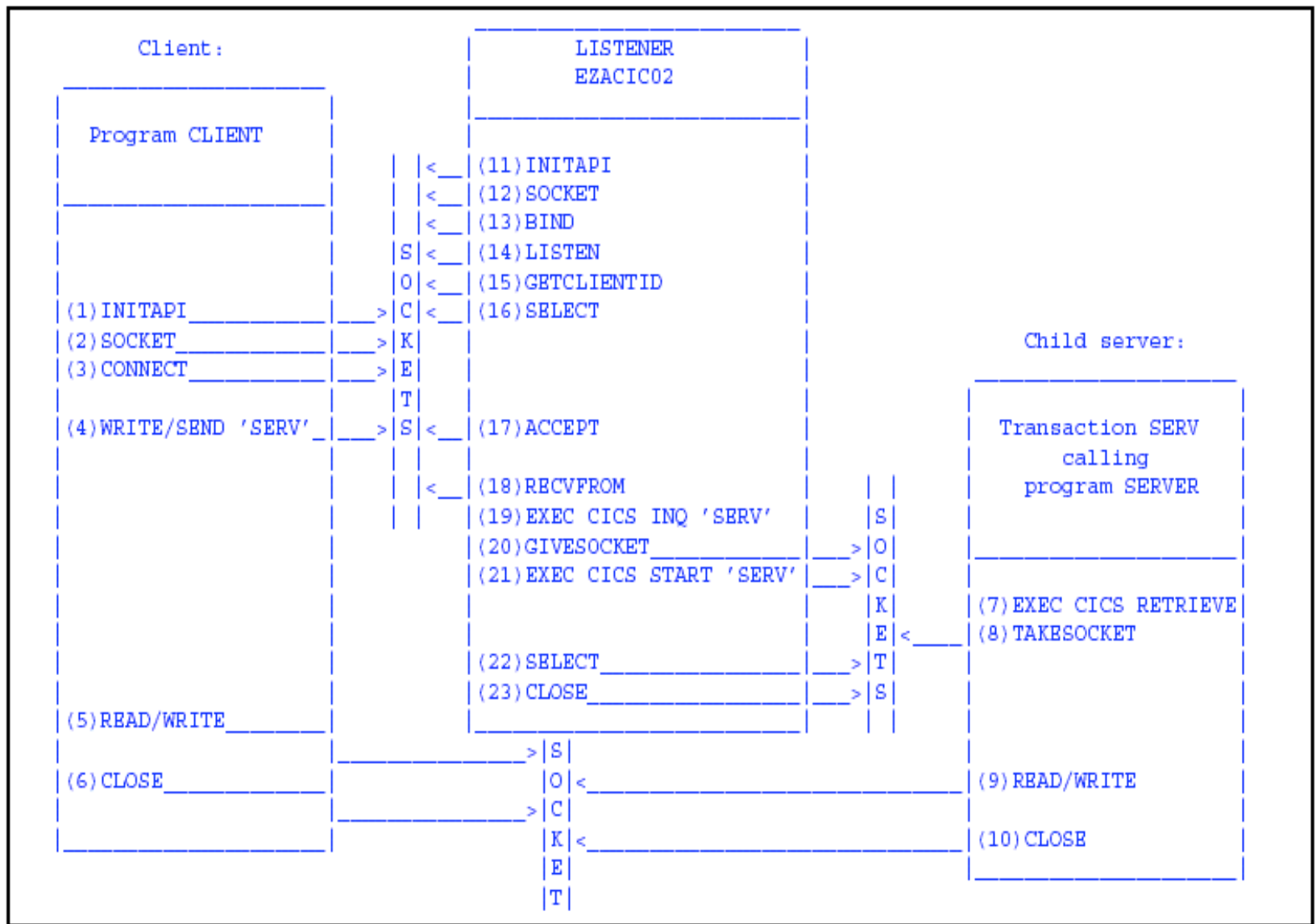
- 1. Call GETCLIENTID
-returns CLIENTID-L
- 2. Call GIVESOCKET
-specifies CLIENTID-CS
- 3. Call EXEC CICS START
-specifies CLIENTID-L
- ...

Child Process under CICS

Child server
(with clientid CLIENTID-CS)

- 4. Call EXEC CICS RETRIEVE
returns CLIENTID-L in the
INTO parameter
- 5. Call TAKESOCKET
specifies CLIENTID-L
- ...





GIVE/TAKE Restrictions

- Both processes must be using the same stack
- There is no capability to transfer between IBM and BSI applications.

Control Functions

- FCNTL
- GETHOSTBYADDR
- GETHOSTBYNAME
- GETCLIENTID
- GETHOSTID
- GETHOSTNAME
- GETPEERNAME
- GETSOCKNAME
- GETSOCKOPT
- IOCTL

SELECT Processing

- Allows a program to wait for multiple actions to occur
- SELECT
 - Wait for new ACCEPT at the same time as waiting for a GIVESOCKET to complete
 - Waiting for multiple ports
 - Waiting for timers
 - Wait for a port or a timer at the same time
- SELECTEX
 - Will also wait for an external ECB

Debugging

- IBM
 - Operator command
 - EZAAPI TRACE=ON[,PART=xx][,SYSLST]
 - Help available
 - EZAAPI ?
- BSI
 - // SETPARM IPTRACE='YYY'
 - Output is in LST queue under the partition id
 - EZALOG_{xx}
 - EZALOGF2 (example)

Other Helpful Manuals

- IBM TCP/IP for MVS: Application Programming Interface Reference
 - Version 3 Release 2
 - SC31-7187-03
 - I like this one better than the latter manual mentioned on slide 5
- TCP/IP for VSE/ESA: IBM Program Setup and Supplementary Information
 - As of VSE 2.5
 - SC33-6601-05

Other Helpful Manuals

- Redbook: A Beginner's Guide to MVS TCP/IP Socket Programming
 - GG24-2561-00
 - Although written for MVS and a little dated, it is a very good book to learn the basics.
 - Watch out for the SYNC call used after a SELECT
 - No longer needed or supported in MVS or VSE

Information

- Download this presentation, compatibility spreadsheet, and all the sample programs:
<http://www.vse2pdf.com/coolstuff>
- IBM 2000 VM/VSE Technical Conference presentation
 - TCP/IP for VSE/ESA Socket Programming
(Ingo Adlung)
 - <http://www-1.ibm.com/servers/eserver/zseries/os/vse/pdf/orlando2000/E06.pdf>

Downloads now available

- Batch
 - Server
 - Client
 - Child
- CICS
 - Listener (Server)
 - Client
 - Child
 - Starter/stopper