

# EZA Sockets User Experience



Karl De Vore – WAVV 2009

[kdevore@lakecountyil.gov](mailto:kdevore@lakecountyil.gov)



# Agenda

- Integrated Justice Evolution
- Issues
- First step – external access
- Second step – initial integration
- Web Services
- Exploration
  - CICS2WS            TCP/IP API
  - EZASOKET        SOAP
- Suggestions
- References

# Lake County

## Demographics

- Population 644,356 (2000)
  - 710,241 (2007 - up 10%)
- White persons not Hispanic 67.2%
- Hispanic or Latino 19.2%
- African American 7.0%
- Other 7.0%
- Asian 5.7%
- Two or more races 1.4%
- Median household income:
  - Lake County \$77,904 (2007)
  - Illinois \$54,141 (2007)
- Median home value:
  - Lake County \$198,200 (2000)
  - Illinois \$130,800 (2000)



# Integrated Justice Evolution

- Current system implemented in 1997
  - Originally developed in COBOL II
  - Utilizes DB2 Guest Sharing
- Original design was “Case” oriented
  - Developed specifically for the Circuit Clerk
  - Criminal Record Information Management System (CRIMS)

# Integrated Justice Evolution

- Other agencies have/had their own systems
  - Public Defender
  - States Attorney
  - County Law Enforcement (Sheriff)
  - State Police
  - Local Jurisdictions
  
- Agencies do not necessary “share” data

# Integrated Justice Evolution

- Agencies application choice dictates
  - Hardware
  - Operating System
  - Data structure
- IT provides interfaces between Agencies, Applications and data structures

# Integrated Justice Evolution

- Over the years the orientation of CRIMS was changed from a case management system to a person based system
  - The Circuit Clerk is the “owner” of the data
  - CRIMS contains information not necessarily replicated or available in other applications
  - Inquiry capabilities have been given to other agencies

# Issues...

- CRIMS – a green screen application
  - The Circuit Clerk had no desire to modify the existing application
  - Judges wanted direct access to data – *but not through a Green Screen interface*
  - Outside agencies needed to input data into the CRIMS but had no access
    - Data would be sent to the Circuit Clerk
    - Court clerks would input the data

# First step – external access

- Cold Fusion intranet application
  - Interfaced directly with the DB2 database(s)
  - Developed to satisfy the Judges’ and extended some availability to outside departments
- 3<sup>rd</sup> party middleware software was brought in for mainframe integration
  - Required no code changes to the CRIMS application
  - Is not a screen scraper
  - Is used to develop “packaged” queries and updates to the DB2 database(s) from remote (java) based applications

# Second step – initial integration

- Host Integration
  - States Attorney “package” application was replaced
  - A new system was developed in java (SAMS)
  - Solely a GUI Interface
  - States Attorney data resides in DB2 on the mainframe
    - The reliability and integrity of mainframe DB2 platform
    - Accesses both CRIMS and SAMS DB2 databases
    - Generates customized legal PDF documents and updates databases

# Web Services

- The need to further exploit and integrate data
- Take advantage of reusable services in other systems and applications
- Application to application communication
- Business to Business communication
- Seamless appearance

# Exploration

- Options:
  - CICS2WS Toolkit
  - TCP/IP API
  - EZASOCKET
  - SOAP

# CICS2WS Toolkit

- No difficulty in creating WSDLs for export
- Had difficulty with imported WSDLs from other systems:
  - Because of the complexity of some WSDLs
  - Tools used on foreign systems to create the WSDLs did not fully qualify all of the information required for CICS2WS
  - Human intervention was going to be required to manipulate imported WSDLs

# TCP/IP API

- Took a look at the TCP/IP API
  - EXEC TCPIP .....
  - Decided not to use (portability issues)

# EZASOKET programming

- Obtained sample programs from Tony Thigpen
  - For CICS and Batch
  - Had problems getting them to work
    - Could start the Server but was unable to get client apps to work
    - Examples:
      - socket being “in-use”
      - replying on wrong socket (or beyond the number of allocated sockets)
  - Attributed some of the problems potentially to the differences in socket support between BSI and CSI stacks

# EZASOKET programming

- Referred to the MVS manual:
  - *A Beginner's Guide to MVS TCP/IP Socket Programming GG24-2561-00* June 1995  
(Bookmanager format only)
  - Found simple samples
  - Copied and made minimal changes to MVS code
  - Still could not make the applications work out of the box

# EZASOKET programming

- Learning curve
  - Reviewed presentations from past WAVV, IBM and Share Conferences
  - Made extensive use of *TCP/IP for VSE/ESA – IBM Program Setup and Supplementary Information SC33-6601-08* (Chapter 10)
    - To understand the functionality of calls
    - To determine the required sequence of calls

# EZASOCKET calls

## ■ INITAPI

- Prerequisite call for most other calls

The INITAPI call connects an application to the TCP/IP interface. Almost all sockets programs that are written in COBOL, PL/I, or assembler language must issue the INITAPI macro before they issue other sockets macros.

The exceptions to this rule are the following calls, which, when issued first, will generate a default INITAPI call.

- GETCLIENTID
- GETHOSTID
- GETHOSTNAME
- SELECT
- SELECTEX
- SOCKET
- TAKESOCKET

# EZASOCKET calls

- READ – RECV – RECVFROM
  - Deciding which calls to use ?
  - Affiliated calls
    - FCNTL ?
    - IOCTL ?
  - Is the protocol TCPIP or UDP ?
  - Is the Socket Blocked or Non-Block ?
  - Is socket streaming in effect ?

# EZASOCKET calls

## ■ IOCTL

- TCPIP for VSE/ESA Setup and Supplementary Guide:

*Table 6. IOCTL Macro Arguments*

COMMAND/CODE	SIZE	REQARG	SIZE	RETARG
FIONBIO X'8004A77E'	4	Set socket mode to: X'00'=blocking; X'01'=nonblocking	0	Not used

- Cobol Program

```
*01 IOCTL-COMMAND-STRING      PIC X(16) VALUE 'FIONBIO'.  
 01 IOCTL-COMMAND-STRING      PIC X(16)  
    VALUE X'8004A77E'.  
  
*01 IOCTL-REQARG-NON-BLOCKING  PIC 9(8) BINARY VALUE 1.  
 01 IOCTL-REQARG-NON-BLOCKING  PIC 9(8) BINARY VALUE 0.
```

# EZASOCKET calls

## ■ FCNTL

Keyword	Description	REQARG						
S	Input parameter. A value or the address of a halfword binary number specifying the socket descriptor for the socket that you want to unblock or query.	A fullword binary field containing a mask that TCP/IP uses to set the FNDELAY flag. <ul style="list-style-type: none"><li>• If COMMAND is set to 3 (query) the REQARG field should be set to 0.</li><li>• If COMMAND is set to 4 (set),<ul style="list-style-type: none"><li>– Set REQARG to 4 to turn the FNDELAY flag on. This places the socket in nonblocking mode.</li><li>– Set REQARG to 0 to turn the FNDELAY flag off. This places the socket in blocking mode.</li></ul></li></ul>						
COMMAND	Input parameter. A fullword binary field or a literal that sets the FNDELAY flag to one of the following values: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>3 or 'F_GETFL'</td><td>Query the blocking mode for the socket.</td></tr><tr><td>4 or 'F_SETFL'</td><td>Set the mode to nonblocking for the socket. REQARG is set by TCP/IP.</td></tr></tbody></table> <p>The FNDELAY flag sets the nonblocking mode for the socket. If data is not present on calls that can block (READ, READV, and RECV), the call returns a -1, and ERRNO is set to EWOULDBLOCK.</p>	Value	Description	3 or 'F_GETFL'	Query the blocking mode for the socket.	4 or 'F_SETFL'	Set the mode to nonblocking for the socket. REQARG is set by TCP/IP.	
Value	Description							
3 or 'F_GETFL'	Query the blocking mode for the socket.							
4 or 'F_SETFL'	Set the mode to nonblocking for the socket. REQARG is set by TCP/IP.							

- Command 4 and REQARG 4 worked
- Command 4 and REQARG 0 did not work

# EZASOKET debugging aids

- TCP/IP for VSE/ESA V1R5.0 IBM Program Setup and supplementary Information
  - Chapter 8
    - OS/390 or z/OS EZASMI and EZASOKET Calls Supported by z/VSE
      - Table 3. Supported OS/390 Socket Calls since VSE/ESA 2.5
      - Table 4. ERRNO Values Sorted by Value & Table 5. ERRNO Values sorted by Name
  - Chapter 10
    - Using the CALL Instruction Application Programming Interface (EZASOKET API)
      - Describes description, syntax, parameters and other related information about calls

# EZASOKET outcome

- Once we got over the initial hurdles
  - We performed the following *within* the z/VSE guest:
    - Batch
      - Successfully brought up the batch server
      - Successfully tested the batch client
      - Successfully shutdown the batch server
    - CICS
      - Developed a CICS Client
      - Brought up the batch server
      - Successfully tested the CICS client against the server
      - Shutdown the batch server

# EZASOKET outcome

- Socket testing externally
  - The CICS program was modified to communicate with an external server
  - Connection to an outside server
    - Opened a foreign port successfully
    - Application failed initially due to protocol (UDP)
    - Fixed protocol issue (TCPIP)
    - Application still failed with no response from the server
    - Ran a Wireshark trace to confirm connection, failure and transmission protocol
    - Realized that there was no “Listener” on the foreign system

# EZASOKET outcome

- Based on preliminary tests, the following decisions were made:
  - Developing and modeling Cobol programs using EZASOKET calls for different scenarios could be difficult
  - Application folks did not want the responsibility of developing “Listeners” on foreign systems
  - Decided to explore the possibility of using SOAP

# SOAP in progress...

- Referenced:
  - *z/VSE e-business Connectors User's Guide*
    - Version 3.1 *SC33-8231-01*
    - Version 4.1 *SC33-8310-00*
  - z/VSE Download Page:  
<http://www-03.ibm.com/servers/eserver/zseries/zvse/downloads/>
- Defined appropriate RDO entries to CICS/TS for CWS support
  - TRANSACTIONS      PROGRAMS
  - TCPIP SERVICE

# SOAP in progress...

- Pulled down IESSOAPO from the z/VSE ftp site:

<ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/download/vsecon/iessoapo.job>

- Assembled IESSOAPO.A (SOAP Trace)

# SOAP in progress...

- Applied all available maintenance for SOAP
- Punched out the following LIBR members from PRD1.BASE:
  - IESXMLPH.H (vse XML parser)
  - IEXSMLAH.H (vse XML parser)
  - IESHTTPH.H (HTTP client interface)
  - IESSOAPH.H (vse SOAP interface)

# SOAP in progress...

- Review IESSOAPH.H
  - Good internal documentation discussing:
    - SOAP Server
    - SOAP Client
    - SOAP Interface - encoder & user program
    - SOAP Interface – decoder & user program
    - SOAP Interface – user program & SOAP client
      - Documents: *SOAP Action Codes*
      - SOAP Error Codes*

# SOAP in progress...

## ■ Status

- Developed a model program for outbound requests
  - Using standard encoder (IESSOAPE)
- In the process of developing a “handler” program that will receive all inbound requests
  - It may or may not use the standard decoder (IESSOAPD) and then will route appropriately within the CRIMS application

# Suggestions...

- CICS/TS startup deck for web services:

```
// OPTION, LOG,...,SYSPARM = '00'
```

(for TCPIP services – match stack INIT parm)

```
// LIBDEF *,SEARCH=(xxxx.TCPIPLIB,PRD2.SCEEBASE,...)
```

The sublibrary where TCPIP resides *must* precede the LE Runtime sublibrary to prevent DFHSO0117 message and CICS/TS web initialization failure

- TCP/IP requirements:

```
DEFINE NAME,NAME=IJPOC1.LAKECO.ORG,IPADDR=10.3.220.201
```

```
DEFINE NAME,NAME=LCJN.LAKECO.ORG,IPADDR=10.3.220.201
```

# Suggestions...

## ■ Debugging tools

- \$SOCKDBG.A (PRD1.BASE)
  - Configurable – see EZASOKET
- EZASOKET
  - EZAAPI Trace (Appendix C - in the Program Setup and Supplementary Information guide)

With the activation of the EZAAPI trace, the BSD-C trace of TCP/IP for VSE/ESA (called \$SOCKDBG trace) is automatically activated in addition. In all those cases where the TCP/IP socket call is passed over to the underlying TCP/IP for VSE/ESA product, this BSD-C trace produces additional messages, like

```
BSD001I IPNRCONN 01.04.00 10/25/00 22.41      01DB0590 00800000  
BSD004I CONNECTING 009.164.155.122,00000 TO 009.164.155.122,0400  
BSD002I IPNRCONN R15=00000000 RETCD=GOOD      ERRNO=NONE
```

# Suggestions...

- Debugging tools

- SOAP Trace (IESSOAPO.A)

<ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/download/vsecon/iesoapo.job>

# References

**A Beginner's Guide to MVS TCP/IP Socket Programming GG24-2561-00**

**IBM TCP/IP for MVS Application Programming Interface Reference Version 3.2**

SC31-7187-03

**TCP/IP for VSE/ESA – IBM Program Setup and Supplementary Information**

SC33-6601-08

**CICS TS for VSE/ESA, Internet Guide SC34-5765-00**

**CICS TS for VSE/ESA, Web Support and 3270 Bridge SG24-5997-00**

**CICS TS for VSE/ESA, Enhancements Guide GC34-5763-05**

**z/VSE e-business Connectors User's Guide(s)**

Version 3.1 SC33-8231-01, Version 4.1 SC33-8310-00, **Version 4.2 SC33-8310-02**

***New: (05/05/09)* How to use Web Services with z/VSE:**

<ftp://ftp.software.ibm.com/eserver/zseries/zos/vse/pdf3/HowToUseWebServicesWithzVSE.pdf>

**z/VSE Homepage:**

<http://www-03.ibm.com/servers/eserver/zseries/zvse/documentation/ebusiness.html#soap>