z/TPF V1.1

2013 TPF Users Group

# Secure Socket Layer (SSL) Hints, Tips, and Futures

Mark Gambino
Communications Subcommittee

**AIM Enterprise Platform Software**
**IBM z/Transaction Processing Facility Enterprise Edition 1.1**

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

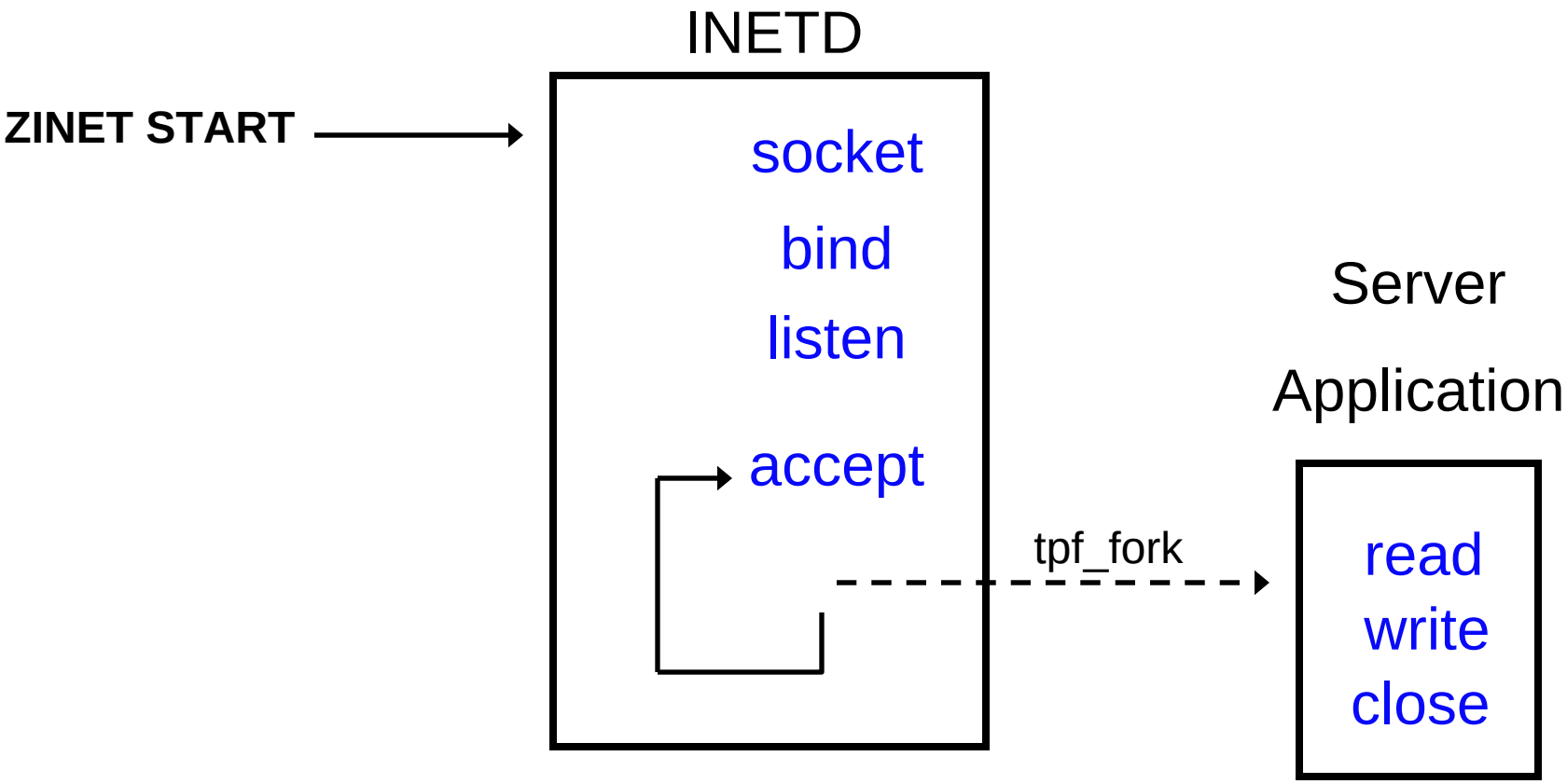
# TCP Server Application Communications Layer Logic

1. **Create the listener socket**
   - ***socket***, ***bind***, and ***listen*** APIs
2. **Accept connections from clients on the listener socket**
   - ***accept*** API
3. **Exchange data over the connected socket**
   - Send application data using the ***write*** (or ***send***) API
   - Read application data using the ***read*** (or ***recv***) API.
4. **End the TCP connected socket**
   - Issue the ***shutdown*** API (optional)
   - Issue the ***close*** API

# Internet Daemon (INETD)

- **System service that enables you to define, start, and stop your TCP/IP server applications**
  - ZINET operator commands
- **Optionally, can monitor and manage TCP listener sockets. For example:**
  - NOWAIT model – issues all the ***accept*** APIs on the listener socket to accept client connections
  - AOA2 model - issues all the ***activate_on_accept*** (AOA) APIs on the listener socket to accept client connections

# INETD TCP NOWAIT Model Overview



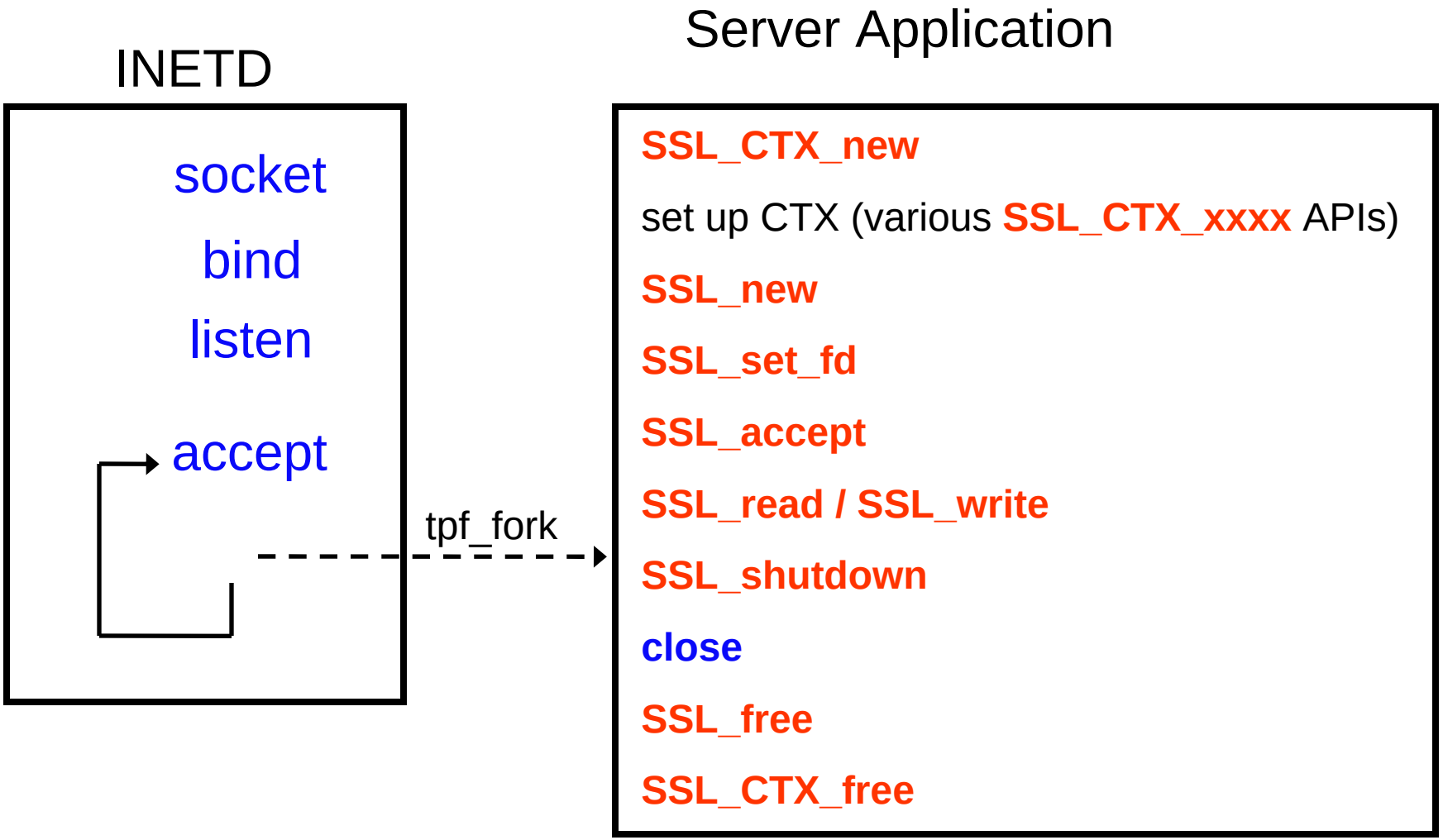
# Steps for Starting an SSL Session

1. **Create an SSL context (CTX) for this application**
   - ***SSL_CTX_new*** or ***SSL_CTX_new_shared*** API
2. **Set up the CTX with information needed to start an SSL session**
   - Ciphers supported (***SSL_CTX_set_cipher_list*** API)
   - Server certificate (***SSL_CTX_use_certificate_file*** API)
   - Private key (***SSL_CTX_use_PrivateKey_file*** API)
3. **Create an SSL structure and assign it to the connected socket**
   - ***SSL_new*** and ***SSL_set_fd*** APIs
4. **Start an SSL session over the connected socket**
   - ***SSL_accept*** API

# After an SSL Session Started

- **To exchange data across an SSL session**
  - Replace ***read*** APIs with ***SSL_read*** APIs
  - Replace socket ***activate_on_receipt (AOR)*** APIs with ***SSL_aor*** APIs
  - Replace ***write*** APIs with ***SSL_write*** APIs
- **To end an SSL session**
  1. Shutdown the SSL session (***SSL_shutdown*** API)
  2. End the connected socket (***close*** API)
  3. Return the SSL structure (***SSL_free*** API)
  4. Return the CTX structure (***SSL_CTX_free*** API) if no more SSL sessions are using this CTX

# INETD TCP NOWAIT Model with Sample Server Application Using SSL

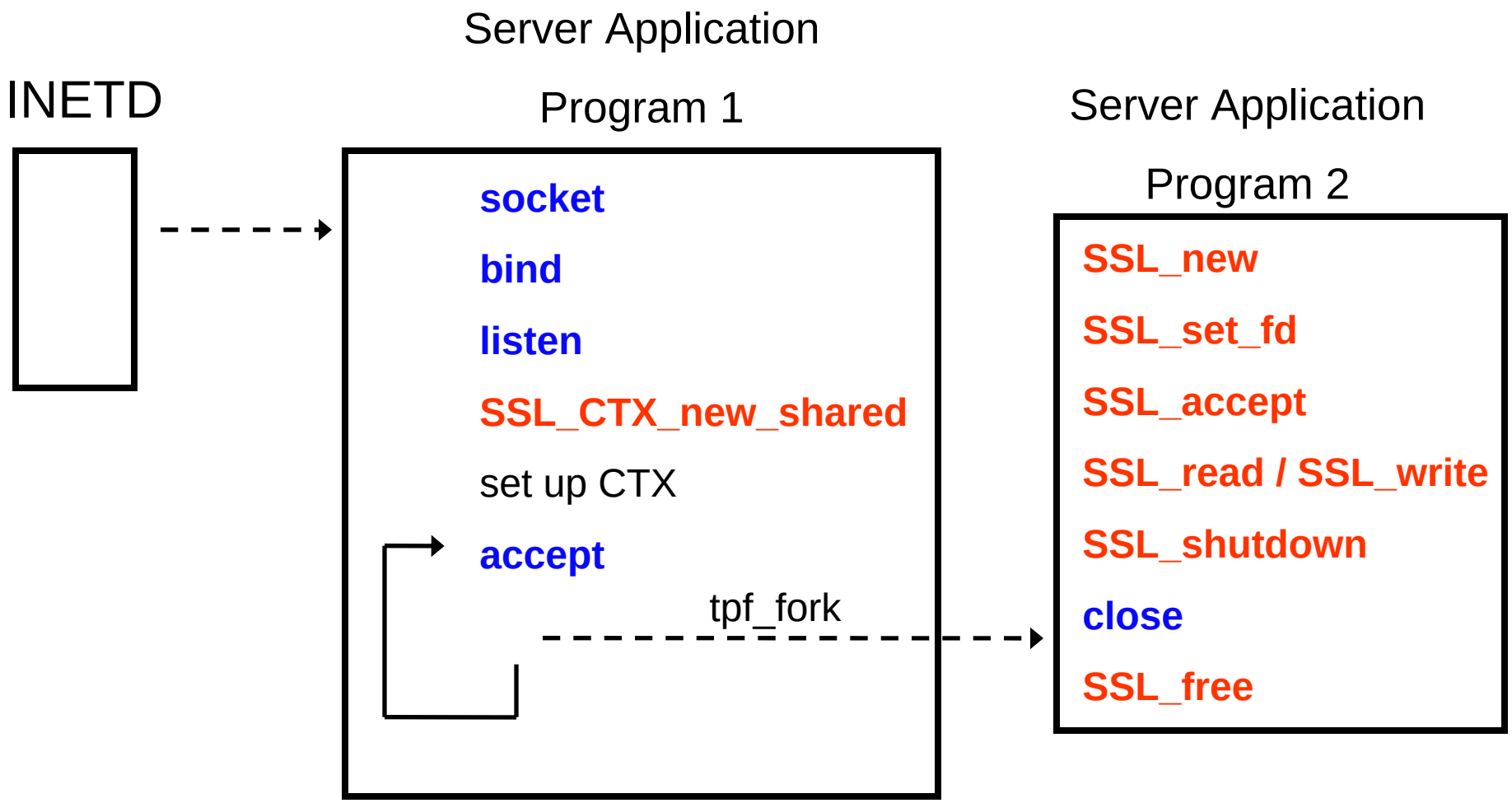


# INETD TCP NOWAIT Model with Sample Server Application Using SSL

**This works, but…**

- **All of the SSL logic is in the server application**
- **Inefficient for short lived sessions**
  - A new CTX is created and set up for each SSL session
- **Server application program changes may be required if:**
  - Ciphers supported by the server change
  - A new certificate, private key, or both are assigned to this server application

# INETD DAEMON Model with Sample Server Application Using SSL



# INETD DAEMON Model with Sample Server Application Using SSL

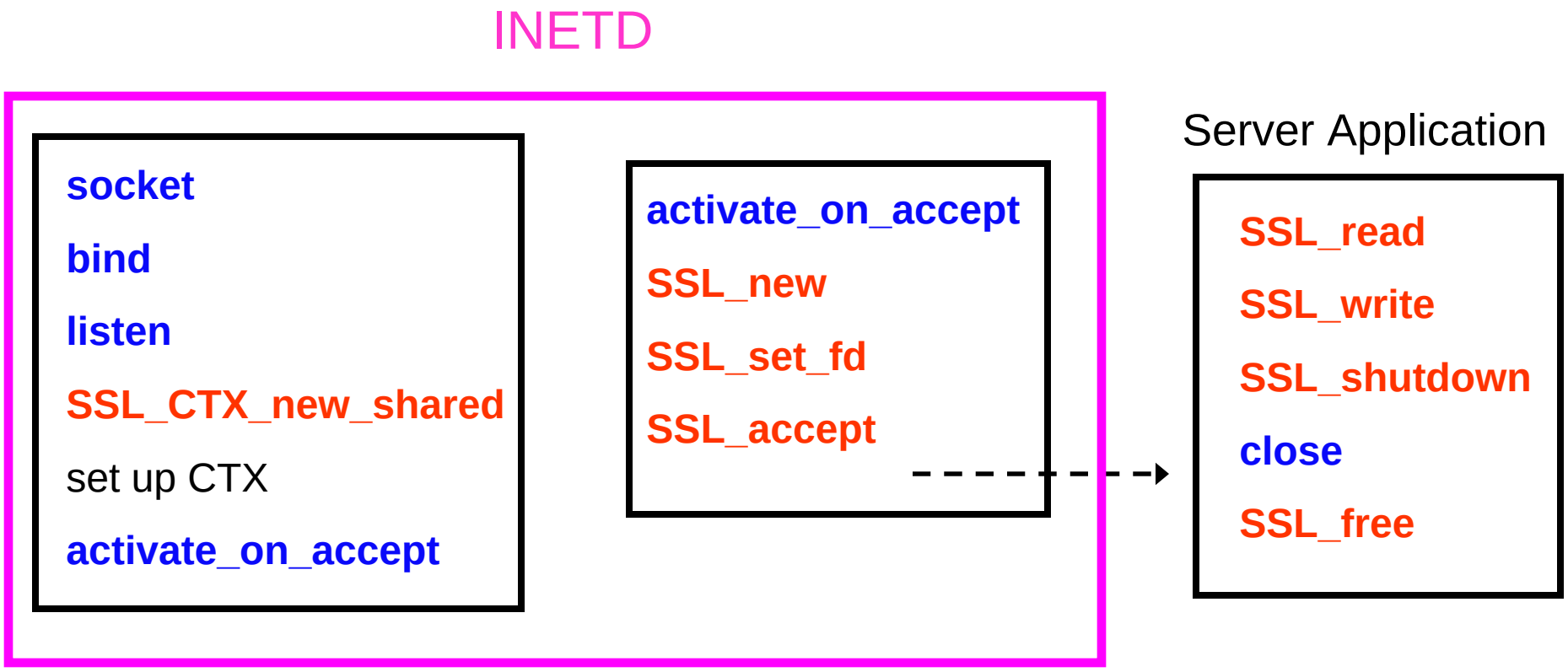
**This works, but…**

- **All of the listener socket logic is in the server application**
- **All of the SSL logic is in the server application**
- **Having one CTX shared by all SSL sessions for this application is higher performance, but…**
  - Need to have recovery logic in the server application for the case where the shared SSL daemons recycle and a new CTX is needed
- **Server application program changes may be required if:**
  - Ciphers supported by the server change
  - A new certificate, private key, or both are assigned to this server application

# Proposed New INETD Model for SSL

- **INETD would:**
  - Create and manage the listener socket
  - Create a single CTX that would be used by all SSL sessions using this server application
  - Create SSL structures
  - Start SSL sessions
  - Automatically restart the listener socket if it fails
  - Automatically create a new CTX if the shared SSL daemons recycle
- **Server application would:**
  - Send and receive data over the SSL session
  - Shutdown the SSL session and close the connected socket

# Proposed New INETD Model for SSL



# SSL Configuration Information for Proposed New INETD Model for SSL

- **SSL configuration information for the server application would be defined using the existing *Application Configuration Files for SSL* mechanism**
- **Sample configuration file for SSL:**

```
USESSL=YES
CIPHER=DES-CBC3-SHA
VERIFYPEER=NO
CERTIFICATE=/certs/tpfprodcert.pem
CERTTYPE=PEM
KEY=/tpfpubk/keypair1.pem
KEYTYPE=PEM
VERSION=TLSV1
```

# New User Exit for Proposed New INETD Model for SSL

- **INETD would call a new user exit when an SSL model application is starting or stopping**
- **ZINET START:**
  - User exit will allow you to initialize tables that are used by this application
  - User exit will be called before any SSL sessions are started
- **ZINET STOP:**
  - User exit will allow you to clean up tables that are used by this application

# Secure HTTP Server

- **z/TPF HTTP Server Support (APARs PJ39252 and PJ39550) was delivered on PUT 9**
  - TPFUG requirement "SOA00002 – *Lightweight HTTP Server for SOAP Messaging*"
  - Subset of the HTTP to be an efficient message transport
- **IBM is currently looking at enhancing this support to provide z/TPF Secure HTTP Server Support**
  - HTTPS protocol (HTTP over SSL)
  - Would use the proposed new INETD model for SSL

# Other Hints and Tips for SSL

# Use Application Configuration Files for SSL

- **To avoid having to make application program changes when parameters for your SSL sessions change, create application configuration files for SSL applications**
- **Server application can use the tpf_SSL_getConfig function to parse a configuration file and put all the necessary parameters into a C structure that the application program can then use when issuing SSL_CTX_xxxxx APIs**

# Use Shared CTXs

- **Create one CTX per server application and use/share it for all SSL sessions with that application**
  - Especially for short lived SSL sessions
- **Setting up a CTX requires file system I/O and parsing that you want to do once per application rather than once per session**

# Use z/TPF PKI Support

- **Create RSA key pairs on z/TPF using PKI support**
  - **ZPUBK GENERATE KEYPAIR-KEYPAIR1 CIPHER-RSA1024**
- **Input to the *SSL_CTX_use_PrivateKey_file* API includes a pointer to the file name containing the private key**
  - Set the file name to **/tpfpubk/keypair1.pem**
    - Values starting with **/tpfpubk** tells z/TPF that what follows is the name of an RSA key pair rather than the name of a file containing the key pair
  - If using application configuration files for SSL, code this on the KEY= statement
    - **KEY=/tpfpubk/keypair1.pem**

# Use ZSSLD DISPLAY Information to Tune Shared SSL (Sample ZSSLD DISPLAY Output)

```
SSLD0007I 15.52.02 CSSLZD - SSL STATISTICAL INFORMATION

                      LAST MINUTE  HIGH WATER MARK
                      -----------  ---------------
  SESSIONS STARTED             39              39
  SSL_WRITES ISSUED         96769          104419
  SSL_READS ISSUED          96730          104386
  MEGABYTES SENT            46.14           49.79
  MEGABYTES RECEIVED        46.12           49.77

   SSL     ACTIVE   MAX ACT  CURRENT  MAX      HEAP      AVAIL   MAX HEAP
  DAEMON   THREADS  THREADS   SESS    SESS   IN USE      HEAP      IN USE
  ------   -------  -------  -------  -----  --------  --------  --------
     1        1        2        12      14       1.53     28.53      1.69
     2        0        2        12      14       1.54     28.52      1.70
     3        1        3        10      14       1.62     28.44      1.84
     4        0        2         9      14       1.32     28.74      1.73

END OF DISPLAY
```

# Tuning Memory Limits for Shared SSL

- **Shared SSL uses ECB heap for all control block structures**
  - CTX, SSL, certificates, data buffers, and so on
- **For each SSL daemon, ZSSLD DISPLAY shows the current amount of ECB heap in use, the amount of ECB heap available, and the maximum amount of ECB heap that was in use**
  - Message SSLD0055A occurs whenever an SSL daemon uses 80% of its ECB heap
- **If SSL daemon(s) are running low on ECB heap, do one of the following:**
  - Increase the amount of ECB heap a given process can use (ZCTKA ALTER EMPS and ZCTKA ALTER MMES)
  - Define more SSL daemon processes (ZNKEY SSLPROC) and spread the sessions across all processes to reduce the load on each individual process

# Increasing Throughput of Shared SSL

- **As your use of shared SSL increases, you may need to increase the number of threads available to process APIs on shared SSL sessions**
- **If ZSSLD DISPLAY consistently shows that the number of active threads in SSL daemon(s) is at or near the number of threads defined, do one or both of the following:**
  - Increase the number of threads per SSL daemon process (ZNKEY SSLTHRD)
  - Define more SSL daemon processes (ZNKEY SSLPROC) and spread the sessions across all processes to reduce the load on each individual process

# Trademarks

- **IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.**
- ***(Include any special attribution statements as required – see Trademark guidelines on https://w3-03.ibm.com/chq/legal/lis.nsf/lawdoc/5A84050DEC58FE31852576850074BB32?OpenDocument#Developing%20the%20Special%20Non-IBM%20Tr)***

**Notes**

- **Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.**
- **All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.**
- **This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.**
- **All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.**
- **Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.**
- **Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.**
- **This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.**