z/TPF High Speed Connector and Enhanced HTTP Client

Jamie Farmer
z/TPF Development

High Speed Connector
What is it?

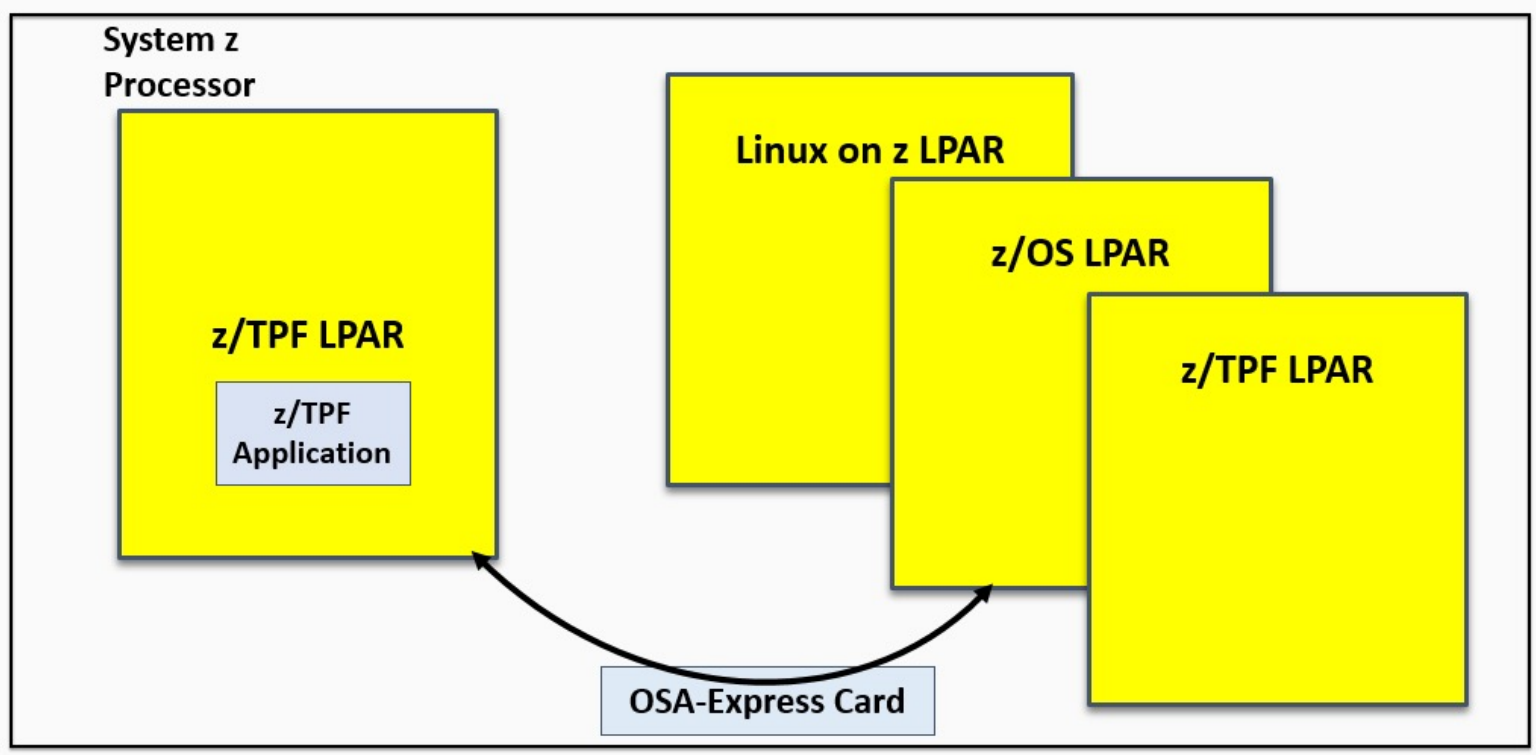
- z/TPF applications can send messages to remote servers efficiently and without knowledge of the connections between z/TPF and the remote servers.

- Could use standard middleware, but many times it is too heavy compared to the request processing.

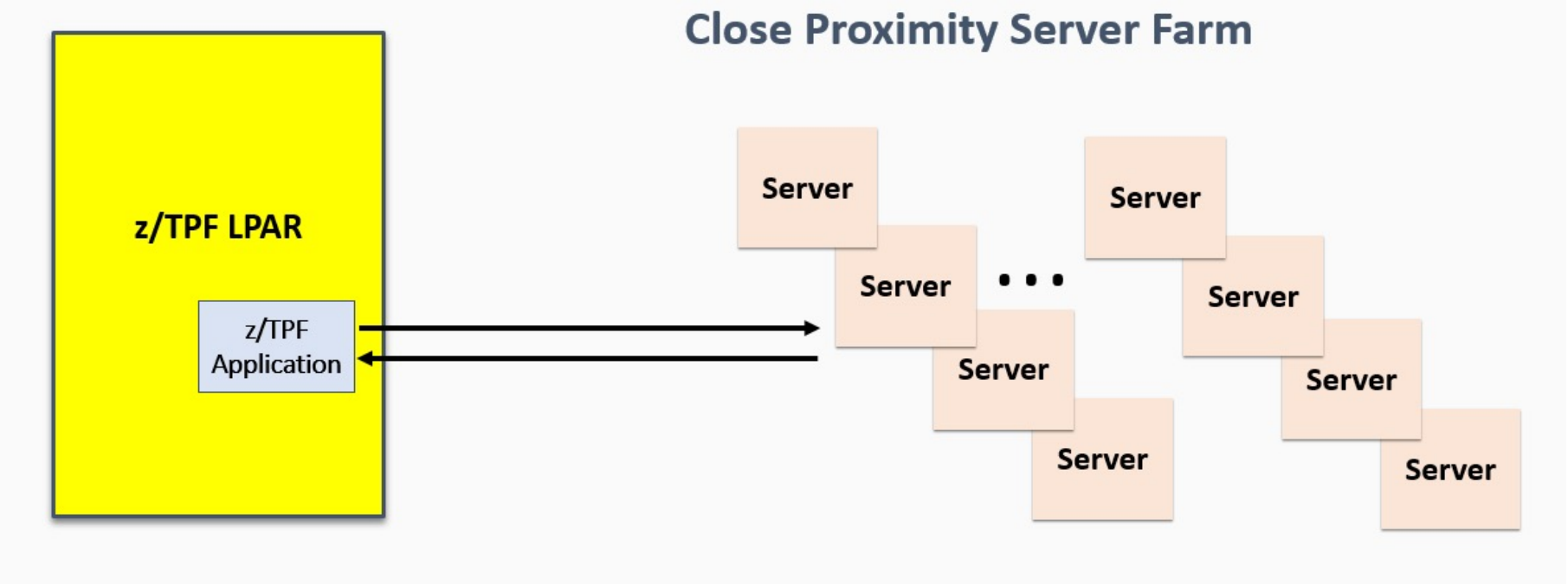
- z/TPF application would need to handle the following
  - Load balancing of servers
  - Primary and Fallback scenarios
  - Error Handling
  - Managing pools of persistent connections
  - Queueing when no servers are available

- Changing the topology or number of servers may require application updates.

High Speed Connector
z/TPF Communicating With Other LPARs in CEC



High Speed Connector
z/TPF Communicating With a Server Farm



High Speed Connector
How Does it Work?

- Through configuration, an administrator can define groups of servers
- From an application
  - Send a request to a “group”
- The High Speed Connector processing handles
  - Load balancing requests across the group of servers
  - Ability to define servers as primary and backup (only used when primary is not available)
  - Error Handling and automatic session establishment
  - Ability to display statistics and provide management of endpoints
    - Handle maintenance on any one server non-disruptively.
  - Queueing requests when no servers are available
  - Dynamic increase of connections to endpoint or number of endpoints is immediate and non-disruptive.

High Speed Connector
Defining An Endpoint Group

```
<tns:EndpointGroup
  <tns:Endpoint>
     <tns:endpointName>PRCRYP1</tns:endpointName>
     <tns:role>PRIMARY</tns:role>
     <tns:destination>remHost.ibm.com:15000</tns:destination>
     <tns:startSocket>25</tns:startSocket>
     <tns:maxSocket>100</tns:maxSocket>
  </tns:Endpoint>
  <tns:Endpoint>
     <tns:endpointName>BKCRYP1</tns:endpointName>
     <tns:role>BACKUP</tns:role>
     <tns:destination>9.57.13.155:15000</tns:destination>
     <tns:startSocket>25</tns:startSocket>
     <tns:maxSocket>100</tns:maxSocket>
  </tns:Endpoint>
  <tns:groupName> CRYPSVR1 </tns:groupName>
  <tns:qMaxDepth>400</tns:qMaxDepth>
  <tns:qThreshold>45</tns:qThreshold>
  <tns:syncTimeout>500</tns:syncTimeout>
  <tns:heartbeatInterval>300</tns:heartbeatInterval>
</tns:EndpointGroup>
```

TPF Endpoint Definitions
- Name
- Primary/Backup
- Host/Port
- Starting/Max Sockets

TPF Group Definitions
- Name
- Queue Depth
- Queue Threshold
- Sync Timeout
- Heartbeat Interval

***Endpoint descriptor is loaded using standard loaders**

High Speed Connector

Supported Commands for High Speed Connector

- ZCONN START GROUP-ept_grp [ENDPOINT-ept]
- ZCONN STOP GROUP-ept_grp [ENDPOINT-ept]
- ZCONN QUIESCE GROUP-ept_grp ENDPOINT-ept
- ZCONN DISPLAY (ALL|GROUP-ept_grp)
- ZCONN STATS GROUP-ept_grp
- ZCONN MAXSTATS GROUP-ept_grp
- ZCONN CLEARSTATS (ALL|GROUP-ept_grp)
- ZCONN CONFIG GROUP-ept_grp
- ZCONN TOPOLOGY GROUP-ept_grp

Introduced in July 2020

High Speed Connector
Displaying Group Information

```
User:   ZCONN DISPLAY GROUP-CRYPSVR1

System: CONN0020I 11.13.59 ENDPOINT GROUP DISPLAY

        CURRENT QUEUE SIZE    -          0
        QUEUE HIGH WATER MARK -          0
        MAX QUEUE ALLOWED     -        400

        SERVER
        ENDPOINT  ROLE STATUS SESSIONS MAXSESS INUSE APIS/SEC API TIME TIMEOUTS ERRORS
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        PRCRYP1   PRIM ACTIVE       32     100    27      882    1.133        0      0
        BKCRYP1   BACK ACTIVE       25     100     0        0    0.000        0      0
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        TOTALS                      57     200    27      882    1.133        0      0

        END OF DISPLAY
```


High Speed Connector
Displaying Group Statistics

```
User:   ZCONN STATS GROUP-CRYPSVR1

System: CONN0022I 13.15.17 ENDPOINT GROUP STATS

        SERVER                  APIS/
        ENDPOINT        APIS      SEC API TIME TIMEOUTS ERRORS
        --------- ---------- -------- -------- -------- ------
        PRCRYP1      4567890      882    1.133        0      0
        BKCRYP1            0        0        0        0      0
        --------- ---------- -------- -------- -------- ------
        TOTALS       4567890      882    1.133        0      0

        END OF DISPLAY
```


High Speed Connector
Overloading Remote Endpoints

```
User:   ZCONN DISPLAY GROUP-CRYPSVR1

System: CONN0020I 11.13.59 ENDPOINT GROUP DISPLAY

        CURRENT QUEUE SIZE    -        20
        QUEUE HIGH WATER MARK -        29
        MAX QUEUE ALLOWED     -        400

        SERVER
        ENDPOINT  ROLE STATUS SESSIONS MAXSESS INUSE APIS/SEC API TIME TIMEOUTS ERRORS
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        PRCRYP1   PRIM ACTIVE      100     100   100      877    1.139        0      0
        BKCRYP1   BACK ACTIVE      100     100   100      876    1.141        0      0
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        TOTALS                     200     200   200     1753    1.140        0      0

        END OF DISPLAY
```


High Speed Connector
Increasing Group Capacity

- Update the group's endpoint group descriptor
- Load the file through the version control file system
- New endpoints in the group or increasing maximum sockets will automatically take effect
- No application changes required.

High Speed Connector
Adding Endpoints to an Endpoint Group

```
<tns:EndpointGroup
  <tns:Endpoint>
     <tns:endpointName>PRCRYP1</tns:endpointName>
     <tns:role>PRIMARY</tns:role>
     <tns:destination>remHost.ibm.com:15000</tns:destination>
     <tns:startSocket>25</tns:startSocket>
     <tns:maxSocket>100</tns:maxSocket>
  </tns:Endpoint>
  <tns:Endpoint>
     <tns:endpointName>PRCRYP2</tns:endpointName>
     <tns:role>PRIMARY</tns:role>
     <tns:destination>remHost2.ibm.com:15000</tns:destination>
     <tns:startSocket>25</tns:startSocket>
     <tns:maxSocket>150</tns:maxSocket>
  </tns:Endpoint>
  <tns:Endpoint>
     <tns:endpointName>BKCRYP1</tns:endpointName>
     <tns:role>BACKUP</tns:role>
     <tns:destination>9.57.13.155:15000</tns:destination>
     <tns:startSocket>25</tns:startSocket>
     <tns:maxSocket>100</tns:maxSocket>
  </tns:Endpoint>
  <tns:groupName> CRYPSVR1 </tns:groupName>
  <tns:qMaxDepth>400</tns:qMaxDepth>
  <tns:qThreshold>45</tns:qThreshold>
  <tns:syncTimeout>500</tns:syncTimeout>
  <tns:heartbeatInterval>300</tns:heartbeatInterval>
</tns:EndpointGroup>
```

New Primary Endpoint

- **Name**
- **Primary/Backup**
- **Host/Port**
- **Starting/Max Sockets**

High Speed Connector
Statistics After Adding Endpoints

```
User:   ZCONN DISPLAY GROUP-CRYPSVR1

System: CONN0020I 11.13.59 ENDPOINT GROUP DISPLAY

        CURRENT QUEUE SIZE     -          0
        QUEUE HIGH WATER MARK  -         29
        MAX QUEUE ALLOWED      -        400

        SERVER
        ENDPOINT  ROLE STATUS SESSIONS MAXSESS INUSE APIS/SEC API TIME TIMEOUTS ERRORS
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        PRCRYP1   PRIM ACTIVE      100     100    88      887    1.121        0      0
        PRCRYP2   PRIM ACTIVE      123     150    97      888    1.139        0      0
        BKCRYP1   BACK ACTIVE      100     100     0        0    1.141        0      0
        --------- ---- ------ -------- ------- ----- -------- -------- -------- ------
        TOTALS                     323     350   185     1775    1.134        0      0

        END OF DISPLAY
```


High Speed Connector
Invoking the Send Message API

```
#include <tpf/tpfapi.h>

hsc conn_parms;                                  ← Structure that contains the parameters for tpf_send_message
char endpoint_in[9], endpoint_out[9];
char* endpoint_group_name = "CRYPSVR1";                  ← The group name defined in Endpoint Descriptor
connector_parms.version              = HSC_VERSION_1;    ← The version of the API to use.
connector_parms.endpointGroup = endpoint_group_name;     ← The name of the endpoint group to send a message to.
connector_parms.request              = request_buffer;   ← Buffer where the request message is stored.
connector_parms.timeout              = 2000;             ← Time in ms until the API times out
connector_parms.resp_len            = 256;               ← Length of the response.  0 if no response expected
connector_parms.response           = malloc(256);        ← Allocating storage for the response buffer.
connector_parms.endpoint_in       = endpoint_in;         ← Specific endpoint to send a message to
connector_parms.endpoint_out     = endpoint_out;         ← Endpoint that a message was sent to.
int rc;

if ((rc = tpf_send_message(&conn_parms))!=TPF_SEND_MESSAGE_OK)
    printf("error");
else
    printf("success");
```

High Speed Connector
tpf_send_async_message Example

Example Issuing Request

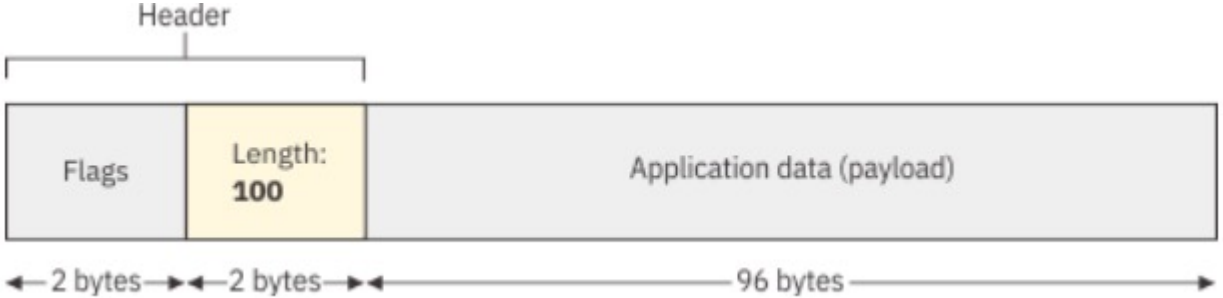
```
#include <tpf/connapi.h>

hsc connector_parms;
t_asyncRequestParms asyncParms;

char *endpoint_group, *request_buffer;
char endpoint_in[9], endpoint_out[9];
int rc;
…
connector_parms.version         = HSC_VERSION_1;
connector_parms.endpointGroup   = endpoint_group;
connector_parms.request         = request_buffer;
connector_parms.timeout         = 1000;
connector_parms.response        = NULL;
connector_parms.resp_len        = 0;
connector_parms.endpoint_in     = NULL;
connector_parms.endpoint_out    = NULL;

memset (asyncParms, 0, sizeof(asyncParms));
memcpy (asyncParms.asyncProgram,"QHSA", sizeof("QHSA"));
asyncParms.asyncData = NULL;
asyncParms.asyncDataLength = 0;

rc = tpf_send_async_message(&conn_parms, &asyncParms, 0);
if (rc != TPF_SEND_MESSAGE_OK)
   printf("error");
else
   printf("success");
```

Example Response Program

```
void QHSA(t_asyncResponse *asyncParams) {
        …
}
```

Configurable Header Support – February 2020

- When configurable header support is not defined, the z/TPF application is required to use the IBM proprietary HSC header in front of every message.
  - This inhibits the use of HSC for existing servers that require their own header format.
- A user can define the format of the header in the endpoint configuration file.



The first 2 bytes are flag bytes while the last 2 bytes of the header contain the length of the entire message.

- The <lenOffset> element is set to a value of 2, which specifies that the location of the length field is 2 bytes from the start of the message.
- The <lenFieldLen> element is set to a value of 2, which specifies that the length of the length field is 2 bytes.

Allows HSC to communicate with any server in your enterprise that supports TCP/IP!

High Speed Connector
Summary

- Complexity of z/TPF applications communicating with remote servers is greatly reduced.
- Dynamic increase of capacity as workload increases
- Allows for monitoring and management of endpoint groups and the associated connections
- z/TPF High Speed Connector code is TE-Eligible!
- High Speed Connector Starter Kit Available

  - Contains sample endpoint group descriptor files, remote server application code, z/TPF driver code to send high speed connector messages.

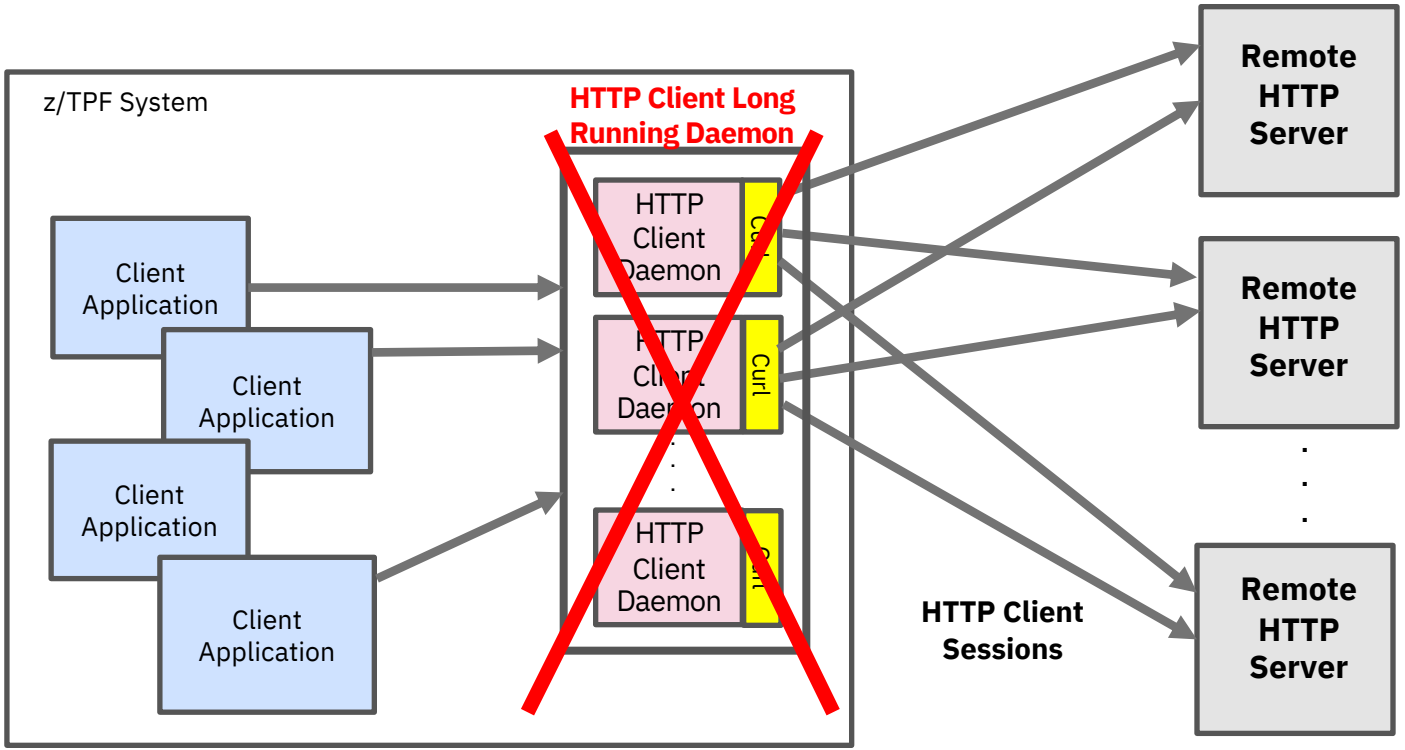
    http://www-01.ibm.com/support/docview.wss?uid=swg24043067

Agenda

- High Speed Connector
- Enhanced HTTP Client

Enhanced HTTP Client

The “original” HTTP Client

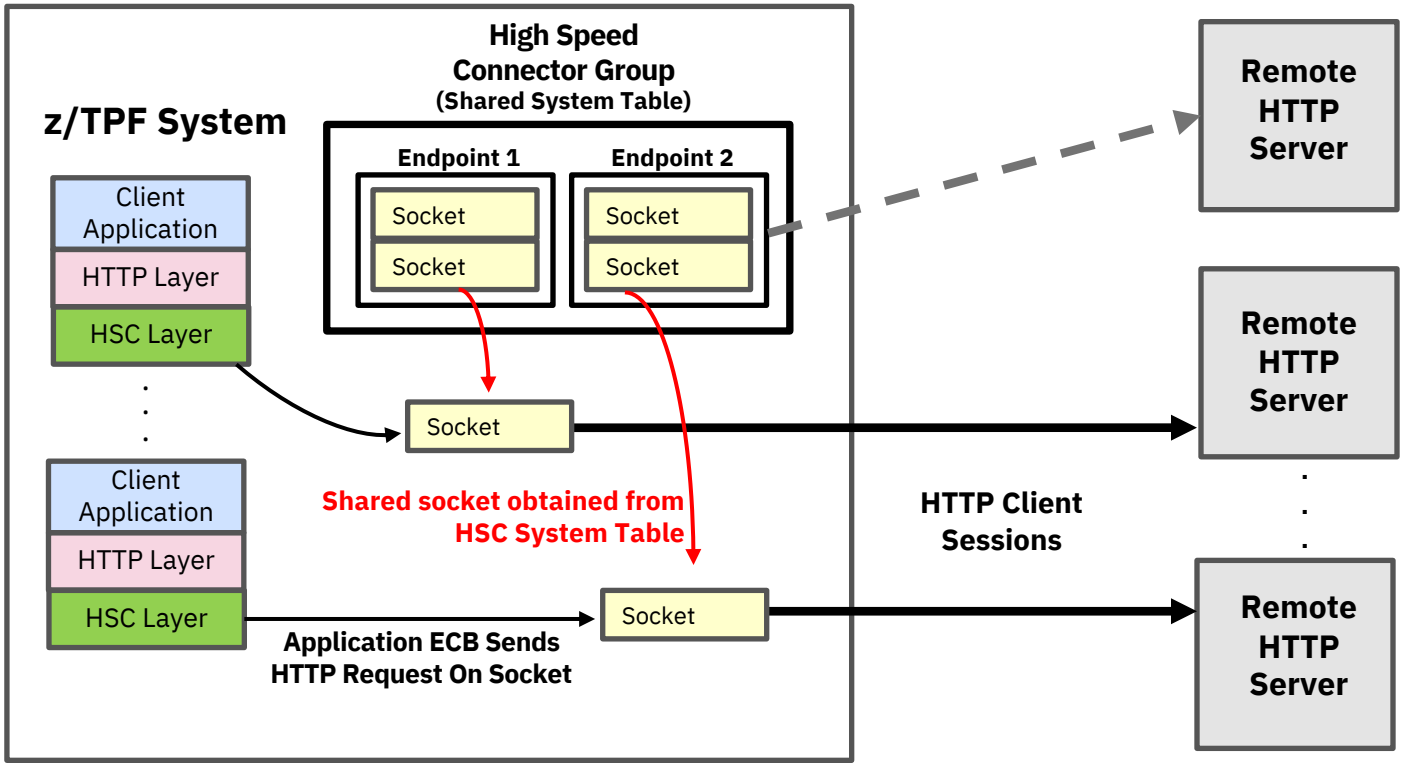


Problems with Original HTTP Client

- Long Running Daemons are not automatically recycled.
- Inefficient model to pass data across processes.
- Current libCurl package does not support the latest TLS standards.
- REST consumer would require significant changes to original HTTP client support.

Enhanced HTTP Client

Enhanced HTTP Client Architecture



Enhanced HTTP Client

- No long running daemons
- No longer passing data across processes
- Leveraging high speed connector

Enhanced HTTP Client
Terminology

HTTP Client Session Types

HTTP Client Persistent Sessions

Long running HTTP client sessions that can be shared by any z/TPF application ECBs and managed by the z/TPF high speed connector.

HTTP Client Non-Persistent Sessions

HTTP client session that is established and subsequently torn down after processing a single request.

HTTP Client API Requests

Synchronous HTTP Client Requests

The application ECB issuing the HTTP client request does not receive control back until the response is received (or a timeout occurs)

Asynchronous HTTP Client Requests

The application ECB issuing the HTTP client request can exit and a new application ECB is created when the response is received.

Enhanced HTTP Client
tpf_httpSendRequest Format

```
LIBS := CHTE
#include <tpf/c_https.h>

int tpf_httpSendRequest(char *host, t_httpClientRequest *requestParms,
                        tpf_httpsvr_resp **response, t_httpClientConnect *connectParms,
                        int options);
```

host: Host or IP address to establish session to
requestParms: A t_httpClientRequest structure containing the HTTP client request information
- **HTTP Version:** only supports HTTP 1.1
- **Request Type:** GET, HEAD, PUT, POST, DELETE
- **uri:** The service on a host to be accessed
- **timeout:** How long to wait to send request and receive response (in milliseconds)
- **headers:** User defined headers to include in request
- **body:** User defined body to send to remote (Ignored if not PUT / POST)

response: Pointer to an tpf_httpsvr_resp structure containing the response
connectParms: Non-persistent connect options (ignored for persistent sessions)
options: Not used – will be used for future extensions

Enhanced HTTP Client
tpf_httpSendAsyncRequest Format

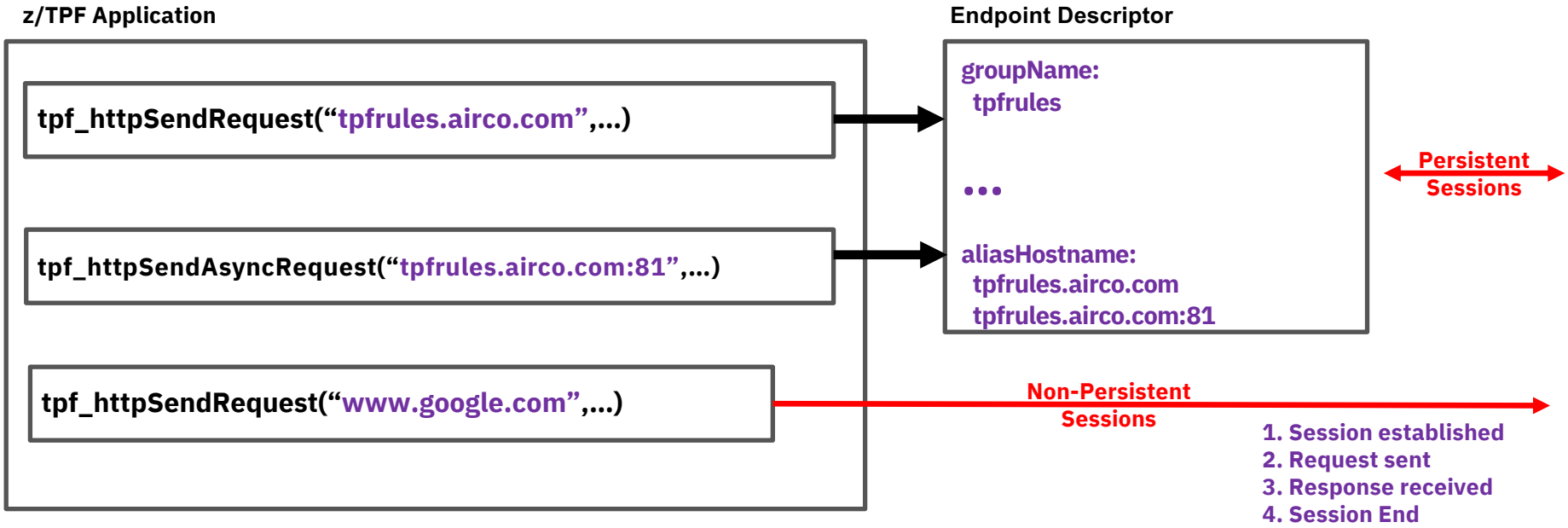
```
LIBS := CHTE
#include <tpf/c_https.h>

int tpf_httpSendAsyncRequest(char *host, t_httpClientRequest *requestParms,
                tpf_httpsvr_resp **response, t_asyncRequestParms *asyncParms,
                t_httpClientConnect *connectParms,  int options);
```

asyncParms: The asynchronous parameters for the request containing the program to invoke when the response is received as well as user data to pass.

options: TPF_HTTP_KEEP_REQUEST option to save original HTTP request across ECBs during the asynchronous application call

Enhanced HTTP Client

Persistent vs Non-Persistent Connections

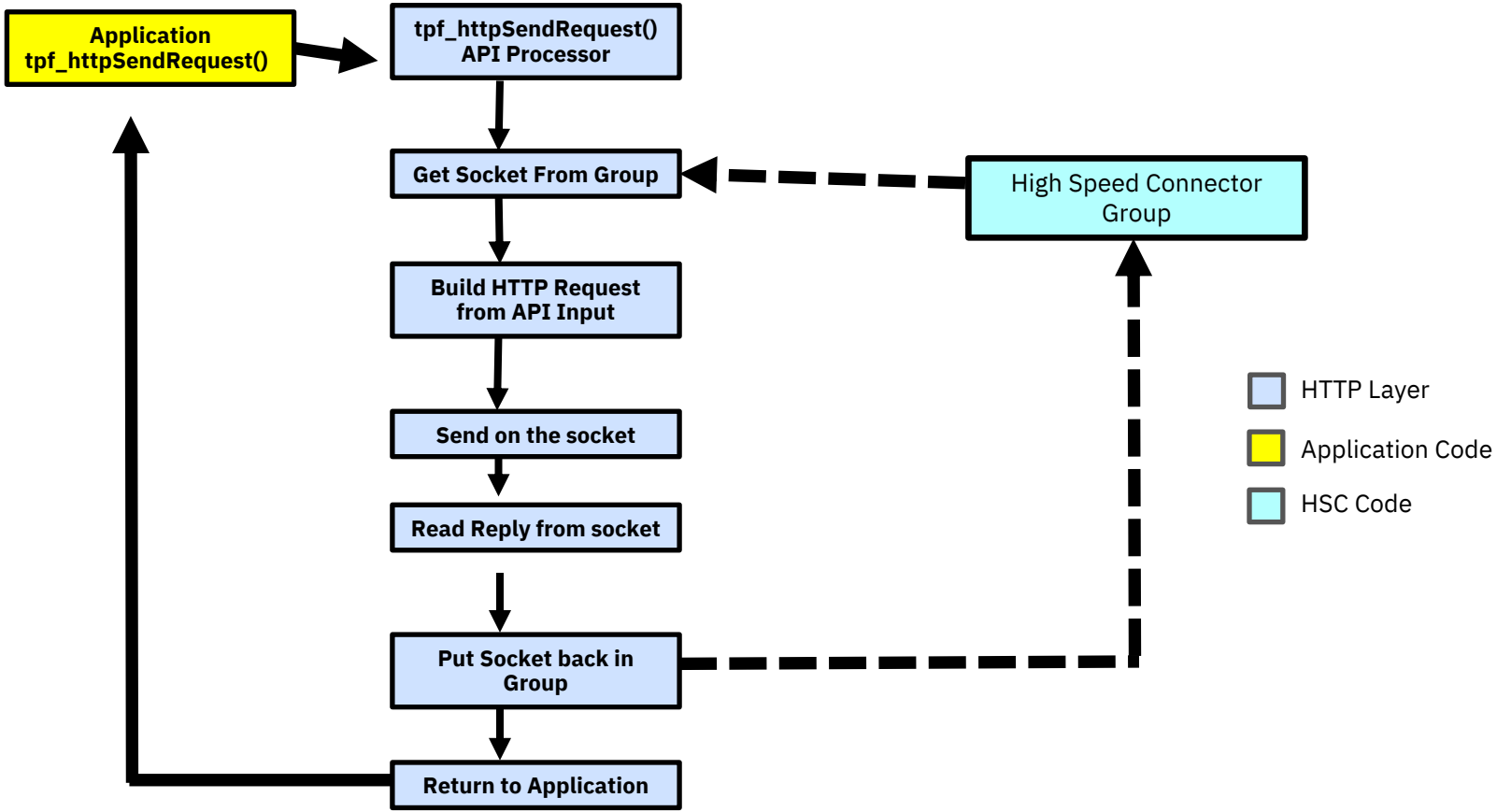


Persistent vs Non-Persistent sessions are transparent to the application

- Allows an administrator switch from non-persistent to persistent sessions with no application changes!

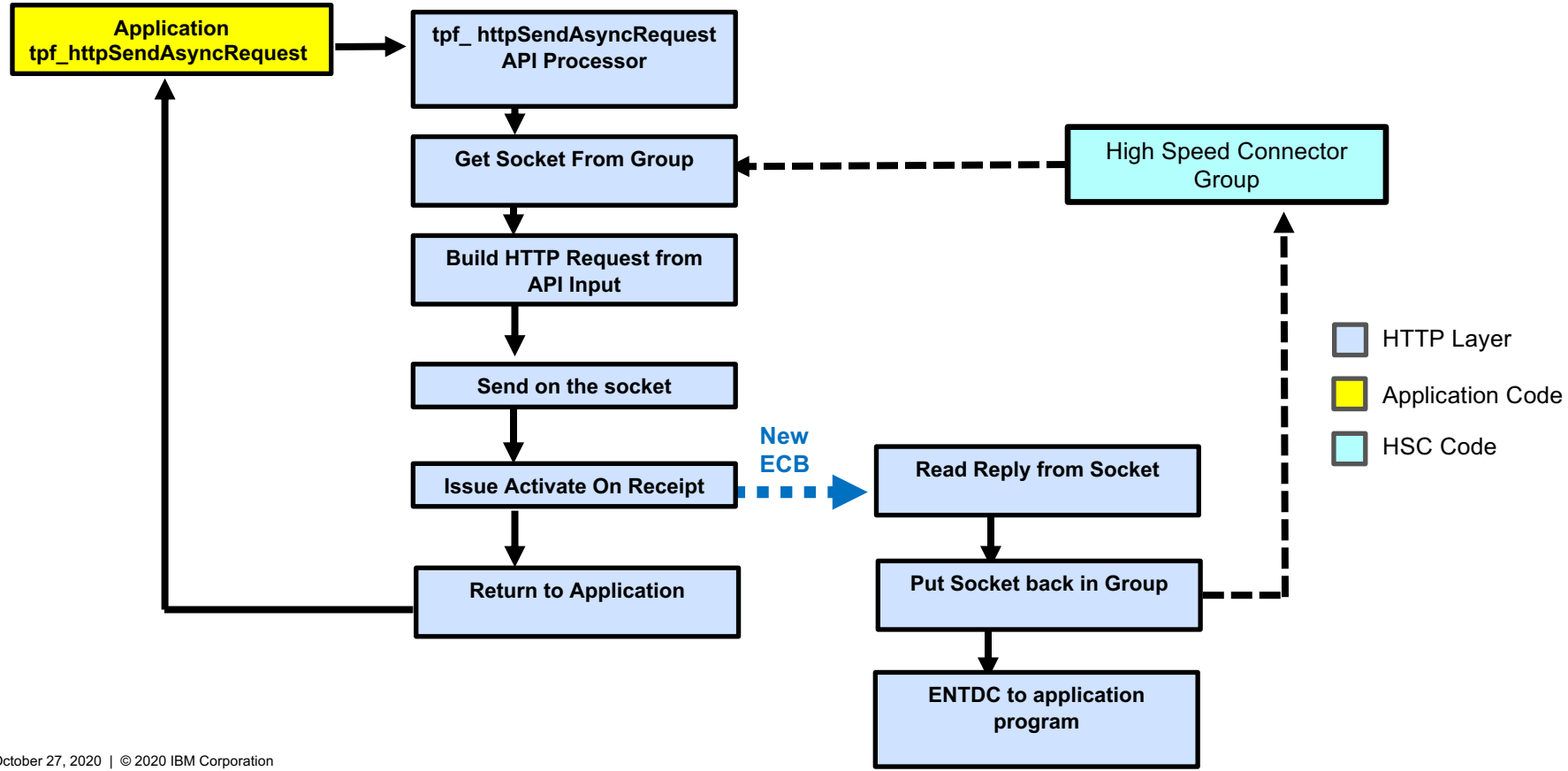
Enhanced HTTP Client

Synchronous HTTP Client Requests Functional Flow



Enhanced HTTP Client

Asynchronous HTTP Client Requests Functional Flow



Enhanced HTTP Client

Defining Persistent Sessions

```
<tns:groupName>tpfrules</tns:groupName>
 <tns:groupType>HTTP</tns:groupType>
 <tns:qMaxDepth>100</tns:qMaxDepth>
 <tns:qThreshold>80</tns:qThreshold>
 <tns:syncTimeout>200</tns:syncTimeout>
 <tns:maxAsyncData>10000</tns:maxAsyncData>
 <tns:maxAsyncMessage>10000</tns:maxAsyncMessage>
 <tns:aliasHostname>tpfrules.airco.com</tns:aliasHostname>
 <tns:aliasHostname>tpfrules.airco.com:81</tns:aliasHostname>
                .
                .
                .
 <tns:Endpoint>
  <tns:endpointName>httprulp</tns:endpointName>
  <tns:role>PRIMARY</tns:role>
  <tns:destination>httprulp.airco.com</tns:destination>
  <tns:startSocket>10</tns:startSocket>
  <tns:maxSocket>20</tns:maxSocket>
  <tns:bufferSendSize>262144</tns:bufferSendSize>
  <tns:bufferReceiveSize>262144</tns:bufferReceiveSize>
 </tns:Endpoint>
```

Endpoint Descriptor File

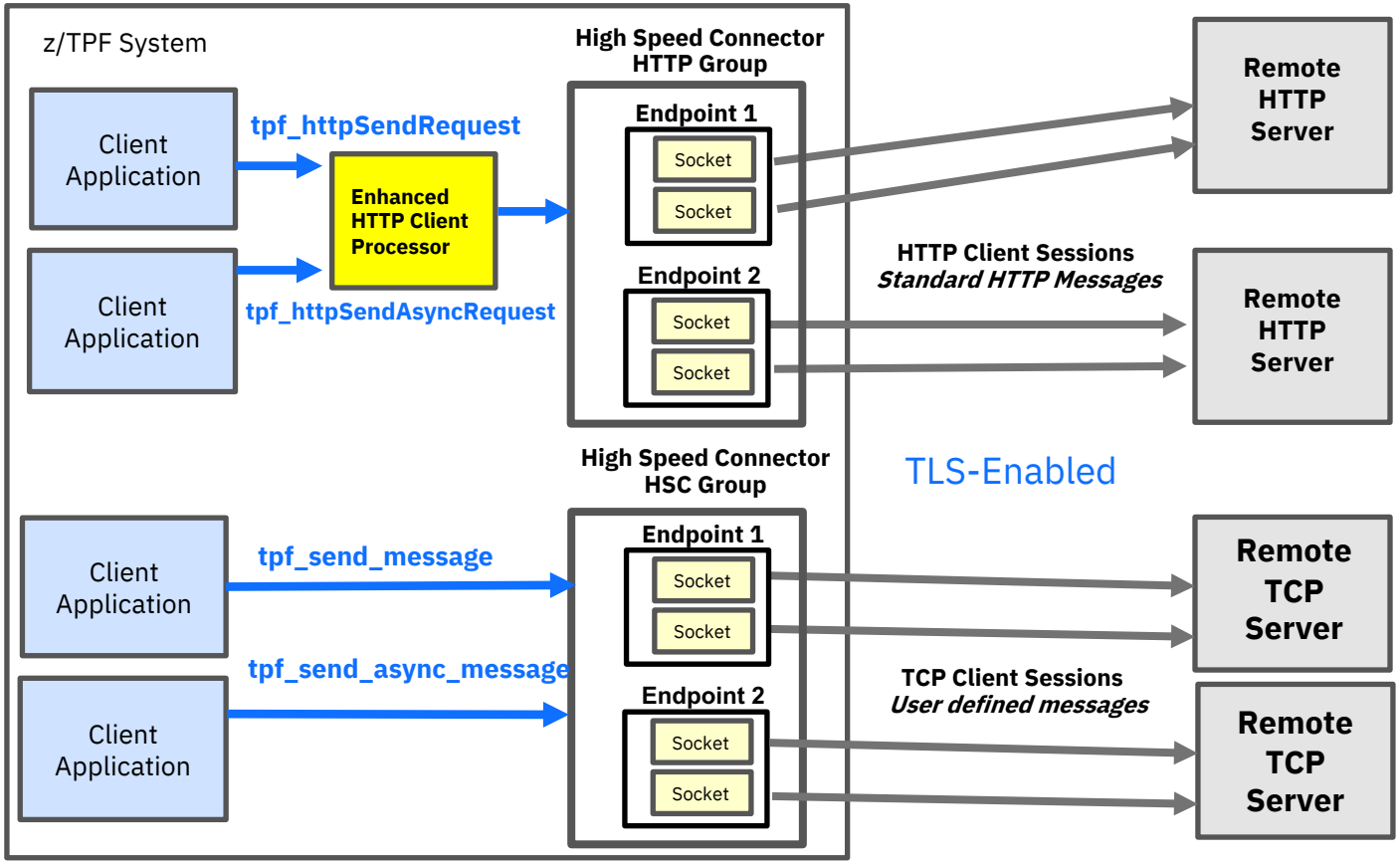
Load – Version Control File System

Enhanced HTTP Client
Changing Persistent Session Configuration

- Transparent to the z/TPF applications
  - Increase capacity (adding more endpoints/socket)
  - Change endpoint definitions (ie. socket buffer sizes)
  - Change group definitions (queue sizes, timeouts, thresholds)

- Simply load an updated version of the file through the z/TPF loader package to apply updates
  - **Changes take effect immediately for existing endpoint groups!**

Enhanced HTTP Client
High Speed Connector Usage



High Speed Connector

- z/TPF managed persistent client sessions
- Ability to send standard HTTP messages
- Ability to send user defined messages asynchronously

Trademarks

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.