z/TPF V1.1

TPF Users Group - 2011

Title: z/TPF HTTP Server Preview

Name: Mark Gambino

Venue: SOA Subcommittee

AIM Enterprise Platform Software

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

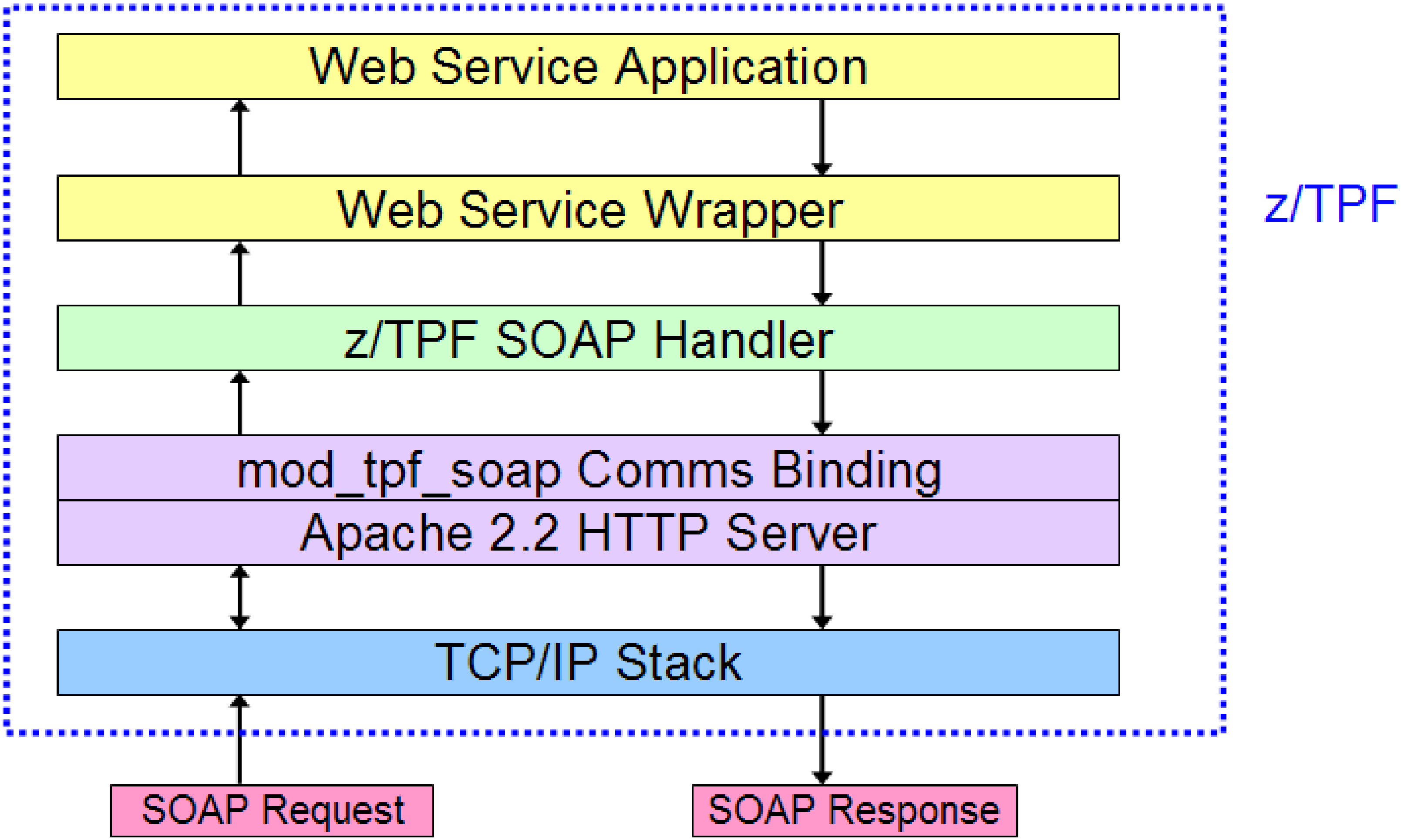
The 2010 TPFUG

- **Meetings were held to discuss HTTP usage (including SOAP usage of HTTP) on z/TPF**
  - Several customers (and IBM) attended
- **The result was that requirement "SOA00002 – *Lightweight HTTP Server for SOAP Messaging*" was drafted and submitted at the 2010 TPFUG**

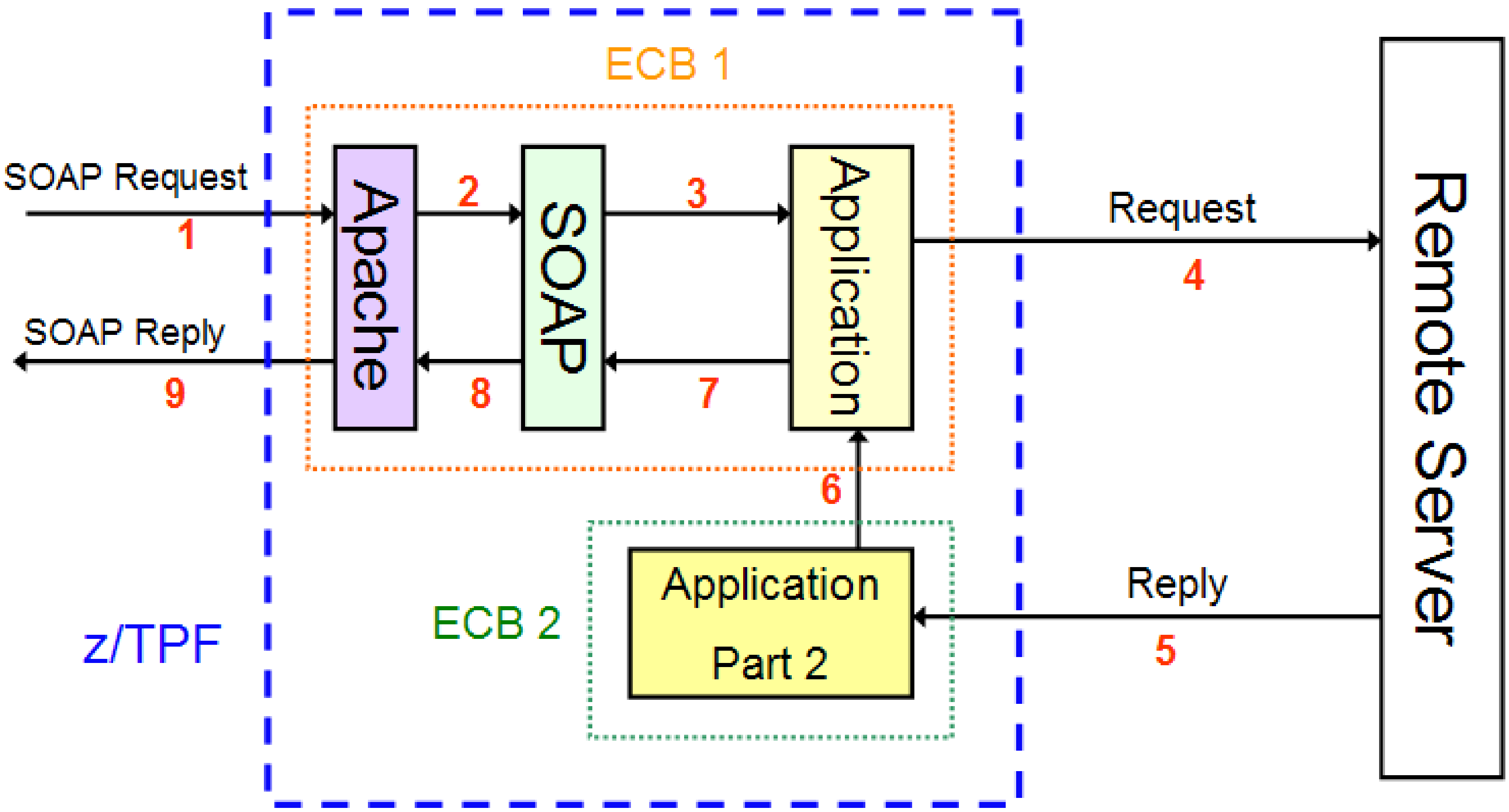
Driving Factors for the Requirement

- **Apache is a full function, very powerful Web server**
  - More lightweight server (lower path length) is desirable when HTTP is being used only as a simple message transport
- **Apache requires that the process (ECB) that receives the HTTP request must also send the response to that request**
  - Asynchronous option is desired for cases where the z/TPF application processing one request spans multiple ECBs (allow any ECB to send the response)

z/TPF SOAP Provider Message Flow Using Apache



z/TPF SOAP Provider using Apache Application Invokes Service on a Remote Platform - Example



SOAP Provider using Apache, Distributed Transaction Details (part 1)

1. **Remote client sends a SOAP request message that is read by Apache**
2. **The message is passed to the SOAP provider layer**
3. **The message is passed to the wrapper program, then is passed to the application**
4. **The application sends a request to a remote server (could use SOAP consumer, MQ, sockets, and so on), then the application suspends the ECB (ECB #1)**
5. **The remote server sends a response that causes ECB #2 to be created and read the response**

SOAP Provider using Apache, Distributed Transaction Details (part 2)

6. **The application in ECB #2 then wakes up the original ECB (ECB #1) passing it the response data from the remote server. ECB #2 exits.**
7. **The application in the original ECB (ECB #1) gets posted and returns to the SOAP handler**
8. **The SOAP provider builds the SOAP response and returns to Apache through the comms binding (mod_tpf_soap)**
9. **Apache builds the HTTP response and sends it back to the remote client**

What is the z/TPF HTTP Server

- **The "z/TPF HTTP Server" is being developed to address TPFUG requirement SOA00002**
- **The remainder of this presentation discusses the proposed design for this new z/TPF HTTP server**
  - Details are subject to change
- **Your feedback is greatly appreciated!**

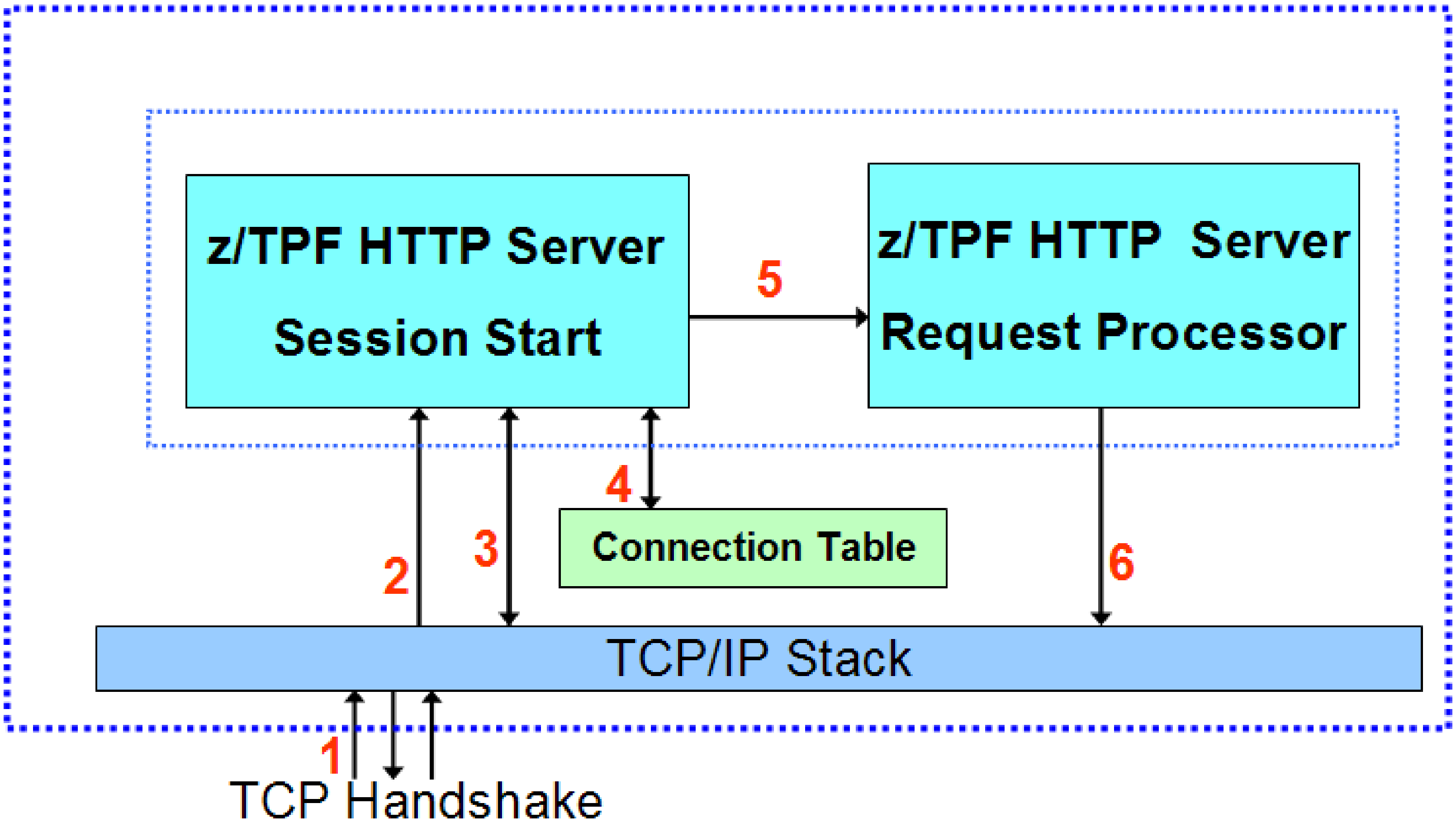
z/TPF HTTP Server Exploits z/TPF Strengths

- **Uses activate_on_accept (AOA) and activate_on_receipt (AOR) APIs such that there are no long running ECBs used by this support**
  - Long life (persistent) HTTP sessions do not tie up any ECBs while waiting for the next request to arrive
- **HTTP session state information and URL to program mapping information maintained in memory for performance reasons**
  - Connection table can reside above the 2G bar
- **HTTP session is not tied to a specific ECB**
  - Allows asynchronous processing where one ECB receives a request but a different ECB sends the response

z/TPF HTTP Server Key Components

- **HTTP Server Initialization – initializes control blocks and the listener socket**
- **HTTP Server Session Start – processes a new HTTP session**
- **HTTP Server Request Processor – processes an HTTP request message**
- **HTTP Server Response Processor – builds and sends an HTTP response message**
  - Invoked via the new ***tpf_httpSendResponse*** API
- **HTTP Server URL-Mapping File – defines which z/TPF application program to activate to process a request message based on URL specified in the request**
- **HTTP Server Connection Table – contains state information for each active HTTP session with this server instance**

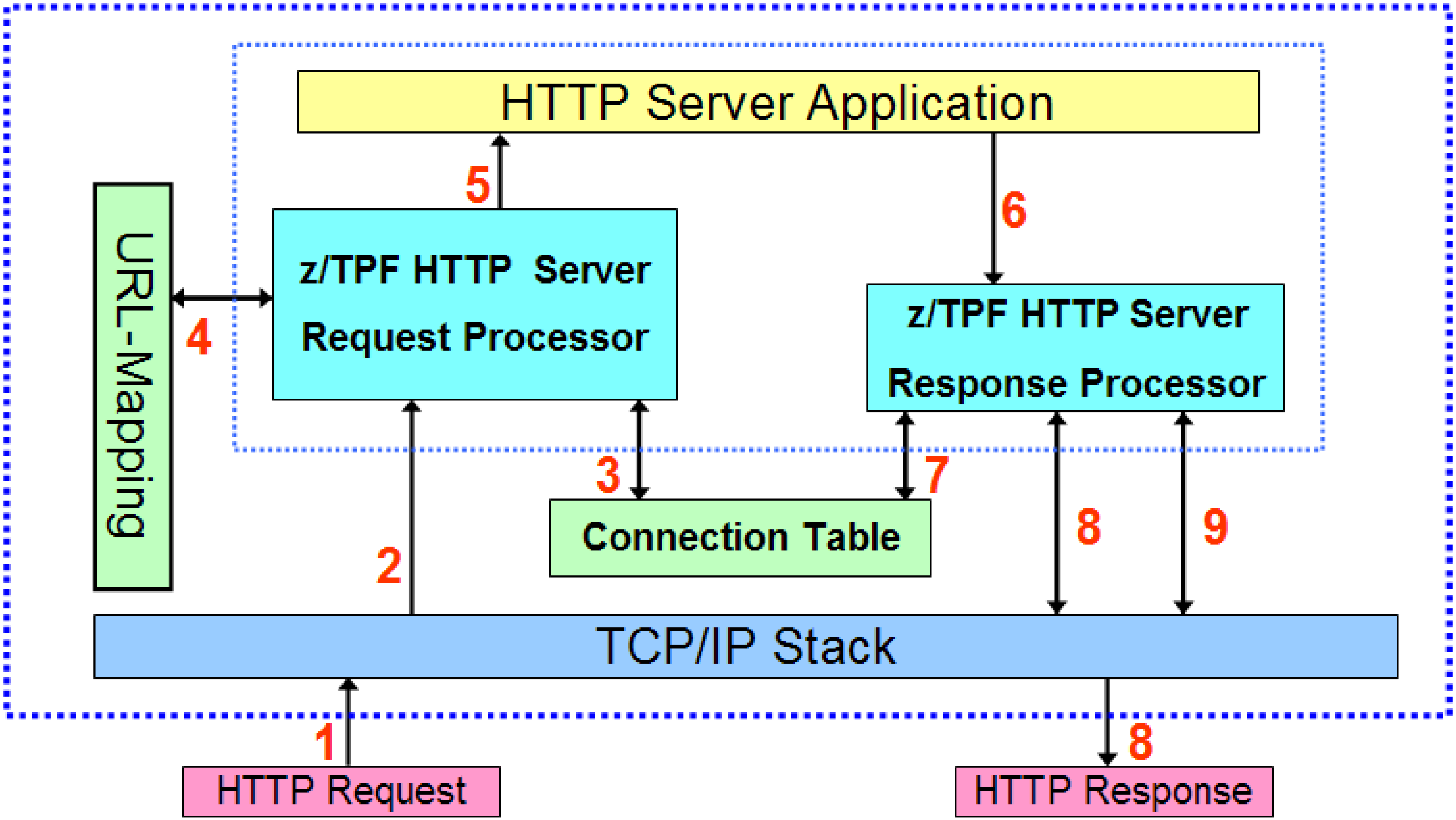
z/TPF HTTP Server - New Session Starts



New HTTP Session Starts - Details

1. Remote HTTP client starts a socket with z/TPF (normal TCP handshake flows)
2. activate_on_accept (AOA) is pending so the TCP/IP stack creates a new ECB activating the z/TPF HTTP Server Session Start program
3. AOA is issued again to handle the next session
4. An entry is added to the z/TPF HTTP Server Connection Table for the new session that just started
5. Control is passed to the z/TPF HTTP Server Request Processor
6. A socket read API is issued to read the first message for this new HTTP session

z/TPF HTTP Server Receives a Message that is Processed in a Single Application ECB



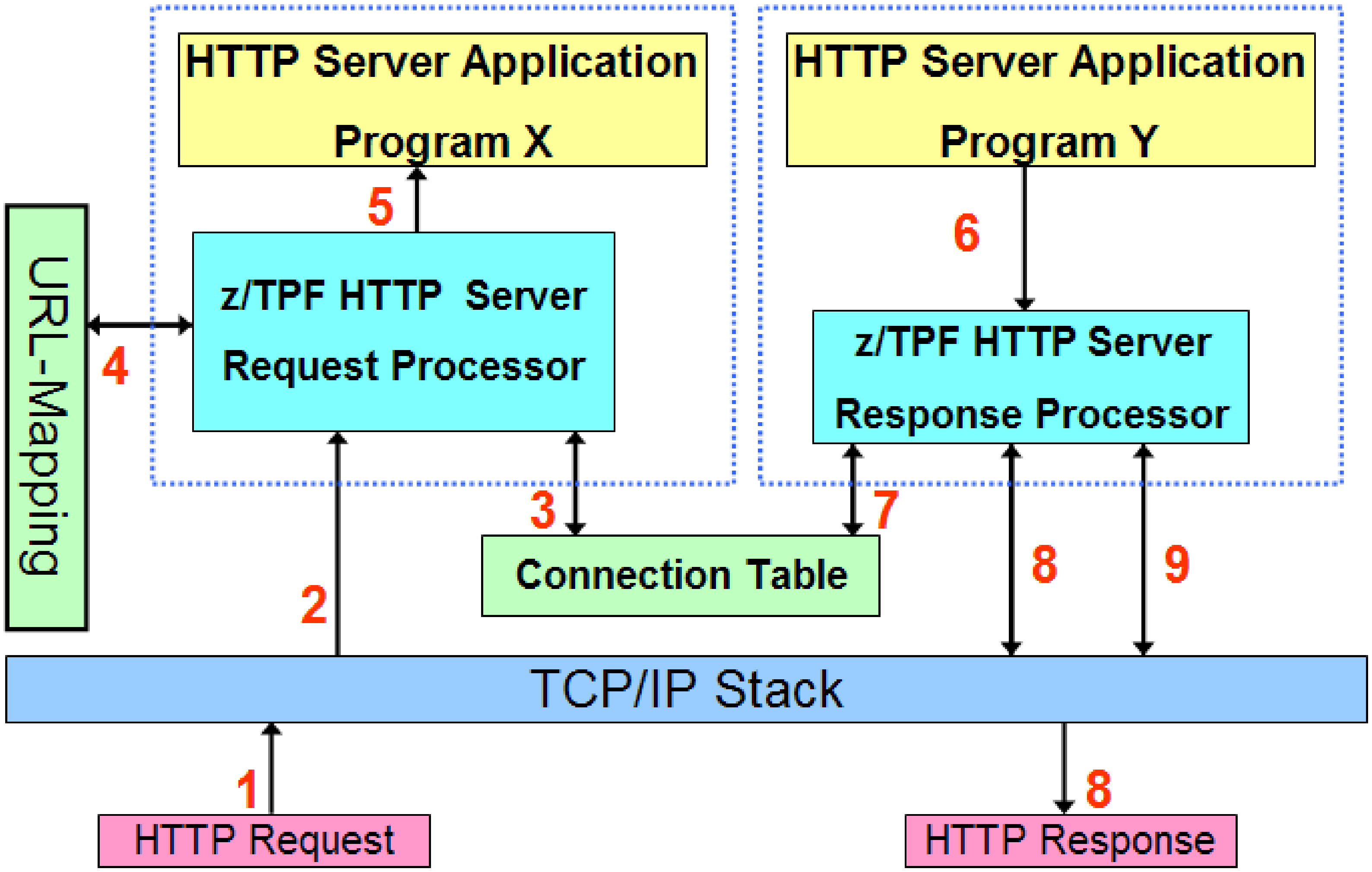
Process HTTP Message – Details (part 1)

1. **Remote client sends an HTTP request message**
2. **TCP/IP stack passes the message to the z/TPF HTTP Server Request Processor that previously issued read API**
3. **Status for this session is updated in its Connection Table entry**
4. **URL in the HTTP request is looked up in the URL-Mapping Table to determine which application program to activate**
5. **Application program is activated and processes the request**
6. **Application program builds the response data and issues the tpf_httpSendResponse API**

Process HTTP Message – Details (part 2)

7. **z/TPF HTTP Server Response Processor looks up the HTTP session information in the Connection Table**
   - If not a persistent HTTP session, then delete the entry. Otherwise, update the session state
8. **The HTTP response message is built and a socket write API is issued to send it**
9. **If this is not a persistent session, close the socket. Otherwise, issues activate_on_receipt (AOR) to kick off z/TPF HTTP Request Processor in a new ECB when the next request arrives on this persistent HTTP session**

z/TPF HTTP Server Receives a Message – Different Application ECB Sends the Response



Multiple Application ECB Example

- **The details of this flow are identical to single ECB details except that the request is given to the application (program X) in ECB #1, but that ECB exits and the response is sent by the application (program Y) from a ECB #2.**

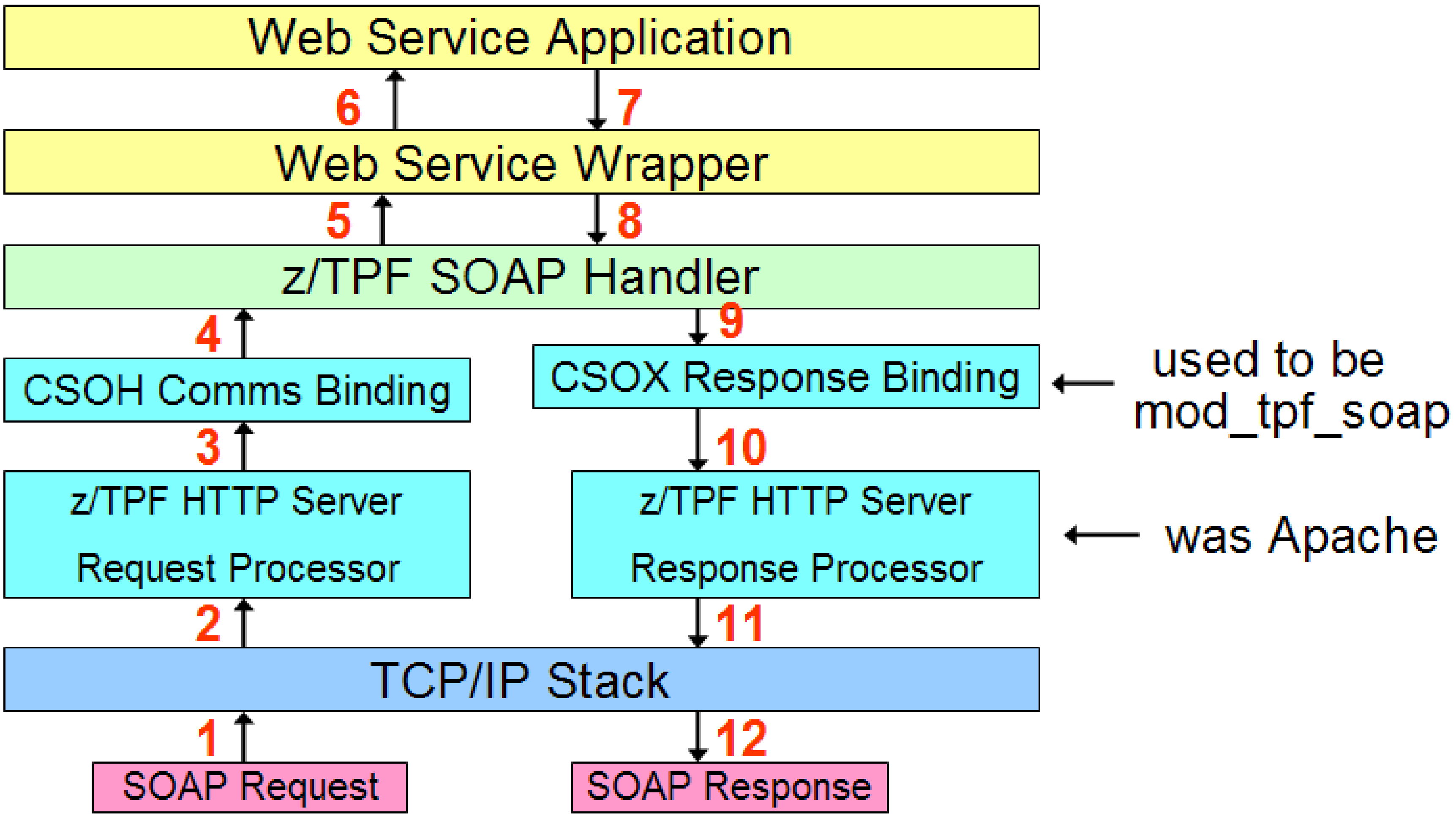
SOAP Provider using the z/TPF HTTP Server

- **In the URL-mapping file, define SOAP URLs as being processed by the z/TPF HTTP Server Comms Binding segment (CSOH)**
- **CSOH does for the z/TPF HTTP Server what mod_tpf_soap does for Apache for requests**
  - Comms binding that interfaces with the z/TPF SOAP Handler
  - CSOH also does Web Service Interoperability (WSI) conformance checks if necessary
- **Use the new *tpf_soapSendResponse* API to send the SOAP response**
- **z/TPF HTTP Server Response Binding (CSOX) passes the SOAP response back to the z/TPF HTTP Server using the new *tpf_httpSendResponse* API**
  - CSOX does for the z/TPF HTTP Server what mod_tpf_soap does for Apache for responses

Migrating a SOAP Server Application from Apache to the z/TPF HTTP Server

- **The next example shows how to take an existing SOAP provider application that was using Apache and have it now use the z/TPF HTTP Server instead**
- **No changes are required to your application or to your wrapper program**

z/TPF SOAP Provider Message Flow Using z/TPF HTTP Server and a Single Application ECB



Process SOAP Message – Details (part 1)

1. Remote client sends a SOAP request message
2. TCP/IP stack passes the message to the z/TPF HTTP Server Request Processor that previously issued read API
3. URL in the HTTP request is looked up in the URL-Mapping Table to determine which application program to activate. The entry for this URL says to use CSOH.
4. The CSOH comms binding segment sets up the interface for and calls the z/TPF SOAP Handler
5. z/TPF SOAP Handler calls the Wrapper program
6. Wrapper program calls the Application program
7. Application program processes the request and returns to the Wrapper
8. The Wrapper returns to the z/TPF SOAP Handler

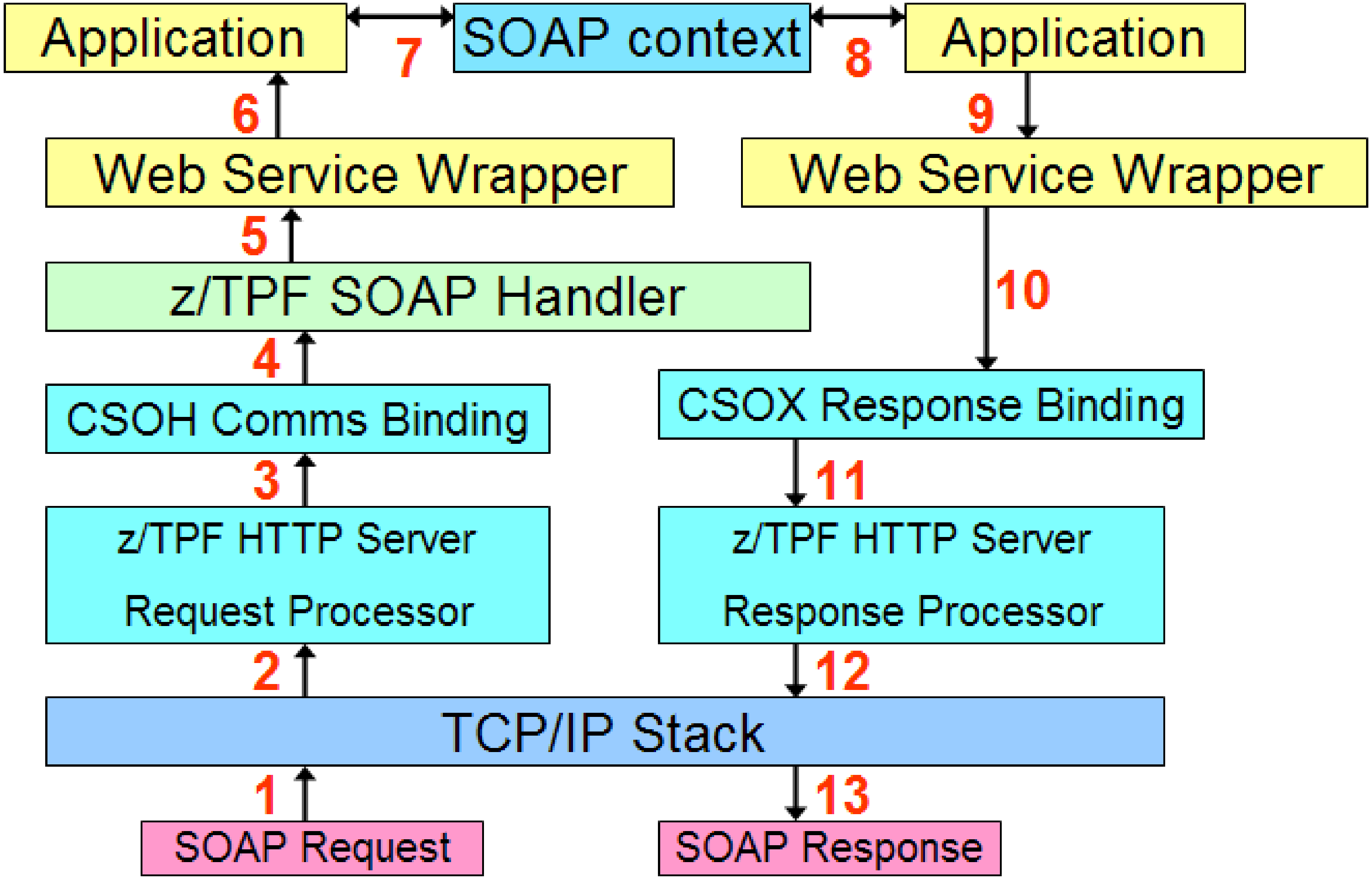
Process SOAP Message – Details (part 2)

9. **z/TPF SOAP Handler builds the SOAP response message, sees that the z/TPF HTTP Server comms binding was used, then calls CSOX**
10. **CSOX Response Binding program issues the tpf_httpSendResponse API to pass the response to the z/TPF HTTP Server**
11. **z/TPF HTTP Response Processor builds the HTTP response and issues a socket write API**
12. **The SOAP response is sent to the remote client**

Asynchronous SOAP Provider Application

- **If a SOAP provider application spans multiple ECBs:**
  - Original ECB that is passed the request message must save the SOAP context information using the new **tpf_soapSaveCtx** API
  - Application ECB that is going to send the SOAP response (via the **tpf_sendSoapResponse** API) must first issue the new **tpf_soapRetrieveCtx** API to restore the saved SOAP context

z/TPF SOAP Provider Message Flow Using z/TPF HTTP Server and a Multiple ECB Application



Process SOAP Message – Details (part 1)

1. **Remote client sends a SOAP request message**
2. **TCP/IP stack passes the message to the z/TPF HTTP Server Request Processor that previously issued read API (ECB #1)**
3. **URL in the HTTP request is looked up in the URL-Mapping Table to determine which application program to activate. The entry for this URL says to use CSOH.**
4. **The CSOH comms binding segment sets up the interface for and calls the z/TPF SOAP Handler**
5. **z/TPF SOAP Handler calls the Wrapper program**
6. **Wrapper program calls the Application program**
7. **Application program issues tpf_soapSaveCtx to save the SOAP context and the application continues in a new ECB (ECB #2). ECB #1 exits.**
8. **Application program in ECB #2 issues tpf_soapRetrieveCtx to restore the SOAP context and finishes processing the request message**
9. **Application program in ECB #2 calls the wrapper program**

Process SOAP Message – Details (part 2)

10. **The wrapper program (ECB #2) builds the SOAP response message and issues the tpf_sendSoapResponse API**
11. **CSOX Response Binding program issues the tpf_httpSendResponse API to pass the response to the z/TPF HTTP Server**
12. **z/TPF HTTP Response Processor builds the HTTP response and issues a socket write API**
13. **The SOAP response is sent to the remote client**

z/TPF HTTP Server Requirement Characteristics

- **Alternative to Apache 2.2 HTTP server**
  - Can run both HTTP servers at the same
- **HTTP as a transport only**
  - **Not** a full function web server
- **Supports a subset of the HTTP protocol (RFC 2616) needed for a transport layer**
  - HTTP 1.1 only
  - POST and GET methods
  - Persistent connections
  - Request timeouts

RESTful Web Services

- REpresentational State Transfer (REST) architecture was developed in parallel with HTTP 1.1
- RESTful Web services is an alternative to using SOAP
- Uses HTTP (HTTP 1.1) exclusively
- Stateless where all parameters needed to process the request are passed as part of the URL. Example:

http://www.fake.com/Search?type=bar&city=SFO

RESTful Web Services on z/TPF

- Can use RESTful Web services with the z/TPF HTTP server
- Entire URL (and entire HTTP request header) is passed to the application program
- Example:

  http://www.fake.com/Search?type=bar&city=SFO

- URL passed to application is Search?type=bar&city=SFO
- HTTP header request passed to application includes:

  GET Search?type=bar&city=SFO HTTP/1.1

  Host: www.fake.com

Defining a z/TPF HTTP Server Instance to INETD

- **Define the server to the Internet Daemon (INETD) using the ZINET ADD command specifying:**
  - Local IP address and local port number
  - Server name of this instance
  - Program name of the z/TPF HTTP server
    - Must specify **PGM-CHT1**

Defining Multiple TCP ports for the z/TPF HTTP Server

- **Multiple z/TPF HTTP server instances can be defined to INETD and active at the same time**
- **The local IP address and port combination for each active server instance must be unique**
- **Example showing 4 active instances:**
  - `Instance 1, IP = ANY    , port = 80`
  - `Instance 2, IP = 1.1.1.1, port = 5001`
  - `Instance 3, IP = 2.2.2.2, port = 5001`
  - `Instance 4, IP = 1.1.1.1, port = 5005`

Starting/Stopping a z/TPF HTTP Server Instance

- **A z/TPF HTTP server instance is started and stopped just like any other INETD server using ZINET START and ZINET STOP commands**
- **ZINET STOP command:**
  - Prevents any new sessions from starting for this server
  - Does not break existing connections or impact requests currently being processed
- **During cycle down**
  - INETD stop all servers
  - TCP/IP stack ends all sockets

z/TPF HTTP Server Instance Configuration File

- **Each z/TPF HTTP server instance needs its own configuration file defined containing:**
  - Maximum number of active sessions allowed for this instance
  - Whether persistent HTTP sessions are allowed and if yes, how long to wait for the next request before timing out
  - TCP backlog value
  - Maximum HTTP request message size allowed
  - Should Web Service Interoperability (WSI) conformance checks be made against the request message
    - Only applicable for SOAP messages

z/TPF HTTP Server URL-Mapping File

- **Defines which z/TPF application program to activate to process a request message**
- **File is subsystem unique**
- **Each entry in the file contains**
  - URL specified in the HTTP request
  - z/TPF program name to activate
  - How long to wait for the z/TPF application program to send a response before timing out
    - If timeout occurs, the z/TPF HTTP server sends an “internal server error” response (HTTP response status 500)

New Operator Commands

- **Display configuration values for one or all z/TPF HTTP server instances**
- **Display statistics for each server instance, including:**
  - Number of sessions currently active
  - Maximum number of sessions that were active
  - Number of requests that timed out

New User Exit

- **z/TPF HTTP Server Request Timeout User Exit**
- **Called when a request was given to a z/TPF application, but the application did not respond in time resulting in a timeout**
- **Allows you to log the incident**
- **Input to the user exit:**
  - URL in the HTTP request
  - Name of the application program that was given this request message

z/TPF HTTP Server Summary

- **Can co-exist with Apache**
- **Easy migration path from Apache**
- **Integrated with z/TPF SOAP provider support**
- **Can be used outside the scope of SOAP**
- **Synchronous or asynchronous**
- **Supports long running (persistent) sessions, but has no long running ECBs**
- **Can support multiple servers (TCP ports)**

Trademarks

- IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.
- Apache is a trademark of The Apache Software Foundation.
- Other company, product, or service names may be trademarks or service marks of others.

- 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.