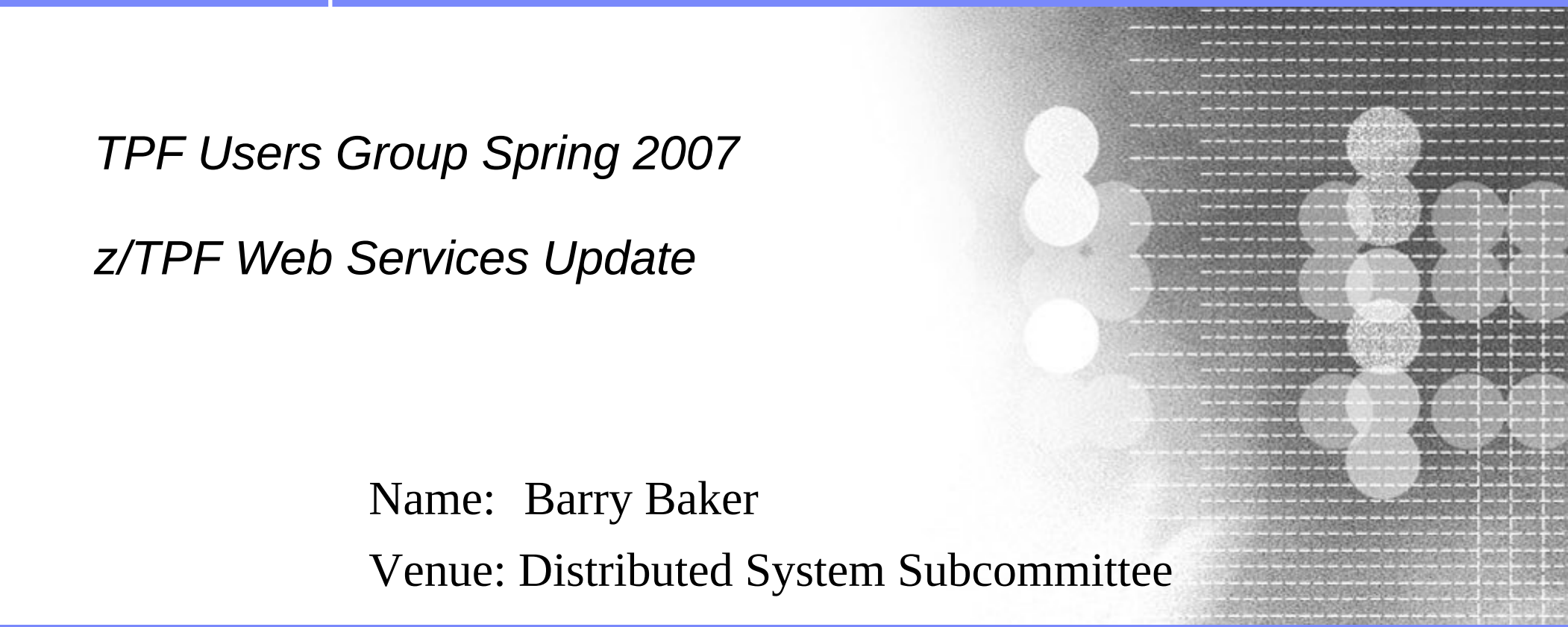
z/TPF EE V1.1
z/TPFDF V1.1
TPF Toolkit for WebSphere® Studio V3
TPF Operations Server V1.2

IBM Software Group

TPF Users Group Spring 2007

z/TPF Web Services Update



Name: Barry Baker

Venue: Distributed System Subcommittee

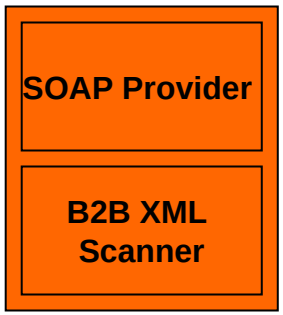
AIM Enterprise Platform Software

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

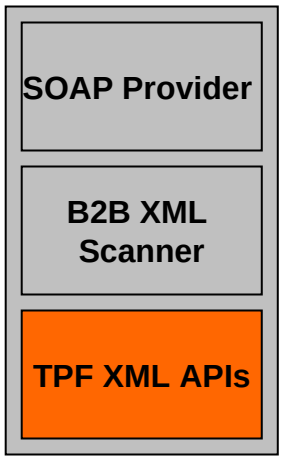
- Where we have been
- Where we are today
- Where we are going…near term
  - HTTP Client
  - SOAP enhancements/infrastructure
  - Web services tooling
- Where we are going…longer term
  - You tell us where to go
    - SOAP Consumer (client) support
    - HTTPS server

Where we have been



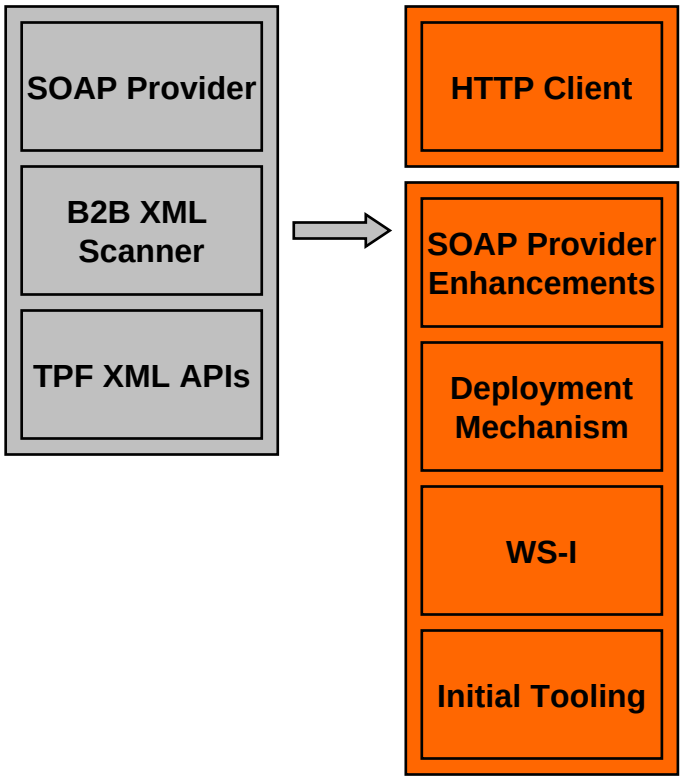
- What
  - SOAP Provider
    - Lightweight home-grown SOAP Handler
    - Transport neutral
  - B2B XML Scanner
    - High performance, non-validating XML scanner
- How
  - SOAP communications bindings
  - SOAP Handler is an API
  - User exit based deployment and routing: tpf_soap_appl_handler() user exit
- Why
  - Nascent technology
  - XML processing is expensive

Where we are today



- What
  - TPF XML APIs
    - Parsing XML
    - Accessing/converting the XML data
    - Building XML
- How
  - ~20 APIs
- Why
  - Raise the level of abstraction from working with a low level data structure to working with XML elements/constructs
  - Insulation from underlying changes to the parser technology
  - Ability to create XML documents

Where we are going…near term



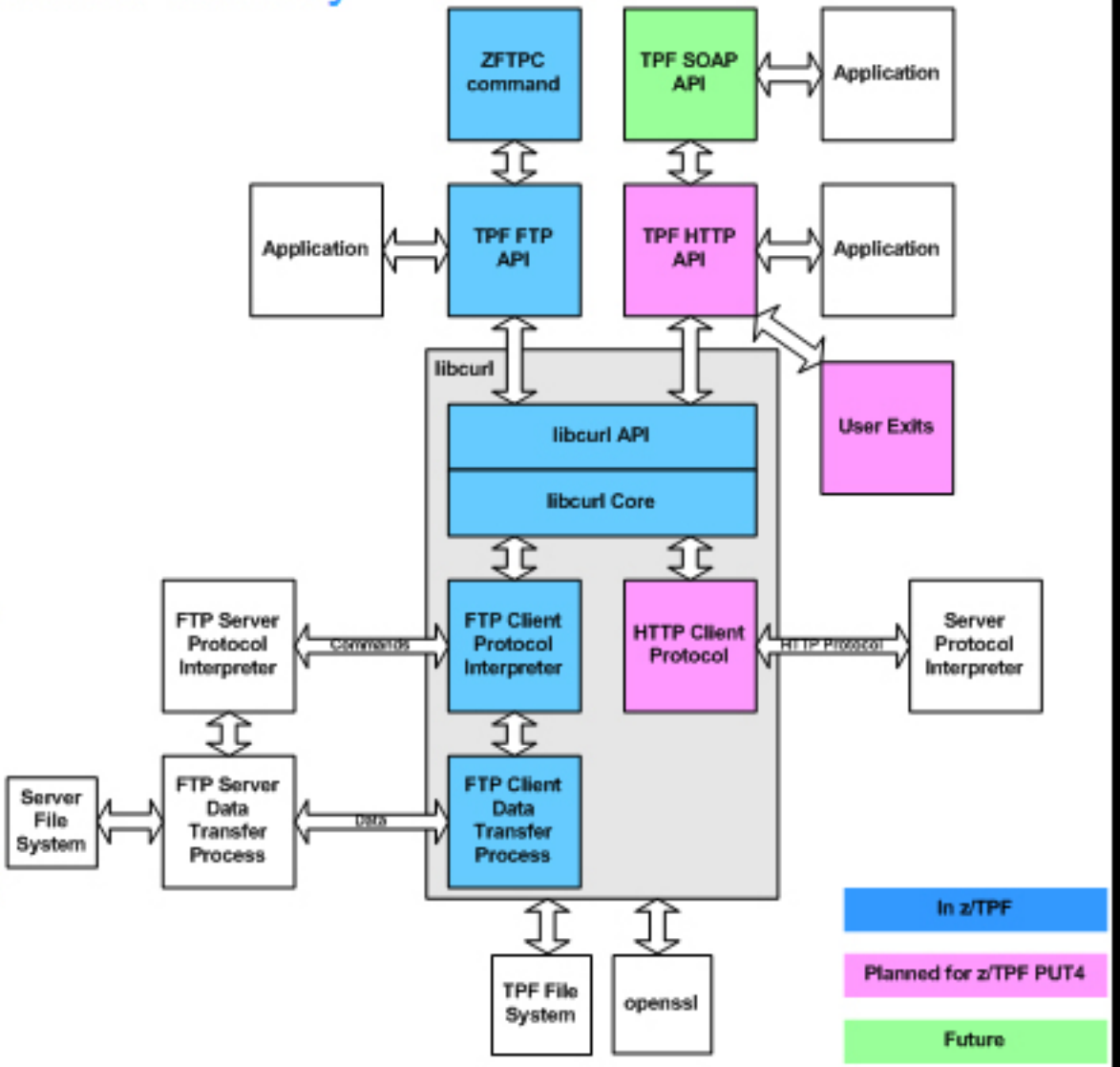
- What
  - HTTP Client (detail to follow)
  - SOAP Provider Enhancements:
    - Web Service wrappers: XML isolation
    - SOAP Message Handlers: WS-* infrastructure
    - SOAP Bridge Support
  - Deployment Mechanism
    - Deployment Descriptors (DDs) + ZWSAT to manage Web service resources (deploy/undeploy all or individual resources) in the Web service deployment table (WSDT)
  - WS-I conformance checking
  - Initial tooling in TPF Toolkit
    - Create/maintain/publish WSDL/DDs/ Web Service wrappers
    - Web service testing tools
- How
  - New home grown support added to the Core support
  - Eclipse Web Tools Platform + custom tools added to TPF Toolkit
- Why
  - Interoperability
  - Ease of use and reduced application changes

HTTP Client Overview

- z/TPF currently provides for Hypertext Transfer Protocol (HTTP) server support
  - Apache 1.3, obtained through the Apache Web site, runs on z/TPF
- We are targeting to address TUG Requirement DS01006 on z/TPF PUT 4
  - "…allow TPF applications to access data available on web based technology. It allows TPF to use business processes that are based in web and application servers."
  - "…open up TPF to participate in exchanging data as defined by the OTA (Open Travel Alliance) using XML under HTTP"
- HTTP client is a pre-requisite to SOAP consumer support
  - HTTP is the most popular transport used with regards to Web services (including both SOAP based and RESTful Web services)
  - Like z/TPFs SOAP provider support, z/TPF SOAP consumer support will be transport neutral. It is expected that sample communication bindings will be provided for SOAP consumer requests over HTTP and Webshere MQ, in a similar fashion to the samples currently provided for SOAP provider support.
- z/TPF HTTP client support
  - Applications can initiate HTTP requests from z/TPF to remote HTTP servers
  - Optionally communicate securely using HTTPS

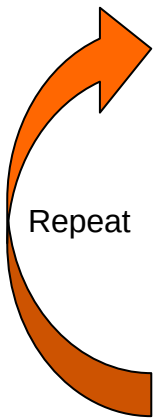
libcurl – Open Source URL Transfer Library

- z/TPF's HTTP client support uses libcurl to handle the HTTP protocol
- libcurl is an open source "client-side URL transfer library"
  - Initially ported to z/TPF to provide for an FTP Client
    - z/TPF PUT 3: PJ31266 and PJ31296
  - Re-port to include the HTTP protocols (HTTP and HTTPS)
  - For additional information about libcurl, see http://curl.haxx.se/libcurl
- The libcurl library is not shipped with the z/TPF product
  - If you wish to use either the FTP or HTTP client, you must download libcurl following the instructions from the z/TPF Downloads Web site
  - If libcurl is *not* installed, both the FTP and HTTP clients are disabled



HTTP Client Application Interface

- Applications can use z/TPF specific application programming interfaces (APIs) to initiate HTTP requests in one of two ways:
  - Persistent connection mode
    - **tpf_httpInitHandle** - Initializes an HTTP client connection handle
      - HTTP version: 1.0/1.1
      - Proxy usage
      - Connect timeout
      - .netrc file location (optional way for dealing with username and passwords)
    - **tpf_httpPerform** – Sends an HTTP client request to a remote HTTP server
      - Input – Handle + Request parameters:
        - HTTP request methods: HEAD/GET/PUT/POST
        - Request URL
        - Request timeout
        - Redirects
        - Data: typically used with PUT and POST
        - Headers
      - Output:
        - Data structure containing HTTP Server return code, all Headers/Footers, and the Body of the response



    - **tpf_httpEnd** – Ends an HTTP client connection with an HTTP server
    - **tpf_httpGetOpts** – Get the options currently set for an HTTP client connection handle
  - Single request mode
    - **tpf_httpPerform1** – Send a single HTTP client request to an HTTP server
- HTTP connection handles *cannot* be shared across processes

Secure HTTP Client Using SSL Application Configuration Files

- SSL Application Configuration Files were introduced with z/TPF's FTP Client support (see *z/TPF FTP Client Support*, TPF Users Group Fall 2006)
  - In order to establish secure sockets using OpenSSL, z/TPF middleware packages must obtain user-specific configuration information
  - SSL application configuration files provide a standardized format for specifying this information
  - z/TPF middleware packages (like FTP client and now HTTP client) make use of the new `tpf_SSL_getConfig` API to read in the desired configuration file

- HTTP Client relies on SSL Application Configuration Files to provide for HTTPS
  - Two types of SSL application configuration files are supported by the z/TPF HTTP client
    - Machine-specific configuration files
      - /etc/ssl/httpc/*machine*.conf
    - Default configuration file
      - /etc/ssl/httpc/httpc.conf
  - z/TPF HTTP Client APIs do **not** rely on the SSL Application Configuration Files to determine if the HTTPS should be used. The Application indicates if HTTPS is requested through the use of "https://" in the request URL

What you can do with HTTP client “out of the box”

- Interact with an HTTP server
  - Programmatically GETing and POSTing/PUTting HTTP resources
  - XML over HTTP -> OTA

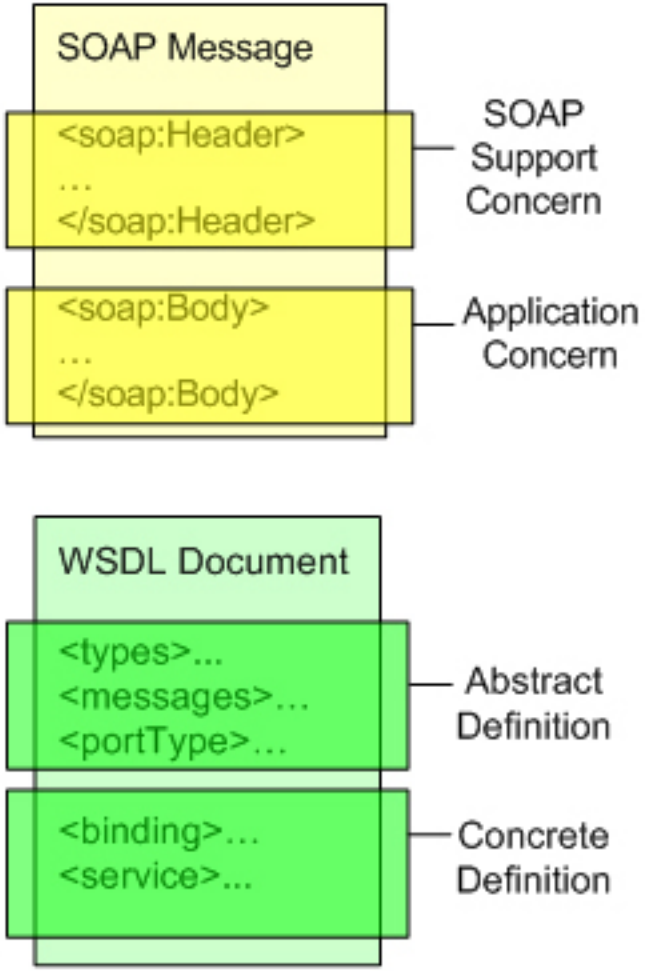
- Interact with Web services
  - Two Styles: SOAP Based Web Services and RESTful Web Services
  - SOAP-based Web services
    - You could implement a SOAP consumer…but we do not suggest this – let us do this!
    - SOAP Web Service with a One-Way Message Exchange Pattern (MEP)
      - In WSDL, a Web service can be defined with an **`input-only operation`**.
      - A fire-and-forget usage requires a mechanism to send a message to a single SOAP provider
      - The SOAP consumer does not require any status information that the message has been sent to or received by the SOAP provider.
      - A complete SOAP request could be built using the TPF XML APIs and the HTTP client could be used to send the request to the Web service and not have to deal with any response data

What you can do with HTTP client “out of the box”

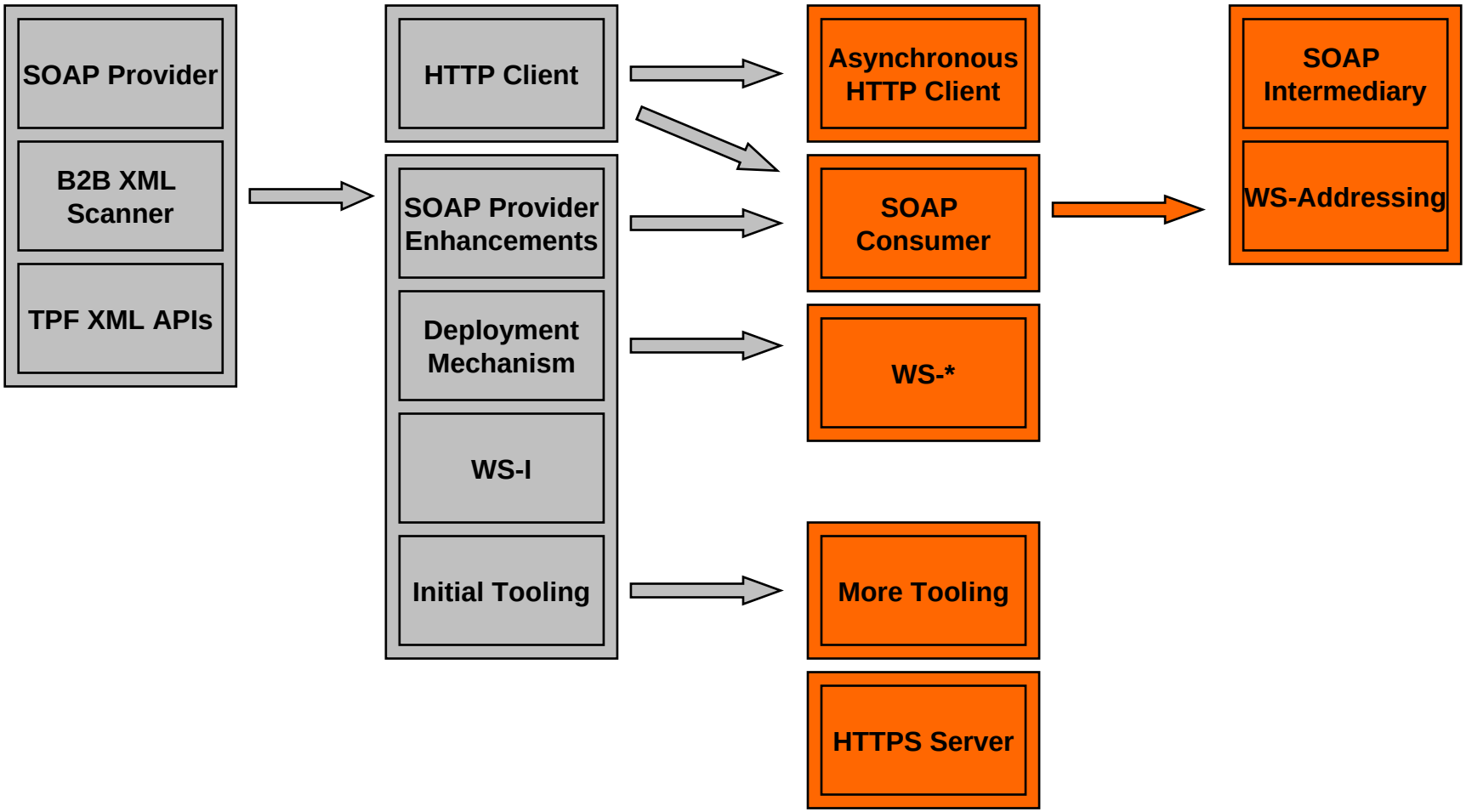
- Interact with Web services (cont)
  - REpresentational State Transfer or RESTful Web Services
    - REST: The architecture of the Web – Roy Fielding’s thesis
      - An architectural style and a distillation of the way the Web already works
      - **Goals of the style:** stateless client-server architecture providing easy to consume Web services that exploit the underlying capabilities of the Web including caching, scaling, and lower coordination costs
      - Resources are identified by URIs, Messages are self-descriptive and stateless
    - RESTful Web Services
      - HTTP 1.1 was designed to conform to REST and is the underlying protocol
      - Nouns: URIs
      - Verbs: Using different verbs for every noun would inhibit widespread use, therefore REST uses *universal* verbs:
        - GET/POST/PUT/DELETE are the verbs, that’s it!…these are the 4 main HTTP request methods
      - REST messages
        - Look like HTTP requests with XML as the body of the messages (can have other payload types)
    - Example RESTful Application/Protocol: Atom publishing protocol
      - Protocol used for many purposes, including syndication via web feeds, journalism, marketing, bug-reports, or any other activity involving periodic updates or publications (i.e. pub/sub)
      - Use Case: Flight schedule changes -> HTTP PUT an Atom Entry to a feed to broadcast the change

SOAP Consumer

- No standard SOAP Consumer API in C/C++
- Design Direction:
  - Handle based API
  - Supports multiple usage modes
    - *Power user* mode: Leaves the application responsible for everything outside of the standard SOAP specification processing
      - i.e. TPF provides the transport and the SOAP conformance checking and presents the response back to the application as an XML structure containing the entire SOAP response message
    - *Smart user* mode: In addition to the standard SOAP specification processing, the SOAP support will:
      - Rely on the new deployment mechanism
      - Allow the application to be coded to the Body portion of SOAP messages (i.e. the Abstract portion of the WSDL definition)
      - Handle the Header portions of the SOAP message through use of the SOAP Message Handlers

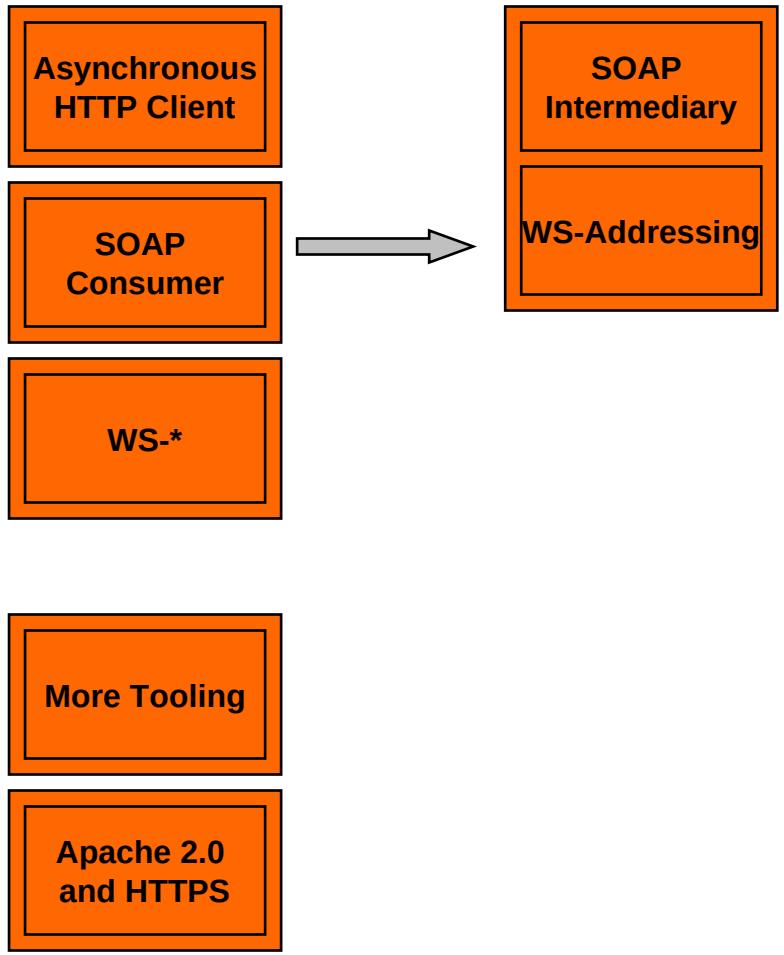


Where we are going…longer term



Where we are going…longer term

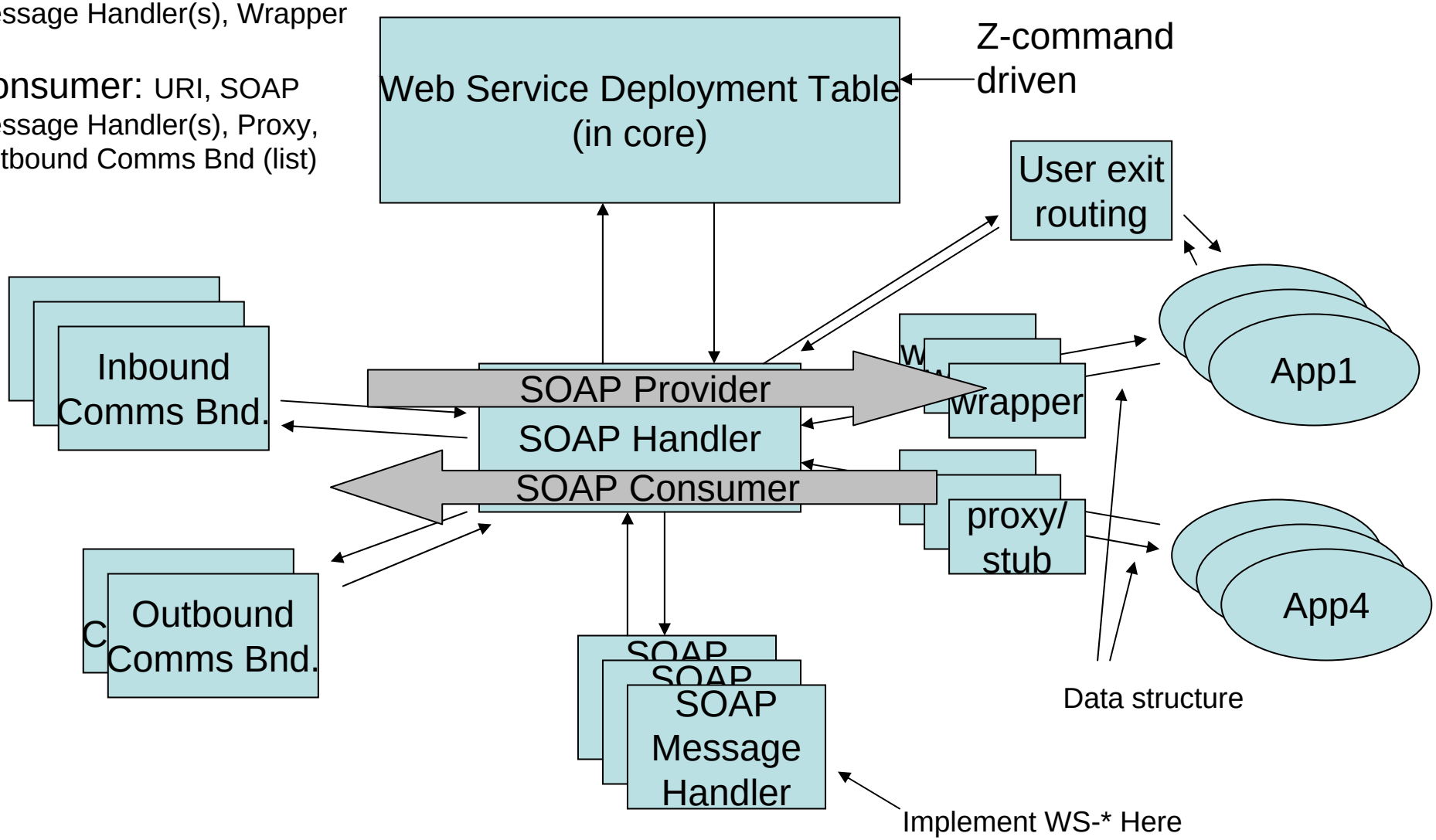
- What
  - Non-blocking or asynchronous HTTP client
  - SOAP Intermediary Support
  - WS-*…there are dozens of these
  - Additional Tooling
  - Provide for HTTPS (current direction is Apache 2.0)
- How
  - Build future support on the infrastructure we are putting in place now
  - WS-*: What's the best way to develop these?
- Why
  - Getting SOAP consumer and SOAP intermediary support allows z/TPF to fully operate in a SOA environment
  - More tooling to help reduce development/maintenance costs



Backup

Provider: URI, SOAP
Message Handler(s), Wrapper

Consumer: URI, SOAP
Message Handler(s), Proxy,
Outbound Comms Bnd (list)



Trademarks

IBM are trademarks of International Business Machines Corporation in the United States, other countries, or both.
Apache is a trademark of The Apache Software Foundation
“WS-I” is a trademark of the Web Services-Interoperability Organization in the United States and other countries.

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.