z/TPF EE V1.1

# *TPF Users Group Fall 2007*

## z/TPF Enhancements for SOAP Provider Support and Tooling for Web Services Development

Jason Keenaghan

Distributed Systems Subcommittee

AIM Enterprise Platform Software

IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

# Developing Web services for z/TPF

- Exposing existing application logic via Web services, or creating new Web services to be run on z/TPF, has been a manual process that required a wide assortment of skills
- In depth knowledge of several Web services-related specifications was needed:
  - eXtensible Markup Language (XML)
  - SOAP 1.1 and/or SOAP 1.2
  - Web Services Description Language (WSDL)
  - Universal Discovery Description and Integration (UDDI)
- In depth knowledge of structures and concepts used by the z/TPF SOAP runtime:
  - Manual interrogation and manipulation of `infoNodes` structure for processing the XML-formatted SOAP request
  - Code updates required to the `tpf_soap_appl_handler` to deploy or undeploy implemented Web services

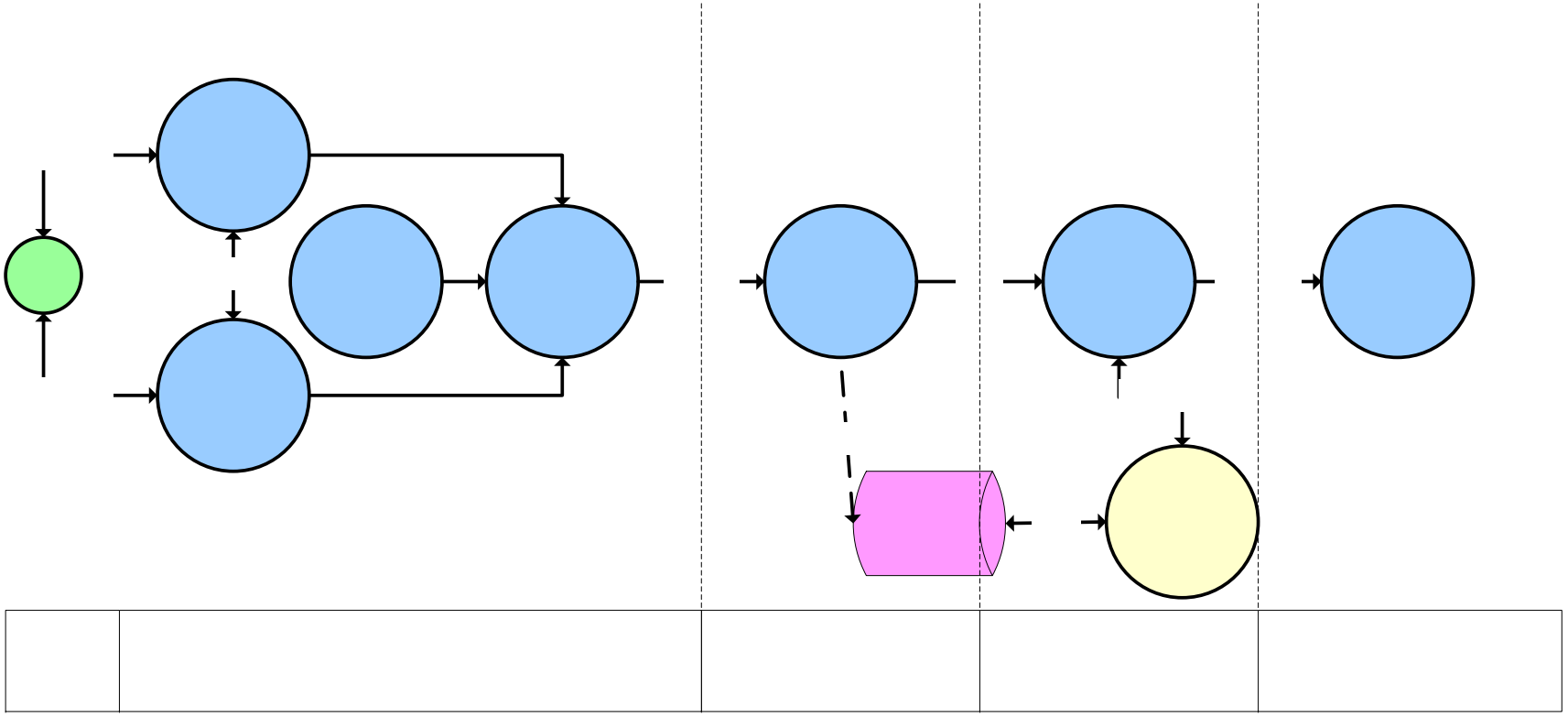
# Improving Web services development for z/TPF

- Earlier efforts had begun to address the complexity involved with developing Web services on z/TPF:
  - z/TPF XML APIs (PJ30866, PUT 2) made it so users no longer had to code to the internal `infoNodes` structure directly
  - Sample communications binding for receiving SOAP requests over WebSphere MQSeries was placed on the z/TPF Web site.
- Our goal is to make the development, deployment, and management of Web services on z/TPF, in both procedures and required skills, similar to those needed on other platforms
  - Knowledge of underlying standards and specifications is still required, but not at the level of detail previously necessary
  - A greater isolation of system and user code, with well-defined interfaces
- The result: *improved usability, greater flexibility,* and *faster time-to-market*

# Enhancements for z/TPF SOAP provider support

- z/TPF APAR PJ32215 (PUT 4)
  - An improved deployment and management mechanism for Web service resources:
    - Provider Web services
    - SOAP message handlers
  - Infrastructure to enable users to implement optional SOAP features (WS-*) with SOAP message handlers
  - Web Services Interoperability Organization (WS-I) conformance checking
  - SOAP bridge support to invoke traditional z/TPF applications
- TPF Toolkit 3.2.3
  - Web Tools Platform (WTP) Eclipse plug-in integrated in TPF Toolkit
  - z/TPF-specific tooling for generating Web service artifacts needed by SOAP runtime

# Typical Web service development process

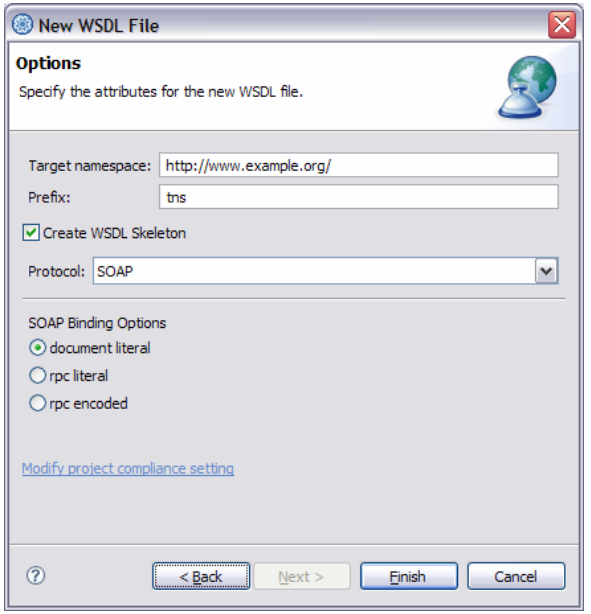


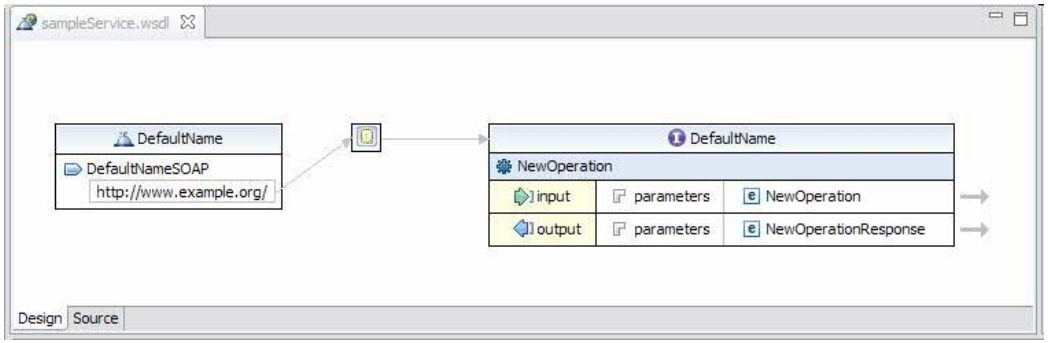
# Building a provider Web service

- 4 primary *artifacts* that each provider Web service should have:
  - **WSDL document:** defines the interface to the Web service for Web service consumers
  - **Provider Web service deployment descriptor:** defines the runtime characteristics of the Web service to the z/TPF SOAP handler
  - **Web service wrapper program:** processes the SOAP request message, invokes the Web service application, constructs the SOAP response message
  - **Web service application:** provides the actual service being requested

# WSDL: Define interface to the Web service

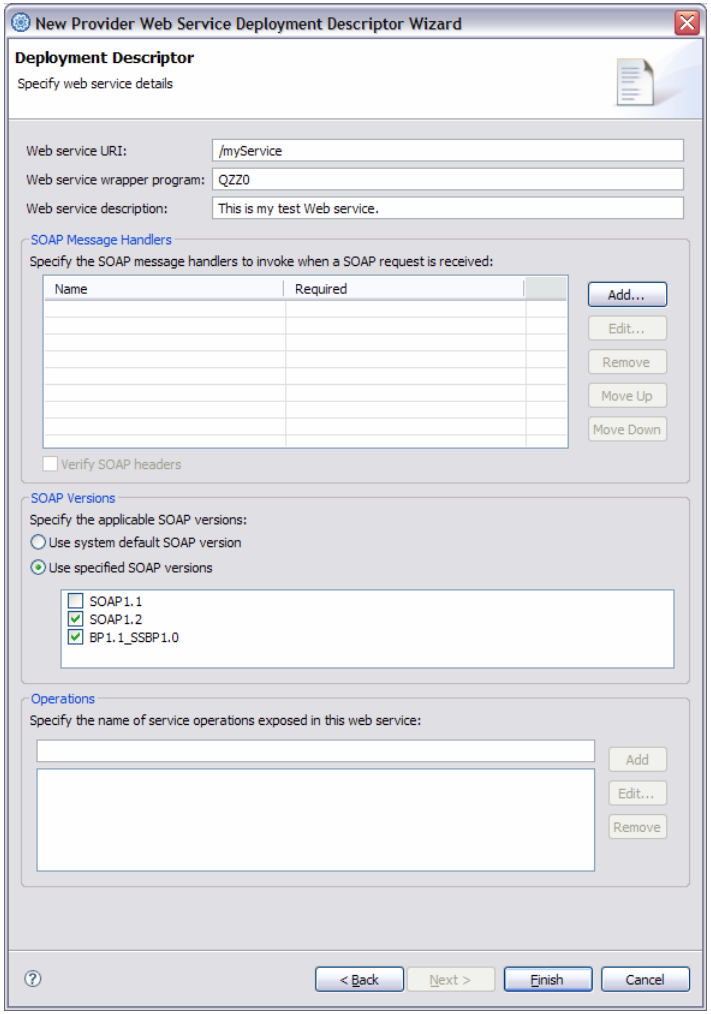
- Use the TPF Toolkit to create a **WSDL document** that defines the interface to the Web service you are exposing on z/TPF
- WSDL creation wizard will generate an initial template that acts as the starting point for your WSDL document
- WSDL editor provides a **Design** view and a **Source** view for editing the WSDL document



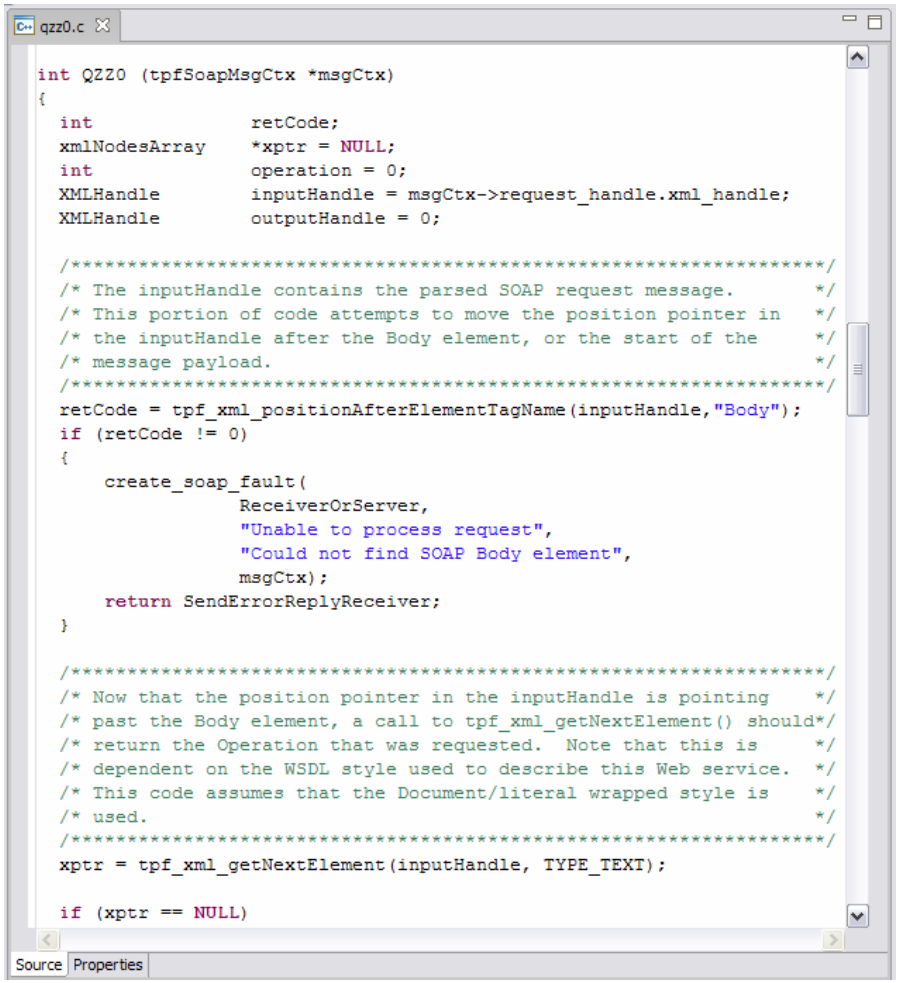
# Deployment descriptor: Define runtime characteristics

- Use the TPF Toolkit to create a **provider Web service deployment descriptor** that defines the runtime characteristics of the Web service to the z/TPF SOAP handler
- Deployment descriptor is an XML document that directs the processing path for an inbound SOAP request:
  - Verify request is in expected SOAP version format (including WS-I conformance checks)
  - Optionally verify that all SOAP headers marked as *mustUnderstand* are actually understood
  - Invoke specified chain of SOAP message handlers
  - Invoke specified Web service wrapper program



# Web service wrapper: Process a SOAP request

- Use the TPF Toolkit to create a template for your **Web service wrapper program** using the WSDL document you previously created
- Edit the generated Web service wrapper program:
  - Use the z/TPF XML APIs to extract any parameters from the SOAP request and transform those parameters from XML into the data representation expected by the underlying application
  - Invoke the correct application
  - Use the z/TPF XML APIs to create a SOAP response or SOAP fault to be returned to the z/TPF SOAP handler
- Compile and link the program like any other z/TPF shared object


qzz0.c

```
int QZZ0 (tpfSoapMsgCtx *msgCtx)
{
  int              retCode;
  xmlNodesArray    *xptr = NULL;
  int              operation = 0;
  XMLHandle        inputHandle = msgCtx->request_handle.xml_handle;
  XMLHandle        outputHandle = 0;

  /*************************************************************************/
  /* The inputHandle contains the parsed SOAP request message.             */
  /* This portion of code attempts to move the position pointer in         */
  /* the inputHandle after the Body element, or the start of the           */
  /* message payload.                                                      */
  /*************************************************************************/
  retCode = tpf_xml_positionAfterElementTagName(inputHandle,"Body");
  if (retCode != 0)
  {
      create_soap_fault(
                  ReceiverOrServer,
                  "Unable to process request",
                  "Could not find SOAP Body element",
                  msgCtx);
      return SendErrorReplyReceiver;
  }

  /*************************************************************************/
  /* Now that the position pointer in the inputHandle is pointing          */
  /* past the Body element, a call to tpf_xml_getNextElement() should      */
  /* return the Operation that was requested.  Note that this is           */
  /* dependent on the WSDL style used to describe this Web service.        */
  /* This code assumes that the Document/literal wrapped style is          */
  /* used.                                                                 */
  /*************************************************************************/
  xptr = tpf_xml_getNextElement(inputHandle, TYPE_TEXT);

  if (xptr == NULL)
```

Source | Properties

# Web service application: Provide requested service

- The **Web service application** may consist of business logic that already exists on the z/TPF system, or it may be newly written code
  - *If the application already exists:* code does not need to change in order to expose it via a new Web services interface
  - *If the application is new:* code does not need to be written to manipulate XML documents
- The application processes the request using any parameters passed to it from the Web service wrapper program and returns the necessary output
  - SOAP message presentation is handled by the Web service wrapper, not the application
- Compile/assemble and link the program like any other z/TPF shared object

# z/TPF SOAP bridge support for Web service wrappers

- Use **z/TPF SOAP bridge support** if the Web service application is a "traditional" z/TPF application:
  - Expects input message in application message format (AM0SG)
  - Activated via the system message router (segment COA4)
  - Sends output message via ROUTC macro or an equivalent
- Support consists of 6 new APIs intended for use in Web service wrapper programs:
  - `tpf_soapBridge_register`: Register a line number, interchange address, terminal address (LNIATA) for use by SOAP bridge support
  - `tpf_soapBridge_route`: Route an input message to a traditional z/TPF application
  - `tpf_soapBridge_receive`: Receive an output message from a traditional z/TPF application
  - `tpf_soapBridge_unregister`: Unregister a LNIATA from use by SOAP bridge support
  - `tpf_convertToAMSG`: Utility function to convert a byte array into AM0SG format
  - `tpf_convertFromOMSG`: Utility function to convert an output message in OMSG format to a byte array

# Packaging a provider Web service

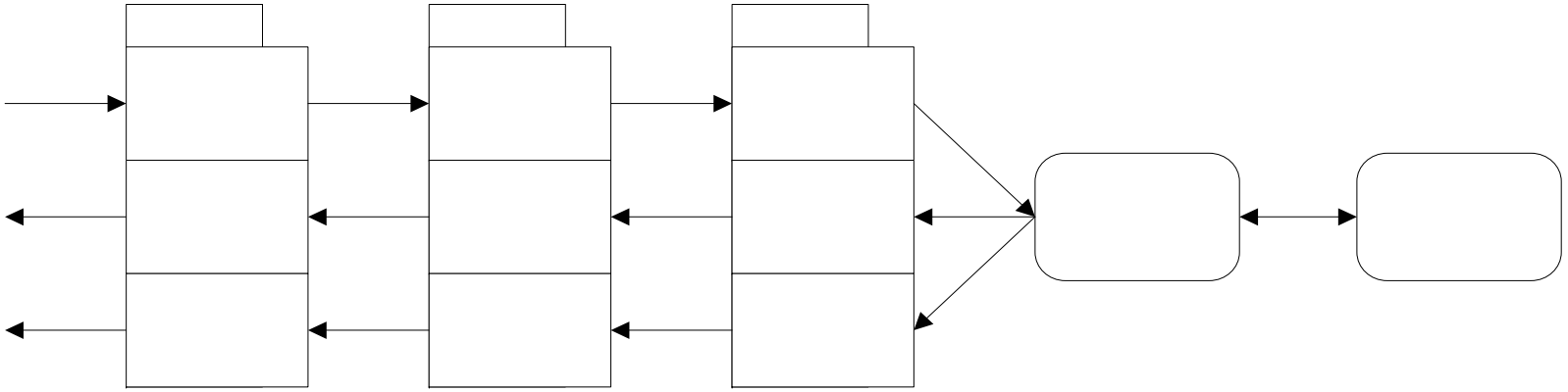
- Some of the artifacts built for a provider Web service are used online by z/TPF, others are intended for use offline only
  - **WSDL document:** used offline only
  - **Provider Web service deployment descriptor:**
    - FTP manually (in ASCII) to the `/etc/tpf-ws/` directory of the z/TPF file system
    - Use the TPF Toolkit to automatically transfer to the z/TPF file system
  - **Web service wrapper program:** load as part of E-type loader loadset
  - **Web service application:** load as part of E-type loader loadset (if not already on the z/TPF system)

# SOAP message handlers: Extend basic SOAP protocol

- **SOAP message handlers** provide an infrastructure that allows many different Web services to take advantage of common processing that is used to extend the basic SOAP protocol
- SOAP message handlers can be written to provide an interface to various Web service extensions:
  - *Standard SOAP features:* WS-Security, WS-Addressing, WS-*
  - *User-specific functionality:* request/response logging, etc.
- SOAP message handler artifacts (packaging and usage is identical to artifacts associated with provider Web services):
  - **SOAP message handler deployment descriptor:** defines the runtime characteristics of the SOAP message handler
  - **SOAP message handler program:** performs necessary processing using the inbound SOAP request and the outbound SOAP response
- Each provider Web service that requires a SOAP message handler to be called must include the message handler's name in the `SOAPMessageHandlerChain` element of its own deployment descriptor

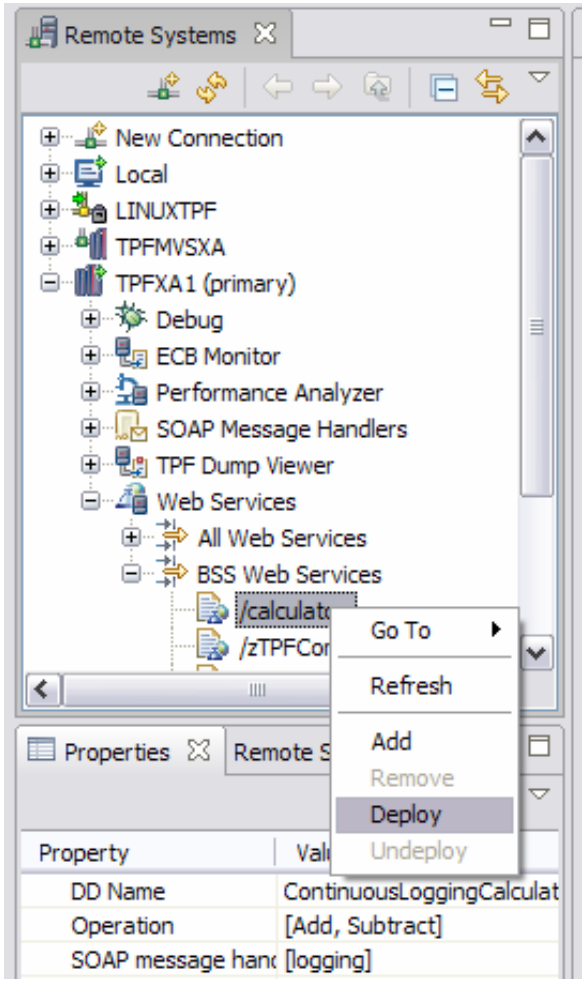
# SOAP message handler processing flow

- SOAP message handlers are invoked by z/TPF SOAP handler in the following situations:
  - **SOAP request:** called in the order they are listed in the Web service's SOAP message handler chain *before* calling the Web service wrapper
  - **SOAP response:** called in the reverse order they are listed in the Web service's SOAP message handler chain *after* calling the Web service wrapper
  - **SOAP fault:** called in the reverse order they are listed in the Web service's SOAP message handler chain



# Deploying Web service resources

- 2 alternatives for deploying Web service resources to the z/TPF SOAP handler:
  - Use the **TPF Toolkit's Remote System Explorer (RSE) perspective** to generate a list of available Web service resources
    - Right-click on one or more Web service resources listed and select the "Deploy" action
  - Use the **`ZWSAT DEPLOY`** command directly on z/TPF
- All programs that will be called to process a SOAP request must be loaded and active for a deployment to be successful:
  - SOAP message handler program
  - Web service wrapper program

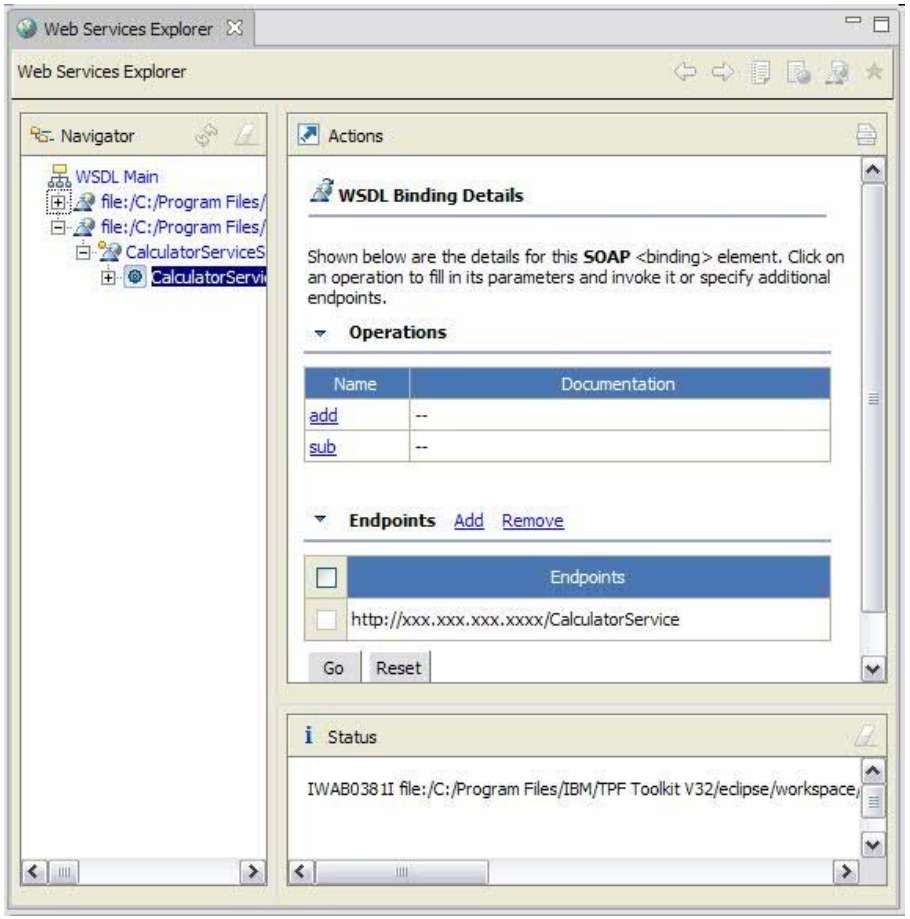


# Web services deployment table

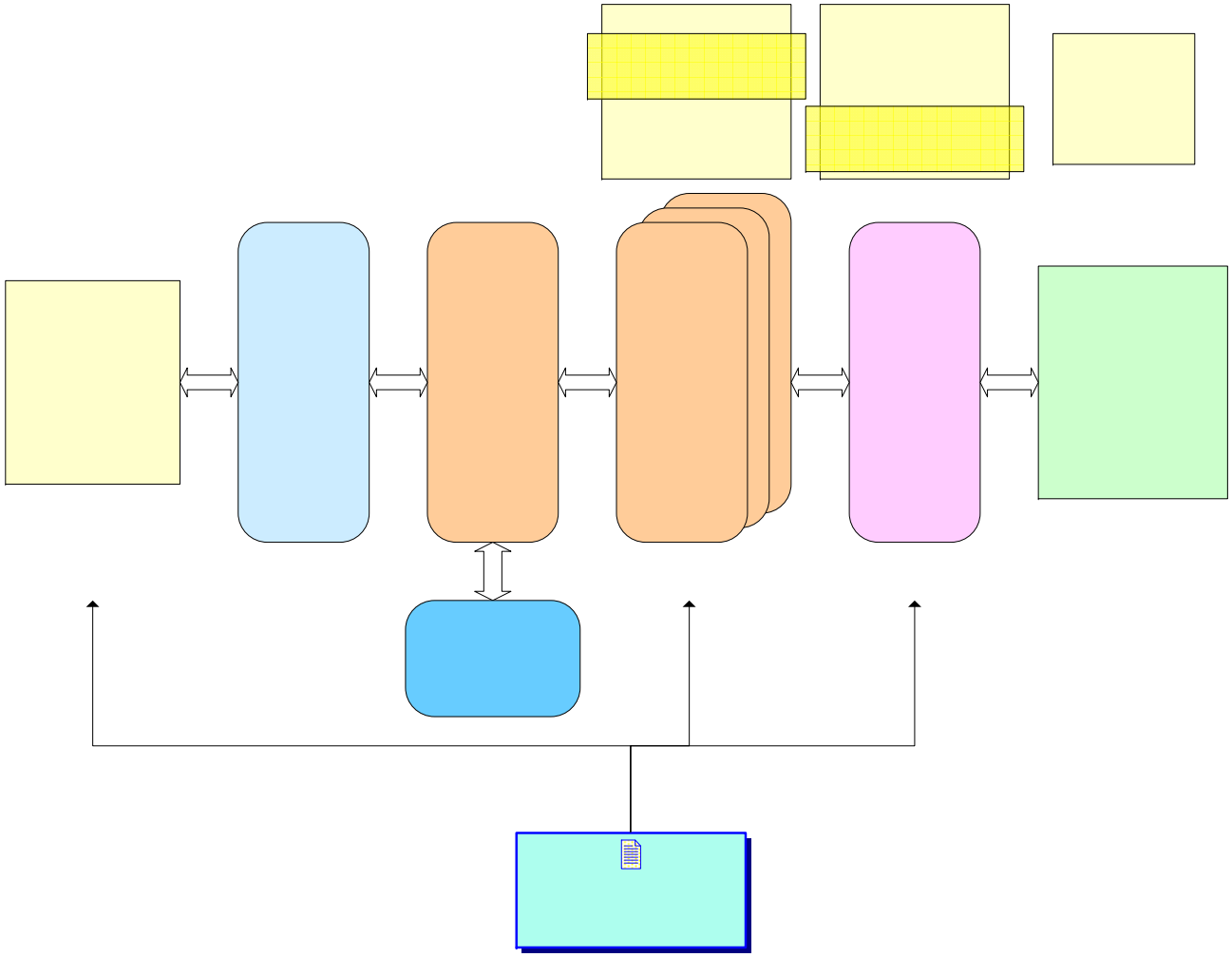
- z/TPF SOAP handler uses the new **Web services deployment table (WSDT)** when processing inbound SOAP requests
- The WSDT keeps track of all **Web service resources** (provider Web services and SOAP message handlers) that have been *deployed* or have been marked as *undeployed*
- The WSDT contains an internal representation of the data contained in each Web service resource's deployment descriptor
  - z/TPF SOAP handler does not interrogate deployment descriptors in XML format while servicing SOAP requests
- If a SOAP request is received for a Web service that does not have an entry in the WSDT, the existing `tpf_soap_appl_handler` user exit will be invoked to route the request

# Testing a provider Web service

- Use the TPF Toolkit to test the newly created Web service interface
- Register the desired program for debugging using the existing **Debug Subsystem**
  - SOAP message handler program
  - Web service wrapper program
- Generate a test SOAP request message using the new **Web Services Explorer**
  - Use the WSDL document that defines the Web service interface to generate a sample HTML form that acts as a SOAP client
  - Sends a complete SOAP request to z/TPF using HTTP



# Running a provider Web service



# Managing Web service resources

- Use the **ZWSAT** command to manage the Web service resources in a z/TPF complex
  - DEPLOY: adds a Web service resource to the WSDT *and* makes it available for use by the z/TPF SOAP handler
  - UNDEPLOY: marks a previously deployed Web service resource as unavailable to the z/TPF SOAP handler
  - ADD: adds a Web service resource to the WSDT *but* leaves its status as undeployed so it is not available to the z/TPF SOAP handler
  - REMOVE: removes a Web service resource from the WSDT
  - DISPLAY:
    - display the attributes and status of a specific Web service resource
    - display a summary of all Web service resources based on resource type or deployment status

# Redeploying Web service resources

- During the life cycle of a Web service resource, it may become necessary to change the runtime characteristics that are associated with that resource. For example:
  - Change the program that implements the resource
  - Update the SOAP message handler chain for a provider Web service
  - Update the supported SOAP versions for a provider Web service
- These updates can be made in real time with *no downtime* for the resource
  - Transfer the updated deployment descriptor to the `/etc/tpf-ws/` directory in the z/TPF file system
  - Use the `ZWSAT DEPLOY` command to “redeploy” the updated deployment descriptor that defines the Web service resource
  - When prompted, enter `ZWSAT CONTINUE` to continue with the deployment even though the resource is already deployed
  - All subsequent SOAP requests that make use of this Web service resource will begin using the new runtime characteristics

# Summary of new features in z/TPF

- Web services deployment mechanism
  - Deployment descriptor defines processing attributes for each Web service (for example, “wrapper” program, supported SOAP versions, etc.)
  - ZWSAT command:  Deploy, undeploy, and display Web service resources
- SOAP message handler infrastructure
  - Programs for extending one or more provider Web services
  - Used to implement standard SOAP features (WS-*) or user-specific extensions
- Web Services Interoperability Organization (WS-I) conformance checking
  - Basic Profile 1.1
  - Simple SOAP Binding Profile 1.0
- SOAP bridge support
  - 6 z/TPF-unique APIs for use in Web service wrapper programs for routing input messages to and receiving output messages from traditional z/TPF applications

# Sample Web service artifacts available for download

- 4 new samples have been created to demonstrate the usage of this new support, and are available for download at the z/TPF Web site
- Sample Web service wrapper (`/CalculatorService`)
  - Implements a simple calculator service with add and subtract operations
  - Includes: WSDL, deployment descriptor, Web service wrapper program, and HTML client
- Sample SOAP message handler (`logging`)
  - Implements a SOAP message handler that writes info to syslog (intended for use with `/CalculatorService` sample)
  - Includes: SOAP message handler deployment descriptor, provider Web service deployment descriptor, and SOAP message handler program
- Sample SOAP bridge wrapper (`/zTPFCommand`)
  - Implements a Web browser interface to z/TPF system message processor (SMP)
  - Includes: WSDL, deployment descriptor, Web service wrapper program, and HTML client
- Sample `mod_tpf_soap` (Apache 1.3 SOAP communications binding) updated to include additional message validation checks for WS-I Simple SOAP Binding Profile 1.0 conformance

# Summary of new features in IBM TPF Toolkit 3.2.3

- Web Tools Platform (WTP) Eclipse plug-in integrated in TPF Toolkit:
  - XML editor
  - Web Services Description Language (WSDL) creation wizard and editor
  - Web Services Inspection Language (WSIL) creation wizard
  - Publish WSDL documents to a Universal Discovery Description and Integration (UDDI) registry
  - Web services explorer (SOAP client for testing)
- Tooling specific to z/TPF added to TPF Toolkit:
  - Deployment descriptor creation wizard
  - Wrapper template creation wizard
  - Web Services and SOAP Message Handlers subsystems added to Remote systems explorer (RSE)

# Reminder

- Wednesday
  - **TPF Education Session:** *Creating Web Services for Your z/TPF System* (Edwin van de Grift)

**Trademarks**

- IBM and z/TPF are trademarks of International Business Machines Corporation in the United States, other countries, or both.
- 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.