z/TPF V1.1

TPF Users Group Spring 2009

Accessing Web Services From Your z/TPF System

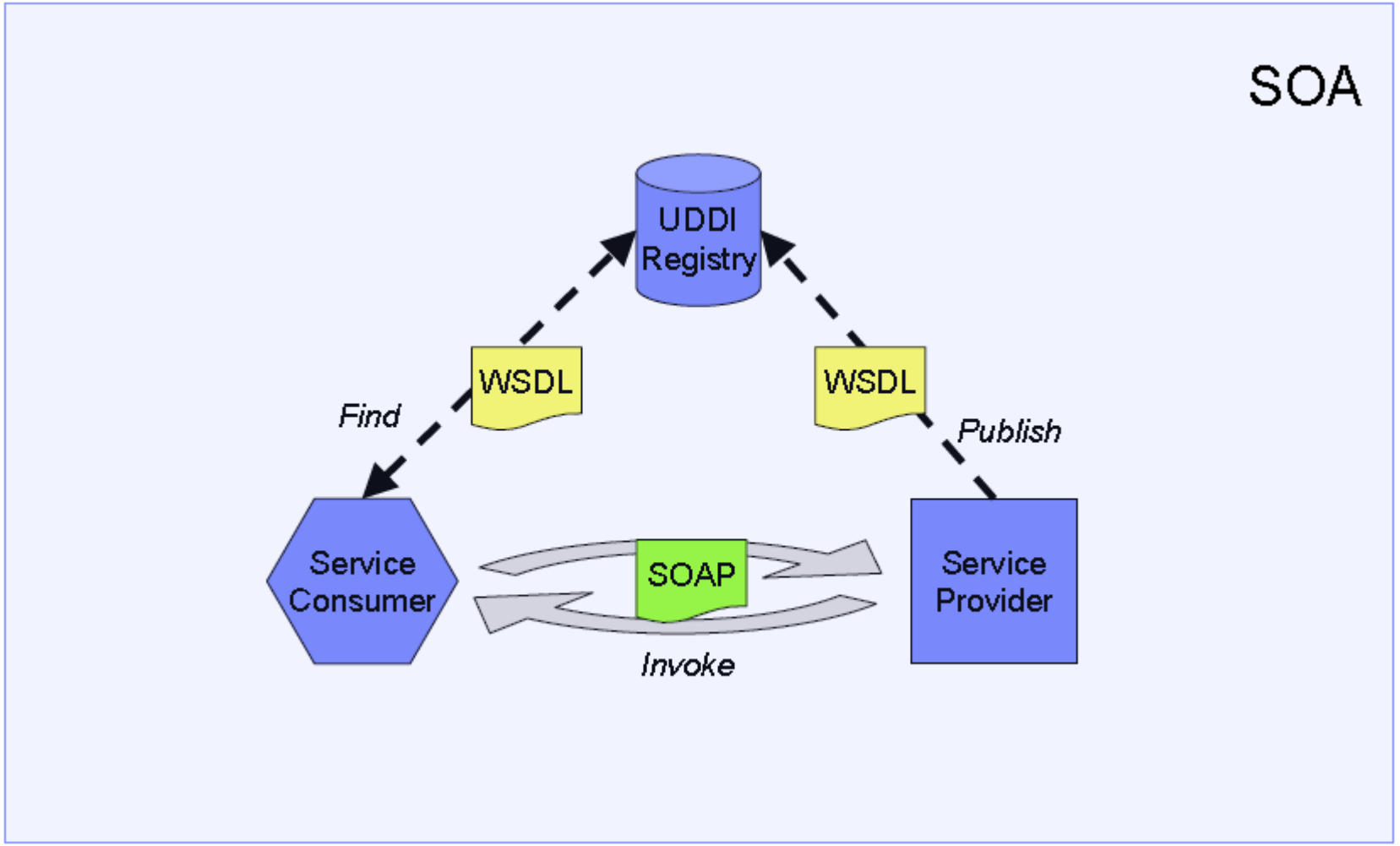
Name: Edwin van de Grift
Venue: Ongoing TPF Education

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

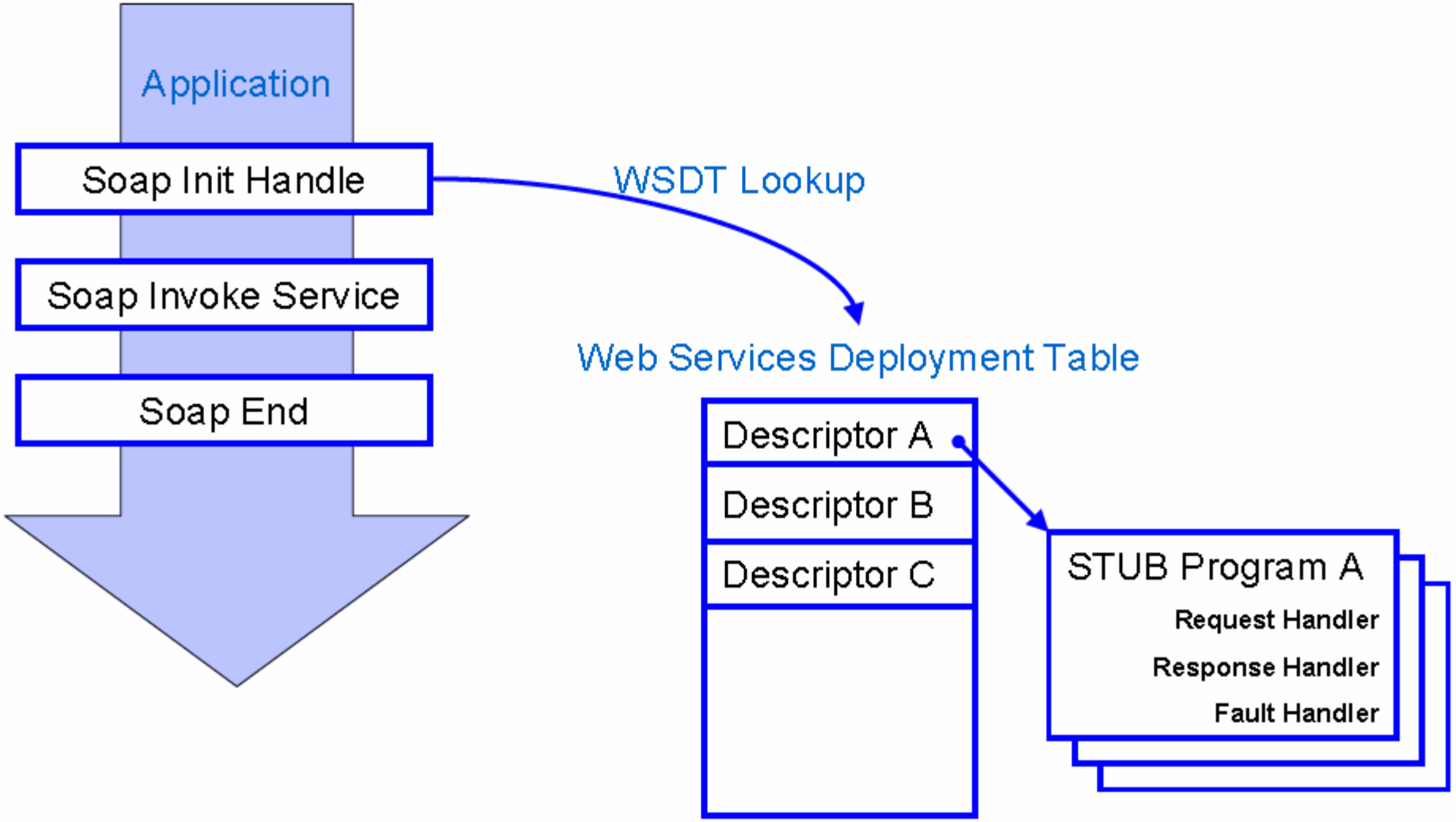
Contents

- **Web Services**
- **Soap Consumer Flow**
- **Accessing a Web Service from your z/TPF application**

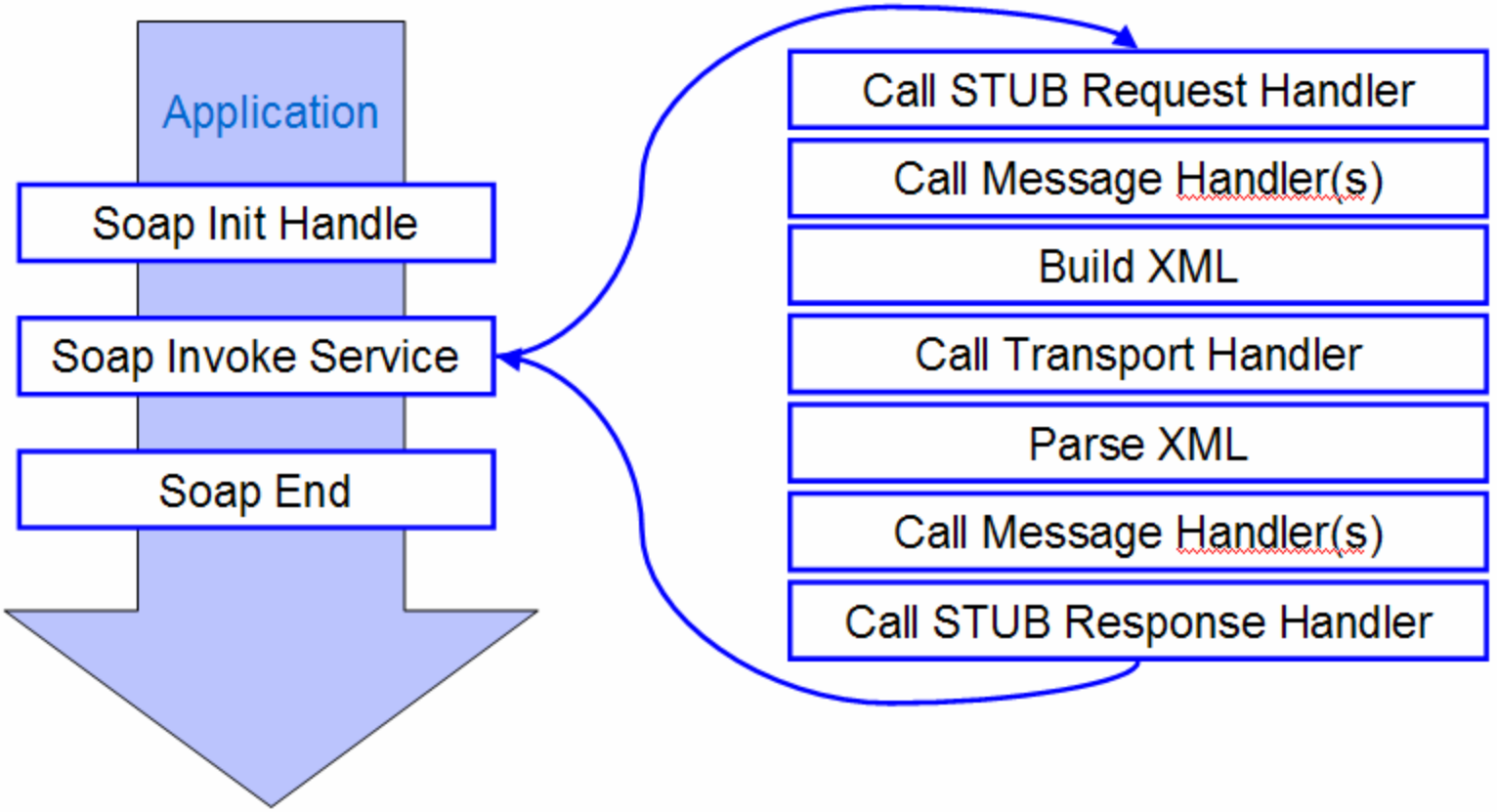
Web Services



Soap Consumer Flow



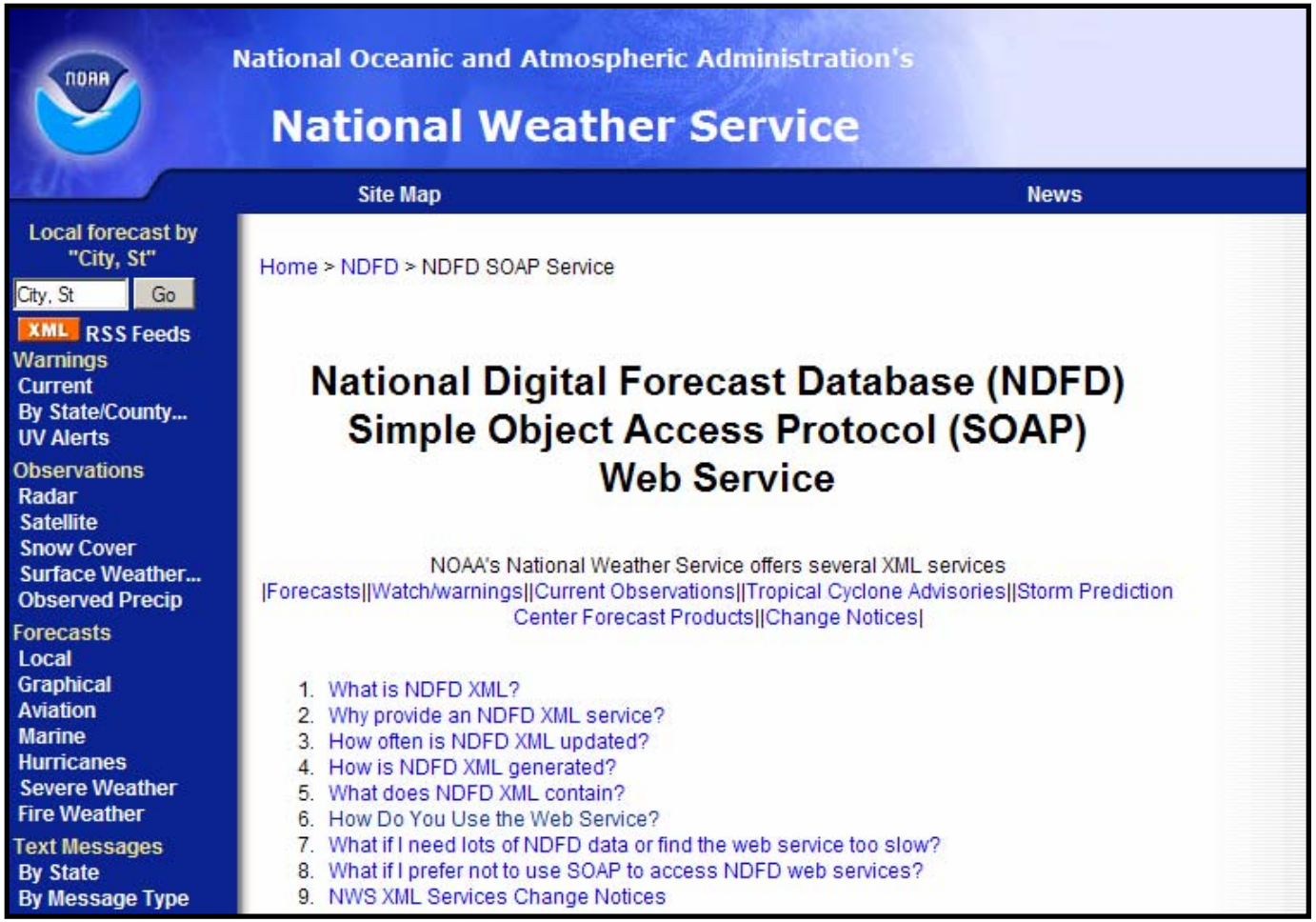
Soap Consumer Flow



Summary

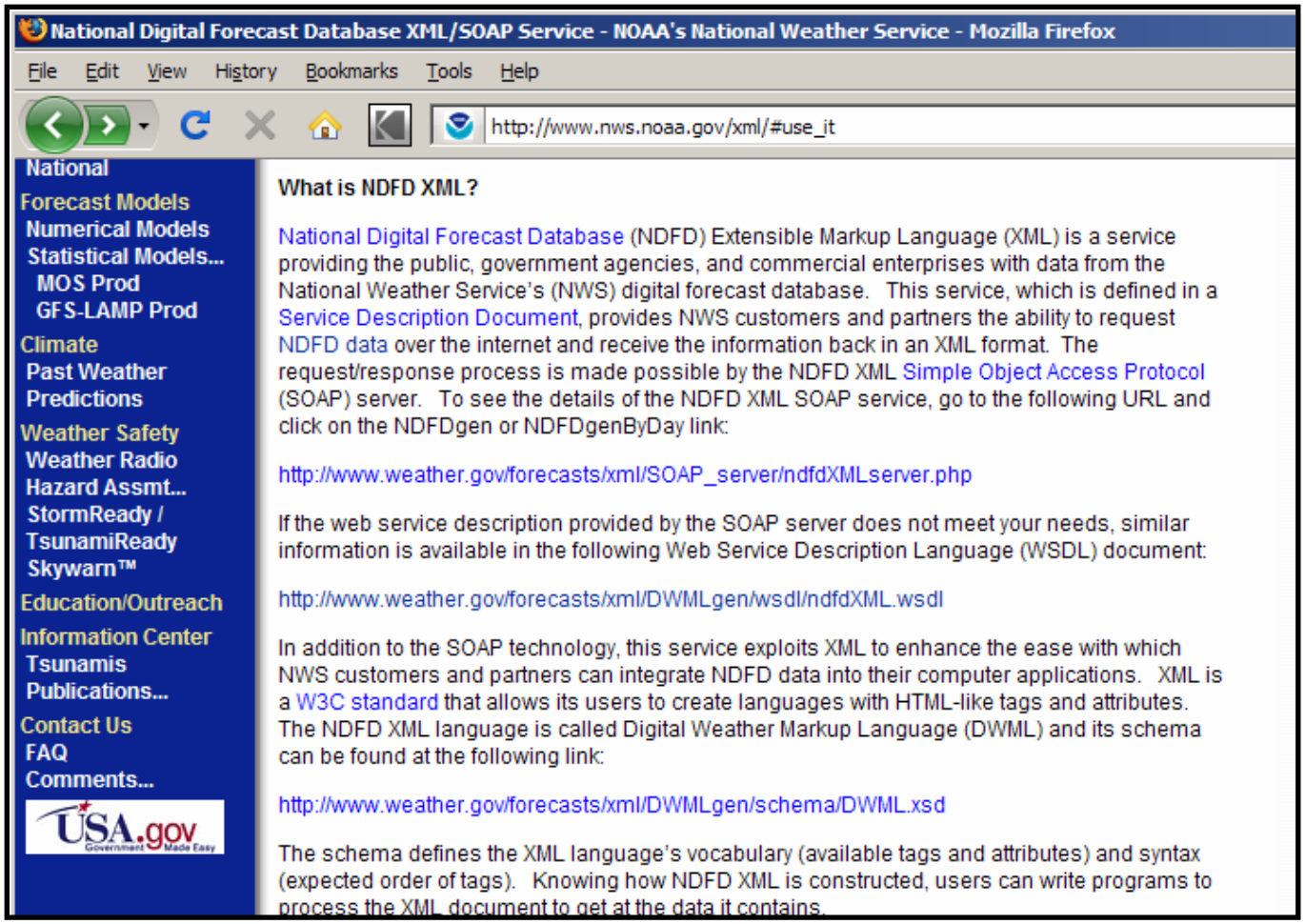
- **Get the Service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**
- **Invoke the Web Service**

What We Will Consume



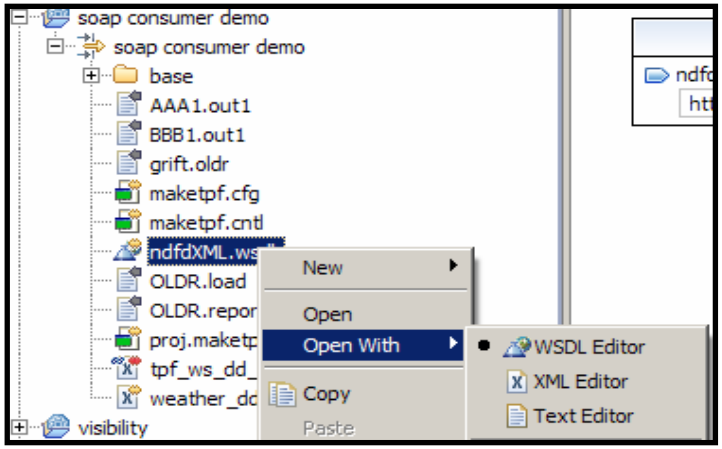
The NWS material incorporated is not subject to copyright protection.

Download the WSDL

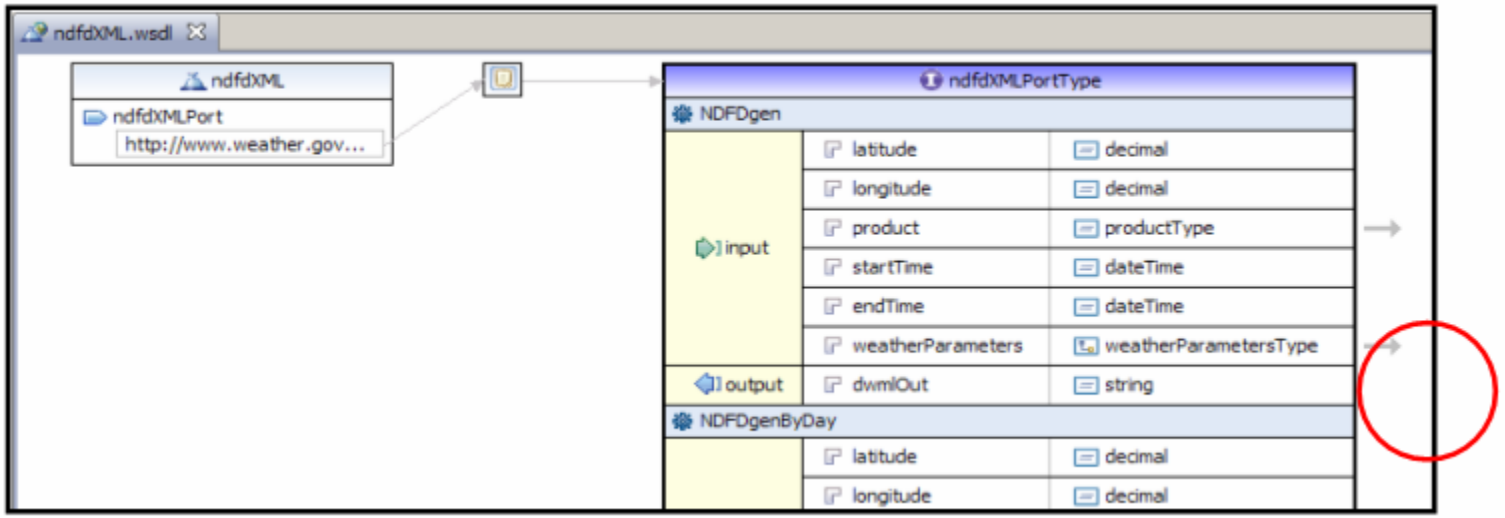


The NWS material incorporated is not subject to copyright protection.

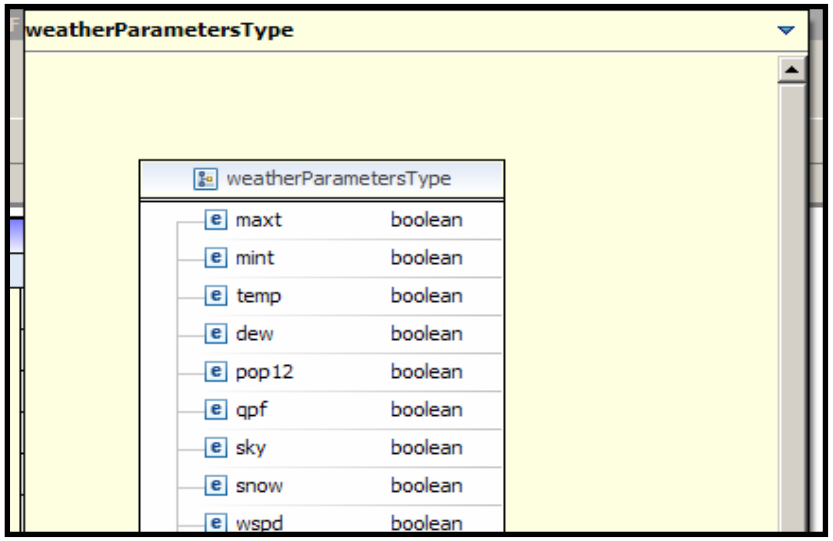
WSDL Management



WSDL Editor



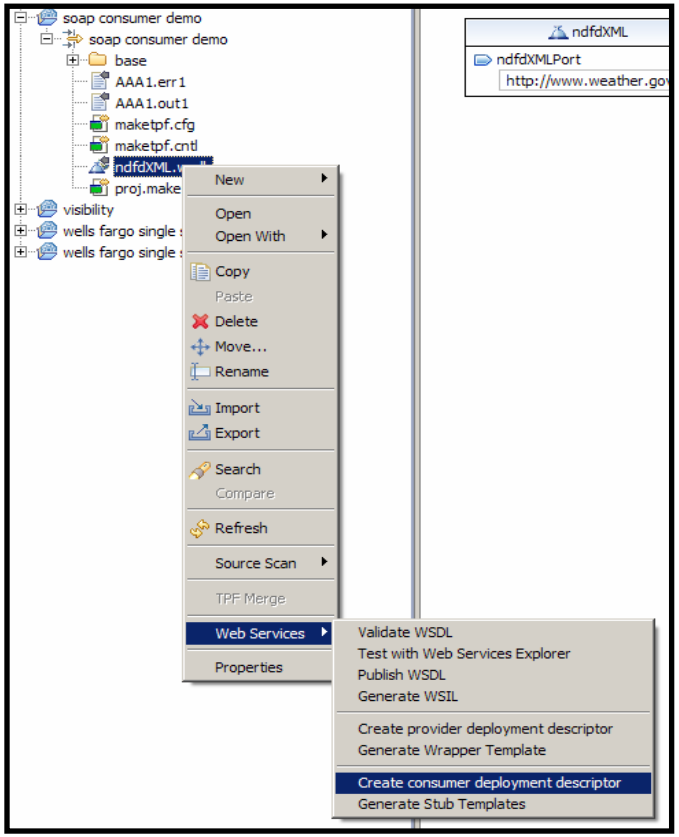
WSDL Editor



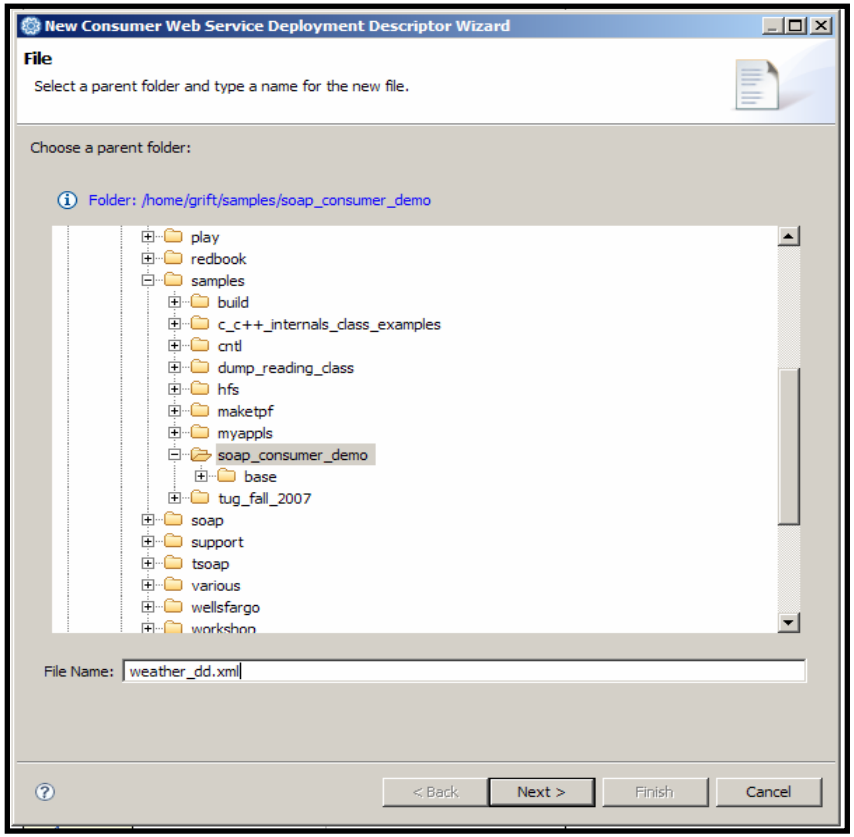
Summary

- **Get the Service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**
- **Invoke the Web Service**

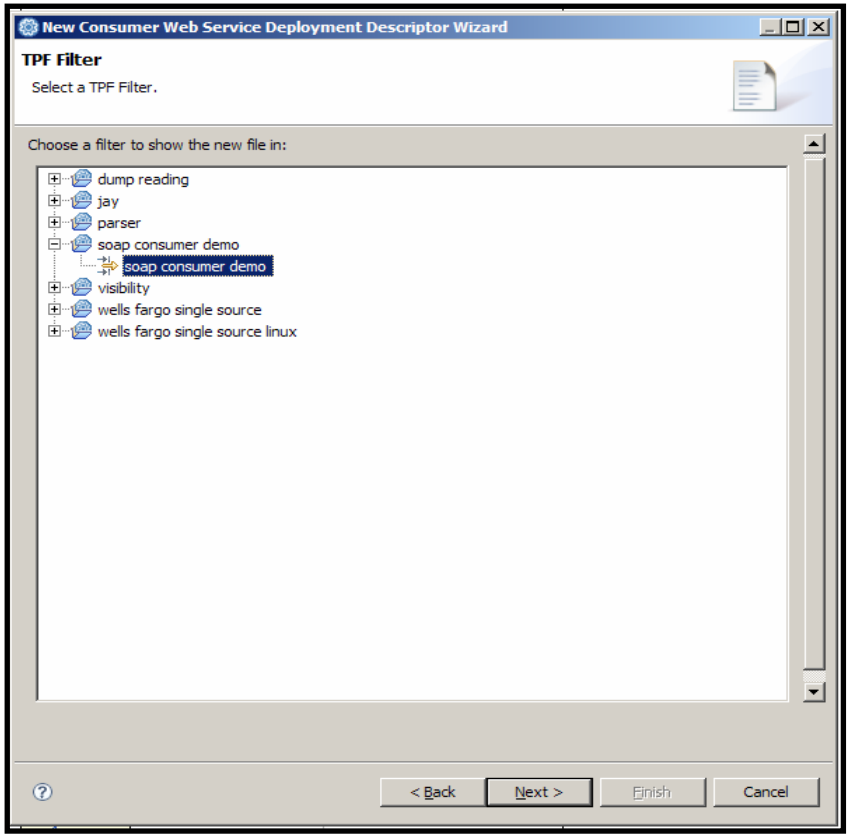
Create Consumer Deployment Descriptor



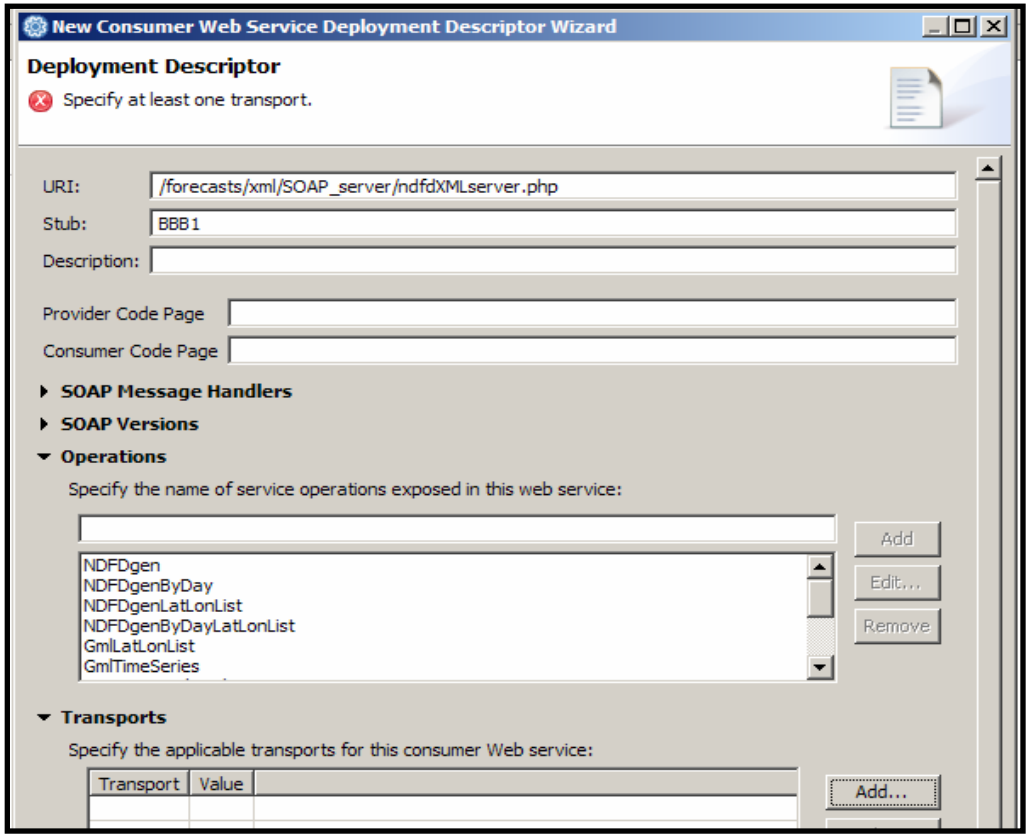
Name the Deployment Descriptor



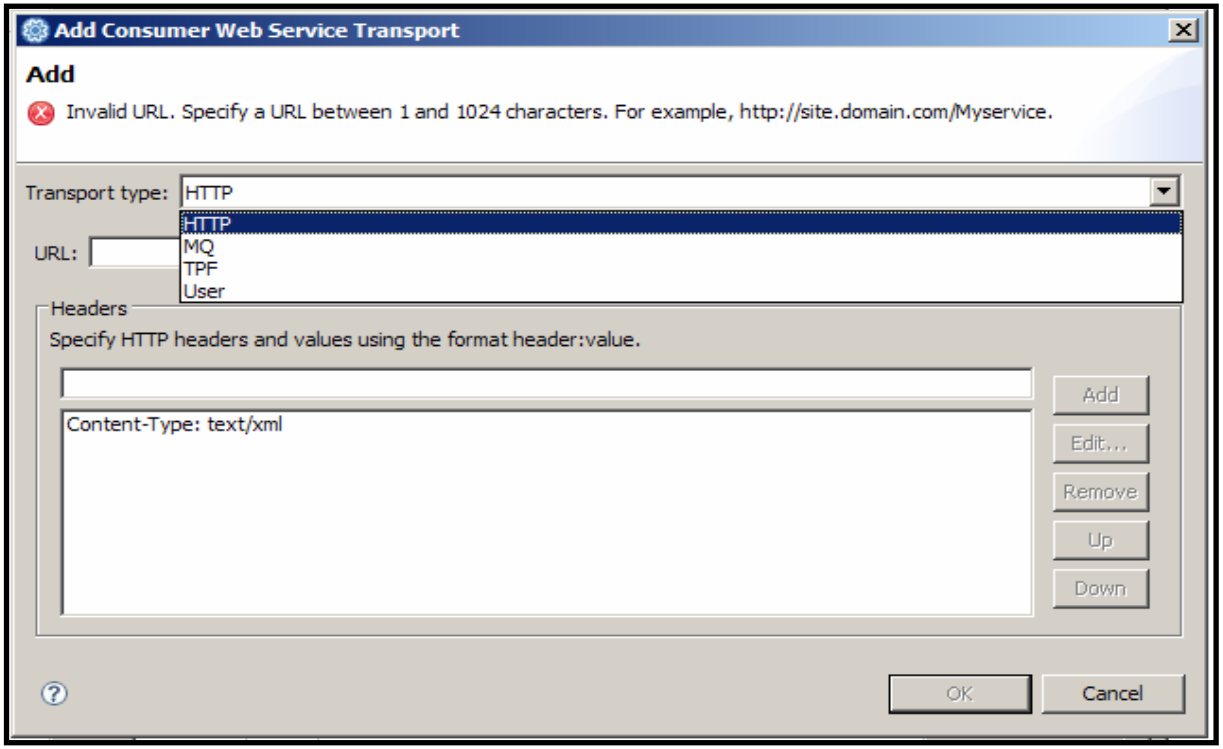
Select a TPF Filter



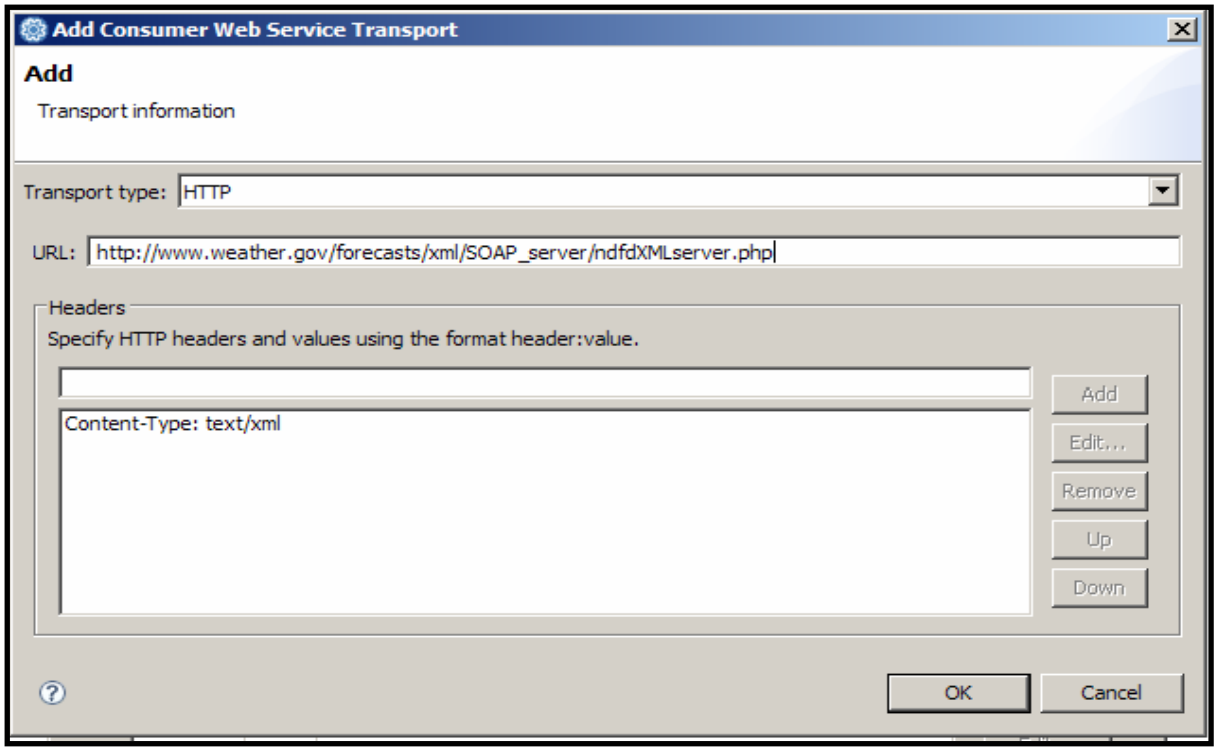
Specify the STUB Name



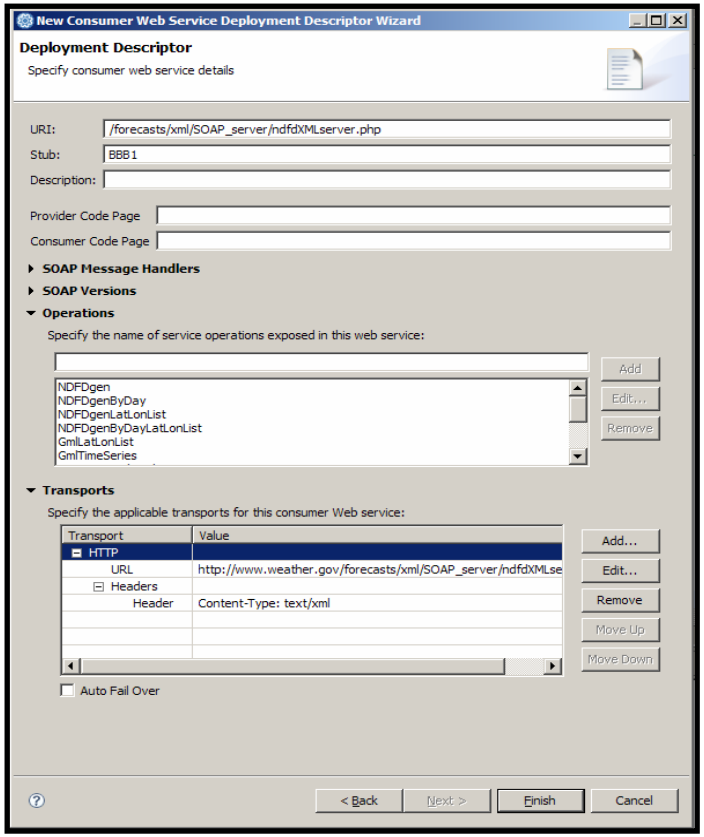
Add a Transport



Add a Transport



Finish the Descriptor Wizard

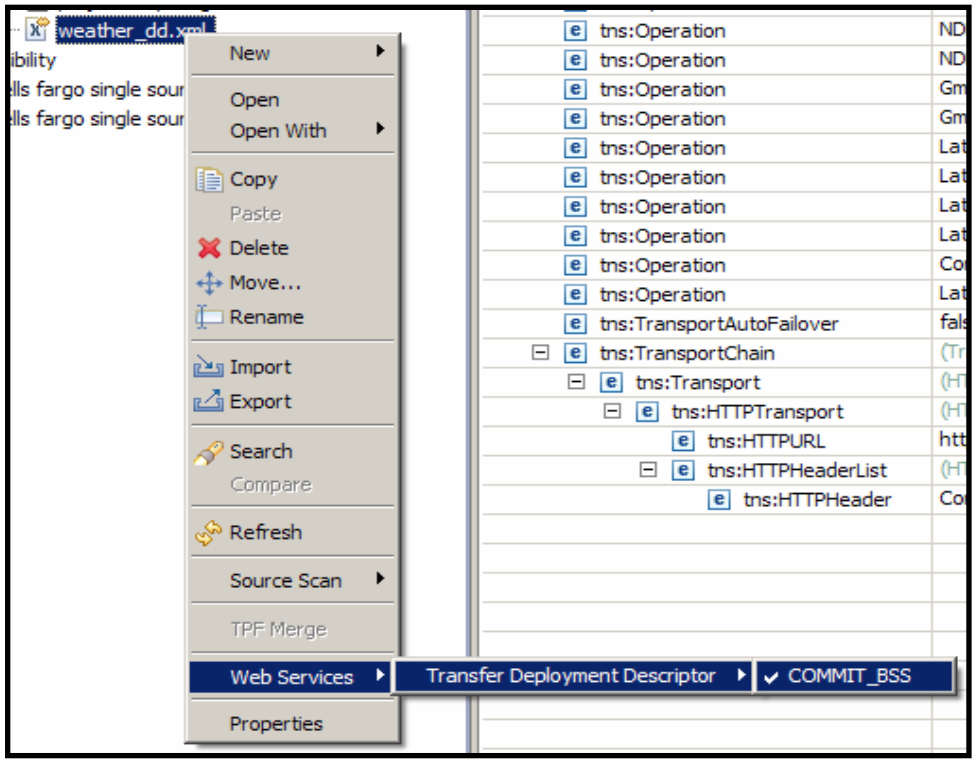


The Deployment Descriptor

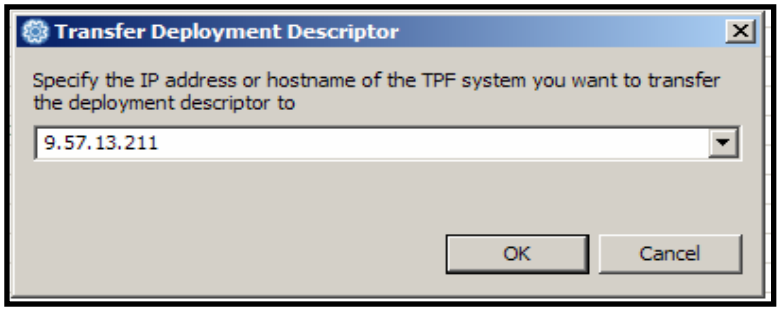
aaa1.cpp | bbb1_request.c | bbb1_response.c | weather_dd.xml | weather_dd.xml

xml	version="1.0" encoding="UTF-8"
tns:ConsumerDeploymentDescriptor	(ServiceURI, Stub, SOAPProcessing, Operation*, Description?, TransportAutoFailover, Co
xmlns:tns	http://www.ibm.com/xmlns/prod/ztpf/soap
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation	http://www.ibm.com/xmlns/prod/ztpf/soap file:/etc/tpf-ws/schema/tpf_ws_dd_v2.xsd
tns:ServiceURI	/forecasts/xml/SOAP_server/ndfdXMLserver.php
tns:Stub	BBB1
tns:SOAPProcessing	(SOAPMessageHandlerChain*, VerifySOAPHeaders, SOAPVersion*)
tns:VerifySOAPHeaders	false
tns:SOAPVersion	SOAP1.1
tns:Operation	NDFDgen
tns:Operation	NDFDgenByDay
tns:Operation	NDFDgenLatLonList
tns:Operation	NDFDgenByDayLatLonList
tns:Operation	GmlLatLonList
tns:Operation	GmlTimeSeries
tns:Operation	LatLonListSubgrid
tns:Operation	LatLonListLine
tns:Operation	LatLonListZipCode
tns:Operation	LatLonListSquare
tns:Operation	CornerPoints
tns:Operation	LatLonListCityNames
tns:TransportAutoFailover	false
tns:TransportChain	(Transport+)
tns:Transport	(HTTPTransport | MQTransport | TPFTransport | UserTransport)
tns:HTTPTransport	(HTTPURL, HTTPHeaderList?)
tns:HTTPURL	http://www.weather.gov/forecasts/xml/SOAP_server/ndfdXMLserver.php
tns:HTTPHeaderList	(HTTPHeader+)
tns:HTTPHeader	Content-Type: text/xml

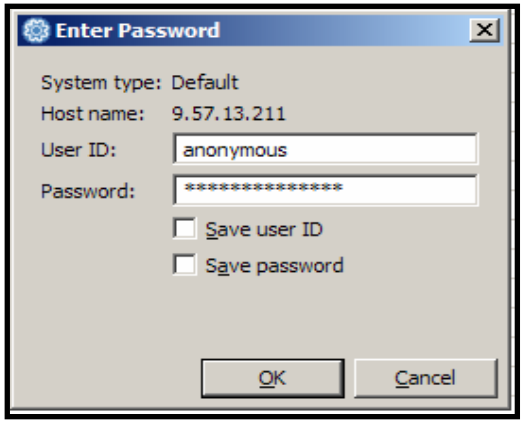
Transfer the Deployment Descriptor



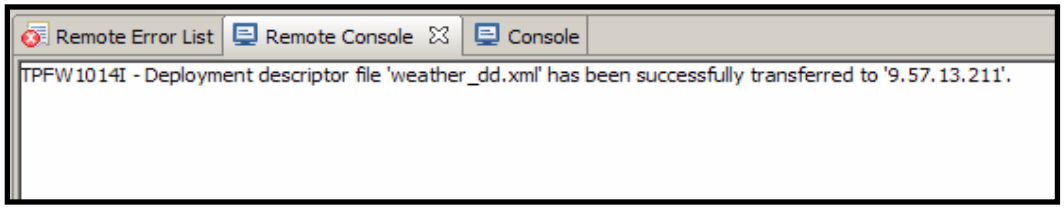
Specify the TPF System's Address



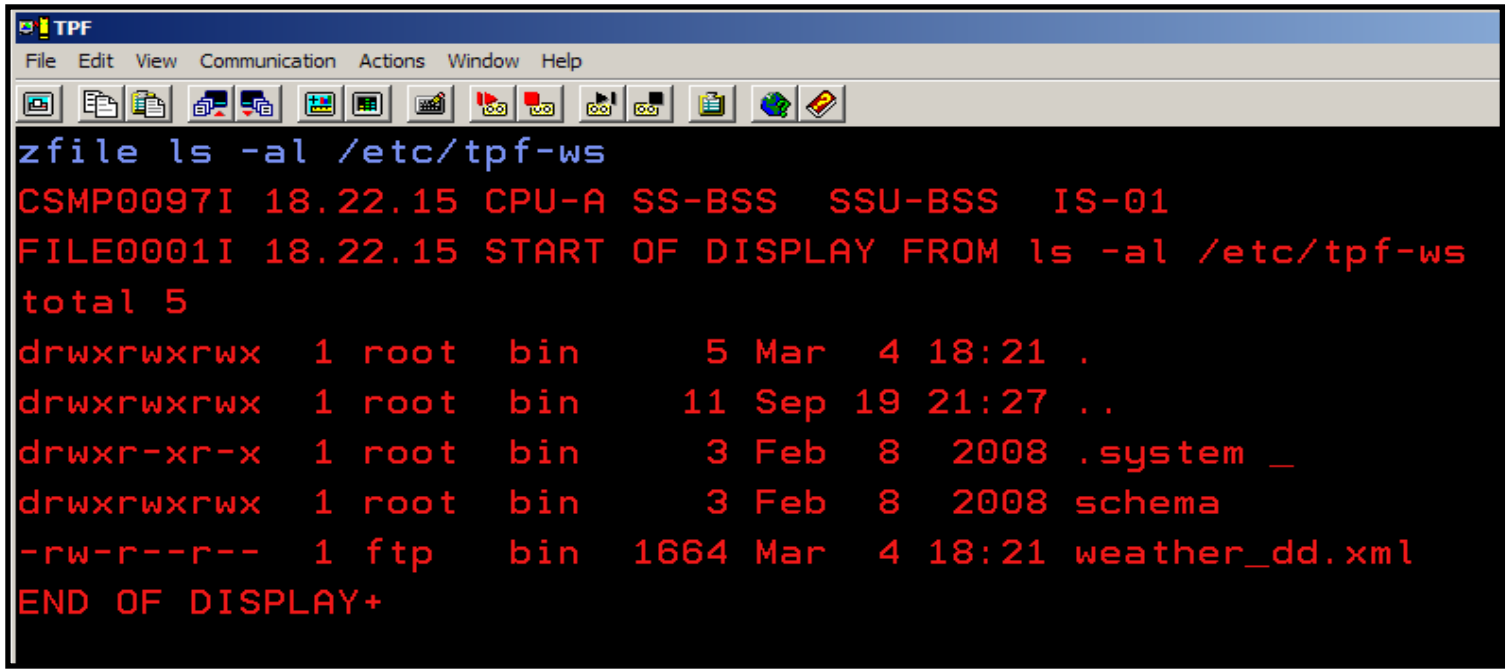
Authenticate with FTP



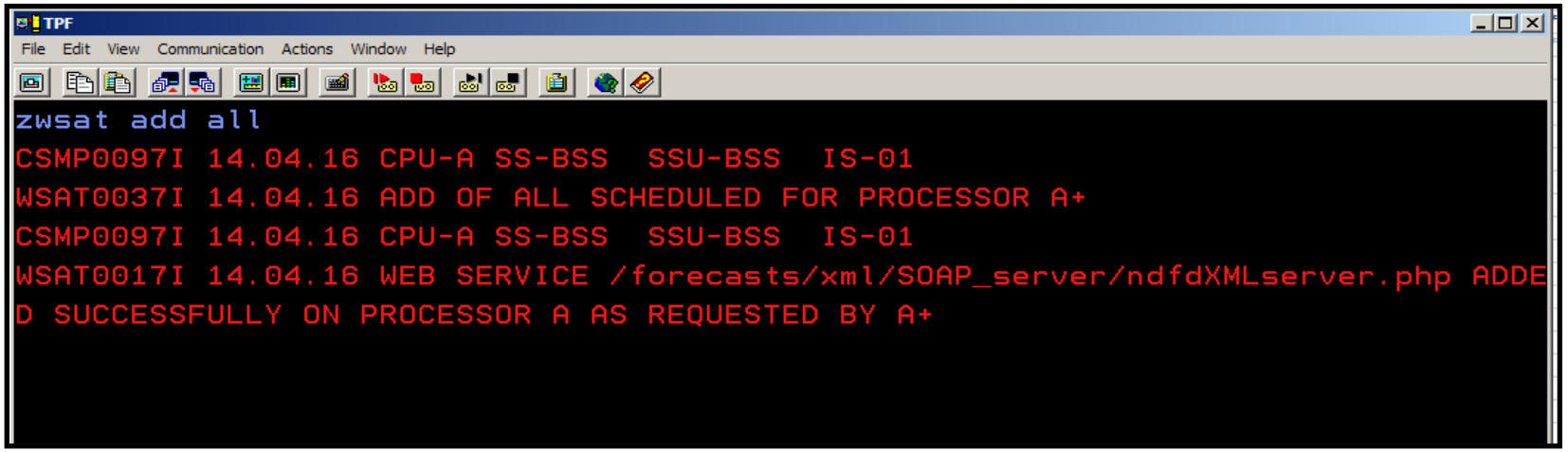
FTP Results



The Descriptor on TPF



Add the Descriptor



Summary

- **Get the Service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**
- **Invoke the Web Service**

The STUB

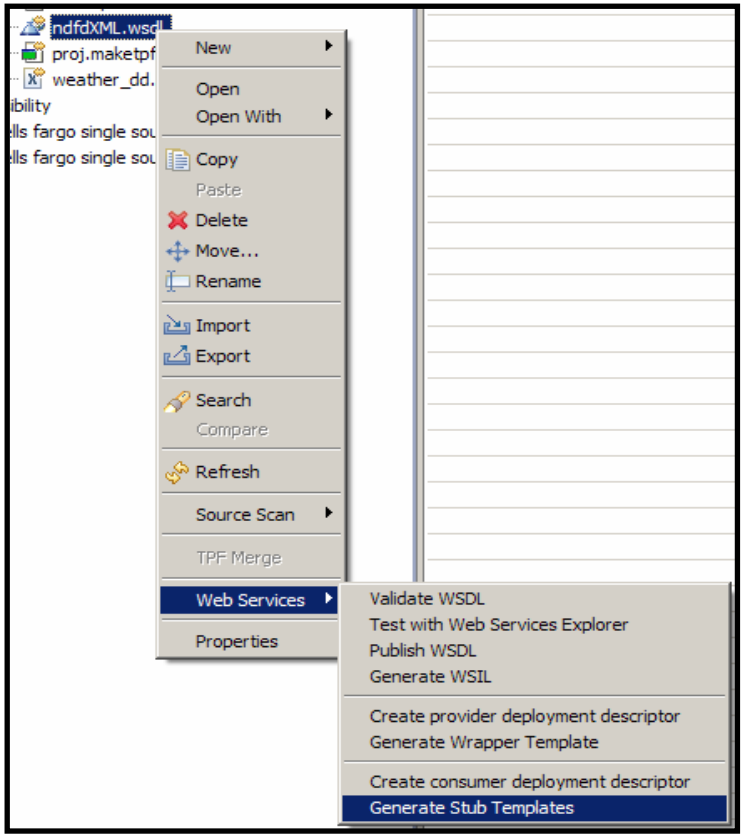
- **An isolation point for the portion of code that:**
  - Serializes the operation and parameter data into the body portion of the SOAP request.
  - Deserializes the return from the service provider from the parsed SOAP response.

- **This hides the following details from the application:**
  - Message formats
  - Service locations
  - Communication protocol information

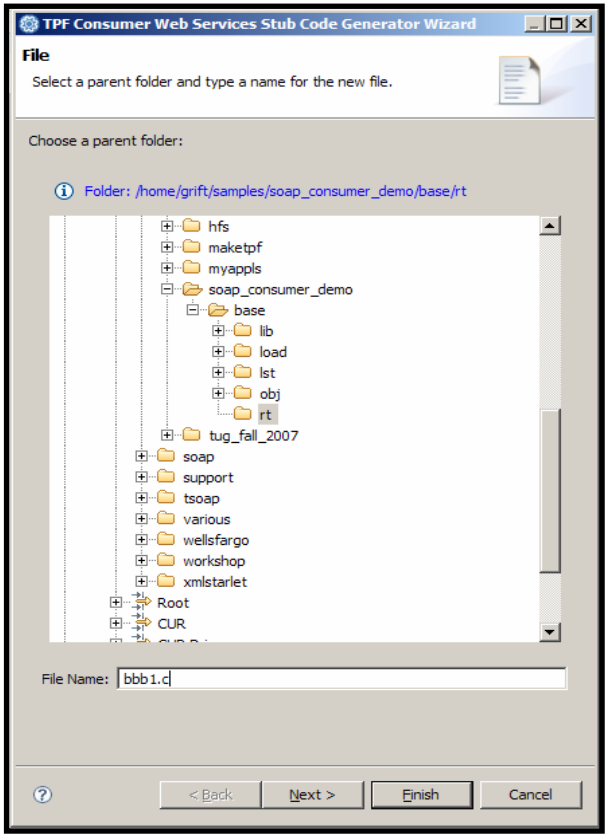
The STUB

- **Uses TPF_CALL_BY_NAME to call the Web service stub program name**
- **Processing is implemented in C shared objects**

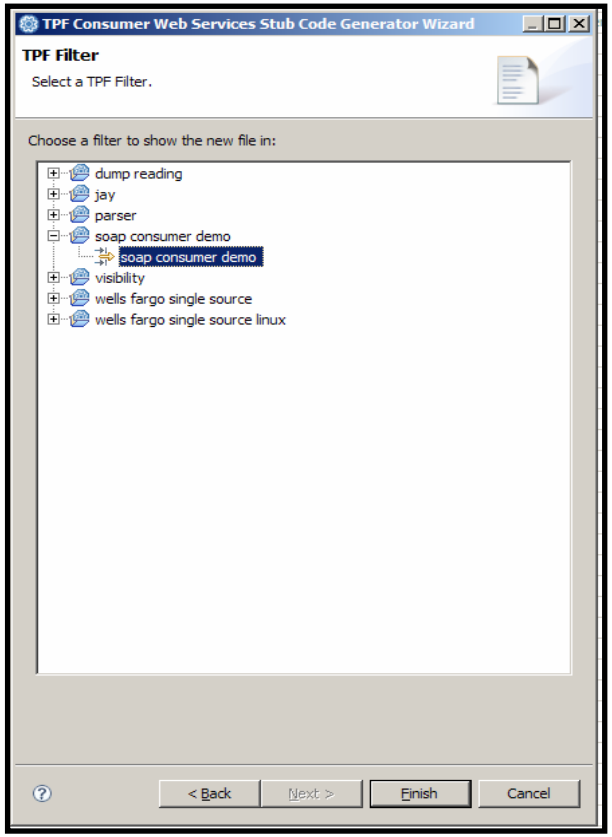
Generate STUB Templates



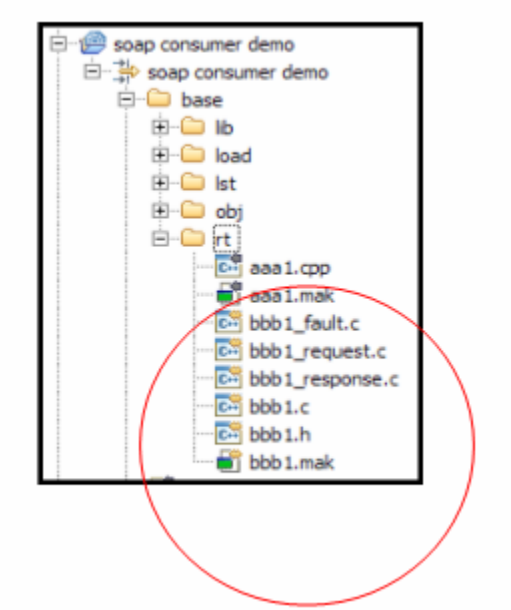
Specify STUB Location & Name



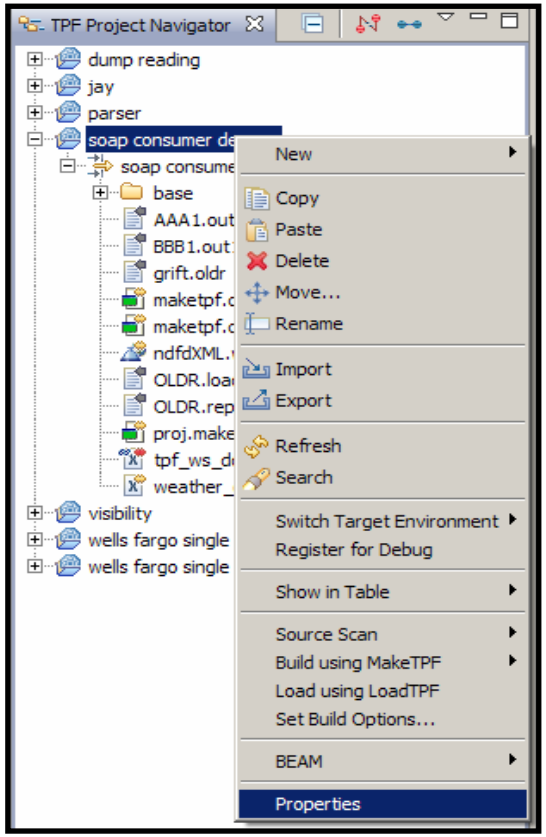
Specify a Filter



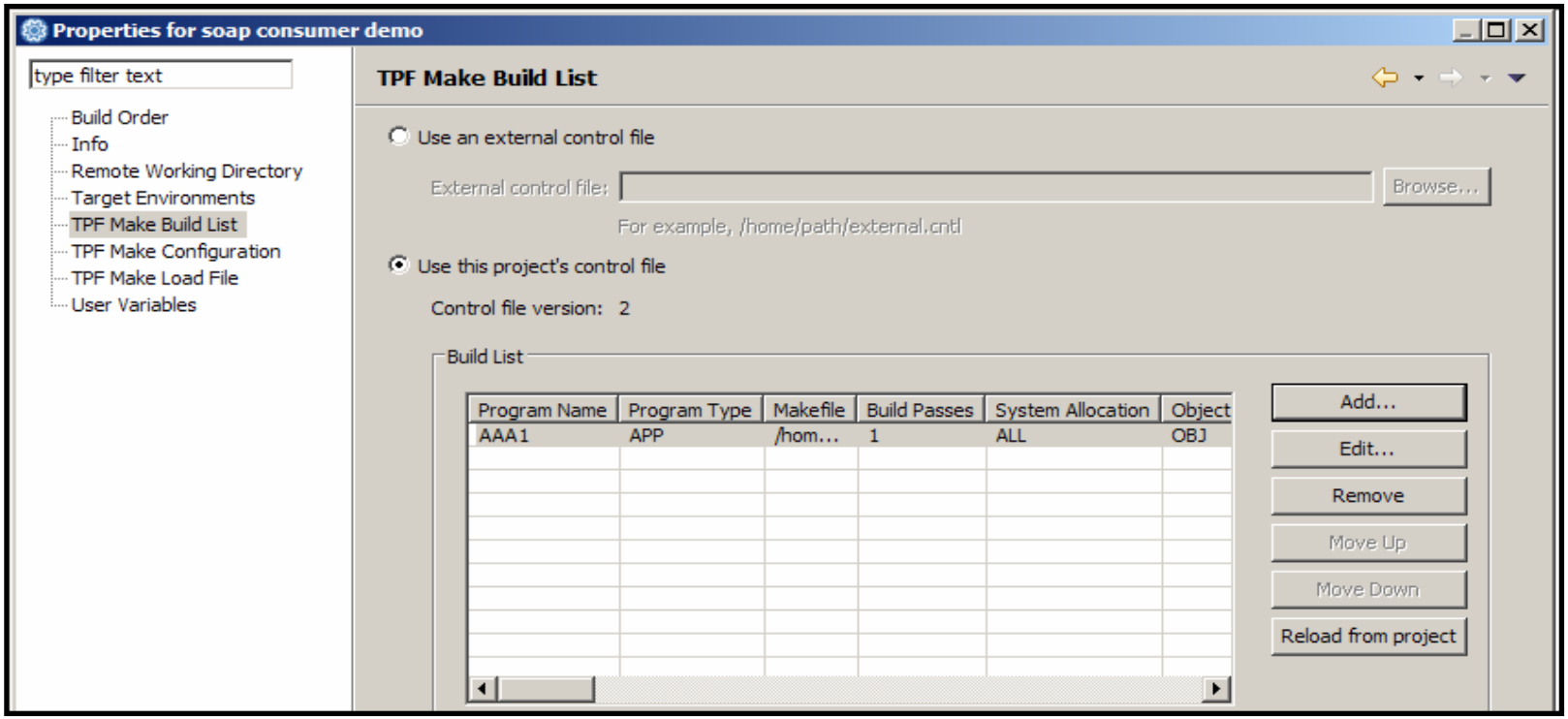
Generated Template Segments



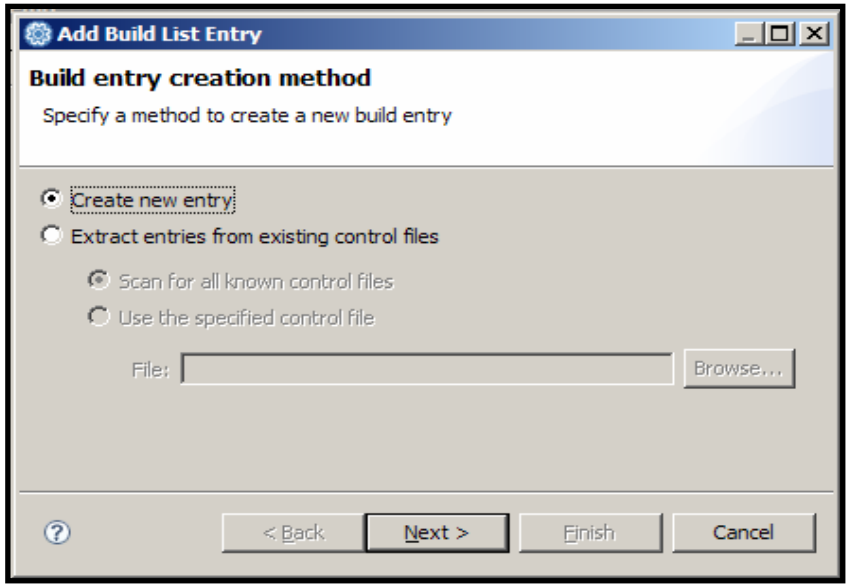
Add the STUB to the Project



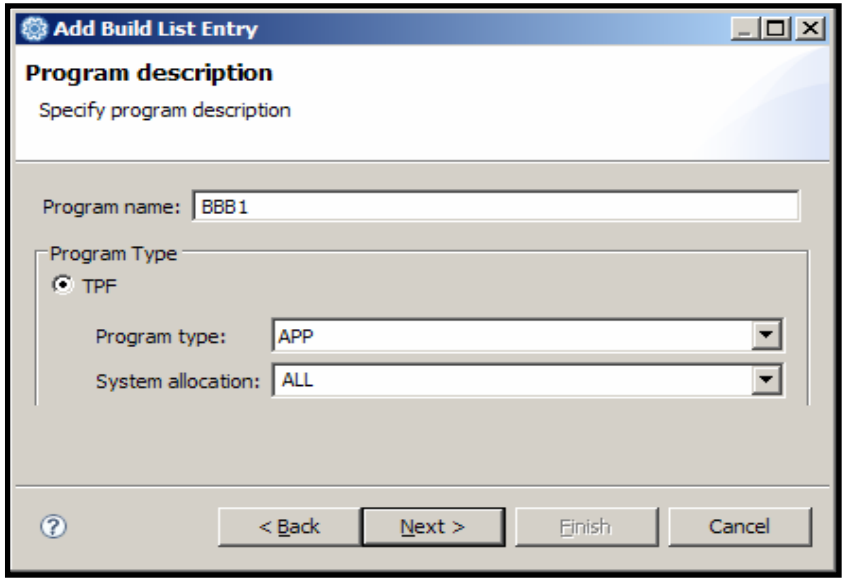
Add the STUB to the Build List



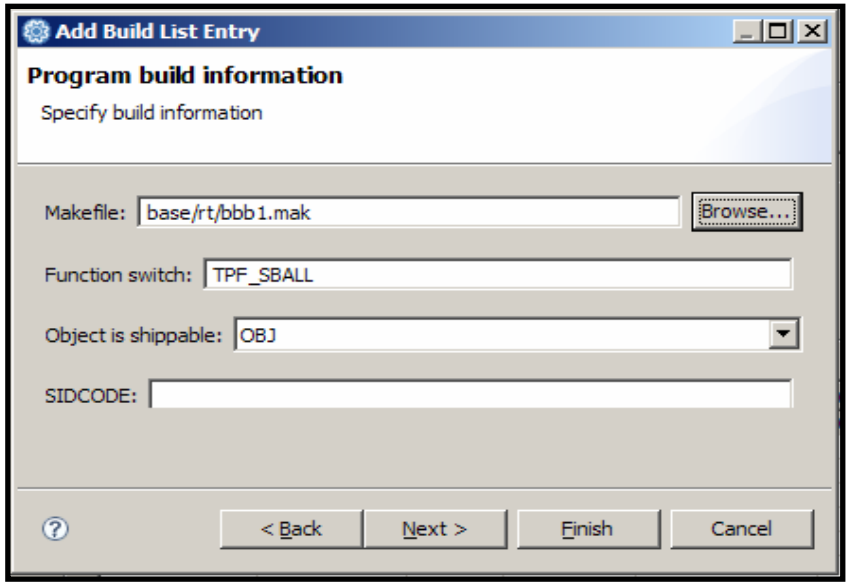
Create a New Build List Entry



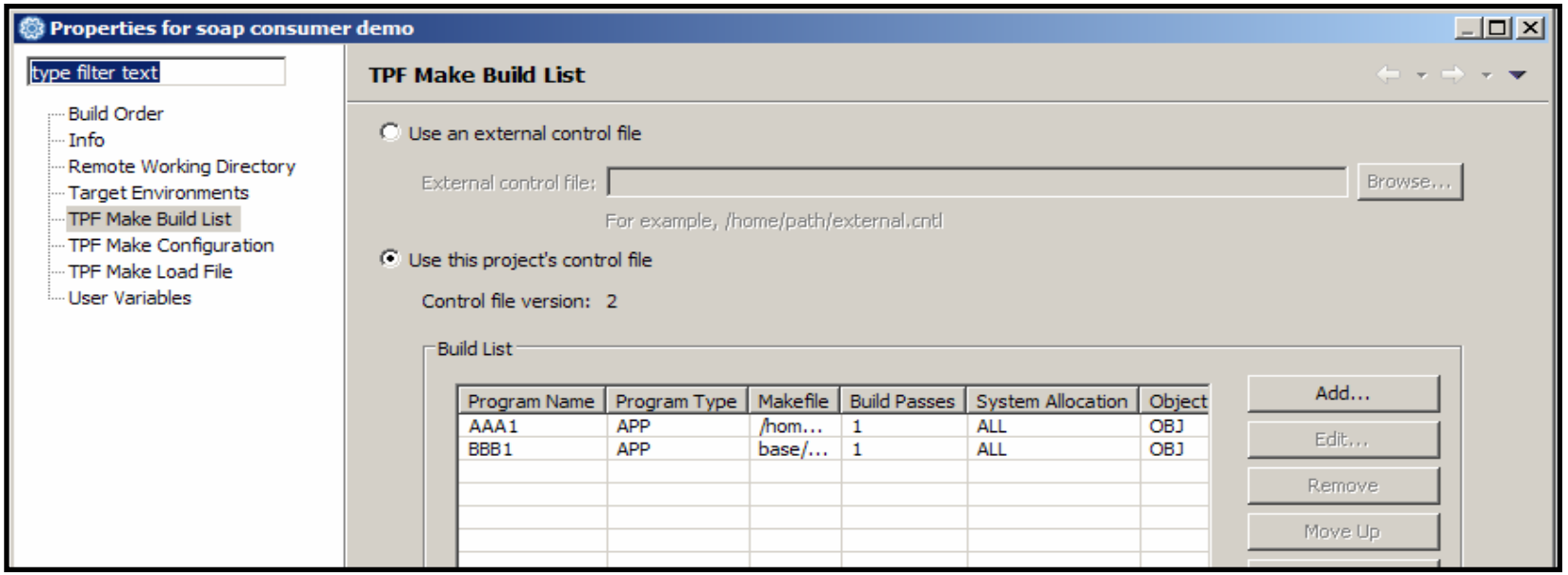
Specify the STUB Program Name



Specify the STUB Make File



Add the STUB to the Control File



Request Message

```
<Body>
   <NDFDGen>
      <latitude>41.90</latitude>
      <longitude>-87.65</longitude>
      <product>glance</product>
      <startTime>2009-03-10T12:00</startTime>
      <endTime>2009-03-10T12:00</endTime>
      <weatherParameters>maxt=TRUE</weatherParameters>
   </NDFDGen>
</Body>
```

Response Message

```
<Body>
   <NDFDGen>
      <dwmlOut>

         ...

         <parameters applicable-location="point1">
            <temperature type="maximum" units="Fahrenheit" time-layout="k-p24h-n1-1">
               <name>Daily Maximum Temperature</name>
               <value>51</value>
            </temperature>
         </parameters>

         ...

      </dwmlOut>
   </NDFDGen>
</Body>
```

Generated Request Handler

```
int bbb1_NDFDgen_request (t_soapHandle soapHandle, XMLHandle requestMsg,
                          t_soapParms *parmsArray, void **returnValue)
{
          int rc = WSSTUB_RET_CONT;
          /* Function return code      */

          /*************************************************************/
          /* ADD YOUR CODE HERE TO APPEND THE NECESSARY ELEMENTS UNDER  */
          /* THE "BODY" ELEMENT TO BUILD A VALID SOAP REQUEST MESSAGE   */
          /* FOR THIS OPERATION.                                        */
          /*************************************************************/

          /*************************************************************/
          /* Return to tpf_soapInvokeService processing.                */
          /*************************************************************/
          return (rc);

} /* end of function bbb1_NDFDgen_request ( )            */
```

Code the Request Handler 1/2

```
int bbb1_NDFDgen_request (t_soapHandle soapHandle, XMLHandle requestMsg,
                          t_soapParms *parmsArray, void **returnValue)
{
   int rc = WSSTUB_RET_CONT;
   int xmlRC = 0;
   xmlRC = tpf_xml_appendElement(requestMsg, "Body", "NDFDgen",
                                 NULL, BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
   if(xmlRC == TPF_SOAP_ERROR) {
      // Error handling
   }
   xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "latitude",
                                 parmsArray->parmArray[0].parmPtr,
                                 BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
   if(xmlRC == TPF_SOAP_ERROR) {
      // Error handling
   }
   xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "longitude",
                                 parmsArray->parmArray[1].parmPtr,
                                 BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
   if(xmlRC == TPF_SOAP_ERROR) {
      // Error handling
   }
```

Code the Request Handler 2/2

```
    xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "product",
                                  parmsArray->parmArray[2].parmPtr,
                                  BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
    if(xmlRC == TPF_SOAP_ERROR) {
       // Error handling
    }
    xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "startTime",
                                  parmsArray->parmArray[3].parmPtr,
                                  BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
    if(xmlRC == TPF_SOAP_ERROR) {
       // Error handling
    }
    xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "endTime",
                                  parmsArray->parmArray[4].parmPtr,
                                  BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
    if(xmlRC == TPF_SOAP_ERROR) {
       // Error handling
    }
    xmlRC = tpf_xml_appendElement(requestMsg, "NDFDgen", "weatherParameters",
                                  parmsArray->parmArray[5].parmPtr,
                                  BBB1_NAMESPACE_PREFIX, BBB1_NAMESPACE, 0);
    if(xmlRC == TPF_SOAP_ERROR) {
       // Error handling
    }
    return (rc);
}
```

Generated Response Handler

```
int bbb1_NDFDgen_response (t_soapHandle soapHandle, XMLHandle responseMsg,
                           void **returnValue)
{
          int            rc = WSSTUB_RET_SUCCESS;
          /* Function return code      */

          /*************************************************************/
          /* ADD YOUR CODE HERE TO EXTRACT THE NECESSARY DATA FROM THE    */
          /* SOAP RESPONSE MESSAGE AND CREATE THE REQUIRED INFORMATION    */
          /* TO BE PASSED BACK TO THE ORIGINAL CALLING APPLICATION.       */
          /*************************************************************/

          /*************************************************************/
          /* Return to tpf_soapInvokeService processing.                  */
          /*************************************************************/
          return (rc);

} /* end of function bbb1_NDFDgen_response ( )          */
```

Code the Response Handler 1/2

```
int bbb1_NDFDgen_response (t_soapHandle soapHandle, XMLHandle responseMsg,
                           void **returnValue)
{
   int rc = WSSTUB_RET_SUCCESS;
   xmlNodesArray* xptr = NULL;
   xmlNodesArray* wptr = NULL;
   int wRc;
   int parserRc;
   XMLHandle w;

   xptr = tpf_xml_getElementsByTagName (responseMsg, TYPE_TEXT, "dwmlOut");
   if(xptr == NULL) || (xptr->nodesArraySize == 0) {
      // Error handling
   }
```

Code the Response Handler 2/2

```
   wRc = tpf_xml_initialize_handle(&w, B2B_XML_SCANNER, NULL);
   if((wRc != 0) || (w == 0)) {
      // Error handling
   }

   wRc = tpf_xml_parseDocument(w, xptr->nodesArray[0].nodeValueStr,
                               TPF_CCSID_IBM1047,
                               strlen(xptr->nodesArray[0].nodeValueStr),
                               &parserRc, 0);
   if(wRc != 0) {
      // Error Handling
   }

   wptr = tpf_xml_getElementsByTagName (w, TYPE_INT, "value");
   if((wptr == NULL) || (wptr->nodesArraySize == 0)){
      // Error handling
   }

   *returnValue = malloc(sizeof(int));
   *(int*)*returnValue = wptr->nodesArray[0].nodeValueDataType.nodeValueInt;
   return (rc);
}
```

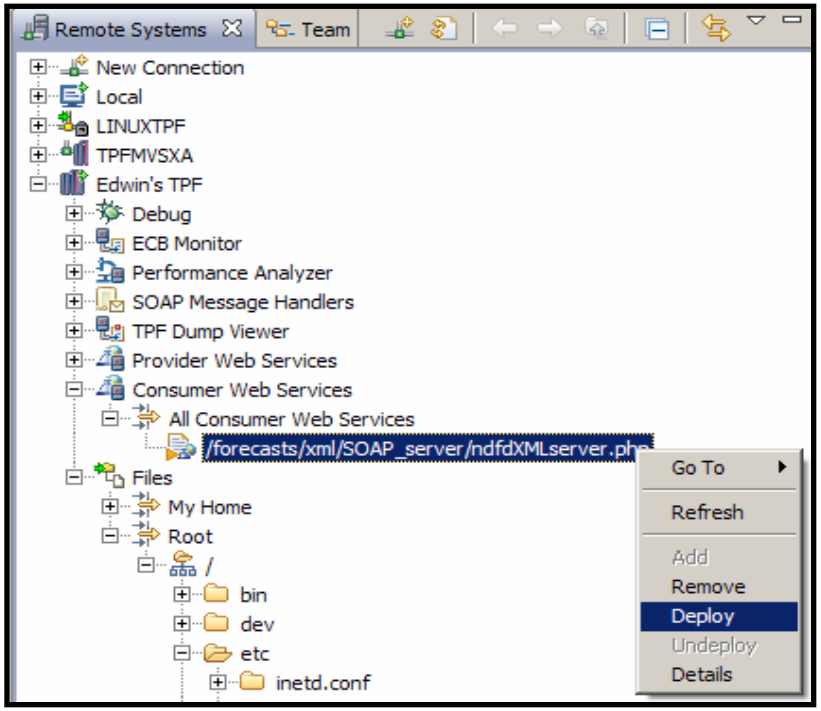
Code the Fault Handler

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


int bbb1_NDFDgen_fault (t_soapHandle soapHandle, XMLHandle faultMsg, void **returnValue)
{
   int rc = WSSTUB_RET_CONT; /* Function return code      */

   /*************************************************************/
   /* ADD YOUR CODE HERE TO EXTRACT THE NECESSARY DATA FROM THE    */
   /* SOAP FAULT MESSAGE AND CREATE THE REQUIRED INFORMATION TO BE */
   /* PASSED BACK TO THE ORIGINAL CALLING APPLICATION.             */
   /*************************************************************/

   wtopc("bbb1 NFDgen fault", WTOPC_EBROUT + WTOPC_PRC, NULL, NULL); /* edwin */

   /*************************************************************/
   /* Return to tpf_soapInvokeService processing.                  */
   /*************************************************************/
   return (rc);
}/* end of function bbb1_NDFDgen_fault ( )              */
```

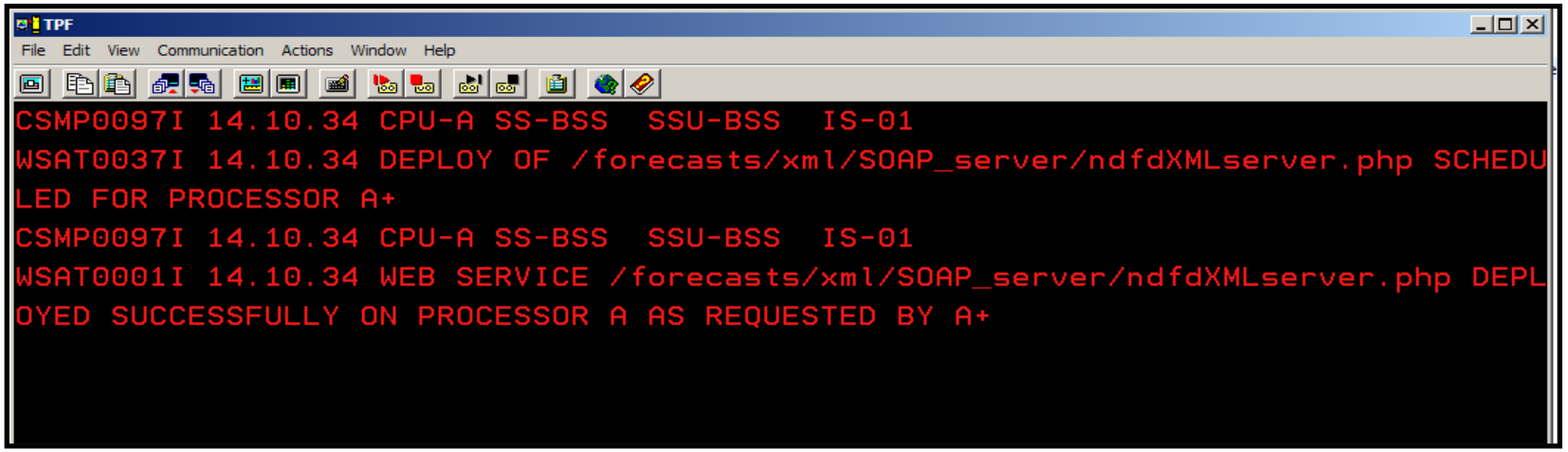
Summary

- **Get the Service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**
- **Invoke the Web Service**

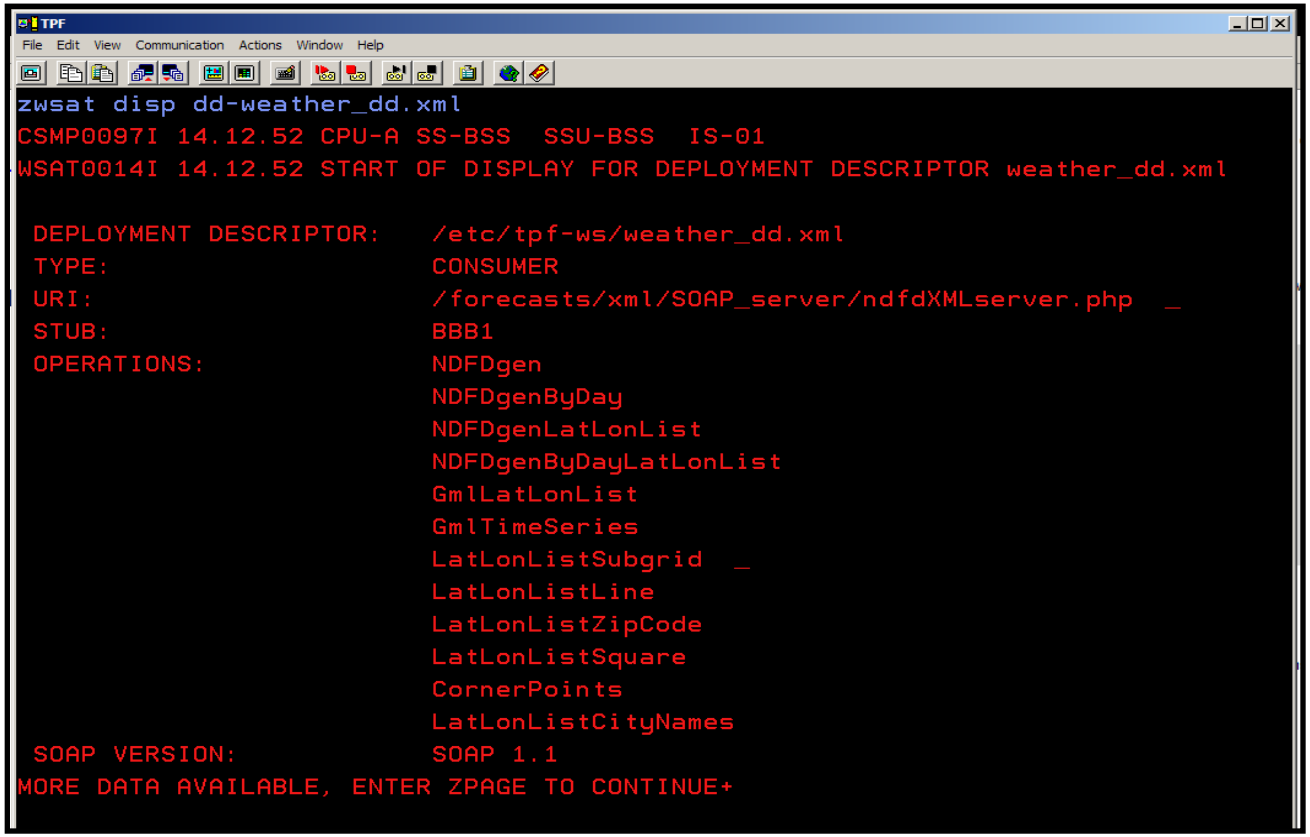
Deploy the Web Service



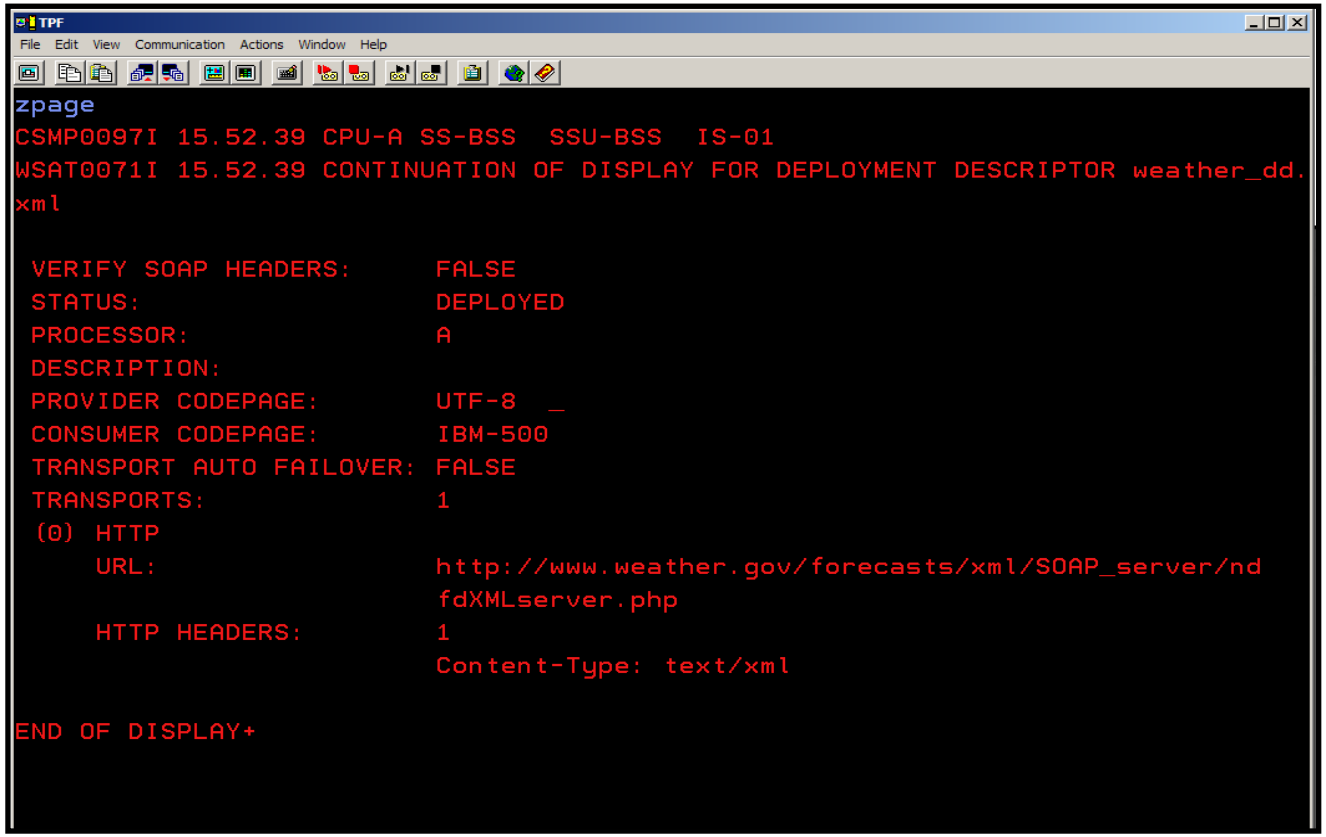
Deploy the Web Service



Display the Web Service



Display the Web Service



Summary

- **Get the Service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**
- **Invoke the Web Service**

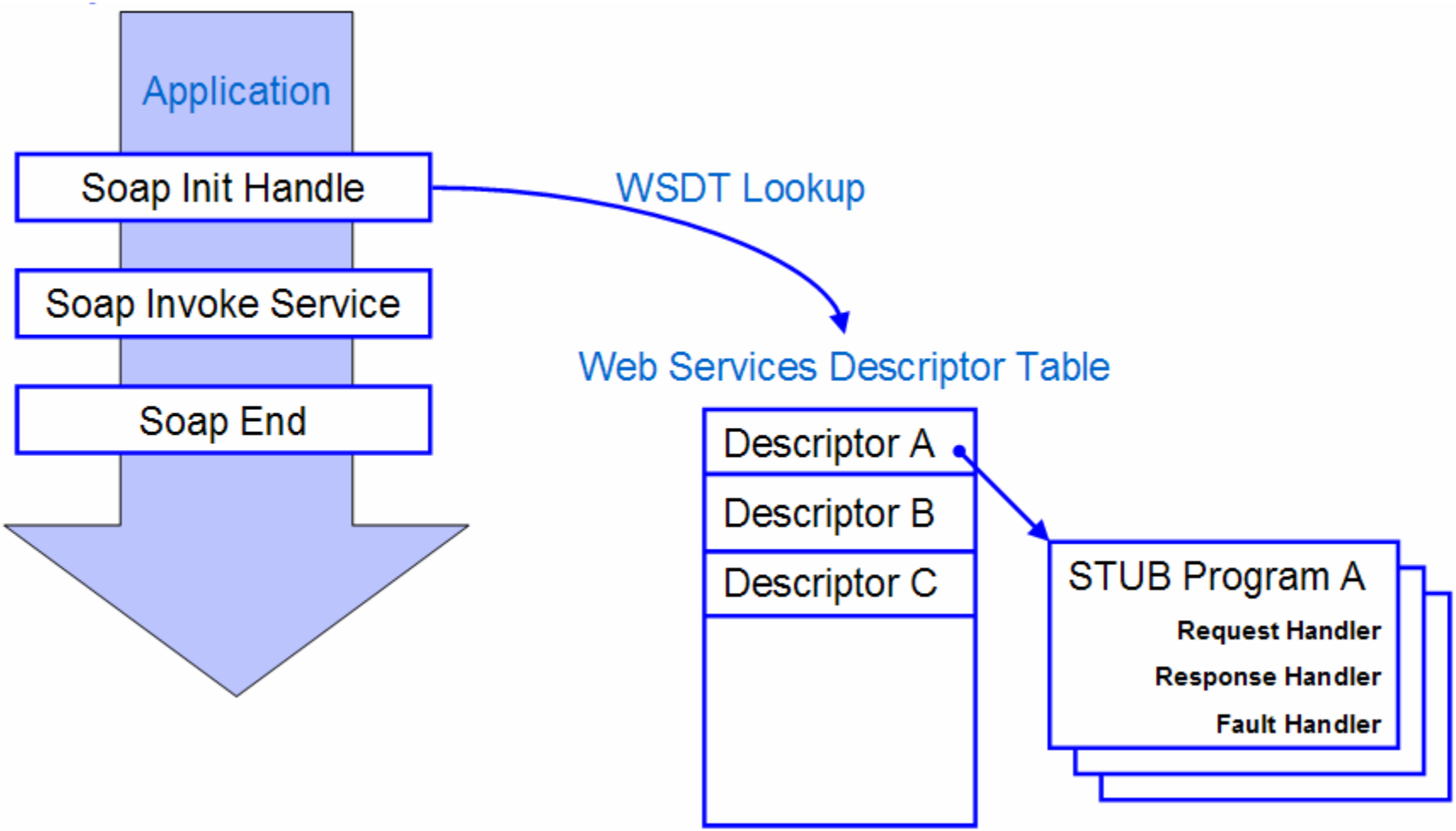
Start Using the Web Service

- **Shared object AAA1**
- **One C++ program aaa1.cpp**
- **To be invoked using ZFMSG**

aaa1.mak

```
##############################################################################
# Define shared object name
##############################################################################
APP := AAA1
APP_EXPORT := ENTRY
APP_ENTRY := AAA1

##############################################################################
# Environments needed for build
##############################################################################
maketpf_env := base_rt
maketpf_env += system

LIBS += CWSA

##############################################################################
# CPP segments
##############################################################################
CXX_SRC := aaa1.cpp

##############################################################################
# Include maketpf build rules
##############################################################################
include maketpf.rules
```

Soap Consumer Flow



Initialize a Soap Consumer Handle

```
LIBS := CWSA
#include  <tpf/c_soap.h>
t_soapHandle tpf_soapInitHandle (char *serviceName);
```

serviceName

A null-terminated string that represents the name of the Web service.

Normal return

A SOAP handle (t_soapHandle) is returned; this represents the SOAP consumer session handle. You must provide this handle for any following SOAP consumer function calls that will use the same session.

Call a Web Service

```
LIBS := CWSA
#include  <tpf/c_soap.h>
int tpf_soapInvokeService (t_soapHandle soapHandle, t_soapInv *invocationParms,
                           void **soapResponse);
```

soapHandle

The SOAP consumer session handle to perform the request for.

invocationParms

A pointer to a fully initialized t_soapInv structure.

soapResponse

A pointer to the SOAP response that will point to different values depending on the dataFormat field.

Normal return

When the SOAP_CONSUMER_SYNC MEP is used, a return of TPF_SOAP_SUCCESS indicates that the tpf_soapInvokeService call was successful.

End Web Service Session

```
LIBS := CWSA
#include  <tpf/c_soap.h>
int tpf_soapEnd (t_soapHandle soapHandle);
```

```
soapHandle

   The SOAP consumer session handle to end.

Normal return

   TPF_SOAP_SUCCESS indicates that the specified consumer Web service session ended.
```

aaa1.cpp 1/4

```
#include <tpf/c_soapc.h>
#include <stdio.h>
#include <stdlib.h>

extern "C" int AAA1() {

   char* date = "2009-03-10T12:00";

   t_soapHandle h = tpf_soapInitHandle("/forecasts/xml/SOAP_server/ndfdXMLserver.php");

   if(h == TPF_SOAP_ERROR) {

      // Error handling
      printf("Soap init handle error\n");
      exit(0);

   } else {
```

aaa1.cpp 2/4

```
t_soapInv parameters;

strcpy(parameters.operation, "NDFDgen");

parameters.timeout = 5;

parameters.messageExchangePattern = SOAP_CONSUMER_SYNC;

parameters.invocParmsVersion = SOAPC_VERSION_1;

parameters.dataFormat = SOAP_PARMS;

t_soapParms* parameterData = (t_soapParms*)calloc(1,
                          (sizeof(t_soapParms) + (sizeof(t_soapParmValues)*6)));

if(parameterData == 0) {
   // Error handling
   printf("Memory allocation failed\n");
   exit(0);
}
```

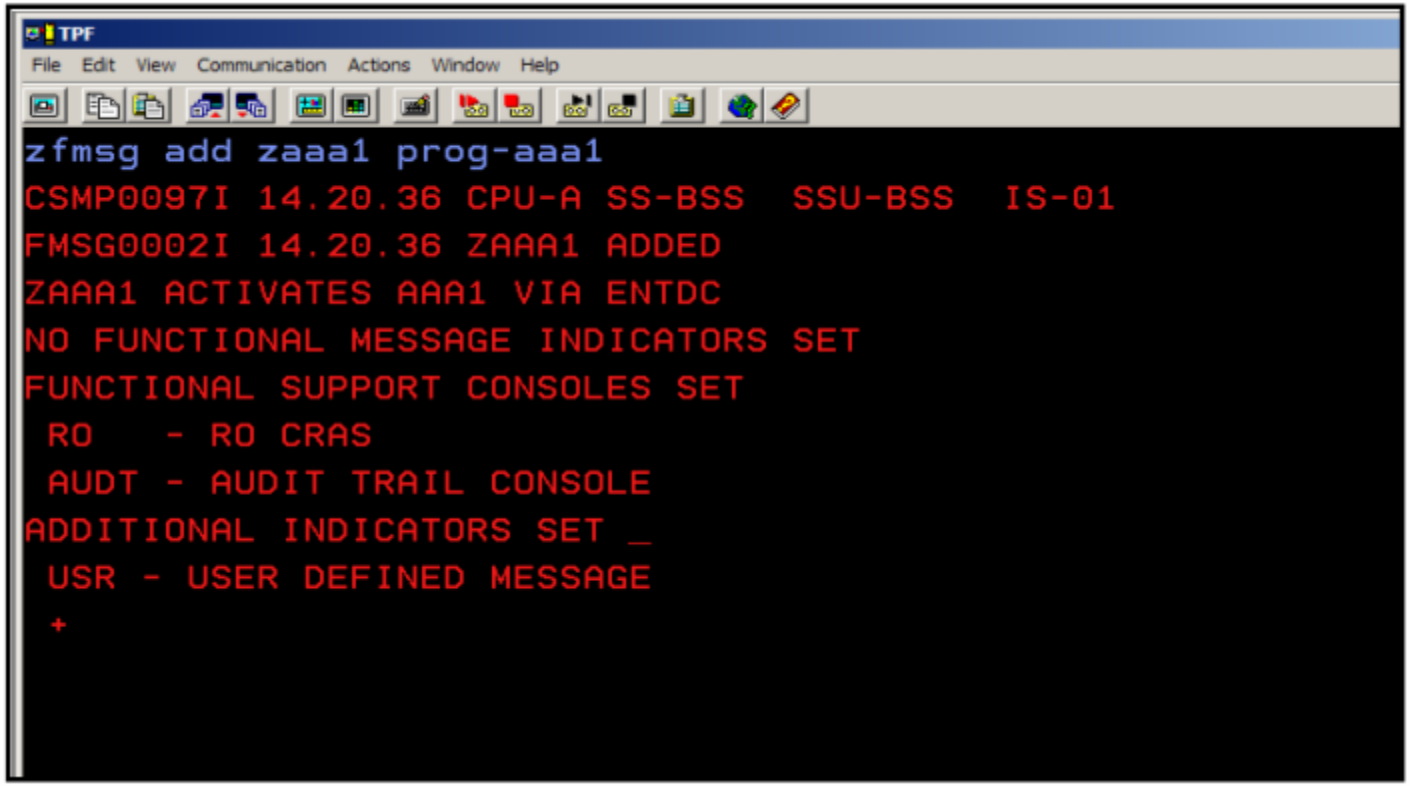
aaa1.cpp 3/4

```
parameterData->numParms = 6;

parameterData->parmArray[0].parmLength = 5;
parameterData->parmArray[0].parmPtr = (void*)"41.90";     // latitude of Chicago, IL
parameterData->parmArray[1].parmLength = 6;
parameterData->parmArray[1].parmPtr = (void*)"-87.65";    // longitude of Chicago, IL
parameterData->parmArray[2].parmLength = 6;
parameterData->parmArray[2].parmPtr = (void*)"glance";    // product
parameterData->parmArray[3].parmLength = 16;
parameterData->parmArray[3].parmPtr = (void*)date;        // startTime
parameterData->parmArray[4].parmLength = 16;
parameterData->parmArray[4].parmPtr = (void*)date;        // endTime
parameterData->parmArray[5].parmLength = 9;
parameterData->parmArray[5].parmPtr = (void*)"maxt=TRUE"; // weatherParameters

parameters.data = parameterData;
```

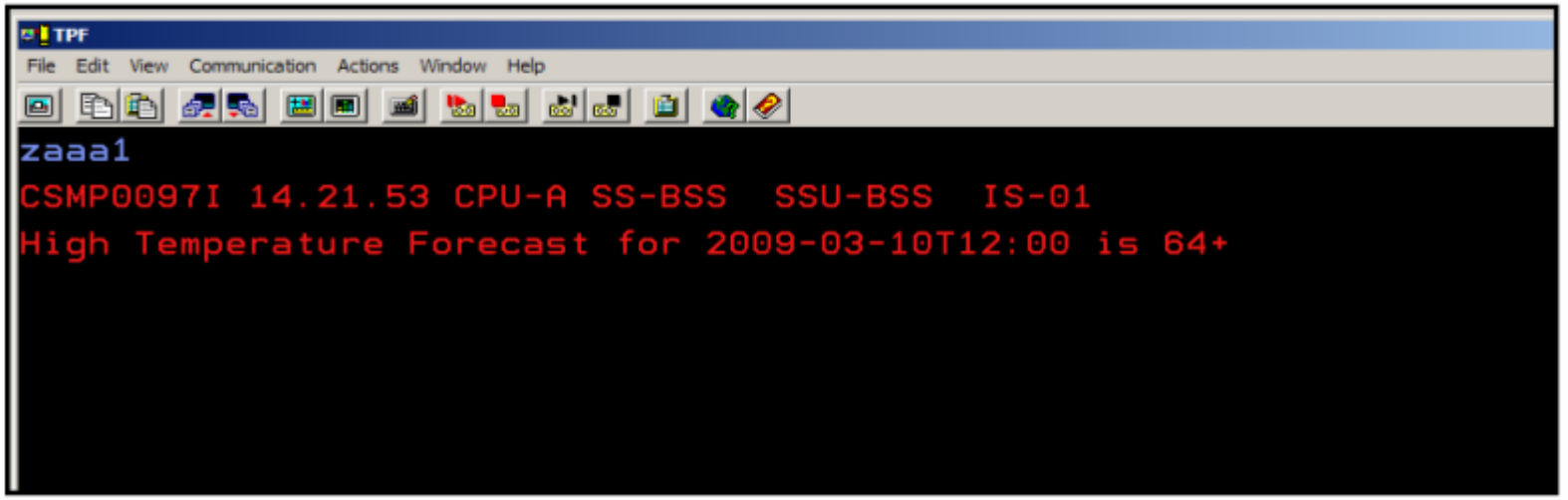
aaa1.cpp 4/4

```
      void* response;
      if(tpf_soapInvokeService(h, &parameters, &response) != TPF_SOAP_SUCCESS) {

         // Error handling
         tpf_soapEnd(h);
         printf("Soap invoke service error - %s\n", strerror(errno));
         exit(0);

      }

      free(parameterData);
      printf("High Temperature Forecast for %s is %d\n", date, *(int*)response);
      free(response);
      tpf_soapEnd(h);

   }

   return 0;
}
```

ZAAA1



ZAAA1



Summary

- **Get a Web service's WSDL**
- **Create a Deployment Descriptor**
- **Create STUB code**
- **Deploy the Descriptor**

Travel Home Safely!

Edwin W. van de Grift

edwinvandegrift@us.ibm.com
IBM Software Group
Application & Integration Middleware

Trademarks

- **IBM and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.**
- **Notes**
- **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.**
- **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.**