Programming Challenge Wrap Up Session

Edwin W. van de Grift

Legal Notices

Overview

- Programming Challenge Setup
- Discussion of a Solution
- Aside: a Z Command Server

Programmers Challenge Setup

- Client
- Server
- echoInfonodes Application
- Verify XML

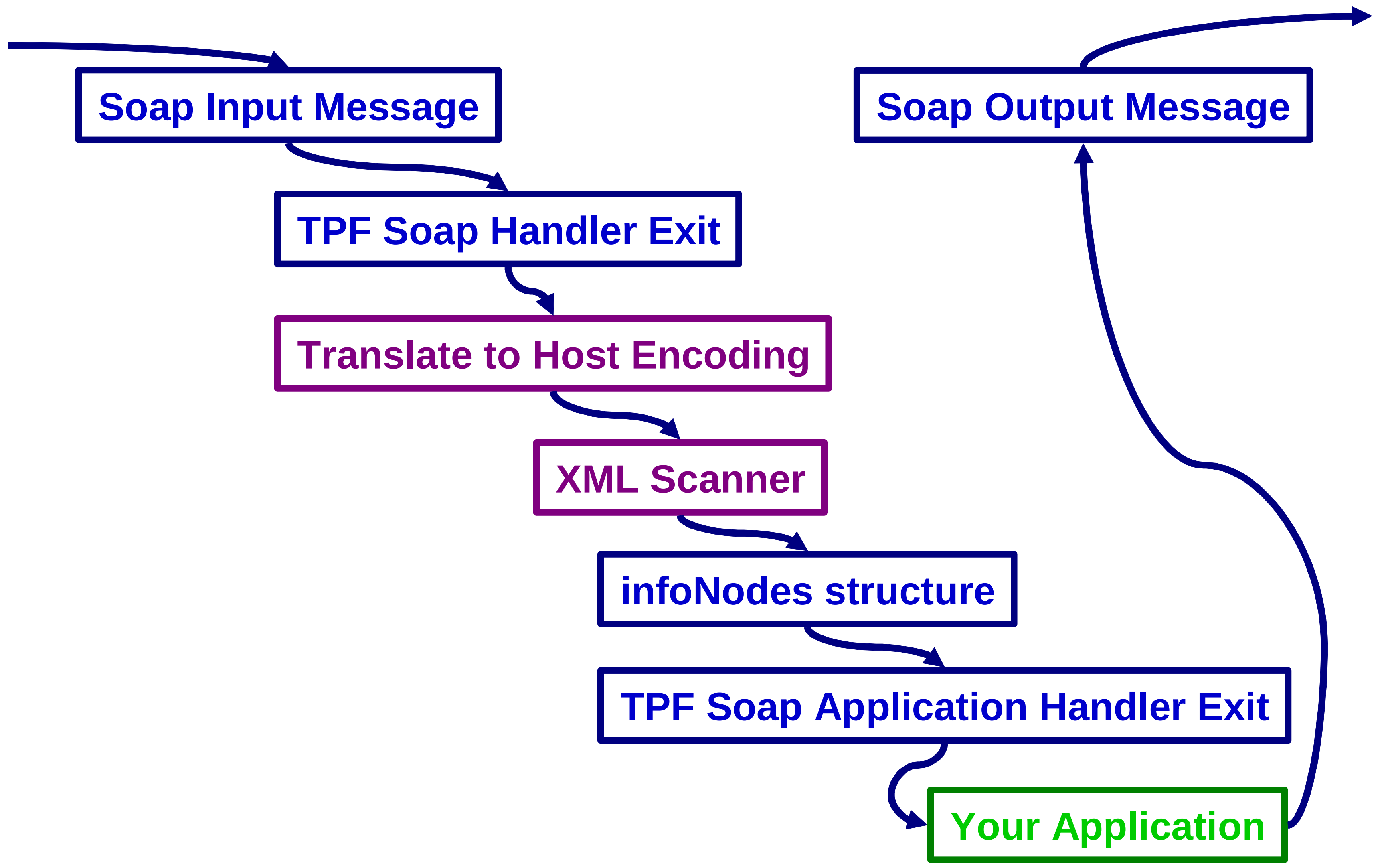
SOAP Client

- HTML page
  - Sends a SOAP message to TPF
    - Input the IP address and your TPF application name
    - Application name is used for routing the message
    - Response displayed is the output from your application
  - Verify message using XML4C parser on TPF

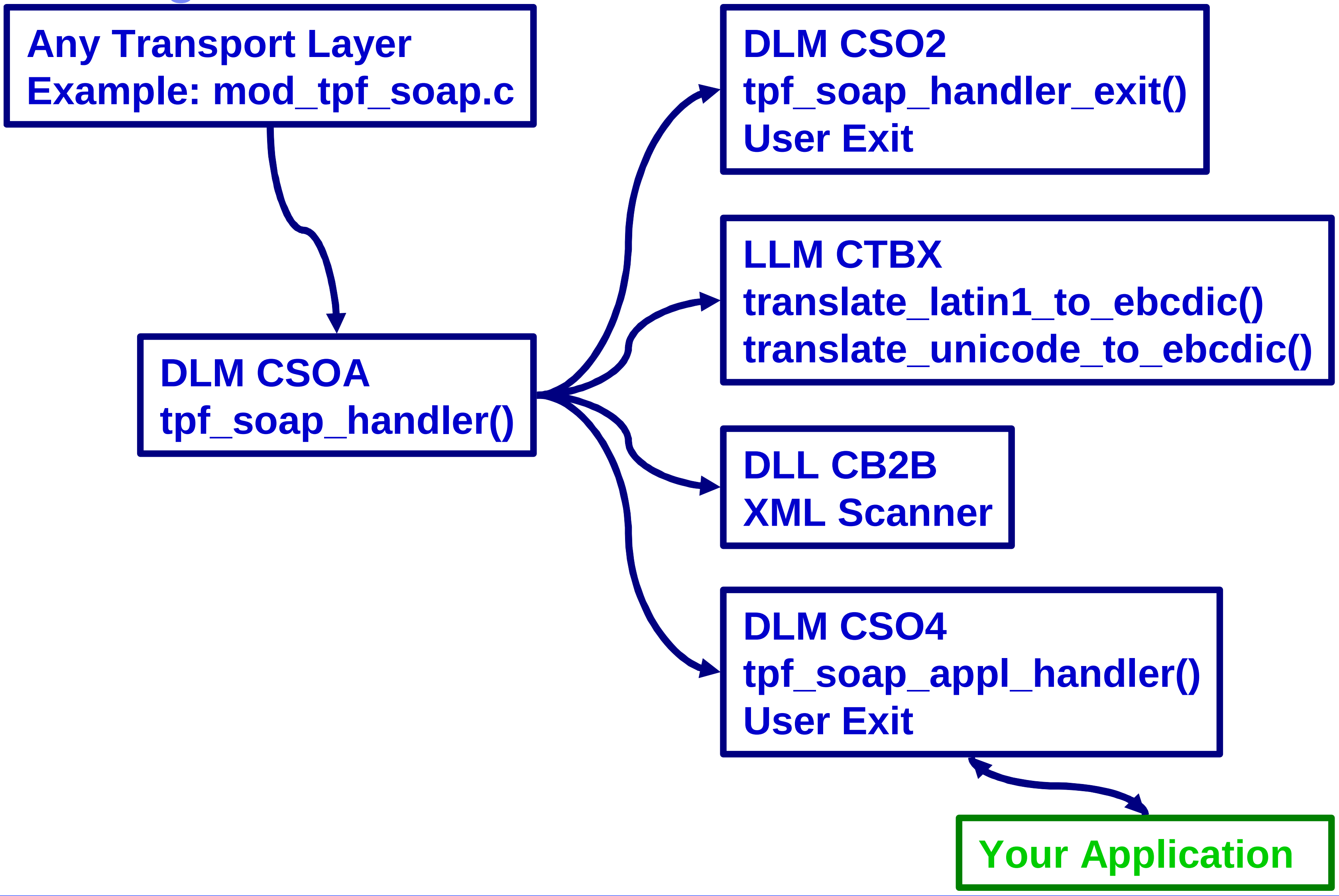
SOAP Server

- z/VM® System with following APARs applied
  - SPE PJ29396
    - PUT18
    - TPF SOAP 1.2 Server Support
  - PJ29500, PJ29681, PJ29716
    - Small fixes

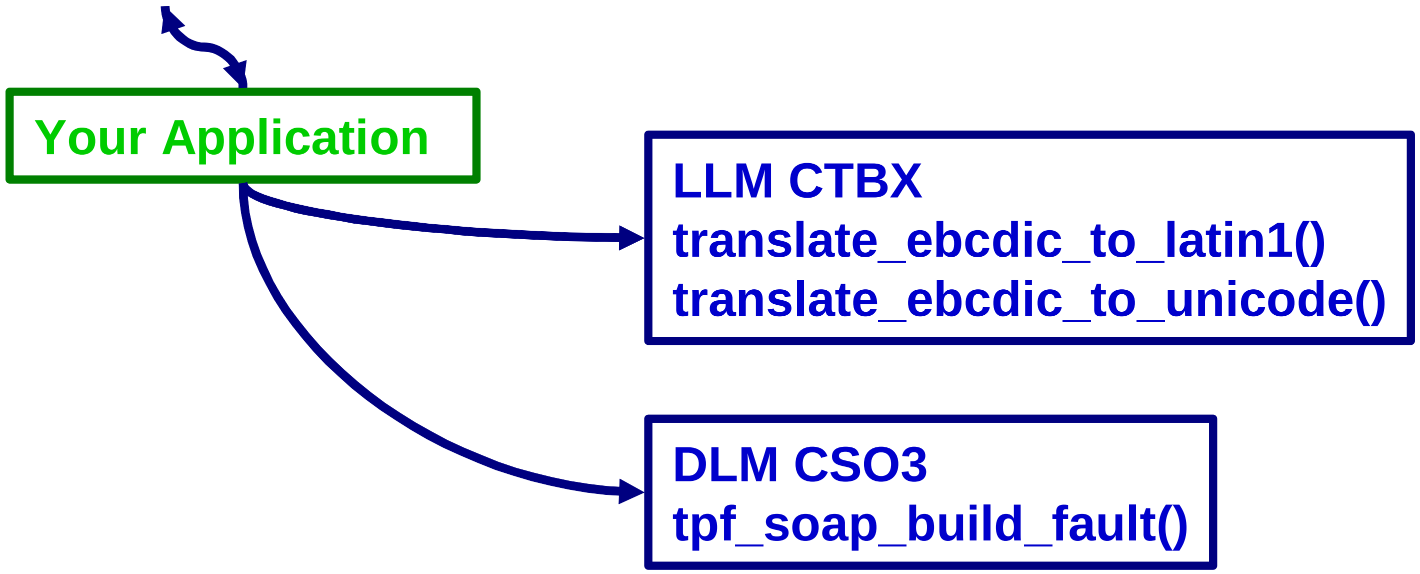
SOAP Message Flow



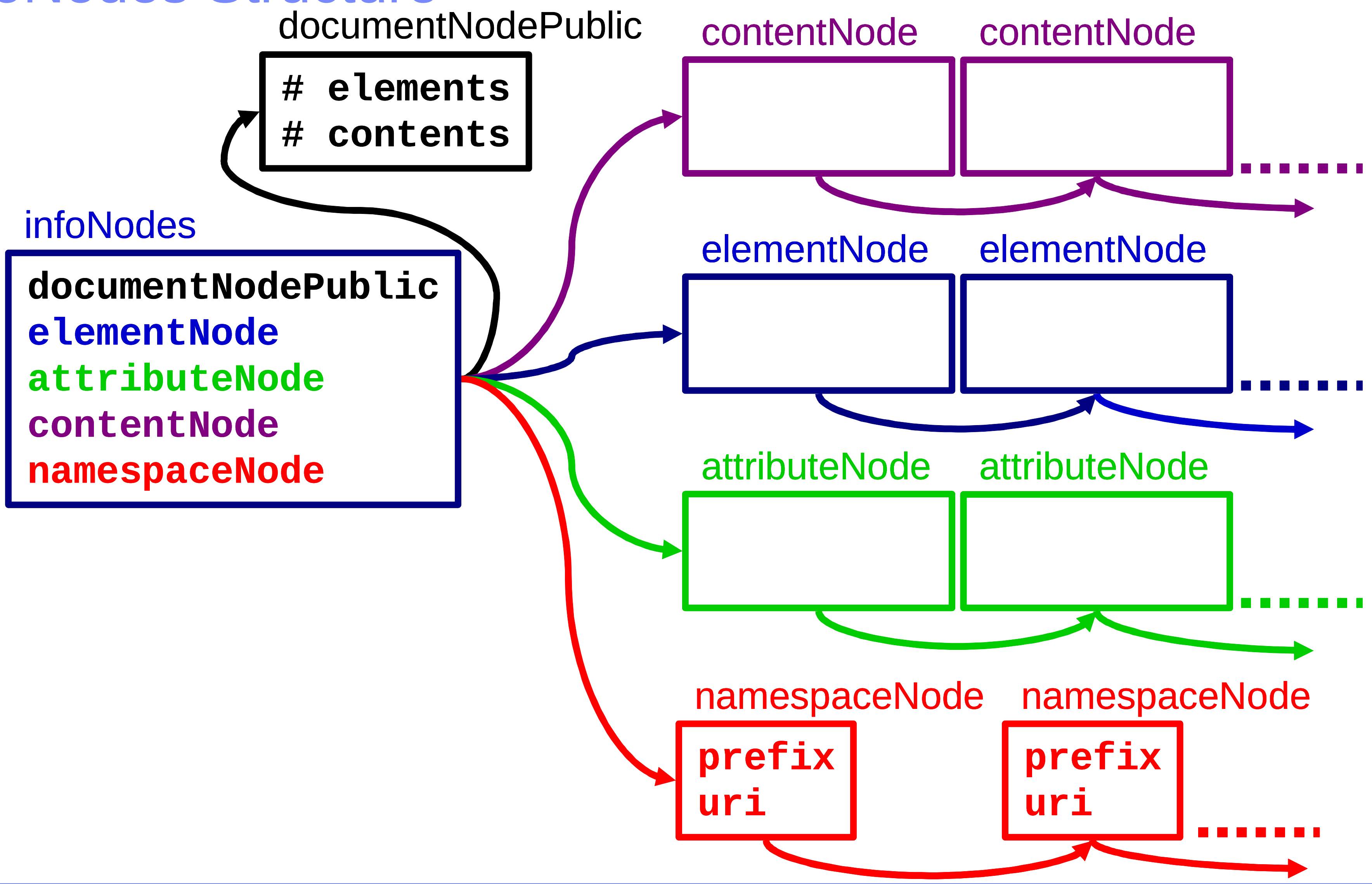
SOAP Program Flow Part 1



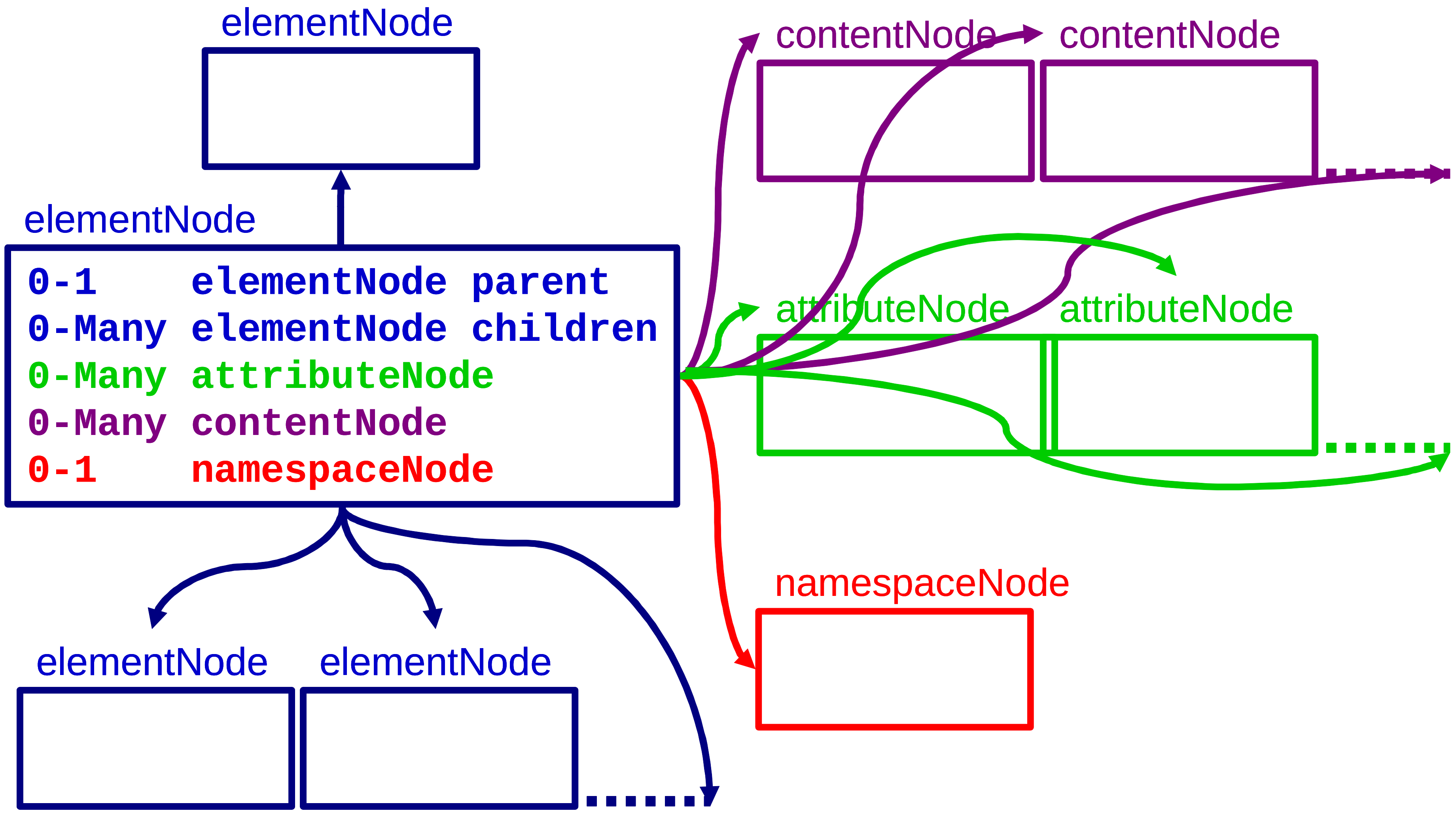
SOAP Program Flow Part 2



infoNodes Structure



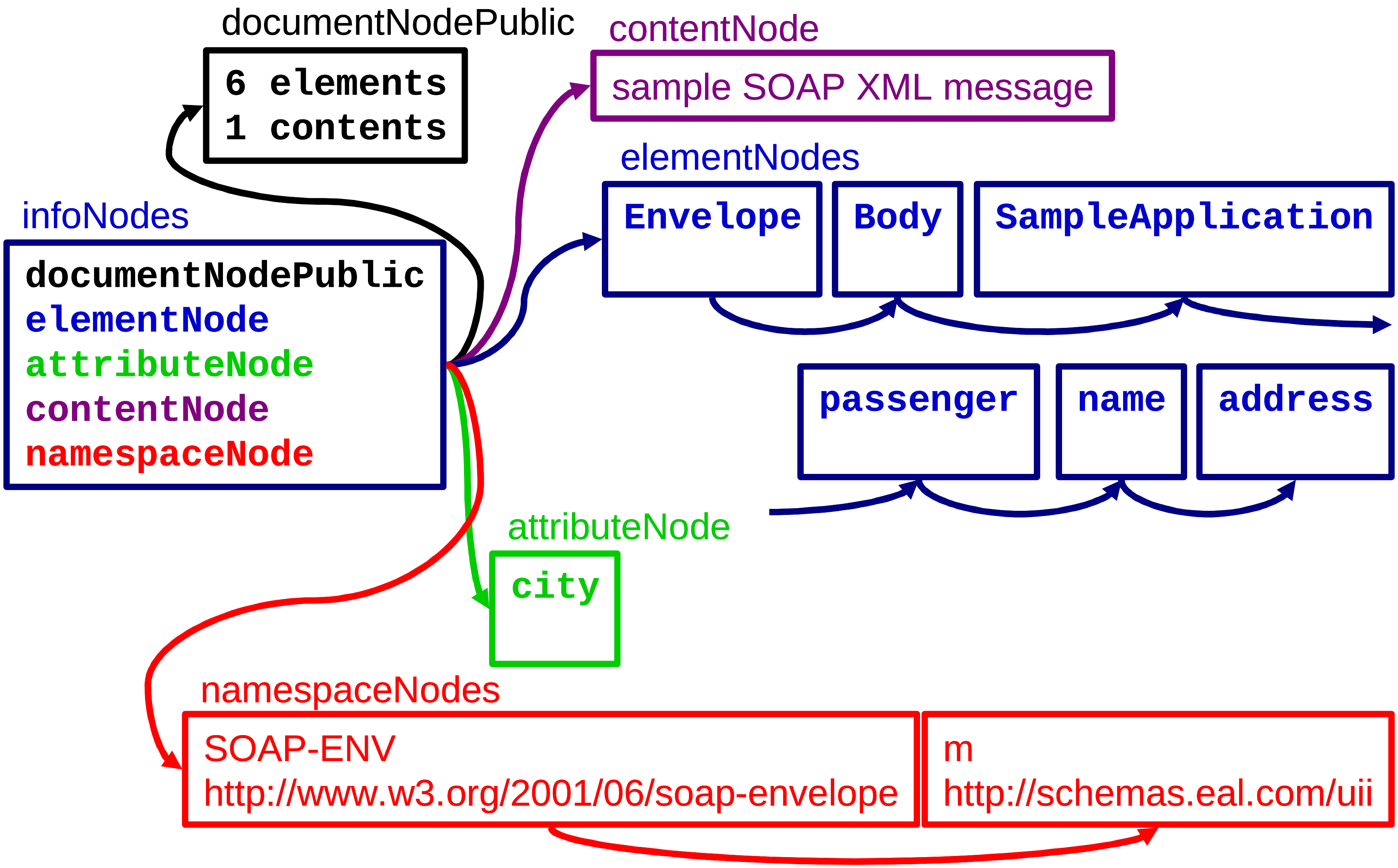
infoNodes Relations



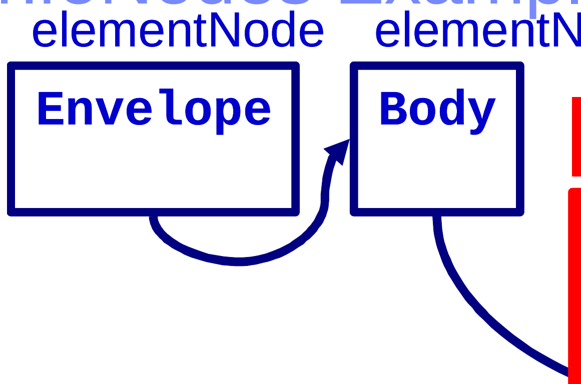
SOAP Message Example

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!--sample SOAP XML message-->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">
<SOAP-ENV:Body>
  <m:SampleApplication xmlns:m="http://schemas.eal.com/uii">
    <m:passenger>
      <m:name>John Doe</m:name>
      <m:address city="Anytown">123 Main Street</m:address>
    </m:passenger>
  </m:SampleApplication>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

infoNodes Example 1



infoNodes Example 2



[redacted]

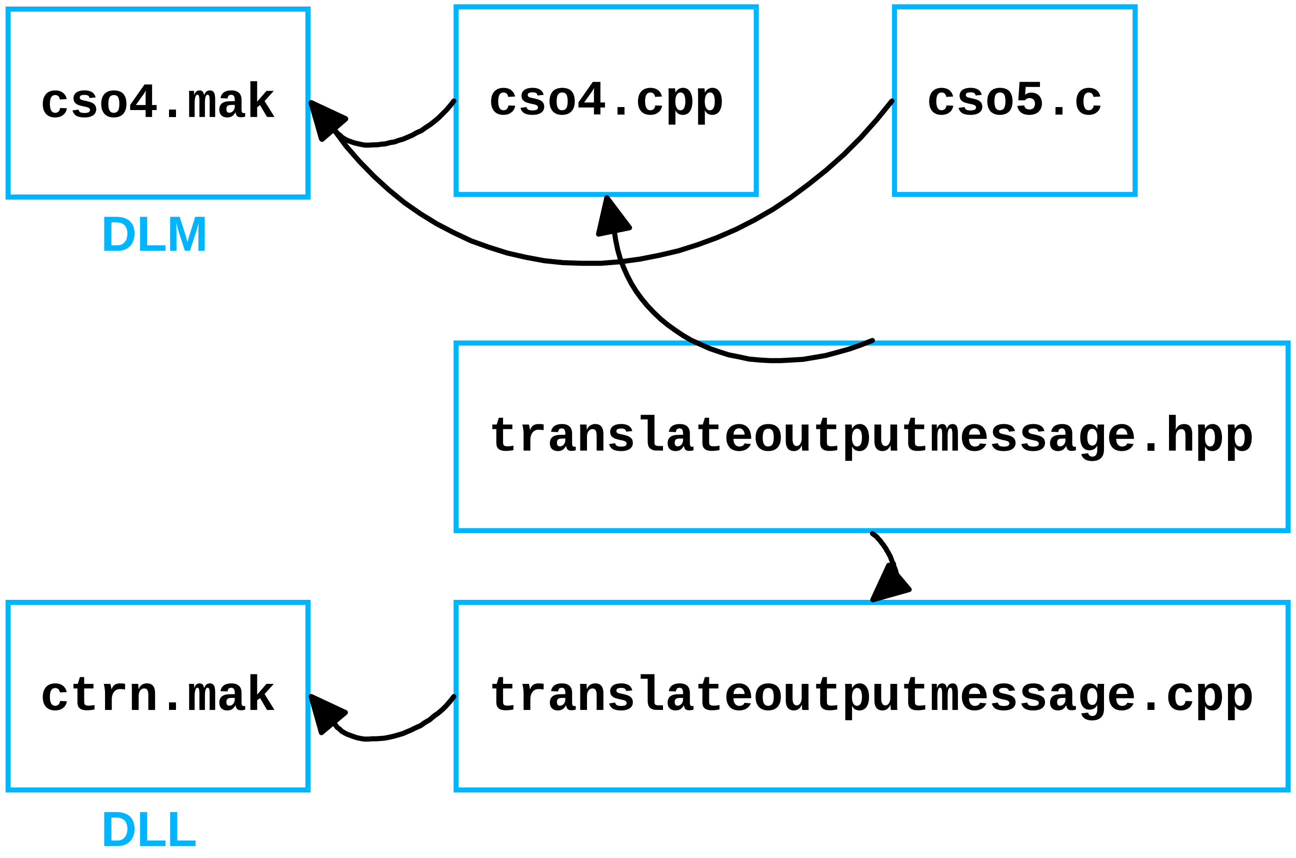
[redacted]

echoInfonodes Application

- URL
  - http://x.xxx.xxx.xx/echoInfonodes
- SOAP Message

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!--sample SOAP XML message-->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">
  <SOAP-ENV:Body>
    <m:SampleApplication xmlns:m="http://schemas.eal.com/uii">
      <m:feature m:type = "test doc">
        traverse &amp; display
      </m:feature>
    </m:SampleApplication>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

echoInfonodes Application Packaging



echoInfonodes Application CSO4

```
int CSO4(infoNodes **infoNodes,
         soapMsg *inputMsg,
         soapMsg *outputMsg,
         commsBinding *binding ) {
   if(strstr(binding->applRoutingInfo,"echoInfonodes") != 0) {
      // Call echoInfonodes()
      // If not OK generate build fault message
      // Translate output into clientEncoding
   } else {
      return SendErrorReplySender;
   }
}
```

echoInfonodes Application echoInfonodes

- Basic algorithm:
  - Display the document
    - Display the contents
  - For all elements
    - Display the element
    - Display the elements namespace
    - For all attributes
      - Display the attribute
      - Display the attributes namespace
      - For all contents
        - Display the contents
    - For all contents
      - Display the contents

echoInfonodes Application echoInfonodes 1

```
int CSO5(infoNodes** info,
         soapMsg* inputMsg,
         soapMsg* outputMsg,
         commsBinding* comms) {
   // Initialization
   documentNodePublic* doc = (documentNodePublic*)(*info)->docnode_ptr;
   elementNode* element = (*info)->element_ptr;
   attributeNode* attribute = (*info)->attribute_ptr;
   contentNode* content = (*info)->content_ptr;
   namespaceNode* namespees = (*info)->namespace_ptr;
   // Display the document
   for(int i=0; i < doc->numContents; ++i) {
      // Display the content
   }
```

... continued on the next page

echoInfonodes Application echoInfonodes 2

```
    for(int i = 0; i < doc->numElements; ++i, ++element) {
        // Display the element
        if(element->indNamespace) {
            // Display the elements namespace
        }
        for(int j = 0; j < element->numAttributes; ++j) {
            // Display the attribute
            if(attribute->indNamespace) {
                // Display the attributes namespace
            }
            for(int k = 0; k < attribute->numContents; ++k) {
                // Display the content
            }
        }
        for(int l = 0; l < element->numContents; ++l) {
            // Display the content
        }
    }
}
```

Building an Output Message in C

- Allocating buffer space
  - Calculate first
  - Bigger than ever needed, using malloc(∞)
  - On the fly, using realloc()
- Keeping track of buffer usage
- Formatting of data

```
len += sprintf(&output[len], "\n\nCONTENT NODE  :\n\n data              ");

strncat (&output[len], content->data, content->dataLen);
len = strlen(output);
len += sprintf(&output[len], "\n%s%d\n%s%c\n%s%d\n%s%d\n",
                             " datalen            ", content->dataLen,
                             " contentType        ", content->type,
                             " reference          ", content->reference,
                             " nextPtr            ", content->nextPtr);
```

Building an Output Message in C++

- Let an ostrstream take care of things

```
output << "\n\nCONTENT NODE  :\n\n data                ";
output.write(content->data, content->dataLen);
output << "\n datalen                " << content->dataLen
       << "\n contentType            " << content->type
       << "\n reference              " << content->reference
       << "\n nextPtr                " << content->nextPtr;
```

verifyXML Application

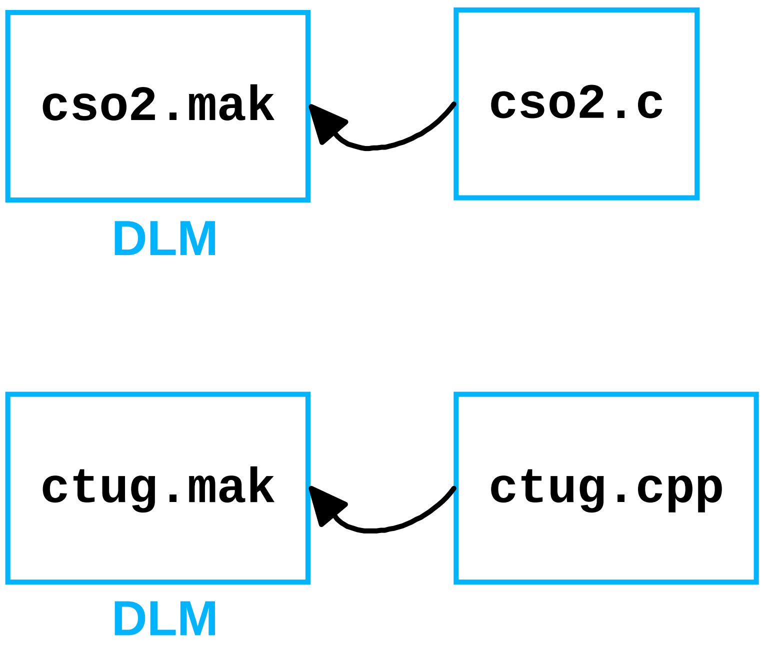
- URL
  - http://x.xxx.xxx.xx/verifyXML
- SOAP Message

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!--sample SOAP XML message-->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">
  <SOAP-ENV:Body>

      Any XML Message

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

verifyXML Application Packaging



verifyXML Application CSO2

```
#pragma map(verify_xml,"CTUG")

int CSO2(soapMsg* inputMsg,
         soapMsg* outputMsg,
         commsBinding* binding ) {
   if(strcasecmp(binding->applRoutingInfo,"/verifyXML") == 0) {
      verify_xml(inputMsg, outputMsg, binding);
      return(222);
   }
   return(TPF_CONTINUE_DO_TRANSLATE);
}
```

verifyXML Application CTUG

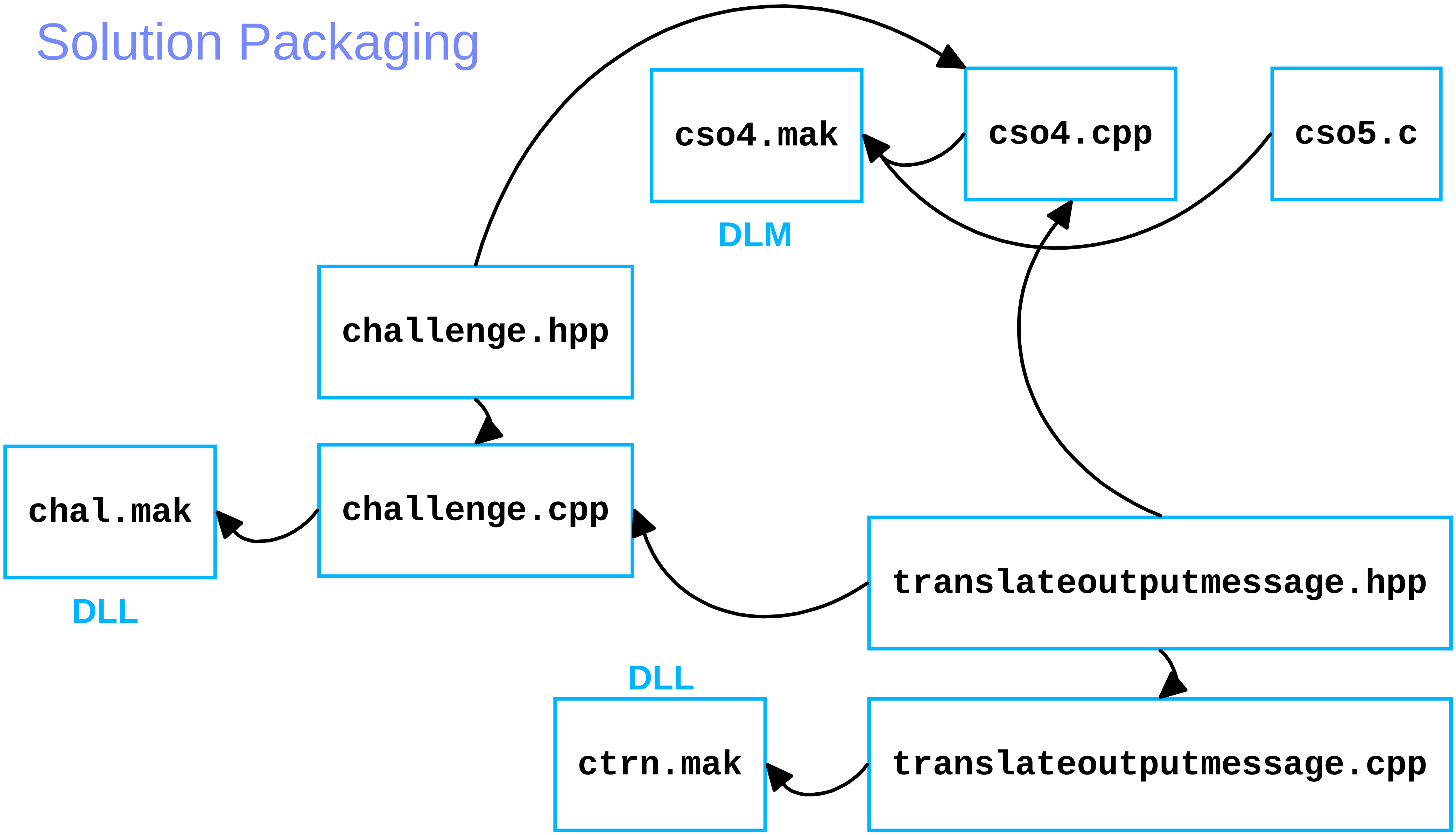
```
int CTUG(soapMsg* input,soapMsg* output,commsBinding* comms) {

   MemBufInputSource* memBuf =
      new MemBufInputSource((const XMLByte*)input->XMLptr,
                                  input->msgLength,"challenge",false);

   DOMParser* parser = new DOMParser;
   try {
      parser->parse(*memBuf);
      if(parser->getErrorCount() == 0) {
         // OK
      } else {
         // Error
      }
   } catch (...) {
      // Error
   }
}
```

Discussion of a Solution

Solution Packaging



CSO4

```
if(strstr(comms->applRoutingInfo,"echoInfonodes") != 0) {
   rc = echoInfonodes(info, input, output, comms);

} else if(strstr(comms->applRoutingInfo,"challenge") != 0) {
   rc = challenge(info, input, output, comms);

} else {
   // Error
}
```

challenge.cpp pseudocode

- Some basic error handling
  - Input parameters are pointers!
    - Ensure there is an XML input message
    - Ensure there is an infoNodes structure
- Iterate through the infoNodes structure's elementNodes
  - Find an element with localName "Iniata"
  - From its contentNode find the data (for dataLen)
- If found
  - Build SOAP output message
  - Translate output message

challenge.cpp Part 1 of 4

```
#include "challenge.hpp"
#include "translateoutputmessage.hpp"
#include <strstream.h>
#include <c_soap.h>
#include <ctype.h>
#include <stdlib.h>
#include <strings.h>

#pragma export (challenge)

int challenge(infoNodes** info, soapMsg* input, soapMsg* output,
                                                        commsBinding* comms) {
   if(input->XMLptr == 0) {
      return SendReply;
   } else if(*info == 0) {
      return ErrorReplyNeeded;
   }
```

continued on the next page...

challenge.cpp Part 2 of 4

```
documentNodePublic* document =(documentNodePublic*)(*info)->docnode_ptr;
elementNode* element =(*info)->element_ptr;
attributeNode* attribute =(*info)->attribute_ptr;
contentNode* content =(*info)->content_ptr;
namespaceNode* namespees =(*info)->namespace_ptr;
if((attribute == 0) || (element == 0) || (content == 0) ||
                                     (document == 0) || (namespees == 0)) {
   return ErrorReplyNeeded;
}
ostrstream lniata;
for(int i = 0; i < document->numElements; ++i, ++element) {
   if(strncasecmp((const char*)element->localName,"lniata",
                                             element->localNameLen) == 0) {
      lniata.write((content+element->contentPtr-1)->data,
                                 (content+element->contentPtr-1)->dataLen);
   }
}
```

continued on the next page...

challenge.cpp Part 3 of 4

```
if(lniata.pcount() == 0) {
   return SendErrorReplySender;
}

ostrstream os;
os << "<?xml version='1.0' encoding='iso-8859-1'?>\n";
if(input->version == SOAP1_1) {
   os << "<SOAP-ENV:Envelope\nxmlns:SOAP-ENV='http://schema.xmlsoap.org"
      << "/soap/envelope'>\n<SOAP-ENV:Body>\n";
} else {
   os << "<SOAP-ENV:Envelope\nxmlns:SOAP-ENV='http://www.w3.org"
      << "/2001/06/soap-envelope'>\n<SOAP-ENV:Body>\n";
}

os << "\n<foundLniata>\n";
os.write(lniata.str(), lniata.pcount());
os << "\n</foundLniata>\n";
```

continued on the next page...

challenge.cpp Part 4 of 4

```
    os << "\n</SOAP-ENV:Body>\n</SOAP-ENV:Envelope>";

    output->msgLength = os.pcount();
    output->XMLptr = os.str();
    output->clientEncoding = TPF_CCSID_LATIN1;

    translateOutputMessage(output, comms);

    if(output->XMLptr) {
       return SendReply;
    } else {
       return ErrorReplyNeeded;
    }
}
```

Aside: a Z Command Server

tpfCommand Application

- URL
  - http://x.xxx.xxx.xx/tpfCommand
- SOAP Message

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<!--sample SOAP XML message-->
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope">
  <SOAP-ENV:Body>

    <lniata>4E0000</lniata>
    <tpfCommand>zstat</tpfCommand>

  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

tpfCommand Application

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/06/soap-envelope"><SOAP-ENV:Body>
    <tpfCommand>
      CSMP0097I 15.38.00 CPU-B SS-BSS  SSU-HPN  IS-01
      STAT0012I 15.38.00 SYSTEM STATUS DISPLAY
                     IOB   FRAME  COMMON     SWB     ECB
      ALLOCATED     1360    5029     235     628     249
      AVAILABLE     1358    3335     230     490     225

      SYSTEM HEAP FRAMES        633
      THREAD FRAME PENDING        0

      ACTIVE ECBS                23
      DLY/DFR ECBS                1
      PROCESSED                3659
      LOW SPEED                   0
      ROUTED                      0
      CREATED                 68984
      SNA                         0
      TCP/IP INPUT               96
      TCP/IP OUTPUT              43
      END OF DISPLAY+
    </tpfCommand>
  </SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Aside: Program Command Routing Service

- Was introduced through APAR PJ28810 on PUT17
  - Provide application with the ability to route TPF Commands to the System Message Processor (SMP) by a given LNIATA
  - PCRS will then route the resulting output to the given LNIATA
  - If the LNIATA is registered with PCRS, PCRS will spawn the registered program with the output

PCRS API

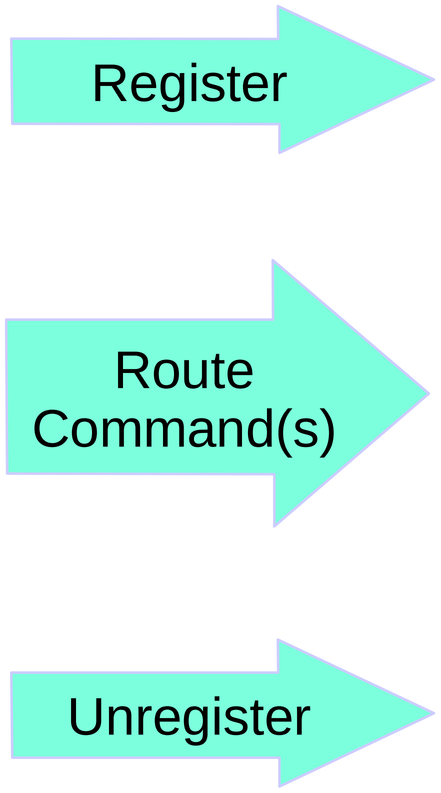
```
#include <tpfapi.h>

PCRS_RC tpf_pcrs(struct pcrs_input* pi);
```

```
typedef struct pcrs {
   int    operation;
   int    lniata;
   char   pgm_name[4];
   char*  command;
   int    cmd_len;
} pcrs_input;
```

```
PCRS_REGISTER
PCRS_UNREGISTER
PCRS_ROUTECMD
```

PCRS Typical Usage



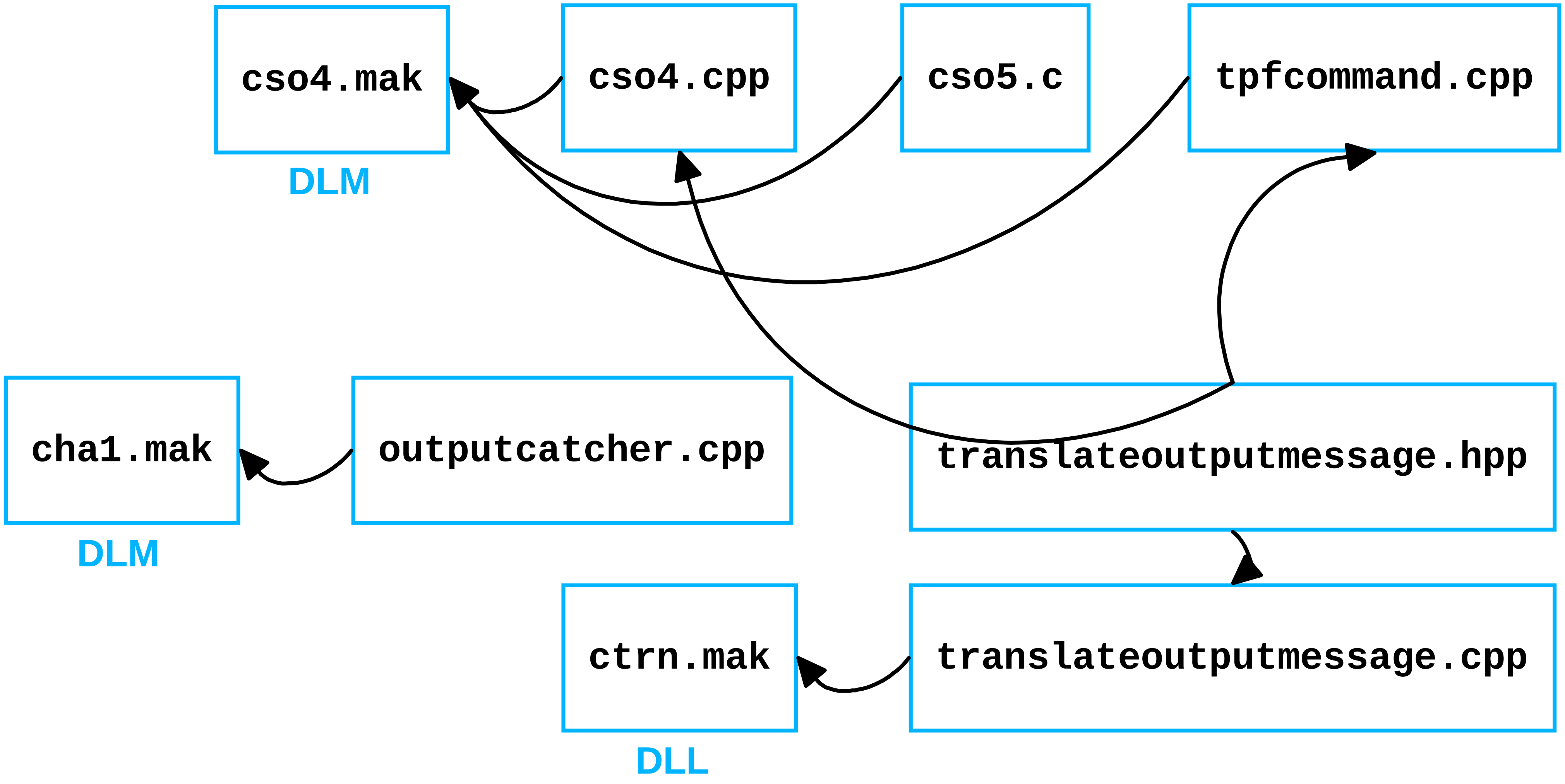
```
#include <tpfapi.h>

pcrs_input pi;
pi.operation = PCRS_REGISTER;
pi.lniata = 0x4E0000;
tpf_pcrs(&pi);

pi.command = "zdsys";
pi.cmd_len = 5;
pi.operation = PCRS_ROUTECMD;
tpf_pcrs(&pi);

pi.operation = PCRS_UNREGISTER;
tpf_pcrs(&pi);
```

tpfCommand Application Packaging



tpfCommand Application Flow

- CSO4()
  - If applicable call tpfCommand()
- tpfCommand()
  - Extract LNIATA and Z-entry from SOAP message
  - Register LNIATA with PCRS
  - Send Z-entry to PCRS
  - Create an event and wait
- outputCatcher
  - Copy the Z-entry output into a heap area
  - Post the event
- tpfCommand
  - Build a SOAP return message containing the Z-entry output

CSO4

```
if(strstr(comms->applRoutingInfo,"echoInfonodes") != 0) {
   rc = echoInfonodes(info, input, output, comms);
} else if(strstr(comms->applRoutingInfo,"challenge") != 0) {
   rc = challenge(info, input, output, comms);

} else {
   rc = tpfCommand(info, input, output, comms);

}
```

tpfcommand.cpp Part 1 of 8

```
int tpfCommand(infoNodes** info, soapMsg* input, soapMsg* output,
                                                       commsBinding* comms) {
   if(input->XMLptr == 0) {
      return SendReply;
   } else if (*info == 0) {
      return ErrorReplyNeeded;
   }
   documentNodePublic* document =(documentNodePublic*)(*info)->docnode_ptr;
   elementNode* element =(*info)->element_ptr;
   attributeNode* attribute =(*info)->attribute_ptr;
   contentNode* content =(*info)->content_ptr;
   namespaceNode* namespees =(*info)->namespace_ptr;

   if((attribute == 0) || (element == 0) || (content == 0) ||
      (document == 0)  || (namespees == 0)) {
      return ErrorReplyNeeded;
   }
```

continued on the next page...

tpfcommand.cpp Part 2 of 8

```
unsigned char* command = 0;
int commandLength;
int lniata = 0;
for(int i = 0; i < document->numElements; ++i, ++element) {
   if(strncasecmp((const char*)element->localName, "tpfcommand",
                                        element->localNameLen) == 0) {
      command = (content+element->contentPtr-1)->data;
      commandLength = (content+element->contentPtr-1)->dataLen;
   } else if(strncasecmp((const char*)element->localName, "lniata",
                                        element->localNameLen) == 0) {
      sscanf((const char*)(content+element->contentPtr-1)->data,"%8X",
                                                              &lniata);
   }
}
if((lniata == 0) || ((command == 0))) {
   return SendErrorReplySender;
}
```

continued on the next page...

tpfcommand.cpp Part 3 of 8

```
pcrs_input pi;
pi.operation = PCRS_REGISTER;
pi.lniata = lniata;
strncpy(pi.pgm_name, "CHA1", sizeof "CHA1");
PCRS_RC rc;
if((rc = tpf_pcrs(&pi)) != PCRS_OK) {
   if(rc != PCRS_LNIATA_ALREADY_REG) {
      return ErrorReplyNeeded;
   }
}
pi.operation = PCRS_ROUTECMD;
pi.command = (char*)command;
pi.cmd_len = commandLength;
if(tpf_pcrs(&pi) != PCRS_OK) {
   return ErrorReplyNeeded;
}
```

continued on the next page...

tpfcommand.cpp Part 4 of 8

```
ev0bk event;
memcpy(event.evnbkn, "CHAL", sizeof "CHAL");
memcpy(event.evnbkn+sizeof "CHAL", &lniata, sizeof lniata);
if(evntc(&event, EVENT_CB_DA, 'Y', 5, EVNTC_NORM) != 0) {
   return ErrorReplyNeeded;
}

if(evnwc(&event, EVENT_CB_DA) != 0) {
   pi.operation = PCRS_UNREGISTER;
   tpf_pcrs(&pi);
   return ErrorReplyNeeded;
}

pi.operation = PCRS_UNREGISTER;
tpf_pcrs(&pi);
```

continued on the next page...

tpfcommand.cpp Part 5 of 8

```
if(!levtest(DA)) {
   return ErrorReplyNeeded;
}
char* levelA = (char*)ecbptr()->ce1cra;
char* systemHeapArea;
memcpy(&systemHeapArea,levelA,sizeof systemHeapArea);
int numberOfPages;
memcpy(&numberOfPages,levelA + sizeof systemHeapArea,sizeof numberOfPages);
char heapToken[8];
memcpy(heapToken,
   levelA + sizeof systemHeapArea + sizeof numberOfPages,
   sizeof heapToken);
int lengthOfMessage;
memcpy(&lengthOfMessage,
   levelA + sizeof systemHeapArea + sizeof numberOfPages + sizeof heapToken,
   sizeof numberOfPages);
relcc(DA);
```

continued on the next page...

tpfcommand.cpp Part 6 of 8

```
ostrstream os;
os << "<?xml version='1.0' encoding='iso-8859-1'?>\n";
if(input->version == SOAP1_1) {
   os << "<SOAP-ENV:Envelope\nxmlns:SOAP-ENV='http://schema.xmlsoap.org"
      << "/soap/envelope'>\n<SOAP-ENV:Body>\n";
} else {
   os << "<SOAP-ENV:Envelope\nxmlns:SOAP-ENV='http://www.w3.org"
      << "/2001/06/soap-envelope'>\n<SOAP-ENV:Body>\n";
}
os << "\n<tpfCommand>\n";
```

continued on the next page...

tpfcommand.cpp Part 7 of 8

```
   char* p = systemHeapArea;
   cinfc(CINFC_WRITE,CINFC_CMMHEAP);
   for(i = 0; i < lengthOfMessage; ++i, ++p) {
      if(!isprint(*p)) {
         if (*p = 0x15) *p = '\n';
         else           *p = 'X';
      }
   }
   p = strstr(systemHeapArea,"XXX");
   while(p) {
      *p = ' ';
      ++p;
      *p = ' ';
      ++p;
      *p = '\n';
      p = strstr(systemHeapArea,"XXX");
   }
   keyrc();
```

continued on the next page...

tpfcommand.cpp Part 8 of 8

```
   os.write(systemHeapArea,lengthOfMessage);
   rsysc(systemHeapArea,numberOfPages,heapToken);
   os << "\n</tpfCommand>\n";

   os << "\n</SOAP-ENV:Body>\n</SOAP-ENV:Envelope>" << '\0';

   output->msgLength = os.pcount();
   output->XMLptr = os.str();
   output->clientEncoding = TPF_CCSID_LATIN1;

   translateOutputMessage(output, comms);

   if(output->XMLptr) {
      return SendReply;
   } else {
      return ErrorReplyNeeded;
   }
}
```

outputcatcher.cpp Part 1 of 4

```
extern "C" void CHA1() {

   if(!levtest(D0)) {
      exit(0);
   }

   am0sg* omsg = (am0sg*)ecbptr()->ce1cr0;
   int lniata = omsg->am0lit;

   // sizeof am0sg::am0lit + sizeof am0sg::am0np1 + sizeof am0sg::am0np2
   int magic = 5;

   ostrstream os;
   os.write(&omsg->am0txt , omsg->am0cct-magic);
```

continued on the next page...

outputcatcher.cpp Part 2 of 4

```
   if((unsigned int*)omsg->am0fch != 0) {
      unsigned int* fileAddress = (unsigned int*)omsg->am0fch;
      char fileId[2];
      memcpy(fileId,omsg->am0rid, sizeof fileId);
      relcc(D0);
      while(fileAddress != 0) {
         omsg = (am0sg*)find_record(D0, (const unsigned int*)&fileAddress,
                                    fileId, '\0', NOHOLD);
         if(omsg == 0) fileAddress = 0;
         else          os.write(&omsg->am0txt , omsg->am0cct-magic);
         if(fileAddress != 0) {
            fileAddress = (unsigned int*)omsg->am0fch;
            memcpy(fileId, omsg->am0rid, sizeof fileId);
         }
         crusa(1,D0);
      }
   }
```

continued on the next page...

outputcatcher.cpp Part 3 of 4

```
int numberOfPages = (os.pcount() + _4K_SIZE) / _4K_SIZE;
char heapToken[8] = { 'E', 'D', 'W', 'I', 'N', '0', '0', '0'};
char* systemHeapArea = (char*)gsysc(numberOfPages, heapToken);
if(systemHeapArea != 0) {
   cinfc(CINFC_WRITE,CINFC_CMMHEAP);
   memcpy(systemHeapArea, os.str(), os.pcount());
   keyrc();
   char* levelA = (char*)getcc(DA,GETCC_TYPE,L4);
   memcpy(levelA, &systemHeapArea, sizeof systemHeapArea);
   memcpy(levelA+sizeof systemHeapArea,&numberOfPages,sizeof numberOfPages);
   memcpy(levelA+sizeof systemHeapArea+sizeof numberOfPages,
          heapToken, sizeof heapToken);
   int lengthOfMessage = os.pcount();
   memcpy(levelA+sizeof systemHeapArea+sizeof numberOfPages+sizeof heapToken,
          &lengthOfMessage,sizeof lengthOfMessage);
```

continued on the next page...

outputcatcher.cpp Part 4 of 4

```
      ev0bk event;
      memcpy(event.evnbkn, "CHAL", sizeof "CHAL");
      memcpy(event.evnbkn+sizeof "CHAL", &lniata, sizeof lniata);
      postc(&event, EVENT_CB_DA, 0);
   }
   exit(0);
}
```

End