APPENDIX B

XML Test Drivers

Revised: July 2010, OL-23045-01

This appendix details the XML test drivers.

CLI to SOAP/CORBA XML Transaction

The following sample test driver executes a normal CLI command, but processes it as a SOAP or CORBA XML transaction.

```
package com.sswitch.oam.drv;

import java.lang.*;
import java.io.*;
import java.util.*;
import java.text.*;
// XML Stuff
import org.apache.ecs.xml.*;
import org.apache.ecs.*;
import org.w3c.dom.*;
import org.xml.sax.*;
import org.apache.xml.serialize.*;
// BTS Utility Code objects...
import com.sswitch.oam.cad.*;
import com.sswitch.oam.xml.*;
import com.sswitch.oam.util.*;
import com.sswitch.oam.ccc.*;

/**
 * XmlCli.java
 * Copyright (c) 2002-2003, 2006 by Cisco Systems, Inc.
 * --Test driver for the XML/CORBA interface...
 *    This test driver executes a normal CLI command and processes the request
 *    as a CORBA XML transaction. The reply is then displayed. I am no as
 *    concerned with complex data show(s) as with the ability to issue
 *    provisioning commands. Note that this example can be built with the
 *    provided tool "oo-cc" this simple script creates the correct CLASSPATH
 *    and invokes the compiler with the correct options. Also, the "oo-idl"
 *    tool can be used to generate the correct IDL output.
 *
 *   @author   A. J. Blanchard
 *   @version  3.1
 *   @since    900-04.01.00
 *   @since    900-03.05.02
 *
```

```
 */

public class XmlCli {

    /*
     * Class private data
     */
    private String []                                   objArgs;
    private XMLAdapter                                  objBts;
    private String                                      objLoginName;
    protected String                                    objPassword;
    protected String                                    objSwitch;
    protected String                                    adapterType;

    /**
     * Generic Constructor for the test driver.
     */
    protected        XmlCli(String[] args)
    {
      // Initialize the BTS ORB interface object.
      try {Log log = Log.getInstance("XmlCli");} catch(Exception e){}
      objArgs = args;
      objLoginName="optiuser";        // Name to login the BTS
      objPassword="optiuser";         // Password to login the BTS
      objSwitch="BTS";         // Default switch name
      parseArgs(objArgs);
      try{
        objBts = XMLAdapterFactory.getInstance().createAdapter(adapterType,args)
;
      }catch(XMLAdapterException e){
        System.out.println("Create Adapter Failure = " + e.err_code +"\n"
                           + e.err_desc);
        System.exit(1);
      }
      return;
    }


    /**
     * This is the main method for the application.
     */
    public static void main(String[] args)
    {
      XmlCli me = new XmlCli(args);
      System.out.println("start me.go()...");
      me.go();
      System.out.println("me.go() done.");
      return;
    }



    /**
     * This is the primary execution method for the object. It performs the
     * actual request and calls for the print of the reply.
     */
    protected void       go()
    {
      //
      // Log into the target machine with generic optiuser
      //
      try {
        System.out.println("Attempt to connect with:"+objLoginName+"/"+objPasswo
rd);
```

```
    objBts.connect(objLoginName, objPassword);
    System.out.println("BTS10200 Login successful...\n");
    System.out.println("Type 'bye' or 'exit' to terminate the program.\n");
    }
  catch (XMLAdapterException e) {
    System.out.println("XMLAdapterException in login = " + e.err_code +"\n"
                       + e.err_desc);
    System.exit(1);
    }
  catch (Exception e) {
    System.out.println("Exception at login = " + e.toString());
    System.exit(2);
    }
  //
  // Read in the file and send request...
  //
  while(true)
      {
      try {
        openCLI();                      // Put out the prompt...
        CommandParser parser = new CommandParser();
        String cmd = readCLI().trim(); // Fetch the request file...
        String reply = "";
        if((cmd.equals("")))
           continue;
        if((cmd.equals("exit"))||(cmd.equals("bye"))||(cmd.equals("quit")))
            break;
        else
          {
          // Issue request to BTS 10200
          reply = objBts.request(parser.toXML(cmd));
          System.out.println("RETURN VALUE: ");
          //parser.prettyPrint(reply);
          System.out.println(printResult(reply));
          }
        }
    catch (XMLAdapterException ce) {
        System.out.println("CIS Command Exception: CODE="+ce.err_code +
                           "\n"+ce.err_desc);
        }
    catch (Exception e) {
        Log.fatal("General Command Exception:"+
                  Util.stackTraceToString(e));
        System.out.println("General Command Exception:");
        e.printStackTrace();
        break;
        }
    } // end while(1)
  closeCLI();
  // Clean up and logout
  try {objBts.disconnect();} catch (XMLAdapterException cad) {}

  return;
} // end go()



/**
 * Open the input file for reading. This allows the read method to suck
 * a line at a time of the CLI style input.
 */
protected void        openCLI()
{
  System.out.print(objLoginName+"@"+objSwitch+"> ");
```

```
  return;
}

/**
 * This is the method that closes and clean up after a file has been
 * processed.
 */
protected void        closeCLI()
{
  System.out.println("\n Bye...");
  return;
}

/**
 * Read in the file provided as the request. Just exit on errors. Don't
 * worry about throwing an error exception.
 */
protected String      readCLI()  throws java.io.IOException
{
  int          temp=0;
  int          idx=0;
  byte []      buf= new byte[256];

  while(true)
      {
      temp = System.in.read();
      if(temp==10)           // <ENTER Key>
        break;
      buf[idx++]=(byte) temp;
      }
  return new String(buf);
}

//=====================================================================
// Tools and utilities...
//=====================================================================

/**
 * This method reads in the startup arguments for the HUB process
 * The only presently supported argument is the HUB slave or master
 * mode assignment. Master is the default. It does not reach out to find
 * the slave HUBs.
 *
 * @param  args  The list of startup arguments
 */
protected void   parseArgs(String[]  args)
{
int  lp=0;

  if(args==null)
     return;

  try {
    for(lp=0;lp < args.length;lp++)
      {
      if(args[lp].startsWith("-n"))
         objLoginName=args[(lp+1)];

      if(args[lp].startsWith("-p"))
         objPassword=args[(lp+1)];

      if(args[lp].startsWith("-b"))
         objSwitch=args[(lp+1)];
```

```
      if(args[lp].startsWith("-t"))
         adapterType = args[++lp].toUpperCase();
      }
    }
  catch (java.lang.NumberFormatException nfe) {
    Log.fatal("Error in args: "+args [lp]+args [lp+1]+"\n"+
              Util.stackTraceToString(nfe));
    System.exit(1);
    }
  catch (java.lang.ArrayIndexOutOfBoundsException e) {
    return;
    }
  return;
}

/**
 * This is the reply data table parser
 */
public String   printResult(String reply)
{

String  output="";
int     size = 0;
HashMap table = null;

try {
  XMLReply parseResult = new XMLReply(reply);
  if(parseResult.getStatus().equals("true"))
     {
     output+="SUCCESS:";
     size = parseResult.getSize();
     }
  else
     output+="FAIL:";
  String temp = parseResult.getReason();
  temp+="\n";
  if((table=parseResult.getReplyData())!=null)
     temp += buildText(parseResult.getReplyData(),0);
  temp += "================================================\n";
  output+=temp;
  for(int lp=1; lp < size; lp++)
    {
    output += buildText(parseResult.getReplyData(),lp);
    output +="================================================\n";
    }

  }
catch (Exception e) {
  System.out.println("Reply Exception = " +e.toString());
  Log.error("Error parsing reply \n"+reply+"\n"+
            Util.stackTraceToString(e));
  }
return output;
} // end printResult

/**
 * This is the text builder for row level data in a reply table.
 */
public String   buildText(HashMap table, int rowNum)
{
  String  data="";
  HashMap rowSet=(HashMap) table.get(Integer.toString(rowNum));

  if(rowSet==null)
```

```
            return "No Data available at row "+rowNum;

          Iterator list = rowSet.keySet().iterator();
          for(;list.hasNext();)
            {
            String key = (String) list.next();
            String value = (String) rowSet.get( key);
            data+=key+"="+value+"\n";
            }
          return data;
        }

} // end XmlCli
```