

XML Test Drivers

Revised: June 24, 2009, OL-15335-04

This appendix describes the XML test drivers.

XML Request Batch File

The following example of an XML/CORBA interface test driver executes an XML request batch file built as a CLI script. This allows compatibility with older CLI scripts used to provision the system.

```
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 Interface Code objects...
import com.sswitch.oam.cad.*;
import com.sswitch.oam.xml.*;
import com.sswitch.oam.util.*;
import com.sswitch.oam.ccc.*;

/**
 * XmlBatch.java
 * Copyright (c) 2002, 2003 by Cisco Systems, Inc.
 * --Test driver for the XML/CORBA interface...
 * This test driver executes a batch file built as a CLI script as XML
 * requests. This allows compatibility with the older CLI scripts used to
 * provision the system. 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 3.0
 */

public class XmlBatch {
```

```

/*
 * Class private data
 */
private String []                objArgs;
private CorbaXmlIntf            objBts;

private File                    objFile;
private RandomAccessFile        objFileHandle;

/**
 * Generic Constructor for the test driver.
 */
protected XmlBatch(String[] args)
{
    // Initialize the ORB.
    objArgs = args;
    objBts = new CorbaXmlIntf(args);
    return;
}

/**
 * This is the main method for the application.
 */
public static void main(String[] args)
{
    //
    // Verify that the argument match what is expected...
    // oo-run XmlRequest <CLI Request File> \
    //     -ORBopenorb.config=../OpenORB.xml"
    //
    if(args.length < 2)
    {
        System.out.println("\nStart program as: oo-run XmlRequest <XML Request File>
-ORBopenorb.config=../OpenORB.xml\n");
        System.exit(0);
    }
    XmlBatch me = new XmlBatch(args);
    me.go();
    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 {
        objBts.connect();
        System.out.println("BTS10200 Login successful...");
    }
    catch (Exception e) {
        System.out.println("Exception in login = " + e);
        System.exit(1);
    }
    //
    // Read in the file and send request...

```

```

//
try {
    String reply = "";

    openCLI();

    while(true)
    {
        CommandParser parser = new CommandParser();
        String cmd = readCLI(); // Fetch the request file...
        if(cmd==null)
            break;

        if(cmd.startsWith("#") || cmd.length()==0)
            continue;

        else
        {
            // Issue request to BTS 10200
            reply = objBts.request(parser.toXML(cmd));
            System.out.println("RETURN VALUE: ");
            parser.prettyPrint(reply);
        }
    } // end while(1)

    closeCLI();
    // Clean up and logout
    objBts.disconnect();
}
catch (CmdExceptions ce) {
    System.out.println("CIS Command Exception: CODE="+ce.error_code +
        "\n"+ce.error_string);
    ce.printStackTrace();
}
catch (Exception e) {
    System.out.println("Batch Command Exception = " + e);
    e.printStackTrace();
}
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()
{
    try {
        objFile=new File(objArgs[1]);
        objFileHandle=new RandomAccessFile(objFile,"r");
    }
    catch(Exception e) {
        // In the event of an error. just bail out of the program
        System.out.println("Error in processing file:\n"+e.toString());
        System.exit(1);
    }
}

/**
 * This is the method that closes and clean up after a file has been
 * processed.
 */
protected void      closeCLI()  throws java.io.IOException

```

```

    {
        objFileHandle.close();
    }

    /**
     * Read in the file provided as the request. Just exit on errors. Don't
     * worry about throwing an error exception.
     */
    protected String    readCLI()
    {
        String data=null;

        try {
            if((data = objFileHandle.readLine())!=null)
                data.trim();
        }
        catch (Exception e) {
            System.out.println("Unable to read "+objFile.toString()+
                " with error:\n"+e.toString());

            System.exit(1);
        }
        // May return a valid string or a NULL object...
        return data;
    }

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

} // end XmlRequest

```

CLI to CORBA XML Transaction

The following example of a test driver executes a normal CLI command but processes it as a 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 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 not 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  3.0
*
*/

public class XmlCli {

    /*
     * Class private data
     */
    private String []                objArgs;
    private CorbaXmlIntf            objBts;

    /**
     * Generic Constructor for the test driver.
     */
    protected XmlCli(String[] args)
    {
        // Initialize the BTS ORB interface object.
        objArgs = args;
        objBts = new CorbaXmlIntf(args);
        return;
    }

    /**
     * This is the main method for the application.
     */
    public static void main(String[] args)
    {
        //
        // Verify that the argument match what is expected...
        // oo-run XmlCli -ORBopenorb.config="./OpenORB.xml"
        //
        if(args.length < 1)
        {
            System.out.println("\nStart program as: oo-run XmlCli
-ORBopenorb.config=\"./OpenORB.xml\" \n");
            System.exit(0);
        }
        XmlCli me = new XmlCli(args);
        me.go();
        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 {
    objBts.connect();
    System.out.println("BTS10200 Login successful...");
}
catch (Exception e) {
    System.out.println("Exception in login = " + e);
    System.exit(1);
}
//
// Read in the file and send request...
//
try {

    while(true)
    {
        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"))
            break;
        else
        {
            // Issue request to BTS 10200
            reply = objBts.request(parser.toXML(cmd));
            System.out.println("RETURN VALUE: ");
            parser.prettyPrint(reply);
        }

    } // end while(1)

    closeCLI();
    // Clean up and logout
    objBts.disconnect();
}
catch (CadExceptions ce) {
    System.out.println("CIS Command Exception: CODE="+ce.error_code +
        "\n"+ce.error_string);
    ce.printStackTrace();
}
catch (Exception e) {
    System.out.println("CIS Command Exception: \n"+e.toString());
    //e.printStackTrace();
}
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("CORBA-CLI> ");
    return;
}

/**

```

```

    * This is the method that closes and clean up after a file has been
    * processed.
    */
protected void      closeCLI()  throws java.io.IOException
{
    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...
//=====

} // end XmlCli
```

