

## Applications Note: Connecting to Device Servers using Java

The following is a Java example which will help you understand what is involved in using Java to extract data dynamically from the NetPort/Cobox/Xport products. Some links to sources of information and tools are included to assist you but it is beyond the scope of this document to provide a crash-course in Java, Borland or Sun are far better at this!

Writing a Java applet to communicate with a serial device attached to the Device Server is straightforward, however, you will need to be familiar with Java programming, and need a Java compiler.

Try the following for typical tools:

Java DDK 1.4.1: <http://java.sun.com/j2se/downloads.html>

Code Editor: <http://www.sourceedit.com/>

If a development environment is necessary try Java Studio from Borland:

<http://www.borland.com/estudiojava/index.html>

Don't forget that Java Virtual Machine must be enabled on the client to be able to host applets:

[http://www.java.com/en/download/windows\\_automatic.jsp](http://www.java.com/en/download/windows_automatic.jsp)

### Making a connection.

Like any network application, you must open a communication channel to the remote device. In our example, we open a socket, and create two data streams to perform the actual sending and receiving of data. The following example will use a new Java Class called "tcpip". Simply copy the following code into a file called tcpip.java.

---

```
import java.*;
import java.lang.*;
import java.net.*;
import java.util.*;
import java.io.*;

/*
 * This class opens a TCP connection, and allows reading and writing of byte arrays.
 */

public class tcpip
{
    protected Socket s = null;
    public DataInputStream dis = null;
    protected DataOutputStream dos = null;

    public tcpip(InetAddress ipa, int port)
    {
        Socket s1 = null;
        try {
            // Open the socket
            s1 = new Socket(ipa.getHostAddress(), port);
        }
        catch (IOException e) {
            System.out.println("Error opening socket");
            return;
        }
        s = s1;
        try {
            // Create an input stream
            dis = new DataInputStream(new BufferedInputStream(s.getInputStream()));
        }
    }
}
```

## Applications Note: Connecting to the Device Server using Java

---

```
        catch(Exception ex) {
            System.out.println("Error creating input stream");
        }
        try    {
            // Create an output stream
            dos = new DataOutputStream(new BufferedOutputStream(s.getOutputStream()));
        }
        catch(Exception ex) {
            System.out.println("Error creating output stream");
        }
    }

    public synchronized void disconnect()
    {
        if (s != null) {
            try {
                s.close();
            }
            catch (IOException e){}
        }
    }

    public synchronized void send(byte[] temp)
    {
        try{
            dos.write(temp, 0, temp.length);
            dos.flush();
        }
        catch(Exception ex) {
            System.out.println("Error sending data : " + ex.toString());
        }
    }

    public synchronized void send(byte[] temp, int len)
    {
        try{
            dos.write(temp, 0, len);
            dos.flush();
        }
        catch(Exception ex) {
            System.out.println("Error sending data : " + ex.toString());
        }
    }

    public synchronized void send(String given)
    {
        // WARNING: this routine may not properly convert Strings to bytes
        int length = given.length();
        byte[] retval = new byte[length];
        char[] c = new char[length];
        given.getChars(0, length, c, 0);
        for (int i = 0; i < length; i++) {
            retval[i] = (byte)c[i];
        }
        send(retval);
    }

    public synchronized byte[] receive()
    {
        byte[] retval = new byte[0];

        try{
            while(dis.available() == 0); /* Wait for data */
        }
        catch (IOException e){}
        try{
            retval = new byte[dis.available()];
        }
        catch (IOException e){}
        try{
            dis.read(retval);
        }
    }
}
```

## Applications Note: Connecting to the Device Server using Java

---

```
        }
        catch (IOException e){}
        return(retval);
    }

    public int available()
    {
        int avail;

        avail = 0;
        try{
            avail = dis.available();
        }
        catch (IOException e) {}

        return(avail);
    }
}
```

---

Next, create the Text\_io class as the application. Copy the following code into a file called "Text\_io.java".

---

```
import java.awt.*;
import java.awt.event.*;
import java.lang.*;

public class Text_io extends Panel implements Runnable, TextListener {
    private tcpip gtp;
    String oldmessage = new String("");
    TextArea input_box = new TextArea("", 10, 60, 3);
    TextArea output_box = new TextArea("", 10, 60, 3);
    Thread timer;

    public Text_io(tcpip tp) {
        gtp = tp;

        setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        c.insets = new Insets(5,5,5,5);
        setBackground(java.awt.Color.lightGray);
        setSize(561,380);

        c.gridx = 0; c.gridy = 2; c.gridwidth = 1; c.gridheight = 1;
        c.weightx = 0.0; c.weighty = 0.0; c.anchor = GridBagConstraints.WEST;
        c.fill = GridBagConstraints.NONE;
        add((new Label("To Device Server: (Type Here-->))), c);

        //input_box
        input_box.addTextListener(this);
        c.gridx = 1; c.gridy = 2; c.gridwidth = 3; c.gridheight = 1;
        c.weightx = 0.5; c.weighty = 0.0; c.anchor = GridBagConstraints.CENTER;
        c.fill = GridBagConstraints.BOTH;
        add(input_box,c);

        c.gridx = 0; c.gridy = 4; c.gridwidth = 1; c.gridheight = 1;
        c.weightx = 0.0; c.weighty = 0.0; c.anchor = GridBagConstraints.WEST;
        c.fill = GridBagConstraints.NONE;
        add((new Label("From Device Server:")), c);

        c.gridx = 1; c.gridy = 4; c.gridwidth = 3; c.gridheight = 1;
        c.weightx = 0.5; c.weighty = 0.0; c.anchor = GridBagConstraints.CENTER;
        c.fill = GridBagConstraints.BOTH;
        add(output_box,c);
        output_box.setEditable(false);
        timer = new Thread(this);
        timer.start();
    }
}
```

## Applications Note: Connecting to the Device Server using Java

---

```
public void run() {
    int i;
    byte[] in;
    Thread me = Thread.currentThread();
    while (timer == me) {
        try {
            Thread.currentThread().sleep(200);
        }
        catch (InterruptedException e) { }
        if ( (gtp != null) && ((i = gtp.available()) > 0) ) {
            in = gtp.receive();
            /* remove non-printing bytes */
            for (i = 0; i < in.length; i++) {
                if (in[i] < 0x20)
                    in[i] = 0x20;
            }
            output_box.append((new String(in)));
        }
    }
}

public void textValueChanged(TextEvent e) {
    int len, i;
    String str = new String("");
    String message = input_box.getText();
    len = message.length() - oldmessage.length();
    if (len < 0) {
        for (i = 0; i < -len; i++)
            str += "\b";
        //System.out.println("Backspace");
    }
    else if (len > 0) {
        str = message.substring(oldmessage.length());
        //System.out.println("len = "+str.length()+" str = "+str);
    }
    oldmessage = message;
    if ( (len != 0) && (gtp != null) )
        gtp.send(str);
}
}
```

---

Next, create the actual applet that uses the tcpip and Text\_io classes. Copy the following code into a file called "Test.java".

---

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
import java.net.*;
import java.io.*;
import java.lang.*;
import java.text.*;
import java.util.*;

public class Test extends Applet {
    static private boolean isapplet = true;
    static private InetAddress arg_ip = null;
    static private int arg_port = 0;
    public tcpip gtp = null;;
    InetAddress reader_ip = null;
    int port = 10001;

    public void init()
    {
        gtp = null;
        reader_ip = null;
        port = 10001;
    }
}
```

## Applications Note: Connecting to the Device Server using Java

---

```
}

public void start()
{
    String st = new String("TCP/IP connection status: ");
    setFont(new Font("Dialog",Font.BOLD,16));
    setLayout(new GridBagLayout());
    GridBagConstraints c = new GridBagConstraints();
    c.gridx = 0; c.gridy = 0; c.gridwidth = 1; c.gridheight = 1;
    c.anchor = GridBagConstraints.CENTER;
    c.fill = GridBagConstraints.BOTH;
    c.insets = new Insets(5,5,5,5);
    setBackground(Color.yellow);
    setSize(600,500);

    /* Either get the IP address from the HTTP server if we're
       an applet, or from the commandline (if passed). */

    if (isapplet) {
        try{
            reader_ip = InetAddress.getByName(getCodeBase().getHost());
        }
        catch (UnknownHostException e){}
    }
    else {
        reader_ip = arg_ip;
        if (arg_port != 0) {
            port = arg_port;
        }
    }

    /* Open a socket to the Device Server's serial port */
    if (reader_ip != null) {
        if (gtp == null) {
            gtp = new tcpip(reader_ip, port);
            if (gtp.s == null) {
                st += "Connection FAILED! ";
                gtp = null;
            }
        }
    }
    if (gtp == null) {
        st += "Not Connected";
        add((new Label(st)), c);
        return;
    }
    st += "Connected";
    add((new Label(st)), c);
    /* You may now perform IO with the Device Server via
       * gtp.send(byte[] data_out);
       * byte[] data_in = gtp.receive();
       * functions.
       * In our example we'll use two TextBoxes which have been extended
       * to handle IO to the Device Server. Data typed in the upper
       * text box will be sent to the Device Server, and data received
       * will be displayed in the lower text box.
       */

    /******
    /* Start of custom application code */
    /* ADD YOUR CODE HERE */
    /******
        c.gridx = 0; c.gridy = 2; c.gridwidth = 3; c.gridheight = 1;
        c.anchor = GridBagConstraints.WEST;
        add((new Text_io(gtp)), c);
    /******
    /* End of custom application code */
    /******

}

public void destroy()
{
```

## Applications Note: Connecting to the Device Server using Java

---

```
        if (gtp != null)
            gtp.disconnect();
        gtp = null;
    }

    public void stop() {
    }

    public static void main(String[] args) {
        Frame frame = new Frame("TCP/IP Test");
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        if (args.length > 0) {
            try{
                arg_ip = InetAddress.getByName(args[0]);
            }
            catch (UnknownHostException e){}
            if (args.length > 1) {
                try{
                    arg_port = Integer.valueOf(args[1]).intValue();
                }
                catch (NumberFormatException e) {}
            }
        }
        Test ap = new Test();
        frame.add(ap);
        ap.init();
        isapplet = false;
        ap.start();
        frame.pack();
        frame.show();
    }
}
```

---

In our example we use two TextBoxes, which are in the Text\_io class to handle IO to and from the Device Server. Data typed in the upper text box will be sent to the Device Server, and data received will be displayed in the lower text box.

If you want to use your own code, you'll need to add your application code at the tag `/* ADD YOUR CODE HERE */` which is in the start() routine. You'll also need to replace the section of code between the 'custom code' fences. The syntax for the send and receive functions are listed below. Use `gtp.send()` to send data to the serial device, and `gtp.receive()` to read data from the serial device.

Syntax:

```
gtp.send(byte[] array)
byte[] array = gtp.receive()
```

Now you need to compile the Java source code into class files.

```
C:> javac tcpip.java
C:> javac Text_io.java
C:> javac Test.java
```

To test your Java application:

```
C:> java Test xxx.xxx.xxx.xxx 10001
Where: xxx.xxx.xxx.xxx is the IP address of the Device Server, and
10001 is the TCP port number the Device Server is listening on.
```



## Applications Note: Connecting to the Device Server using Java

---

Once you've successfully connected to the Device Server, and completely debugged your Java application, you can then create a HTML document, which includes your Java application in the form of an Applet. Copy the text below into a file called "test.html".

---

```
<HTML>
<BODY>
<CENTER>
<APPLET CODE="Test.class" WIDTH="862" HEIGHT="512"></APPLET>
</CENTER>
</BODY>
</HTML>
```

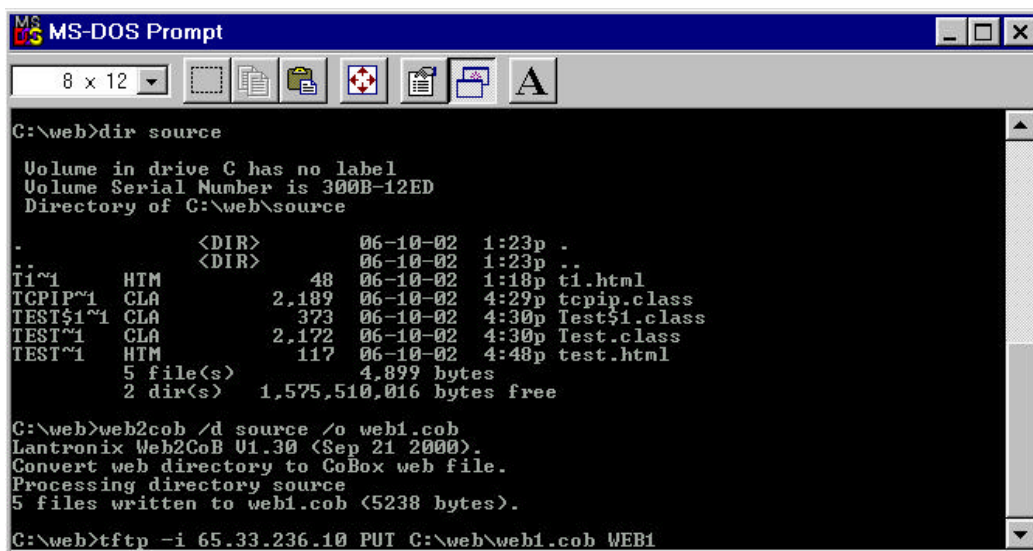
---

Now copy the Java classes and new HTML file into the \web\source directory.

```
C:> copy *.class \web\source
C:> copy test.html \web\source
```

Create a new .COB file, as described in Apps Note AMC-AN-LAN03, and load it into the Device Server via ftp. (you can of course use XportInstaller or DeviceInstaller to do this too)

```
C:> cd \web
C:> web2cob /d source /o web1.cob
C:> tftp -i xxx.xxx.xxx.xxx PUT \web\web1.cob WEB1
```



The screenshot shows an MS-DOS Prompt window with the following text:

```
MS-DOS Prompt
8 x 12
C:\web>dir source

Volume in drive C has no label
Volume Serial Number is 300B-12ED
Directory of C:\web\source

.                <DIR>          06-10-02  1:23p  .
..               <DIR>          06-10-02  1:23p  ..
ti~1             HTM             48        06-10-02  1:18p  ti.html
TCPiP~1         CLa           2,189    06-10-02  4:29p  tcpip.class
TEST$1~1       CLa             373      06-10-02  4:30p  Test$1.class
TEST~1         CLa           2,172    06-10-02  4:30p  Test.class
TEST~1         HTM             117      06-10-02  4:48p  test.html
5 file(s)       4,899 bytes
2 dir(s)       1,575,510,016 bytes free

C:\web>web2cob /d source /o web1.cob
Lantronix Web2CoB 01.30 (Sep 21 2000).
Convert web directory to CoBox web file.
Processing directory source
5 files written to web1.cob (5238 bytes).

C:\web>tftp -i 65.33.236.10 PUT C:\web\web1.cob WEB1
```

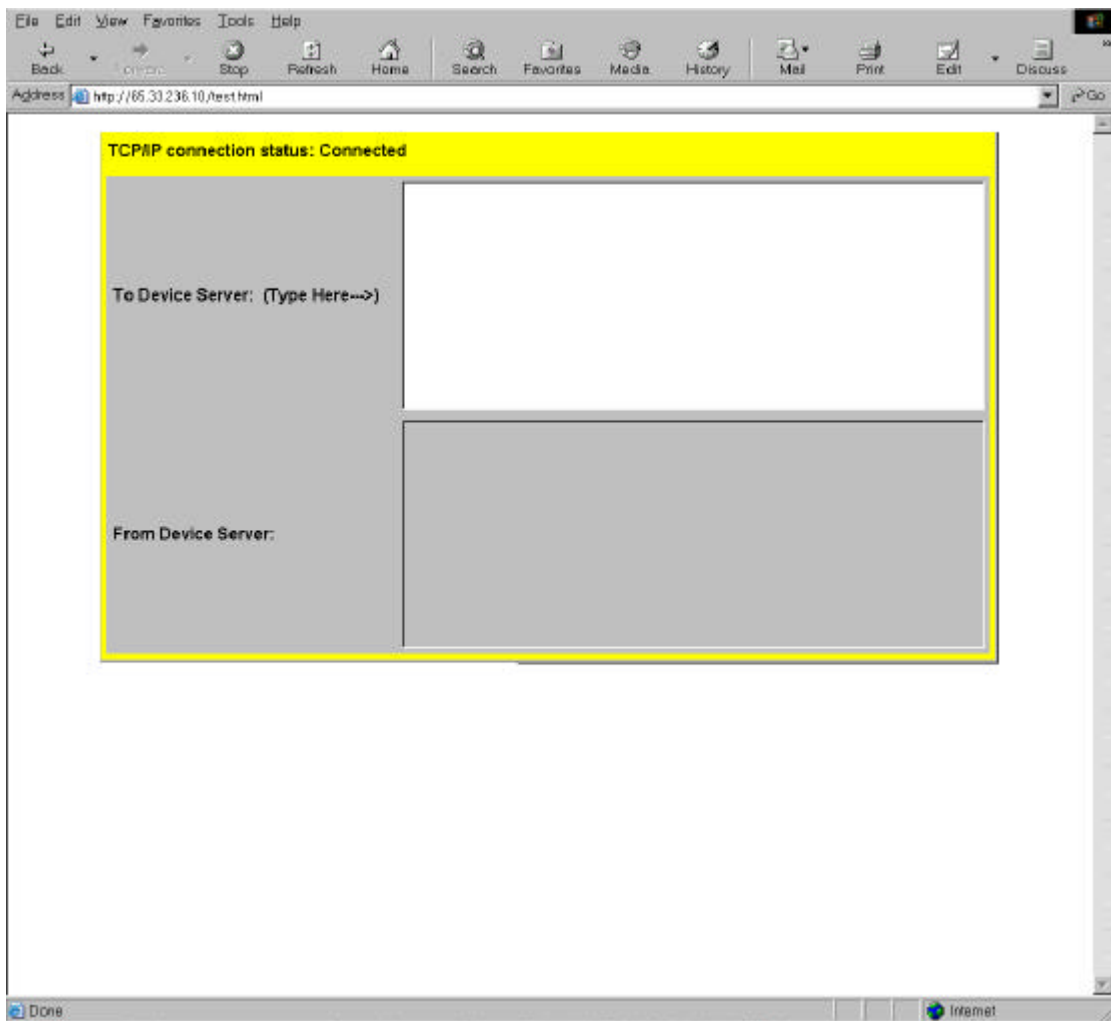
You can now access your new web page from any browser using the new URL:

<http://xxx.xxx.xxx.xxx/test.html>

Where xxx.xxx.xxx.xxx is the IP address of the Device Server.



## Applications Note: Connecting to the Device Server using Java



### **Summary**

The Device Server supports an internal web server that may be utilized for storage and retrieval of documents, images, and Java applets. With a little effort, the experienced web programmer can create custom web pages to interact with any serial device and present the information to a user in a very familiar and friendly user interface.