Java/Web Manual

Table of contents

1. Overview

1.1. Overview

1.1.1. Introduction

This document describes how to use the SoftPLC Web Server and embedded Java language support. The SoftPLC Web Server is a chargeable software option to any SoftPLC based controller. All SoftPLC controllers include the embedded Java language support at no charge.

With the SoftPLC Web Server you can use a web browser as an HMI, or as a user-friendly way to view files (eg: pdf files) stored on the SoftPLC flash disk.

With Java language support, you can run the SoftPLC Web Server (which was written in Java by SoftPLC Corp.). You can also run programs written in Java language, using the Java Application Program Interface (API) described in the SoftPLC Programmer's Toolkit. Additionally, you can use TOPDOC Loadable Modules (TLM's) written in Java such as the free ladder instructions "SendEmail" and "SQL Database Read", both of which are described in this manual.

More than one Java application can run at the same time on SoftPLC.

Note:

This document does not attempt to describe how to develop programs in Java, nor does it attempt to describe HMTL programming. Free documentation on both these topics is available from a number of sources. This document also assumes that you are familiar with SoftPLC configuration via TOPDOC NexGen and .LST files.

1.1.2. Important Concepts

Java is a programming language. Java program code is called "bytecode". In order to run Java bytecode, a Java Virtual Machine (JVM) is required.

SoftPLC uses a JVM called "MachJ", which has been implemented as a TOPDOC Loadable Module (TLM) that runs in SoftPLC. You must load and configure MachJ.TLM in SoftPLC in order to run Java bytecodes or the SoftPLC Web Server. This is described in Chapter 2.

You also need to install the Sun Microsystems version 1.1 Java Runtime Environment (JRE), called RT.JAR. This is included with SoftPLC. Installation is described in the Installation section.

1.1.3. Java Developer Concepts

Java bytecode is developed with any Java compatible compiler or Integrated Development Environment (IDE). The output from the IDE is bytecode that may be executed on SoftPLC.

To develop bytecode, you may use any third party IDE or Java Development Kit (JDK) which produces Java compatible bytecode. Some IDE's to consider are:

- Borland's JBuilder
- Sun's Forte
- NetBeans Developer
- IBM's Visual Age for Java
- Symantec's Visual Café
- KAWA by Tech-tools

There are many others that you can use too. However, users must be cautioned when considering Microsoft's Visual J++. Although it is possible to generate Java compatible bytecode with Visual J++, it is also possible to generate bytecode which is not Java compatible. Only Java compatible bytecode is supported by SoftPLC.

SoftPLC Corp.'s definition of "Java compatible" is given by Sun Microsystem's Java Virtual Machine specification, Sun's Java Language Specification, and Sun's JDK 1.1 API's, including JNI.

1.1.3.1. Supported Java

The Java language environment supported on SoftPLC will run most non-GUI programs that Java 1.1.x will run. AWT is not supported, but most all other classes from the Java 1.1 environment are fully supported.

You can use all the classes in the following Java language packages:

- java.lang
- java.lang.reflect
- java.util
- java.util.zip
- java.net
- java.io
- java.text

More than one Java application can run at the same time on SoftPLC or SoftWIRES.

Note:

Because of the short filename constraint, bytecode running on SoftPLC must be in *.JAR format. SoftPLC supports compressed or non-compressed JAR files.

2. Java Installation/Configuration

2.1. Installation

This chapter describes how to install and configure the software used to run Java bytecode in SoftPLC - the JRE and the JVM.

2.1.1. Directories

SoftPLC is a 32 bit realtime operating system that uses a file system limited to the DOS compatible 8.3 filename.ext (i.e. short filename) format.

Here is a picture of a runtime directory tree that has the Java support and the core SoftPLC support installed:

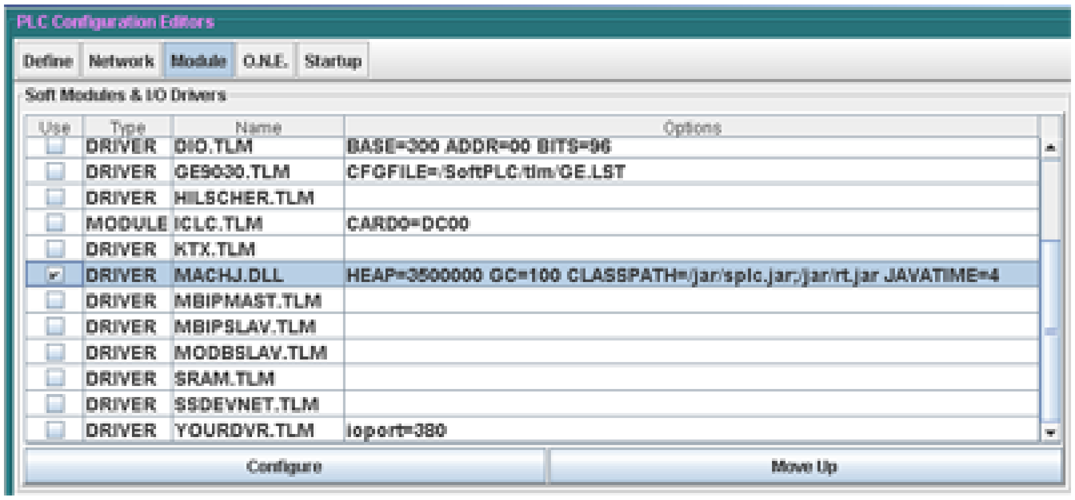
```
\SOFTPLC
    \APP
        * (apps here)
    \FTP
        ALWAYS.LST
        MODULE.LST
        NETWORK.LST
        readme.txt
        STARTUP.LIST
    \RUN
        MODLET.LST
        ALWAYS.LST
        STARTUP.LST
        MODULE.LST
        *.dll
        *.BIN
        *.CFG
        RUNFTP.BAT
        RUNSPLC.BAT
        SOFTPLC.EXE
    \TLM
        *.tlm
\LST
    \SYS
        ALIAS.LST
        RESOURCE.LST
        SERVER.LST
        SERVLET.LST
        USERROOT.LST
    \USR
        ALIAS.LST
        RESOURCE.LST
        SERVER.LST
        SERVLET.LST
        USERROOT.LST
        USERS.LST
\JAR
    RT.JAR
    SPLC.JAR
    \MODLET
        * (modlets here)
    \SERVLET
        * (servlets here)
\HTM
    \USR
        * (website files here)
    \SYS
        * (global webserver files)
```

2.1.2. Obtaining RT.JAR

All SoftPLC's include the required version of RT.JAR (Sun's Java Runtime Environment). The default installation location for RT.JAR is in the \JAR folder in SoftPLC.

2.1.3. Loading the JVM

The SoftPLC JVM, called MachJ.TLM, is loaded as a normal TOPDOC Loadable Module (TLM) under SoftPLC. To load the TLM, you use TOPDOC NexGen's PLC Configuration Editor. Select to USE machj.tlm.



Use	Type	Name	Options
☐	DRIVER	DIO.TLM	BASE=300 ADDR=00 BITS=96
☐	DRIVER	GE9030.TLM	CFGFILE=/SoftPLC/tlm/GE.LST
☐	DRIVER	HILSCHER.TLM	
☐	MODULE	ICLC.TLM	CARD0=DC00
☐	DRIVER	KTX.TLM	
✓	DRIVER	MACHJ.DLL	HEAP=3500000 GC=100 CLASSPATH=/jar/splc.jar;/jar/rt.jar JAVATIME=4
☐	DRIVER	MBIPMAST.TLM	
☐	DRIVER	MBIPSLAV.TLM	
☐	DRIVER	MODBSLAV.TLM	
☐	DRIVER	SRAM.TLM	
☐	DRIVER	SSDEVNET.TLM	
☐	DRIVER	YOURDVR.TLM	ioport=380

Then, you need to set the module options. You simply edit the text string in the Options column with your desired parameter values. The supported options may occur in any sequence. Each option consists of a NAME and VALUE pair that has the general format:

```
NAME=VALUE
```

with no spaces between NAME, equal sign, and VALUE.

A full example showing several options (all one line):

```
DRIVER=MACHJ.TLM JAVATIME=8 HEAP=16000000
CLASSPATH=c:\jar\splc.jar;c:\jar\rt.jar;c:\jar\modlet\mod.jar; JSTACK=12000
```

The machj.tlm options are:

CLASSPATH=<list of paths>

<list of paths> is a semicolon separated list of places that the JVM is to look for bytecode for the Java programs you want to run. Each element in the list may be either 1) a JAR file or 2) a subdirectory. Normally your classpath will always have `c:\jar\splc.jar;c:\jar\rt.jar` as the first two paths.

Note:

Bytecode comprising a "Servlet" is an exception. It is loaded by the SoftPLC web server from information that is supplied in its configuration files, so there is **never a reason to add a path to the \jar\servlet directory**.

The JVM first looks to see if there is an environment variable CLASSPATH= and if so, that is the default classpath. If the CLASSPATH option is supplied in the module editor (and in MODULE.LST), it fully overrides the CLASSPATH environment variable. Most users use the CLASSPATH option and not the environment variable.

HEAP=<maximum Java heap in bytes>

<maximum Java heap in bytes> must be at least 7000000 and can be much larger if you have enough RAM. If this option is not provided, the default value is 7000000. This value leaves about 7 MB remaining on a 16MB system for the core (non-Java) SoftPLC software. You must have at least 16MB of RAM in your CPU to run the JVM under SoftPLC. For the Tealware CPU (Cat No SPLC-2) the recommended HEAP value is 3500000. Commas are not allowed.

Example: "HEAP=56000000"

JAVATIME=<java time>

<java time> is the amount of “pause” time that the Ladder Thread yields to Java Threads at each “check point”. A check point is a point in the Ladder Logic Interpreter’s scan where it checks on the O.N.E. queues (and may call some of the TLM’s giving them periodic control of the CPU). A check point occurs unconditionally at the end of each Ladder Program scan. In addition to that one check point, you may get additional check points every <X> number of rungs of ladder logic scanned. <X> is configured in the STARTUP options tab in TOPDOC. If <X> is greater than or equal to the total number of

rungs per scan, then you will only get one check point per Ladder Program scan.

At each check point, you can cause the main Ladder Thread to yield to normal priority Java Threads for <java time> milliseconds. Since the Ladder Thread runs at a higher priority than normal priority Java Threads, without this mechanism, only the main Ladder Thread will actually run Java code. All normal priority Java Threads will be starved for CPU time. By yielding from the Ladder Thread for specific amounts of time to all the normal priority Java Threads, you can specify the (nearly exact) amount of time that is spent running Java code collectively by all the normal priority Java Threads.

When <java time> is actually applied, SoftPLC rounds it up to the nearest clock tick. On SoftPLC this is currently a 4 msec granularity.

Example 1: Say you had a 4000 rung program and you wanted to spend no more than 16 msecs executing Java code (in normal priority Java Threads) per scan of Ladder. You could set the comm check interval to 1000, giving you a total of 4 check points per scan. Then at each check point you would choose to yield 4 msecs to normal priority Java Threads. Do this with the JAVATIME=4 option.

Example 2: Say you had no Ladder Logic and were doing everything in normal priority Java Threads. You still have a Ladder Thread scanning the end of program statement, and it will have a single check point pertaining to the end of program. You could supply JAVATIME=20 which would give you 20 msecs of Java time per ladder scan. Since the ladder scan is so fast in this case, the CPU would almost immediately yield back to Java code for the next 20 msecs. This essentially gives the Java Virtual Machine almost all the CPU time, but every 20 msecs the main Ladder Thread will run and service the O.N.E. queues for TOPDOC and any MMI.

Example 3: This example assumes you are familiar with the Java Toolkit for SoftPLC. Say you had no Ladder Logic and were doing everything in Java language, but doing it from the main Ladder Thread in the Modlet scan() method. In this case, there is no competition for CPU time since there is only one Thread. It will still scan the end of program statement in ladder, but your Java code will get control in that check point via the scan() method. You need supply no comm check interval since there is no CPU time competition. You will however, have to regularly return from the scan() method so the Ladder Thread can do the O.N.E. queue checks.

Warning:

Since the JAVATIME=<java time> option is so important to system performance, it is best to get help from an expert. Call SoftPLC for advice or consult your in-house expert.

`JSTACK=<maximum Java stack size in bytes>`

<maximum Java stack size in bytes> must be at least 2000 and defaults to 8000 if this option is not provided. This is the size of each Java thread’s stack. You need only adjust this larger if you see a StackOverflowException while running your application.

Example: "JSTACK=12000"

`NSTACK=<maximum Native stack size in bytes>`

<maximum Native stack size in bytes> must be at least 150000 and defaults to 150000 if this option is not provided. This is the size of each Native thread’s machine stack. You need only adjust this larger if you see a StackOverflowException while running your application.

Example: "NSTACK=200000"

`COMPORTS=<list of ComPorts>`

<list of ComPorts> is a comma separated list of serial ports that you wish to make available to Java programs under SoftPLC. If you are not using Java for serial ports, do no use this option.

ComPorts are numbered from 1 to 36. Each ComPort number is implemented by hardware residing at a particular IRQ and IOPORT, as given by COMGENIE.TXT. ComPorts which are not to be used by Java applications should not be given in this option. The <list of ComPorts> will cause an array of Java ComPort instances to be created in SPLC.port[], such that array element SPLC.port[0] will be a ComPort assigned to the first port number in the list, element SPLC.port[1] will be a ComPort assigned to the 2nd number in the list, etc.

Example 1: "COMPORTS=1" will assign SPLC.port[0] to COM1.

Example 2: "COMPORTS=3,28,1" will assign SPLC.port[0] to COM3, SPLC.port[1] to COM28 and SPLC.port[2] to COM1.

Do not put in a trailing comma. Again, spaces are not allowed. The default is to create SPLC.port[] with length zero, which means you will not be able to use the serial ports with Java applications.

3. Java Language Modules

3.1. Applets

An applet is a Java compliant software component that runs in the context of a web application on a client computer. The applet must run in a container, which is provided by a browser, or through a plug-in, or a variety of other applications including mobile devices that support the applet programming model.

Unlike a program, an applet cannot run independently (has no main entry point). Applets feature display and often interaction with the human user, and are usually stateless and have restricted security privileges. An applet characteristically performs a very narrow function that has no independent use.

Applets can be used with SoftPLC, but their creation and use is beyond the scope of this manual.

3.2. Modlets

3.2.1. Introduction

Modlets are the top most Java entry points within SoftPLC. All Java language execution within SoftPLC has its origin within a Modlet. A Modlet is somewhat like an Applet without a GUI. Although Modlets have the ability to print textual output to the screen, it is not GUI, and they cannot take keyboard or mouse input. Modlets are intended to interact mainly with the SoftPLC datatable. There is no limit to how many Modlets you can concurrently run under SoftPLC.

Modlets receive events from SoftPLC, indicating things like end of scan, operating mode transitions, installation or startup, and shutdown.

Modlets are often written to use command line parameters. Command line parameters are specified with a text editor just after the Modlet class name within the text file MODLET.LST.

Here is a sample MODLET.LST file:

```
com.softplc.Weblet C:\LST\SYS
com.softplc.ApplicationLauncher MyApp myparam1 myparam2
MyModlet
```

Modlets and their command line options must go on a single line. Blank lines are ignored, and anything to the right of a semicolon is ignored.

In this example we have three Modlets given, as shown in the table below.

Interpretation of sample MODLET.LST file:

Modlet Class Name	Command Line Parameters
com.softplc.Weblet	C:\LST\SYS
com.softplc.ApplicationLauncher	MyApp myparam1 myparam2
MyModlet	none

Each Modlet class file must be made accessible via the classpath option given to MACHJ.TLM. See the section on classpath.

3.3. Weblets

We use the term Weblet to refer to an HTTP server, a web server. Each SoftPLC can run multiple HTTP servers (Weblets). The default installation of the web server sets up 2 weblets. For example, one can be used for administrative activities, and the other can be used for normal operating personnel activities as you chose.

Because a Weblet is actually an implementation of a Modlet, a Weblet is started by specifying an instance of class com.softplc.Weblet in the MODLET.LST file. This Modlet takes a single command line parameter, the directory path where the Weblet's dedicated configuration files are to be found. Each Weblet must be given its own configuration directory and associated configuration files.

The configuration files are SERVER.LST and RESOURCE.LST. These are text files that you may edit with a text editor. Assuming you will have two Weblets, then you might have the following directory tree fragment:

```
+-\LST
. +-\SYS
. | + RESOURCE.LST
. | + SERVER.LST
. +-\USR
. + RESOURCE.LST
. + SERVER.LST
```

In your MODLET.LST file, in addition to any other entries you might have, you would have the following two entries:

```
com.softplc.Weblet C:\LST\SYS
com.softplc.Weblet C:\LST\USR
```

More Weblets can be added as needed. Simply create another subdirectory, say usr2, below the \LST subdirectory shown above, and then add another entry to MODLET.LST.

3.3.1. Weblet Configuration

What makes a Weblet unique are its configuration parameters. Each Weblet listens on a particular TCP/IP port number on a particular TCP/IP interface, has its own document root, and its own name. Your SoftPLC CPU might have more than one TCP/IP interface (ethernet card/port). Each Weblet may only listen on one interface. More than one Weblet can listen on the same interface. Each Weblet needs a unique port number on any given interface. If two Weblets are using different interfaces, only then may they use the same port number.

Each parameter belongs to a specific configuration file, either RESOURCE.LST or SERVER.LST. Remember that each Weblet instance has its own set of these files. These files may be edited by a text editor to modify the desired configuration parameters. **The**

editable text is always to the right of the first semicolon. There are more parameters in these files than are described here. More information can be found in the configuration files themselves. The important parameters are listed here as are the file in which they belong.

Weblet.inetaddress:<IP address> in SERVER.LST

IP address is an optional parameter that is only needed if you have more than one TCP/IP interface. If you have two or more ethernet cards in the SoftPLC and each are on a TCP/IP network, then you have more than one TCP/IP interface. If you don't supply this parameter, then the first non-loopback interface will be assumed. Otherwise, provide the IP address for the interface in numeric form (e.g. 130.1.1.65) that this Weblet instance is to use. SoftPLC currently supports up to 5 TCP/IP interfaces, the loopback interface, and up to 4 ethernet cards. PPP and SLIP over RS-232 or modem are not supported at this time.

Weblet.port:<port number> in SERVER.LST

Port number is the TCP/IP port number that this Weblet listens on for new connections. The normal port number for the World Wide Web Internet HTTP servers is 80. Each Weblet must have its own unique combination of interface and port number, as described above. Since browsers default to 80, that port number is normally used for the USR Weblet. The SYS Weblet, defined by the configuration files in LST\SYS, defaults to 9090. A port number can be any number between 1 and 65535.

Weblet.name:<server name> in SERVER.LST

This is the name of this Weblet instance. It should generally be set to same name as what the client web browser must use to find this machine on the TCP/IP network. It may also be set to an IP address if there is no Domain Name Server on the network.

Weblet.document.root:<document root> in SERVER.LST

A document root is the directory that the web browser user will perceive as the root directory. He points at the document root directory by providing the shortest URL for this server, with the least amount of path information.

For example, say <document root> is set to \htm\usr, and the client requests from URL http://ipaddress/page.htm. In this example ipaddress is used by the client machine to find the SoftPLC machine from the Domain Name Server on your network, so the only part of the URL pertaining to the filename is page.htm. Notice there is no path information as part of this filename. So the actual file retrieved under these circumstances would be \htm\usr\page.htm.

All file references will have the <document root> path prefixed, so a request for

http://ipaddress/subdir/file.htm

would result in \htm\usr\subdir\file.htm being retrieved.

Weblet.servlet.path:<servlet path> in RESOURCE.LST

Servlet path is a semicolon separated list of directories, each of which will contain one or more *.JAR files. JAR files are not listed in this parameter, but rather the directories in which they reside are listed. For example, a valid <servlet path> would be C:\jar\servlet;C:\myserv; Any and all *.JAR files in these two directories will be searched for servlets that are referenced by a URL.

Weblet.server.admin:<admin email address> in SERVER.LST

Whenever a file not found error comes back to the client because it requests a file that does not exist, the email address of the person that adminstrates this Weblet may be displayed. This way the client can simply click on that hyperlink to be able to send an email message to that administrator. For this to work you need a SMTP server on the network and the client's browser must be configured to send email through that SMTP server. The default email address is "joeplantengineer@company.com"

3.4. Servlets

A Weblet is an HTTP server running under SoftPLC. SoftPLC Weblets support the JavaSoft Servlet API. Servlets are dynamically loaded Java code fragments that get invoked on demand by a client side web browser, but actually run in the server. For a tutorial on Servlets, see:

http://java.sun.com/docs/books/tutorial/servlets/index.html

The web browser actually has no idea it is doing this, but depending on the URL that is requested, a different Servlet is invoked on the server in response to a particular URL. SoftPLC supports an unlimited number of Servlets.

User's may write their own Servlets to handle any number of activities on the server side. The Servlet API is documented in the SoftPLC Java Toolkit as well as on the Sun website.

This manual is not intended to be a programming manual. There are plenty of skilled developers in this field of server side development, as there are over 5,000 websites being added to the Internet each day.

3.4.1. Uses for Servlets

Servlets can dynamically create web pages that contain live data. They are also very useful for processing form data that was entered into a web browser page.

They are also used as a partnering agent to interact with network transactions of any kind initiated by client side Applets. A common scenario will use an Applet to process a series of data entry screens, and at completion this data is sent to a URL on the server using the HTTP POST request. The URL is carefully chosen so that the request gets routed to a Servlet that has been specially written to handle data in the format created by the Applet on the other end of the wire.

Servlets may write to disk, so there is almost no limit to what they can perform.

3.4.2. Invoking a Servlet

The client web browser may invoke a Servlet by referencing a URL (either manually keyed in, or occurring as a hyperlink), that contains the basic form:

http://<ipaddress>/servlet/<servletname>

Examples:

http://softplc1/servlet/ShiftReport	would invoke the Servlet "ShiftReport" on the

	box whose IP Address is softplc1
http://130.1.1.155/servlet/com/company/Servlet	would invoke the Servlet implemented by class "com.company.Servlet" on the box whose IP Address is 130.1.1.155

Another way to invoke a servlet is to add an entry to the \LST\USR\SERVLET.LST file so that the servlet is tied to a particular file extension. Here is an example for the SmartPage servlet:

ServletByRequest(GET|*|stp): SmartPage

This causes any URI that ends in .stp to invoke SmartPage. See the comments in SERVLET.LST for more info.

3.4.3. Symbolic IP Addresses

There are two forms for IP addresses, numeric or symbolic. An example of a numeric IP address is 130.1.1.55 and an example of a symbolic IP address is softplc1, each shown in the table above.

In order to use a symbolic IP address, sometimes called a domain name, you must have a Domain Name Server (DNS) somewhere on your network, and the lookup table in the DNS must have an entry which maps your symbolic name to its numeric IP address. Many UNIX boxes have this capability and NT server does too. If you have lots of browser clients, this is the most maintainable scheme to use.

For single client environments, or in the event you don't have a DNS, you can edit the text of your "HOSTS" file on your local disk. On WinNT the file is C:\WIN32\SYSTEM32\HOSTS. On WinXP and higher, the file is C:\WINDOWS\SYSTEM32\drivers\etc\hosts. The file HOSTS.SAM can be copied to HOSTS and then simply add your symbolic entries there.

3.4.4. Query Strings

By using query strings that may be tacked onto the end of URL's, any given Servlet can respond in different ways. A query string has a series of <name>=<value> pairs, each separated by the '&' character. As an example URL, consider:

http://softplc1/servlet/mydata?address=S23&length=4

Query strings are always introduced at the end of a URL by a question mark character '?'. So in this example there are two name/value pairs: "address=S23" and "length=4".

Query strings are normally generated by the web browser as the result of processing an

HTML form.

4. Sendmail Modlet

4.1. SENDMAIL

SoftPLC provides a freely modifiable and deployable TOPDOC Loadable Instruction (TLI) that adds the ability to send email from a SoftPLC system. The TLI is Java code that gets called from Ladder, and the packaging of the TLI is in the form of a Modlet.

A similar instruction could be used to receive email as well, although that would require a development effort by somebody familiar with Java programming as well as the POP3 protocol. SendMail only supports the sending of email and not receiving.

4.1.1. Technical Overview

A dedicated Service Thread is started which blocks on a com.softplc.Queue. This is sort of an "outbox" for the SoftPLC. Mail messages are sent from this Queue so that the main control Thread is not involved in sending the messages. It is however, involved in formatting the text for the messages, and since this happens in the context of the Ladder Thread, this needs to be done quickly. Therefore we chose a simple fast scheme where all message text could be produced without consulting the disk.

The Service Thread is the one that interacts with the TCP/IP socket and normally runs at a priority lower than the ladder thread. It can operate on a leisurely basis since the text has already been captured in the outbox Queue by the time the message is actually sent.

Note:

If your SMTP server is not on your local subnet, then for each SoftPLC, you need to set "GATEWAY=" to your gateway machine in NETWORK.LST, see below.

4.1.2. Hardware Requirements

SoftPLC must be placed on an ethernet to send email.

SendMail uses the SMTP protocol over ethernet TCP/IP to send mail. SoftPLC is the SMTP client. Somewhere in the picture there must be an SMTP server. This server can be 1) in-house, or 2) at your Internet Service Provider (ISP).

When running in a SoftPLC, it is recommended that you not try and use a modem, but rather use an in-house ethernet where one of the boxes on the in-house ethernet is running an SMTP mail server program. The SMTP server will then forward the email message as needed.

1. When using an In-House SMTP server, there are several possibilities:

   1. Any Linux machine has this capability, and most Unix boxes do too.
   2. Mercury/32 is a freeware SMTP server that can also dialout over modem to your ISP. It only runs on Windows. Information on Mercury/32 can be found in the usenet news group: bit.listserv.pmail.

   Note that having an internal SMTP server can be handy even if you have an external one as well. Departmental internal emails can be routed via this server.
2. When using your ISP as the nearest SMTP server, then you need some connection to your ISP. If by modem, then you can use our product Gatecraft Junction (GCJ-2xx) which acts as both: a) ethernet to ppp router, and b) a dialup modem. Several SoftPLC's can share the same Gatecraft Junction (GCJ-2xx). **You may not use a modem in your SoftPLC machine, the TCP/IP interface from the SoftPLC machine must be via an ethernet interface.**

4.1.3. Software Configuration

The SENDMAIL TLM is packaged for you as a downloadable set of files here. The package includes some tools for configuring the TLM, and the TLM itself which consists of a number of source files. However, you need to edit only two of them - MAILDATA.LST and MailData.java to fully configure the TLM for your application. You can use any text editor to modify the files. Detailed instructions for editing are included in the files themselves.

In brief, the steps required to configure the SENDMAIL TLM are as follows. Each step is described in more detail in this section.

1. Edit MAILDATA.LST
2. Edit MailData.java
3. Run MAKEJAR.BAT
4. Copy SENDMAIL.JAR into C:\JAR\MODLET on the SoftPLC CPU
5. Add C:\JAR\MODLET\SENDMAIL.JAR to the end of your CLASSPATH in MODULE.LST
6. Add the line SendMail to MODLET.LST

4.1.3.1. MAILDATA.LST

MAILDATA.LST is a simple text file that provides the information on the message recipients, your sending address, the SMTP server, and Authorization. Configuration is done by changing the following fields in the \SoftPLC\TLM\MAILDATA.LST file:

- FromAddr
- ServerAddr
- AuthMethod
- AuthID
- AuthPasswd

4.1.3.2. MailData.java

MailData.java is a java source file. However, even non-Java programmers should be able to do the required editing of MailData.java. In this file are the message subject(s), message text(s), and attachment(s) detail for the emails you want to send. Each separate message has an index number that the Ladder instruction can be used to reference.

- getSubject() method changes the subject for emails sent with the given msgIndex
- getMessageBody() method changes the body for emails sent with the given msgIndex
- getMessageAttachment() method changes any attachments for emails sent with the given msgIndex

4.1.3.3. MAKEJAR.BAT

MAKEJAR.BAT is a batch file you can run from a Windows command prompt. It invokes a supplied free Java compiler, which will compile and create a new SENDMAIL.JAR using your edited files. Unless your edits introduced Java syntax errors, you should see an updated SENDMAIL.JAR file with a current time stamp on it in your working directory.

After compiling and jarring SENDMAIL.JAR, you need to:

1. Copy SENDMAIL.JAR into the C:\JAR\MODLET directory on the SoftPLC machine. You can use FTP or a simple COPY to the flash disk to perform this.
2. Using TOPDOC NexGen, add C:\JAR\MODLET\SENDMAIL.JAR to the end of your CLASSPATH option in MODULE.LST, separating it from the rest of the list with a semicolon. CLASSPATH is an option to the MACHJ.TLM discussed elsewhere in this document. Send the edited MODULE.LST to the SoftPLC.
3. Add SendMail to the MODLET.LST file, using FTP or editing the flash disk directly. When it comes to Java classnames, they are case sensitive, so you must add exactly this line to that file:
   ```
   SendMail
   ```

4.1.4. Authorization

The SENDMAIL TLM supports SMTP AUTH. The support consists one of the following: LOGIN or PLAIN. PLAIN is the recommended one to use when available.

Please see MAILDATA.LST for information on how to configure the TLM with authorization.

4.1.4.1. Choosing AuthMethod

Open a command prompt and type `telnet [hostname] 25`, which would connect on port 25 to the SMTP server [hostname]. Then type `EHLO anyTestName` and check for a line containing **AUTH**. Afterwards, you can type `QUIT` to terminate the connection.

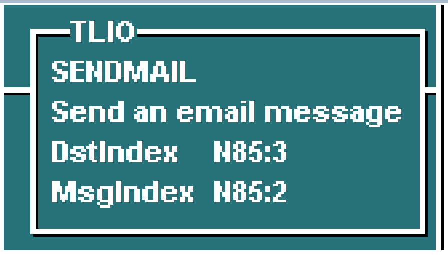
If no AUTH line is found, then the server does not support or is not configured for SMTP authorization, and you should set AuthMethod to AUTH_NONE. If you do see an AUTH line, but it does not contain either LOGIN, PLAIN or ANONYMOUS, then the SENDMAIL TLM is not compatible with that server. If the line does contain one of the previously mentioned types, then use the table below to determine the proper value for AuthMethod:

AUTH Line	AuthMethod
PLAIN	AUTH_PLAIN
LOGIN	AUTH_LOGIN
ANONYMOUS	AUTH_NONE

4.1.5. Deployment

The ladder instruction shows up as "SENDMAIL" in TOPDOC. It takes two integer parameters:

dstIndex	an integer which indexes into a hardcoded list of email addresses within MailData.lst
msgIndex	an integer that controls which of several potential email messages to send.



The SendMail TLI is level triggered, not rising edge triggered. This means that it will send another email every time the rung containing the instruction is energized true when it is scanned. This is likely to be a problem, but one you can overcome by placing a ONS instruction in series with the SENDMAIL instruction. The ONS instruction should be just to the left of the SENDMAIL instruction. This pair of instructions will serve to make the SENDMAIL function rising edge triggered as desired.

The consequences of not using a rising edge mechanism is that you will fill up the internal queue within SendMail.java, and somebody will get 50 or so email messages all saying the same thing.

5. Database Read Modlet

5.1. Database Read Modlet

This document describes a TLI, which resides in a SoftPLC Java Modlet (Modlet) that is provided at no charge with all SoftPLC licenses. The Modlet is called **JDBC_TLM** and resides in a file with the name JDBC.java. This Modlet was developed by SoftPLC Corporation, but in order to apply it you will have to modify the source code and compile and zip it to a JAR file. As is, JDBC_TLM can be used to perform simple INT column reads from a SQL-compliant relational database. You will have to decide on the SQL query string to use and the column names to read. A single ladder instruction is implemented in the Modlet, however you can add additional ones that might vary on any of: table name, query string, or even SQL command. That is, you could change the code to do SQL UPDATEs rather than or in addition to SQL SELECTs

Note:

One of the things to consider if you have the need for vastly different operation, is to change the division of responsibility between Ladder Logic and Java. Right now, the Java is called from Ladder Logic and returns the data to the Ladder Logic Datatable. You might want to change the Ladder instruction so it merely signals the Java and then the Java does what it needs to with the retrieved data rather than putting it back into the Datatable.

5.1.1. Overview

The TLI included in JDBC.java is **JDBC_READ**, which is designed handle a single row read of columns given by an SQL query string. As written, only a single row is handled. (However you can modify this to you're liking). The sample source reads only three INT columns but it is a simple matter to extend this to any number of columns, all of which would be INT.

From this point forward, this application note will assume that you intend to use it as is. Customization should be straight forward for someone familiar with Java. If you are not a Java programmer, you will likely need the help of one to do any major modifications to the source code. Refer the Java programmer to the SoftPLC Java API's which are online at http://softplc.com/api/index.htm.

5.1.2. Source Code

The source code to the JDBC Modlet is available at http://dl.softplc.com/pub/JDBC.java. Download this file and save to your development system's disk as JDBC.java (as always, case is significant with Java source code filenames).

5.1.3. Instructions

1. Obtain an SQL-compliant relational database server.

   If you are not constrained to use one provided by the powers that be, you can use MYSQL. This is a free database server that runs on Linux or Windows. Download it from http://www.mysql.com/.

   he JDBC.java is setup for this database. Download the latest server version for your server's operating system.

2. Install the database server software on a server computer.

   Consult the database server documentation on how to do this. If using MYSQL, this is fairly well documented.

3. Create a database called "test" and a table called "recipe". We used a component called "JDBC Explorer" which is part of the ENTERPRISE version of JBUILDER, not the FOUNDATION version of JBUILDER. However there are numerous free MYSQL or JDBC clients that you can use to manipulate your relational database. Here is one for MYSQL:

   http://www.mysql.com/downloads/gui-mysqlgui.html

   Install the client on a development system. It may be the same system. The columns we used were Day, Month, Timer1, Timer2 and Timer3. All were type INT in the SQL vernacular. You will need to define a user and a password, **and possibly need to tell the database from which IP ADDRESS this user is allowed to login from.**

4. Obtain a Java compiler

   You can use JIKES from IBM at http://www10.software.ibm.com/developerworks/opensource/jikes/ or the one in any JavaSoft Java Development Kit (JDK) from http://java.sun.com or you can download Borland's JBUILDER FOUNDATION which is a free Java development environment from http://www.borland.com.

   Jikes is the most direct path but it does not come with any API documentation. The JDK is the most authentic definition of a Java development platform with complete API documentation. JBUILDER is recommended only if you want to incur the additional learning curve of the JBUILDER IDE itself. It includes the JDK documentation and is a much larger download. JBUILDER makes sense if you have plans for Java beyond this Modlet. If you want to become a Java programmer use JBUILDER. If you just want a quick compile, download and use JIKES.

5. Obtain a type 4 JDBC driver (http://industry.java.sun.com/products/jdbc/drivers for you

database. You can most, but not all, JDBC drivers from this website. You want a type 4 driver as a first choice. Type 3 may also work. Type 1 and 2 will not work on SoftPLC.

We used MM MYSQL version 2.0.7. You have to un zip the distribution JAR and then rename the enclosed JAR has a long filename and is the driver itself. Rename this enclosed jar file to shorter name compatible with SoftPLC's 8.3 file system. E.g. JDBC_DVR.JAR

6. Obtain a JAR making tool such as ZIP.EXE or WINZIP.EXE. A JAR file is basically a ZIP file with a different file extension. You can download the public domain zip.exe and unzip.exe command line driven programs for Windows from:

   http://dl.softplc.com/pub/zip.exe
   http://dl.softplc.com/pub/unzip.exe

7. On a windows development machine, edit the JDBC.java file. There is a section in the source bracketed as <Configuration Parameters>. Change the items in this section to match your setup. Make sure the source file retains the filename JDBC.java (case and spelling are significant in this filename).
8. Compile the JDBC.java file to JDBC.class. For example:
   `C:>jikes -classpath .;RT.JAR; JDBC.java`
9. Zip the JDBC.class file to JDBC.zip or JDBC.jar. This is the "JAR FILE". For example:
   `C:>zip JDBC.JAR JDBC.class`
10. Place the JDBC.JAR file on SoftPLC using a 3rd party FTP client like FileZilla in the C:\JAR\MODLET directory.
11. Place the JDBC driver JAR file into the C:\JAR directory on SoftPLC's flashdisk.
12. Tell SoftPLC about the MODLET in the JDBC.JAR file and reboot SoftPLC. The JDBC's main classname (`JDBC`) should be added to your MODLET.LST file.

    And the JDBC driver and JDBC.JAR both should be added to your CLASSPATH= statement in your MODULE.LST file (all one line):

```
driver=machj.tlm classpath =
\jar\splc.jar;\jar\rt.jar;\jar\modlet\jdbc_tlm.jar;\jar\mm.mysql-2.0.7\mm.mysql-2.0.
```

13. Go online with TOPDOC and add the JDBC_READ instruction to the ladder program.
14. Figure out under what conditions the JDBC_READ instruction should be energized and add that supporting logic.

6. Browser Based HMI's

6.1. Overview

The SoftPLC Web Server Weblet, along with Java Servlets and/or Applets can be used to build an operator interface that runs in a web browser. You can view various example web pages that utilize these techniques by connecting to a SoftPLC on the internet, a link is provided at http://www.softplc.com/webserver.php.

SoftPLC Corporation provides some very useful Servlets that enable you to build live web pages using only HTML. We've done the Java programming for you, you need only to build your HTML pages and configure the Servlet data.

The remainder of this document describes how to use the Servlets provided with the SoftPLC Web Server. This chapter describes the Servlets and the next chapter provides example HTML code, graphics and more to demonstrate how easy it is to build your browser based HMI.

Applets require Java programming and are beyond the scope of this document. If you are interested in using Applets, contact the SoftPLC Corporation sales department for information on existing applet shells and system integrators who can develop applets for you.

6.1.1. HTML

What is HTML? H-T-M-L are initials that stand for HyperText Markup Language. It is a special kind of text document used by Web browsers to present text and graphics. The text includes markup tags such as <p> to indicate the start of a paragraph, and </p> to indicate the end of a paragraph. HTML documents are often referred to as "Web pages". The browser retrieves Web pages from Web servers, such as the one in SoftPLC.

If you don't know HTML, there are lots of good HTML tutorials, documents, manuals, etc. available for free on the internet. HTML shouldn't scare you, it is just text with tags. Another way to learn is to look at how other people have coded their html pages. To do this, click on the "View" menu and then on "Source". On some browsers, you instead need to click on the "File" menu and then on "View Source".

Many people still write HTML by hand using tools such as NotePad, or use their Word Processing program to save documents to HTML format. There are also many, many HTML WYSIWYG editors (What You See is What You Get) that make it much easier to ensure proper formatting and allow you to format your pages without having to know all the HTML markup tags.

One free, cross-platform WYSIWYG HTML editor is Mozilla Composer, which comes with Mozilla Suite. It is available at http://www.mozilla.org/products/mozilla1.x/.

6.2. SmartPage

Servlets are described above, and in detail at the Sun website Java tutorial listed above. SmartPage is a particular servlet that is quite useful for displaying live data from a SoftPLC. SmartPage works by filtering an HTML page looking for hyperlinks that are in a special format. The modified HTML page is created in memory on the fly and streamed to the browser with live process information in it.

The input HTML files that are processed by this servlet have the same name as the request URI, except the extensions are different. The request URI must end in ".stp", whereas the input files must be the same root filename but with an extension of ".htm". So for example, a URI of somefile.stp would invoke the SmartPage servlet and cause it to look for somefile.htm as its input for this request. You can have any number of input files, so the utility of this servlet is quite good.

Here is the sequence of events:

1. URI: somefile.stp comes into weblet.
2. weblet sees from SERVER.LST that URI's with extensions of ".stp" are to be sent to the SmartPage servlet.
3. SmartPage servlet is run, passing it the request URI somefile.stp.
4. SmartPage servlet swaps the trailing ".stp" with ".htm" and looks for somefile.htm and processes that file through its filter mechanism.

To embed dynamic information you put one of 3 types of special hyperlink URL macros into the input file: @DecimalFmt(), @TextChoice(), or @ImageChoice(). You simply add a hyperlink with your HTML editor and instead of keying in a URL to jump to, you key in one of the macros. The text or image that holds the hyperlink will not be shown, but rather it will be removed in favor of whatever the chosen macro produces.

Pay close attention to spaces within the macro specification, avoiding them unless specifically called for.

6.2.1. DecimalFmt

This macro substitutes the hyperlinked text or image with a numeric decimal string whose current value is determined by a SoftPLC data table word. This macro takes two parameters:

1. **address** is some data table word address such as I:0, B3:34, N23:456, T4:0.ACC, PD13:23.SP or F8:12. Bits, structures, strings and indirect addresses are not supported.
2. **format** specifies how the value is to appear, including whether the displayed value is to have leading zeros and/or a decimal point.

Examples:

`@DecimalFmt(PD13:0.SP,#.0)` will display as an integer the value of the Setpoint word of the structure PD13:0.
`@DecimalFmt(F8:0,#.000)` will display the value of F8:0 as a fixed decimal with 3 places after the decimal

A DecimalFormat comprises a pattern and a set of symbols that formats decimal numbers. It has a variety of features designed to make it possible to parse and format numbers in any locale, including support for Western, Arabic, and Indic digits. It also supports different kinds of numbers including integers (123), fixed-point numbers (123.4), scientific notation (1.23E4), percentages (12%), and currency amounts ($123). The underlying formatting mechanism used is the standard Java API DecimalFormat object, which is described at DecimalFormat.

Warning:

The above link is for the 1.4 version of DecimalFormat, and may have features that are not supported in SoftPLC. For documentation for the 1.1 version of DecimalFormat, you must download the enitre JDK 1.1 at http://java.sun.com/products/archive/jdk/1.1/.

The table below includes some commonly used examples. Refer to the previous link for a very detailed description. Basically, "#" means to place a number in this location, if it is available, and "0" means to place a number if it is available, otherwise place a zero. Other symbols like "%" and "$" can be used normally.

SoftPLC Value	DecimalFmt "format" parameter	Displayed result
12	#.##	12
12	#.00	12.00
12	0000	0012
12	#.00%	12.00%

Note:

It is useful to put placeholder text in your HTML page where the macro'ed text will appear. Then, apply your hyperlink to the placeholder text.

6.2.2. TextChoice

To dynamically display a certain word or words based on a value in the SoftPLC, TextChoice can be used. This macro takes **at least** 6 parameters.

- **address**: PLC Data Table address whose value will determine the text to be displayed.
- **bitmask**: hex value bitmask (given in 0xABCD format) that will be ANDed with the

word at **address** to yield a set of important bits. If all 16 bits are important, then use 0xFFFF.
- **defaultText**: default text to display when none of the other bit comparisons match.
- **compare1**: equality comparison bitmask. The algorithm is:
  ```
  if ( bitmask AND address ) = compare1 then Use text1
  ```
- **text1**: displayed on the HTML page if the above algorithm succeeds.
- [**comparen, textn**]: additional pairs of these may follow, as many as is needed.

For example, the tag below would show "on" when the value PD13:4.0 has the first bit set to 1, "off" if the bit is set to 0, and "error" on any other case.

```
<a href="@TextChoice(PD13:4.0,0x1,'error',0x1,'on',0x0,'off')"> </a>
```

Below are some examples with sample values in a PLC (in binary, hex and decimal), the bitmask, and the result if they were to be ANDed together. Following that is a TextCoice tag and the result if it were used with the given parameters and the mentioned values.

```
      Binary     Hex Dec
    8  4  2  1
    0  1  1  0   0x6  6  Value in PLC
AND 0  0  1  0   0x2  2  Bitmask
  = 0  0  1  0   0x2  2  Result
<a href="@TextChoice(N7:2,0x2,'off',0x2,'on')"> </a>  =  on

      Binary     Hex  Dec
    8  4  2  1
    0  1  1  1   0x7   7  Value in PLC
AND 1  1  0  1   0xD  13  Bitmask
  = 0  1  0  1   0x5   5  Result
<a href="@TextChoice(N7:2,0xD,'off',0x5,'on')"> </a>  =  on

      Binary     Hex  Dec
    8  4  2  1
    1  1  0  1   0xD  13  Value in PLC
AND 0  1  1  0   0x6   6  Bitmask
  = 0  1  0  0   0x4   4  Result
<a href="@TextChoice(N7:2,0x6,'off',0x3,'on')"> </a> = off
```

6.2.3. ImageChoice

This tag is the same as TextChoice except for there is no need for quote marks. For example, the tag below shows the image “/graphics/comfort.gif” when the value PD13:5.1 does not have 0x4 and 0x8 under the mask 0xC

```
<a
href="@ImageChoice(PD13:5.1,0xC,/graphics/comfort.gif,0x4,/graphics/hot.gif,0x8,/graphi
</a>
```

Below is a table that shows the possible resultant images based on certain values in the

SoftPLC.

```
          Bitmask:  1  1  0  0 (0xC)                        Example case
Possible Result 1:  0  1  0  0 (0x4) /graphics/hot.gif      when PD13:5.1
is 4 (0100)
Possible Result 2:  1  0  0  0 (0x8) /graphics/cold.gif     when PD13:5.1
is 8 (1000)
          Default:     N/A           /graphics/comfort.gif  when PD13:5.1
is 2 (0010)
```

6.3. SmartForm

SmartForm is a servlet that allows users to change values within their SoftPLC using only a browser. It uses standard HTML codes that are processed by the servlet. SmartForm determines which value is to be changed by the `name` parameter of the the HTML element called `form`. **It is important to know that you can only have one `input` element with a `name` parameter per `form` block.** Below are some examples of what is improper and proper:

Not allowed:

```
<form action="/servlet/SmartForm" method="get">
  <input type="text" name="N7:0" size="10">
  <input type="submit" name="submit_button">
</form>
```

Allowed:

```
<form action="/servlet/SmartForm" method="get" >
  <input type="text" name="N7:0" size="10">
  <input type="submit">
</form>
```

In the above example, the fact that the submit button had the `name="submit_button"` parameter would prevent the SmartForm servlet from working properly.

Not allowed:

```
<form action="/servlet/SmartForm" method="get">
    <select name="F10:0">
        <option value="1">Up</option>
        <option value="2">Down</option>
        <option value="4">Idle</option>
    </select>
    <input type="hidden" value="true" name="do_addition">
    <input type="submit" value="Change Value" name="submitter">
</form>
```

Allowed:

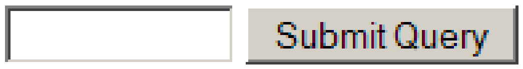
```
<form action="/servlet/SmartForm" method="get">
    <select name="F10:0">
        <option value="1">Up</option>
        <option value="2">Down</option>
        <option value="4">Idle</option>
    </select>
    <input type="submit" value="Change Value">
</form>
```

In the previous example, the 2 `name` tags in the hidden field and submit button are what is disallowed.

6.3.1. Text Field

To change a word or float value (ie: N7:0 or F10:0) in the SoftPLC to a user-specified value, simply use the HTML codes below in your webpage. The user would enter the desired new value and click the submit button or press enter.

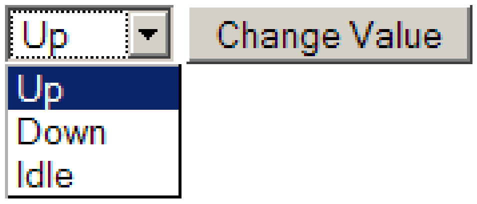
```
<form action="/servlet/SmartForm" method="get">
    <input type="text" name="N7:0" size="10">
    <input type="submit">
</form>
```



6.3.2. List Box

To change a word or float value (ie: N7:0 or F10:0) to a specific value using a listbox, simply add the HTML codes below to your webpage. You can add, remove, change, and/or reorder the `option` elements in order to have a different list of possible, seletable values.

```
<form action="/servlet/SmartForm" method="get">
    <select name="N7:0">
        <option value="1">Up</option>
        <option value="2">Down</option>
        <option value="4">Idle</option>
    </select>
    <input type="submit" value="Change Value">
</form>
```

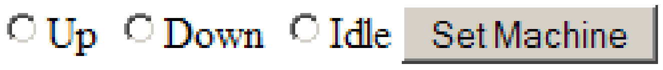


6.3.3. Radio List

To change a word or float value (ie: N7:0 or F10:0) to a specific value using a radio button list, simply add the HTML codes below to your webpage. You can add/remove/change the input tags for different selectable radio buttons. The name parameter is what is used to distinguish the value in the SoftPLC.

```
<form action="/servlet/SmartForm" method="get">
    <input name="N7:0" type="radio" value="1">Up
    <input name="N7:0" type="radio" value="2">Down
    <input name="N7:0" type="radio" value="4">Idle
    <input type="submit" value="Set Machine">
```

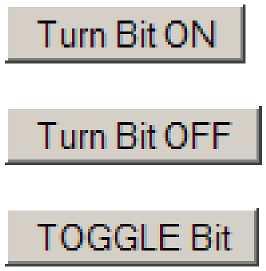
```
</form>
```



6.3.4. Push Buttons

To change a specific bit's state (ie: N7:0/0), you have 3 options. You can set the bit to ON or OFF (1 or 0) or you can toggle the bit. Please notice that there are 3 separate FORM blocks; this is because you can only have one named form element per form block.

```
<form action="/servlet/SmartForm" method="get">
    <input type="SUBMIT" name="N7:0/0,ON"     value="Turn Bit ON">
</form>
<form action="/servlet/SmartForm" method="get">
    <input type="SUBMIT" name="N7:0/0,OFF"    value="Turn Bit OFF">
</form>
<form action="/servlet/SmartForm" method="get">
    <input type="SUBMIT" name="N7:0/0,TOGGLE" value="TOGGLE Bit">
</form>
```



6.4. Security Servlet

SoftPLC's web server supports servlet based security, which allows you to password protect any file in any directory or any servlet. The security model is based on the HTTP protocol's built-in authentication support - called HTTP authentication. Basic HTTP authentication is structured on a simple challenge/response, username/password model. Usernames, passwords, and directories are secured by editing the web server's text based configuration files. Other servlet based security models can be implemented such as custom HTML form authentication via session management or cookies.

To change or add usernames and passwords to the security, edit the SERVLET.LST file (C:\LST\USR\SERVLET.LST) on the SoftPLC (the below code segment). **"duke" would be the username and "texas" would be its corresponding password.** (And correspondingly, the user "joeuser" would have a password of "mypassword".)

```
servlet.Authorization.code=Authorization
servlet.Authorization.initArgs=duke=texas,joeuser=mypassword
```

To enable or define directories to secure, edit the SERVER.LST file (C:\LST\USR\SERVER.LST) on the SoftPLC in this code segment:

```
ServletByRequest(GET|secure|*):Authorization
```

Replace "secure" with whatever directory you wish to require authentication.

7. HMI Examples

7.1. HMI Examples

7.1.1. Overview

Below are some code snippets to help you get started designing HMI screens. Source code that contains SmartPage or SmartForm tags will need to be changed to meet your application's specifications. Also, as mentioned on the SmartForm page, you can have only one value changed in the SoftPLC per submission. This means that each `form` block can have only one field with a `name` parameter. But you can easily get around this by having multiple forms, as shown in the examples. Some of the examples use a hyperlink in order to replace the submit buttons. For example, when there are multiple forms, they can be named f1 and f2. And then to submit the form with a hyperlink, use:

<a href="javascript:document.f1.submit();">Submit</a> for the f1 form, and

<a href="javascript:document.f2.submit();">Submit</a> for the form named f2.

Note:

If you are new to HTML, you can find the meanings of the tags used in these code snippets within any HTML reference. Or, put the code into your WYSIWYG editor to see a friendlier view of the code.

7.1.2. Date/Time

Below is an example of how to display a date/time combination using DecimalFmt.

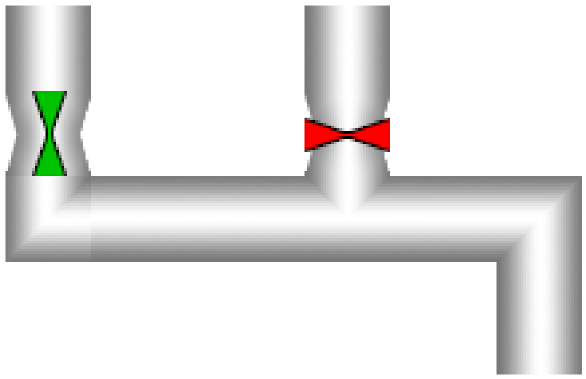
```
<!-- This would show the month/day/year. -->
<a href="@DecimalFmt(S19,#0)">0</a>/<a href="@DecimalFmt(S20,#0)">0</a>/<a
href="@DecimalFmt(S18,#0)">0</a>
<!-- This would show the hour:minute:second. -->
<a href="@DecimalFmt(S21,#00)">00</a>:<a
href="@DecimalFmt(S22,#00)">00</a>:<a href="@DecimalFmt(S23,#00)">00</a>
```

The displayed result would look something like what is shown below. (S19 is the month, S20 is the day, S18 is the year and S21 is the hour, S22 is the current minute, and S23 is the current second.)

11/30/2005 08:55:23

7.1.3. Basic Formatting

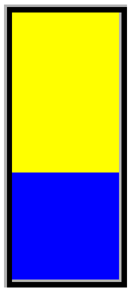
This example simply shows how through the use of HTML tables and our sample images, even HMI screens can be created. Of course, for static graphics, you can always create your own, or find them in graphic libraries.



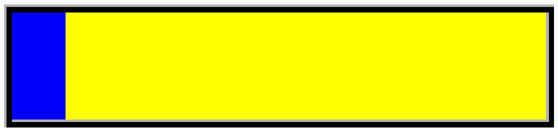
```
<table border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td><img src="pipe_vert.png" width="32" height="32"></td>
    <td> </td>
    <td><img src="pipe_vert.png" width="32" height="32"></td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td><img src="pipe_valve_open.png" width="32" height="32"></td>
    <td> </td>
    <td><img src="pipe_valve_closed.png" width="32" height="32"></td>
    <td> </td>
    <td> </td>
  </tr>
  <tr>
    <td><img src="pipe_2.png" width="32" height="32"></td>
    <td><img src="pipe_horiz.png" width="80" height="32"></td>
    <td><img src="pipe_c_2.png" width="32" height="32"></td>
    <td><img src="pipe_horiz.png" width="40" height="32"></td>
    <td><img src="pipe_4.png" width="32" height="32"></td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
    <td> </td>
    <td> </td>
    <td><img src="pipe_vert.png" width="32" height="42"></td>
  </tr>
</table>
```

7.1.4. Status Bars

Due to the power of HTML and JavaScript, you can easily develop bar graphs or other complex controls. In the examples below, N7:1 is a number between 0 and 100. The bar sizes will be based on the value of N7:1. (In the horizontal status bar example, the values are multiplied by 2 to make it look longer).



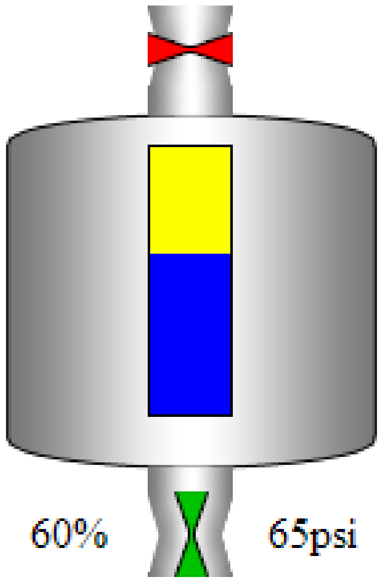
```
<table cellpadding=0 cellspacing=0 border=1 style="border: 3px ridge">
<tr><td><script language="JavaScript">
   // This line is the SmartPage macro. N7:1 is the value that is being
returned
   var h1 = <a href="@DecimalFmt(N7:1,#0)">0</a>;
   var h2 = 100 - h1;
   document.write('<img src="bar_yellow.png" width=40 height=' + h2 + '>');
   document.write('<br>');
   document.write('<img src="bar_blue.png" width=40 height=' + h1 + '>');
</script></td></tr>
</table>
```



```
<table cellpadding=0 cellspacing=0 border=1 style="border: 3px ridge">
<tr><td><script language="JavaScript">
   // This line is the SmartPage macro. N7:1 is the value that is being
returned
   var h1 = <a href="@DecimalFmt(N7:1,#0)">0</a> * 2;
   var h2 = 100*2 - h1;
   document.write('<img src="bar_yellow.png" width=' + h2 + ' height=40>');
   document.write('<br>');
   document.write('<img src="bar_blue.png" width=' + h1 + ' height=40>');
</script></td></tr>
</table>
```

7.1.5. Tank and Valves

This example shows usage of both ImageChoice and DecimalFormat tags to make a graphical display. (Again using only HTML tables and images).

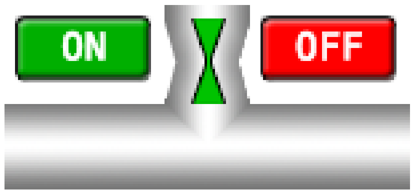


```
<table border="0" cellspacing="0" cellpadding="0">
  <tr>
    <td> </td>
    <!-- N7:3 is the word that will be examined -->
    <td><a
href="@ImageChoice(N7:3,0x1,pipe_valve_closed.png,0x1,pipe_valve_open.png">
</a></td>
    <td> </td>
  </tr>
  <tr>
    <td><img src="tank_01.png" width="59" height="150"></td>
    <td valign="top" background="tank_02.png"> <br>
      <table width="30" height="100" border="0"
            cellpadding="0" cellspacing="0" style="border: 1px solid">
        <tr>
          <td><script language="JavaScript">
   // The value of N7:2 will be inserted below by the SmartPage servlet
   var h1 = <a href="@DecimalFmt(N7:2,#0)">0</a>;
   var h2 = 100 - h1;
   document.write('<img src="bar_yellow.png" width=30 height=' + h2 + '>');
   document.write('<br>');
   document.write('<img src="bar_blue.png" width=30 height=' + h1 + '>');
</script></td>
        </tr>
      </table>
    </td>
    <td><img src="tank_03.png" width="59" height="150"></td>
  </tr>
  <tr>
    <!-- This next macro will insert N7:2. In this example, it is the %
```

```
full of
    the tank. -->
    <td align="center"><a href="@DecimalFmt(N7:2,#0)">0</a>%</td>
    <!-- N7:0 will be examined to determine if the valve is closed/open.
-->
    <td><a
href="@ImageChoice(N7:0,0x1,pipe_valve_closed.png,0x1,pipe_valve_open.png)">
</a></td>
    <!-- N7:1 is the value in the SoftPLC that represents the pressure of
the tank -->
    <td align="center"><a href="@DecimalFmt(N7:1,#0)">0</a>psi</td>
  </tr>
</table>
```

7.1.6. Bit States

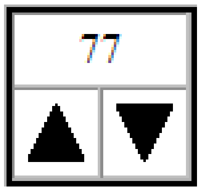
This example uses ImageChoice and SmartForm to allow you to change and display the state of the valve.



```
<table border="0" cellspacing="0" cellpadding="0">
<tr>
  <td><form ACTION="/servlet/SmartForm" NAME="f1" METHOD="GET" >
        <!-- These 2 lines will switch the N7:0/0 bit to ON -->
    <input type="hidden" name="N7:0/0,ON" value="N7:0/0,ON">
        <a href="javascript:document.f1.submit();"><img
            src="on.png" width="60" height="32" border="0"></a></form></td>
  <td valign="bottom" background="pipe_vert.png">
  <!-- Shows the status of N7:0 as an image -->
  <a
href="@ImageChoice(N7:0,0x1,pipe_valve_closed.png,0x1,pipe_valve_open.png)">
    </a></td>
  <td><form ACTION="/servlet/SmartForm" NAME="f2" METHOD="GET" >
        <!-- These 2 lines will switch the N7:0/0 bit to OFF -->
    <input type="hidden" name="N7:0/0,OFF" value="N7:0/0,OFF">
        <a href="javascript:document.f2.submit();"><img
            src="off.png" width="60" height="32"
border="0"></a></form></td>
</tr>
<tr>
  <td background="pipe_horiz.png"> </td>
  <td><img src="pipe_c_2.png" width="32" height="32"></td>
  <td background="pipe_horiz.png"> </td>
</tr>
</table>
```

7.1.7. Dial

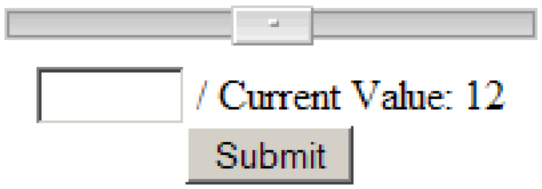
Here SmartPage is combined with SmartForm to make a dial.



```
<table cellpadding=0 cellspacing=0 border=1 style="border: 3px ridge">
<tr><td colspan=2 align="center"><a
href="@DecimalFmt(N7:1,#0)">0</a></td></tr>
<tr><td><script language="JavaScript">
   // This will insert the value of N7:1 so that the JavaScript can create
   // values that will allow the dials to increase/decrease it
   var cur = <a href="@DecimalFmt(N7:1,#0)">0</a>;
   var up = cur+1;
   var down = cur-1;
   document.write('<form ACTION="/servlet/SmartForm" NAME="f1" METHOD="GET"
>');
   // This is a little complicated, but it will have the current value of
N7:1 + 1
   // and submitting the form will thus "increment" the N7:1 word
   document.write('<input TYPE="hidden" NAME="N7:1" VALUE="' + up + '">');
   document.write('<a href="javascript:document.f1.submit();">');
   document.write('<img src="up.png" border=0></a>');
   document.write('</form>');
   document.write('</td><td>');
   document.write('<form ACTION="/servlet/SmartForm" NAME="f2" METHOD="GET"
>');
   // This is a little complicated, but it will have the current value of
N7:1 - 1
   // and submitting the form will thus "decrease" the N7:1 word
   document.write('<input TYPE="hidden" NAME="N7:1" VALUE="' + down +
'">');
   document.write('<a href="javascript:document.f2.submit();">');
   document.write('<img src="down.png" border=0></a>');
   document.write('</form>');
</script></td></tr>
</table>
```

7.1.8. Slider

This example is a little more complex, but it allows you to have a slider component. First, the following needs to be within the HEAD tag of your HTML document. (These files can be downloaded here).



```
<link rel="stylesheet" type="text/css" href="slider.css">
<script type="text/javascript" src="addanevent.js"></script>
<script type="text/javascript" src="slider.js"></script>
<script type="text/javascript" src="slider-setup.js"></script>
```

As the DIV and INPUT tags can be added into the BODY section of the document, you can lay them out however you wish. Because these are standard HTML tags, you can lay them out as standard HTML components!

```
<form name="f1" method="GET" action="/servlet/SmartForm">
<div class="slider" id="slider01">
        <div class="left"></div>
        <div class="right"></div>
        <img src="knob.png" width="31" height="15" >
</div>
<input id="output1" name="N7:3" size="5">
   / Current Value: <a href="@DecimalFmt(N7:3,#0)">0</a>
<br><a href="javascript:document.f1.submit();">Submit</a>
</form>
```

If you want additional sliders per page, then you simply copy the above source onto the page, but use a different id for both the DIV and INPUT (for example: `id="slider02"` and `id="output2"`). Additionally, you would need to change the FORM's NAME parameter and the submission link to match it (ie: from `name="f1"` to `name="f2"` and from `document.f1.submit()` to `document.f2.submit()`). Afterwards, you would need to open the **slider-setup.js** file and add additional lines to match the configuration of slider[1]. You would be adding another stanza to the file for slider[2]. For instance:

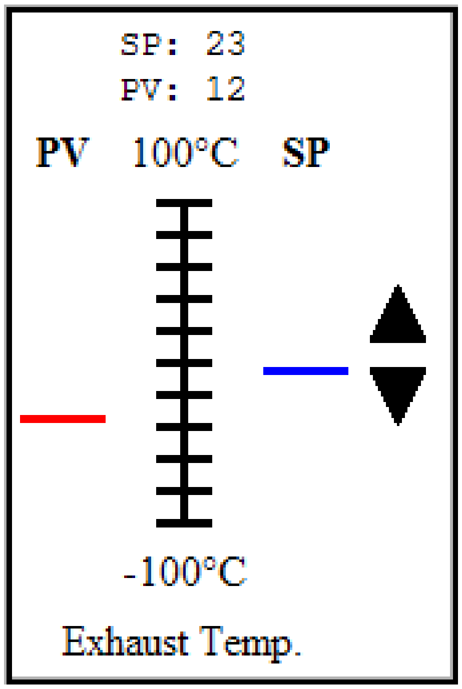
```
slider[2]=new Object();
slider[2].min=-100;
slider[2].max=100;
slider[2].val=100/3;
slider[2].onchange=setBoxValue;
```

And one final possible configuration is to change the width of the sliders. Open **slider.css** and change line 4 to patch your desired dimension.

And of course, all of the images, CSS and JavaScript files must be in the same directory.

7.1.9. PID Faceplate

Below is another complex example, a PID faceplate. It combines the use of status bars and the dial.

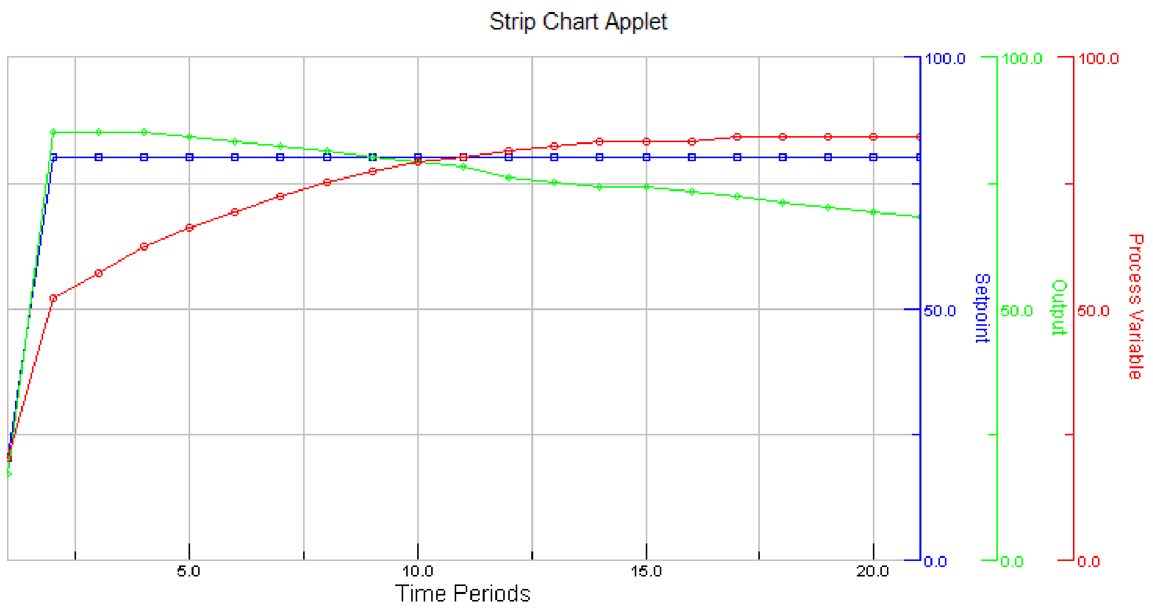


```
<table border="0" cellspacing="0" cellpadding="3" style="border: 3px
ridge">
<tr align="center">
  <td colspan="3"><tt>
    <!-- These next 2 lines will show the values of the
    SP/PV values -->
    SP: <a href="@DecimalFmt(PD13:5.1,#0)">0</a><br>
    PV: <a href="@DecimalFmt(PD13:5.2,#0)">0</a>
    </tt></td>
  <td></td>
</tr>
<tr align="center">
  <th>SP</th>
  <td>100°C</td>
  <th>PV</th>
</tr>
<tr>
  <td align="right" valign="top"><script language="JavaScript">
       // The PV value
       var h1 = 100 - <a href="@DecimalFmt(PD13:5.2,#0)">0</a>;
       document.write('<img src="blank.png" width=32 height=' + h1 + '>');
    </script><br>
    <img src="bar_red.png" width="32" height="3"> </td>
  <td align="center" valign="top"><img src="ticks.png" width="27"
```

```
height="125"></td>
  <td align="left" valign="top"><script language="JavaScript">
       // The SP value
       var h1 = 100 - <a href="@DecimalFmt(PD13:5.1,#0)">0</a>;
       document.write('<img src="blank.png" width=32 height=' + h1 + '>');
    </script><br>
    <img src="bar_blue.png" width="32" height="3"> </td>
  <td><form ACTION="/servlet/SmartForm" METHOD="GET" name="f1">
        <script language="JavaScript">
      // The SP value, again
      var cur = <a href="@DecimalFmt(PD13:5.1,#0)">0</a>;
      var up = cur+1;
      var down = cur-1;
      document.write('<input TYPE="hidden" NAME="PD13:5.1" VALUE="' + up
+'">');</script>
      <a href="document.f1.submit();"><img src="up.png" border="0"></a>
    </form>
    <form ACTION="/servlet/SmartForm" METHOD="GET" name="f2">
    <script language="JavaScript">
      // The SP value, again!
      var cur = <a href="@DecimalFmt(PD13:5.1,#0)">0</a>;
      var up = cur+1;
      var down = cur-1;
      document.write('<input TYPE="hidden" NAME="PD13:5.1" VALUE="' + down
+'">');</script>
      <a href="document.f2.submit();"><img src="down.png" border="0"></a>
    </form></td>
</tr>
<tr>
  <td valign="top"> </td>
  <td valign="top">-100°C</td>
  <td valign="top"> </td>
  <td> </td>
</tr>
<tr>
  <td colspan="3" align="center">Exhaust Temp.</td>
  <td> </td>
</tr>
</table>
```

7.1.10. Trend Graph

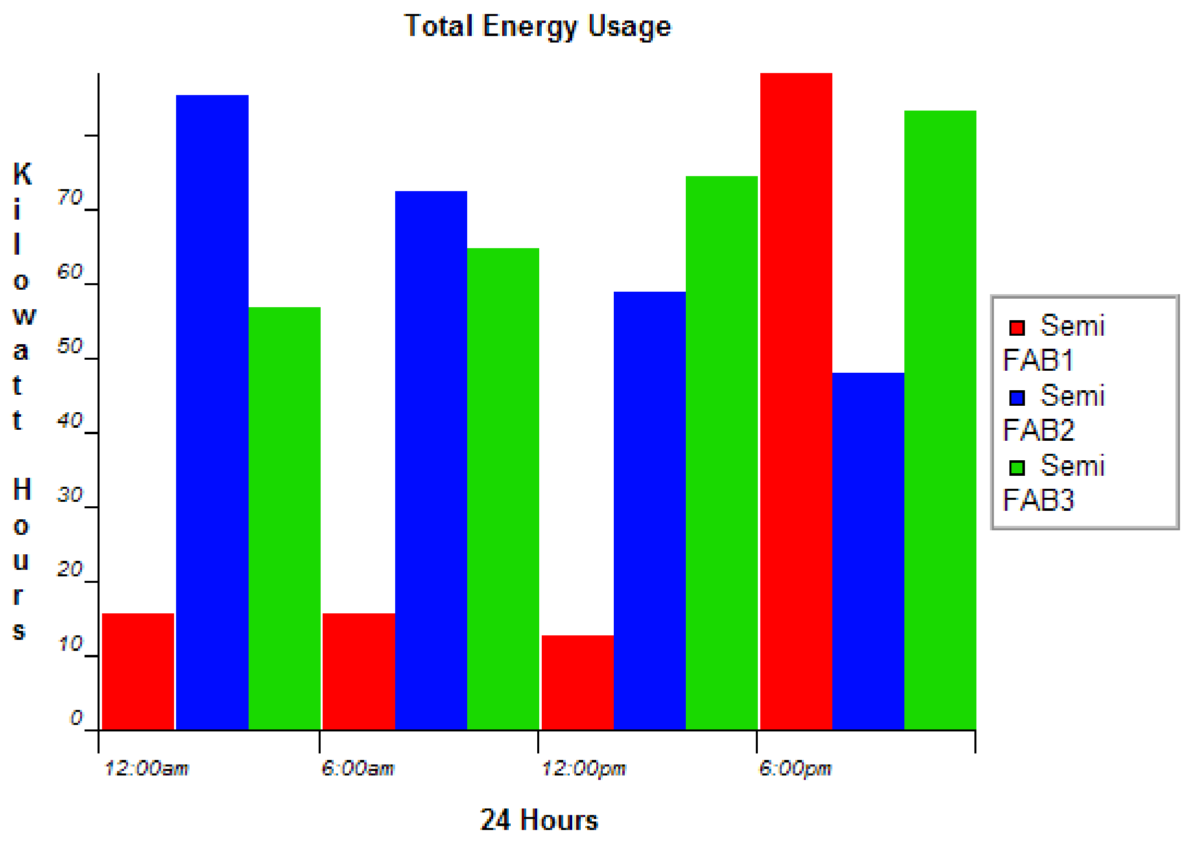
You can download a a trend graph that updates based on the value in the SoftPLC here. Changing the `address` parameter in the `applet` tag inside the **INDEX.HTM** file would alter the graphed variable. This table is based on Leigh Brookshaw's open source Java 2D Graph Package. Below is an example display of the graph:



7.1.11. Bar Chart

Creating a bar chart through JavaScript and SmartPage is simple as well! The files required for this chart can be downloaded here.

In the **chart.html** file, there is a `<script>` section, which is where the configuration is made to change the appearance of the chart.


Total Energy Usage
Kilowatt Hours
0
10
20
30
40
50
60
70
12:00am
6:00am
12:00pm
6:00pm
24 Hours
🟥 Semi FAB1
🟦 Semi FAB2
🟩 Semi FAB3

8. All