The gateway−cgi program

Location

The `gateway−cgi` program may be found in the /usr/lib/cgi−bin directory. However, due to directory aliasing by the HTTP server, it should be referenced as if it were in the /cgi−bin directory from web pages on the EMAC Sever In a Box (SIB). Thus, if your SIB can be found at SIB.yourdomain.com, the gateway−cgi program may be referenced as "http://SIB.yourdomain.com/cgi−bin/gateway−cgi".

Use

The `gateway−cgi` program is designed to act as a WWW proxy for embedded devices which cannot speak internet protocols themselves. The `gateway−cgi` program is can receive an arbitrary data string from a browser, send that data string through a communications port of the SIB, then get a reply and return that reply to a browser. The `gateway−cgi` program should be referenced by the POST method. The POST method of CGI will send a data string to the `gateway−cgi` program in the form:

"field1=data1&field2=data2&field3=data3...".

The string "data1" will be sent to the device identified by "field1". "field1" is specified in a configuration file, described in this document shortly. The program will then wait for a reply for a specified time, and return any received data back to the browser by merging the device reply string(s) with a particular HTML file and returning that merged file to the browser.

In case of program failure, the `gateway−cgi` program is equipped with a self−monitoring feature, called "Time to Live". As soon as the program is executed, it creates a second ("child")process. While the `gateway−cgi` program does its communications jobs, the child / watchdog process waits in the background for a specified length of time. If the main program terminates successfully before the child times out, the watchdog process is also terminated. However, it is possible that the `gateway−cgi` program will block , attempting to wait indefinitely (this usually occurs when the program tries to open a serial port which does not exist.) When this happens, the watchdog process will time out, then terminate the blocked gateway program before exiting. Thus, the `gateway−cgi` program cannot exist for longer than the time specified to the watchdog process.

By default, this timeout value is sixty seconds. The timeout value may also be configured by the a user, when the default is not appropriate. This configuration is explained later. However, it is important that the timeout value be kept long enough such that only a blocked gateway program will be terminated − each time a program is terminated unexpectedly, communications ports can be left open and / or in unknown states. This can cause subsequent communications to unexpectedly fail, though this is uncommon.

The `gateway−cgi` program has the ability to collect data in asynchronous mode, as well as the standard polled mode. Asynchronous mode must be used to receive data from devices which

might send data at any time, rather than only sending data when polled. This functionality is accomplished by a second program, `asyncd`. The `asyncd` program is started at boot time, and runs continuously until the SIB is shut down. `Asyncd` reads the same configuration file as `gateway-cgi`. For each port identified as asynchronous, `asyncd` constantly monitors that port, saving all incoming data. When `gateway-cgi` is run, data is sent from the SIB to the device as usual, but instead of getting data back directly from the device, `gateway-cgi` will read one packet of data from `asyncd`.

It should be noted that `asyncd` separates incoming data into packets based on the timeouts listed in the configuration file. Once a character has been received, `asyncd` will place that and all subsequent characters into the same packet, until a character fails to arrive within the specified inter-character timeout. When the next character does arrive, it will be placed into a new packet. Note that this will not overwrite any previous packets; all incoming data is stored until it is read by `gateway-cgi`, up to the maximum memory of the machine (however, the size limits of `gateway-cgi` as listed at the end of this section remain the same when reading data from `asyncd`). It is also possible to get a packet of data from a device via `asyncd` without actually sending any data out to the device; simply send out a field / data pair with an empty data field.

Configuration

The gateway-cgi program is configured via a file called "cgi.config", located in the same directory as the `gateway-cgi` program (/usr/lib/cgi-bin). As the syntax and grammar of the configuration file are best understood by example, an example cgi.config file has been included in the appropriate directory of the SIB. The example file is extensively commented, and should be straightforward to understand. It is recommended that this example file be renamed and kept on the SIB, in case reconfiguration is required later.

Syntax: The file consists of several configuration and comment lines, separated by carriage returns (CRs). Comments are full-line only. A comment line is marked by a '#' character as the first non-whitespace character in a particular line. Configuration lines consist of an identifier field and several port configuration fields, separated by any combination and number of separator characters. Separator characters are space, tab, and comma.

Grammar: Each configuration line begins with an identifier. The identifier string is free-form, can be any length within the line length limit listed at the end of this section, and may contain any characters other than the separator characters and the CR character. The identifier is how a particular port will be referenced in a POST query. When a POST query arrives with a "field" equal to an identifier, then the port configuration data associated with the identifier will be used in sending out the "data" string associated with "field".

The second item in a configuration line is the synchronous / asynchronous identifier. This determines whether a devices operates in continuous or polled mode, as described previously. Legal values are "AS" and "SN", specifying asynchronous and synchronous modes, respectively.

The third item in a configuration line is the ASCII or binary selector. Legal values are ‘A’ and ‘B’(caps required), for ASCII and binary respectively. If ASCII is selected, then the data portion of the command will be passed unmodified to the target port. Make sure your character sets match when using this option! If binary is chosen, then the data string will be treated as a list of 2−digit hexadecimal values, which will be converted to binary before transmission. Each binary byte *must* be represented by two characters in the data string (the data string must have an even number of characters). If binary is chosen, then the only legal characters for the data string are 0−9, a−f, and A−F.

For example, the string FF2718034b is a legal string for binary format; the bytes 0xFF, 0x27, 0x18, 0x03, and 0x4B will be transmitted. The string FF271834b is illegal because it has an uneven number of characters; the zero placeholder is required. The string 00285SIB is also illegal, because the characters ’S’ and ’I’ are not valid hexadecimal digits.

If ASCII is chosen, then a reply string from a device will be placed directly into the master reply string (discussed later). If binary is chosen, then a reply from a device will be converted to an ASCII string, using two hexadecimal digits to represent each value, exactly reversing the conversion performed on the outgoing data string (the digits A−F will be capitalized).

The third item in a configuration line is the protocol. The `gateway−cgi` program currently recognizes three protocols: TCP, UDP, and RAW. If TCP or UDP is specified, then the data string will be sent via the Internet−standard TCP/IP or UDP/IP protocols. The SIB operating system will determine how IP packets should reach their destination. As such, IP networking must be properly configured before these protocols will work. RAW is usually specified for devices which do not speak any standard protocol, connected directly to the serial ports of the SIB

The format for the rest of the line depends on the protocol specified. In a line where *TCP* or *UDP* has been specified, the next field is the destination IP address for the data string. This may be a fully qualified host name (e.g. www.emacinc.com), or a numeric address (10.0.2.5). Note that if you specify a host name, your SIB must have some way in which to resolve the name into an IP address (either a DNS server or an entry in the static hosts table).

Two fields containing the port numbers follows the IP address field. The first port number field contains the IP port number on the SIB that will be used to communicate with the device. This is generally only used for asynchronous devices. If a device is to be synchronous, then this port should be listed as 0, which means any available port.

The second port number field should contain the TCP or UDP port to which the data string should be sent on the destination machine. Note that port numbers below 1024 are generally reserved for standard Internet applications.

The final field for an IP protocol is an optional timeout value. This is the time in seconds that the `gateway−cgi` program will wait for data from the remote device before returning a timeout. If this field is omitted, a default timeout of one second will be used.

If the *RAW* protocol is specified after the identifier, then the data string will be sent out exactly as it was received (though CGI to ASCII translation will be performed), and a device must

be specified in the field following the protocol (instead of an IP address). Any device (in fact, any file) may be specified. Note that these devices are referenced by their UNIX−style names (e.g. /dev/ttyS0). These device names are case−sensitive, and the "/dev/" preceeding the device name is required. Common DOS equivalents are given below. Remember that the `gateway−cgi` program will need read, write, and execute permission on this device or file; this has been pre−configured for the standard serial devices. The specified device will retain whatever speed and data format with which it was configured on system boot−up.

Common DOS equivalents for UNIX devices:

DOS	UNIX
COM1	/dev/ttyS0
COM2	/dev/ttyS1
COM3	/dev/ttyS2
COM4	/dev/ttyS3
LPT1	/dev/lp1

The next field for the RAW protocol is an optional timeout value. This is the time in seconds that the `gateway−cgi` program will wait for the first character to arrive from the remote device. If no value is specified, a default value of 1 second will be used.

Following this is another optional timeout field, the inter−character timeout value. The `gateway−cgi` program will wait this many microseconds for the next character before deciding that no more data is coming and returning. Note that this means that a maximum possible delay of (reply timeout + (IC timeout* reply length in characters) + 1) is possible. If no timeout value is specified, a default 10 microseconds will be used.

The final field for the RAW protocol is an optional multidrop control. Note that if this field is specified, both timeout values must also have been specified. The multidrop field is designed to allow the use of RS−485 and 9−bit / multidrop protocols. This field, if specified, must always have the value "M" (caps required).

If multidrop is specified for a particular identifier, then all data sent to that identifier will be transmitted in a manner consistent with RS−485 master/slave multidrop networks. When the data string is to be transmitted, the RS−485 transmitter will first be activated. Next, MARK (force 1) parity is set, and *only the first byte* of the data string is transmitted. The port is then set to transmit SPACE (force 0) parity, and the rest of the data string is transmitted. The RS−485 transmitter is then disabled, and `gateway−cgi` waits for a response in the usual manner. At this time, a transmitter shutoff time of 200 milliseconds is required due to operating system and hardware constraints.

Note that the SIB can only be set up to transmit MARK and SPACE parity, not receive them. The ninth bit of any characters received in reply to a data string with MARK or SPACE parity will be ignored, and all received characters will be returned to the browser. Also note that with the standard SIB hardware, RS−485 is only available on ttyS1 (COM2). Make certain that your RS−485 network is connected to the correct hardware connector before using this option. You may also need to change a jumper on the SBC board, and / or use a different serial connector to use RS−422 or RS−485; see the SBC hardware manual for details.

Time to Live: The `gateway−cgi` program allows the user to configure the maximum time that the program will be allowed to run before it will be terminated ("TimeToLive"). This value is sixty seconds by default, acceptable for most applications. However, there may be some instances when a user will need to wait longer for returning data. Conversely, the user may be loading the SIB heavily, and need potential runaway processes terminated more quickly.

The time to live, if present, must be specified on the first line of the cgi.config file. No comments or configuration lines may precede the time to live. The time to live is not specified as a standard configuration line. To specify the time, the identifier "TimeToLive" must be the first text on this line (whitespace is ignored). Capitalization must match exactly. Following this identifier, the next (non−whitespace) item on the line should be the desired time to live, in seconds. All other text on this first line is ignored.

Additional lines in the configuration file beginning with "TimeToLive" will be treated as standard configuration lines, generally producing parse errors (unless they are properly formed configuration lines; "TimeToLive" is a valid identifier for a configuration line.) If "TimeToLive" is not specified as the first text of the first line of the configuration file, then the first line of the cgi.config file may be any comment or configuration line, and the default value of 60 seconds will be used as the time to live.

When your cgi.config file is ready: To test your cgi.config file without a browser, log in as user **www** and make sure that the file is in the correct location (/usr/lib/cgi−bin/, the same directory as the `gateway−cgi` program). Next, you will need to run your file manually, providing it with the input it expects. You will need to use a command line of the form:

```
echo −n <field1>=<data1>\&<field2>=<data2> | ./gateway−cgi
```

Replace the <fieldx> and <datax> with identifiers and data appropriate to your devices. You may specify as many field / data pairs as you like; just separate them with the "\&" pair of characters (and no spaces).

Before you run the program, however, you will need to set an environment variable with the length of the string containing the field / data pairs. Use the command "export CONTENT_LENGTH=<length>", replacing <length> with the length of your field / data pair string, *excluding the '\' characters preceding each '&' character.*

You may then enter the command line listed above. This will run the `gateway−cgi`

program, giving it the input "<field1>=<data1>&<field2>=<data2>". The `gateway-cgi` program will parse the input file, then perform the specified communications. The data that would be returned to a browser will be printed on the screen. A sample cgireply.html file has been supplied for testing purposes. If there are parse errors or unexpected communication errors, fix them and use the test command again until the program functions as expected.

To test the configuration file from a remote browser, create a test HTML page referencing the `gateway-cgi` function (via the POST method) with an appropriate data string. You may also use the included example code. Then simply load the page and click the link. Since all `gateway-cgi` error messages are prefixed with an HTTP "Content/type:" header, any parse or run-time errors should appear in the browser.

Getting data back to the browser

The `gateway-cgi` program forms a reply to the browser by first forming a device reply string, then merging this string with an HTML file and returning the merged file to the browser. The device reply string is similar in form to the POST query string, except it uses commas instead of ampersand characters. Thus, the format would be "field1=replydata1,field2=replydata2…fieldN=replydataN". The "field1" through "fieldN" are the identifiers sent to the gateway-cgi program by the browser. All such fields sent by the browser will be represented in the device reply string, whether or not any data was received from the device. The fields "replydata1" through "replydataN" are the data strings received back from the ports / devices identified by "fieldx", after the incoming data was sent out to the device(s). The individual reply data strings may be inserted into the master reply string verbatim, or they may be converted from binary to ASCII, as described previously.

Once the device reply string has been created, it will be merged with a reply file. This file is found in the HTML root directory (/home/www, which is accessed as the root directory "/" when forming HTTP requests.) This is also the default home directory of the user “www". The file name is “cgireply.html". This file should be a normal HTML file in most respects, and it may use any and all HTML features your browser can support, including java and / or javascript. This file must have one special feature, however: you must tell the gateway-cgi program where to insert the device reply string. The `gateway-cgi` program searches the cgireply.html file for the sequence of characters "-$-" (dash, dollar sign, dash). Each time it encounters this set of characters, the `gateway-cgi` program will replace them with the device reply string. Note that these characters must not be separated by a space or any other character(s). No quotes, spaces, newline characters or carriage returns will be added during the substitution. The cgireply.html file may contain multiple "-$-" sequences, and the device reply string will be substituted for each instance. An example cgireply.html file has been included on the SIB. It is common for the cgireply.html file to contain a call to a java or javascript function, passing the device reply string as an argument to that function.

When designing devices for use with the SIB, keep in mind that a browser will interpret the ’<’ and ’>’ characters as parts of HTML tags, and not as regular text. Unless this is desired, it is recommended that these characters not be returned by devices when possible.

Important Limits

Due to the unknown lengths of user data, certain limits had to be imposed on the lengths of data string passing through the `gateway-cgi` program. In all cases, excess length will be ignored or truncated, and will not affect the stable operation of the program.

Limit 1: A single line of the configuration file "cgi.config" should not be longer than 500 characters before the CR character. Excess length will generate a parse error and the program will exit.

Limit 2: The total length of the string sent to the `gateway-cgi` program by the browser via the POST method should not exceed 500 characters in length (this is approximately six lines of text on a standard page.) Excess data will be ignored. Note that the browser will expand many punctuation characters to escape-character notation, replacing a single punctuation character with a '%' character and the punctuation character's ASCII representation (total of three characters including '%'). This can unexpectedly increase the length of your POST query string.

Limit 3: No individual reply data string sent by your device back to the `gateway-cgi` program should exceed 500 characters in length. Excess length will be truncated.

Limit 4: The total device reply string should not exceed 500 characters in length. This reply will include a field identifier, an '=' character, the returned data string, and a ',' separator character for each device/port the gateway-cgi program communicated with. Remember that binary device replies will be doubled in length when converted to ASCII. Excess length will be truncated, possibly returning incomplete and/or badly formed identifier and/or data fields.

Gateway−cgi errors

If no data could be received for a given "field" identifier, the reply data may be an error message instead of received data. The possible error fields are:

UNKNOWN IDENTIFIER: The identifier "field" in the "field=data" pair could not be found in the configuration file. Check the spelling of "field", use a different "field," or add the entry to the configuration file.

UNABLE TO OPEN COM PORT xxxx: A system error occurred trying to open the specified device. Check the name of the device in the configuration file, and make sure that all users have read and write permission on that device.

FAILED HOST LOOKUP: The specified host name could not be resolved into a numeric IP address. Check the spelling of the hostname in the configuration file, add a static host entry, or add an appropriate nameserver.

FAILED SOCKET OPEN: The system call to open a socket failed. Check the networking settings, and make sure that no blocked or "zombie" processes are running in the background holding network resources.

FAILED TO CONNECT: The system call "connect" failed. For TCP sockets, this could be because the specified host did not respond, or because the network configuration is in error. For UDP sockets, this is only due to SIB network configuration errors.

SEND ERROR: An error occurred while attempting to send data to the specified port. Make sure your data is of reasonable length, and that valid devices and ports are specified in your POST query and configuration file.

READ FAILED: An error occurred while attempting to read data from the specified device. Make sure the device is returning data of reasonable length and proper format, that a proper device is specified, and that all file permissions are correct.

TIMEOUT: A reply from the device was not received within the specified timeout. Make sure your SIB network configuration is correct, and check that your device is sending data back to the SIB. This is also returned if a device is specified as asynchronous, and asyncd has no data for the device.

FAILED PIPE OPEN: The `gateway−cgi` program was unable to open a pipe to read data from `asyncd`. Make sure none of the pipe files in /var/pipes have been modified. Make sure asyncd is running; a manual restart may be necessary.

Revisions

Rev 1:

Cut from SIB 1.0 manual, minor editing.