

Control Solutions Babel Buster 3 User HTML Web Server CGI Overview (3/2021)

This document is an updated version of the user_html_cgi.pdf document found on the introduction page under User HTML Tutorials at <https://info.csimn.com>. This document has been updated per implementation in Babel Buster 3 gateways. Except for changing the link “../index.html” to “/html/index.html” where applicable, the various demos found in the knowledgebase should work in a Babel Buster 3.

The “naked pages” referenced in the 2008 version of the CGI overview are not implemented in Babel Buster 3. They were an attempt at allowing some level of “private labeling” the gateway. If private labeling is desired, Control Solutions does offer that option which will result in fully rebranding the entire preprogrammed web UI.

User HTML

User HTML may be installed as a “wrapper” around the default web pages. To install user HTML, use the File Manager (filter set to *.*) to upload any combination of .html, .txt, .gif, and .jpg files. You can also include .pdf, .xml, .css and JavaScript .js files. PHP, ASP, etc., are not supported.

You can also use FTP to upload files to the /FLASH0 (that’s flash zero) directory. However, do not use smart clients like FileZilla - they try to be too helpful and simply screw things up. Use only a plain dumb ftp session opened from the command prompt in Windows.

The top level user file must be named User.html (case sensitive). If this file is present in /FLASH0, it will be served instead of the default index.html page pre-programmed in the device. Once this page is open, it may link to any other html files in the /FLASH0 directory. All user HTML is filtered as it is served to provide dynamic content.

There are a handful of tricks that must be observed to make user html work. All references to other user pages and to user image files must have the file names preceded with /UE/ as in /UE/User2.html or /UE/mypicture.gif. The UE stands for “user escape” and is treated as a virtual directory that actually points to /FLASH0.

Dynamic Data – Creating a Form

Dynamic access to register data is provided. Dynamic updates of register contents is also supported via the form post method. The form must be defined using the following tags:

```
<form id="UserForm" action="/UE/icanForm" method="post"
name="UserForm">
```

The submit button causing the post must be defined as:

```
<input type="submit" name="submitChange" value="Change">
```

Any other submit buttons will simply result in a page refresh. Only the submit button named “submitChange” with a value of “Change” will result in parsing of the form data. Only a form with action as named above will be parsed.

If you want to redefine the appearance of the button, you can implement a graphic button by including an image as follows, and then including the JavaScript function as shown:

```

```

```
<script type="text/javascript"><!--  
function sendMeAway() {  
    document.UserForm.submitChange.value="Change";  
    document.UserForm.submit();  
}  
//--></script>
```

Input Types: <input type="text">

Two types of data input are recognized by CGI processing of the user post: Text input and option select. The search string keyed upon for text is **<input type="text">** and the search string keyed upon for the select option is **<select name=**

A text input should be constructed as follows:

```
<input type="text" name="reg22" value="%d" readonly size="8">
```

The contents of the register number included in the name (“regX”) will be displayed when the page is served, and the data will be taken dynamically from the register at that point in time, and again each time the page is refreshed. The data will be formatted using the C format string found in the value tag. Integer formats (%d, %04d, %x, etc) should be used for integer registers, and floating point (%f, %.2f, etc) should be used for floating point registers. If “readonly” is specified, data will only be displayed in this window. Otherwise the data returned by the post will be parsed, and the result placed back into the register.

The following keywords are recognized as text input “names”:

- regX – references the value in local register number X
- namX – references the name of register number X
- site – references the SNMP System Name

All of these data elements may be read, and will be written unless you specify “readonly”. The definition of read means take data from the local register when serving the page, and write means write data to the local register if the form was submitted by the appropriately named submit button (see Form above).

Input Types: <select>

An option select should be constructed as follows:

```
<select name="reg25" size="1">
  <option selected value="0">OFF</option>
  <option value="1">ON</option>
</select>
```

The strings corresponding to the values given will be displayed when the register named matches that value, otherwise “---“ will be displayed. When an option is selected and the form posted, the value corresponding to the new selection will be written back into the register. The “selected” tag shown above is not required since it is automatically inserted in the appropriate place (moved around) when the page is served.

Input Types: <input type="hidden">

An additional form of input has been added to filtered HTML. Hidden variables may be defined using the following syntax:

```
<input type="hidden" name="reg22" value="%d" readonly>
```

This will be processed the same as “text” input except the value is not displayed. This is useful as a means of providing non-displayed data to a JavaScript function. Hidden data upon return will be parsed and put back into registers unless readonly is specified. Omit readonly if hidden data should be parsed. This provides a means for JavaScript to get data back into registers.

Page Links

To create a link on the user page to get into the default preprogrammed pages, define a link to “/html/index.html”, for example:

```
<a href="/html/index.html">Log In</a>
```

To link to another user page in the FLASH0 directory, use a link such as:

```
<a href="/UE/pwUserP3.html">Room #1</a>
```

Links to graphic images you want shown on the page are created in similar fashion:

```

```

Note that you preface the page name with “/UE/” when the file is located in the FLASH0 directory, but preface the name with “/html/” when accessing a preprogrammed page.

Password Protection

There are 3 levels of password protection: Restricted, Maintenance, Administrator (root is a special form of administrator). User pages may be password protected at the “Restricted” level. To password protect a user page, simply insert the letters “pw” in front of the name. Therefore, if *User2.html* is a page you wish to protect, rename it *pwUser2.html*.

The top level page for User HTML must still be named *User.html* (and not *user.html* or *USER.html*). If you don’t want anything useful to be completely unrestricted, simply put a plain dumb page in *User.html* with a link that says “log in” and link it to *pwUser.html*.

Note that “restricted” level of password protection means the user can access any “pw” user pages, but cannot access any pages beyond index in the pre-programmed page set.

Other Input Types

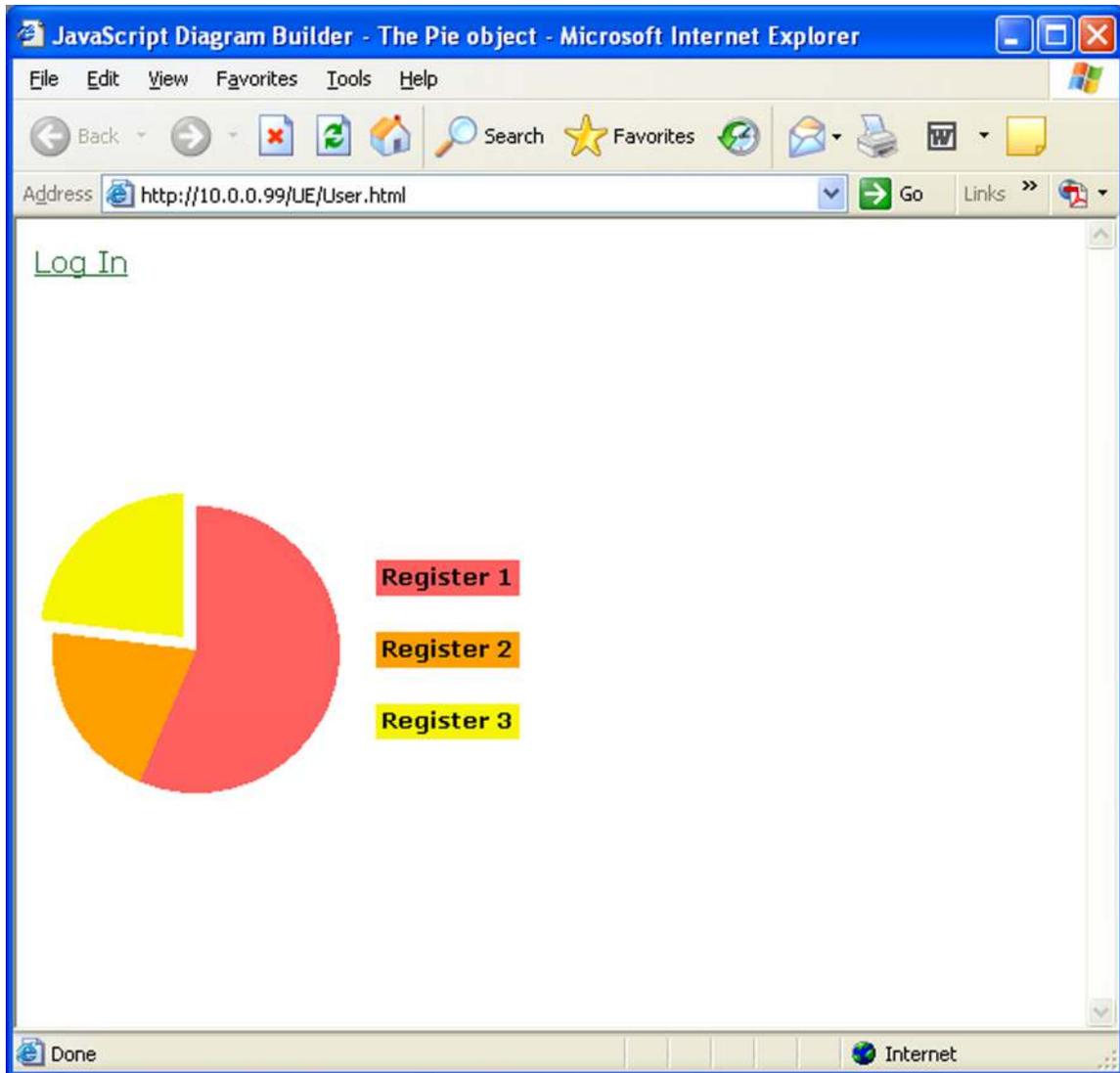
Radio buttons and other forms of input are not supported at this time. The HTML will be passed through, but not filtered and associated with register data. Therefore, you can use radio buttons, etc., with JavaScript, but you must explicitly associate the resulting data with hidden input variables in order to return the data to registers.

JavaScript

An external JavaScript library can be specified. The JS file should be loaded into the /FLASH0 directory along with *User.html*, etc. In the HTML file, the script file is referenced as (using *diagram.js* as an example):

```
<SCRIPT Language="JavaScript" src="/FS/FLASH0/diagram.js"></SCRIPT>
```

The following illustration is an example of a JavaScript library being used to generate a pie chart based on the contents of registers. This example even includes a mouse-over that pulls out a pie slice when the mouse is over its respective label. The proportional size of each slice will change based on the contents of registers 1 through 3 as viewed from the System->Data->Local Registers page.



The freeware JavaScript library `diagram.js` can be found by simply doing a Google search for "diagram.js". Many types of graphics are supported by this library.

The library example was modified as shown in the `User.html` page whose source code is given below:

```

<html>
<head>
<title>JavaScript Diagram Builder - The Pie object</title>
<SCRIPT Language="JavaScript" src="/FS/FLASH0/diagram.js"></SCRIPT>
</head>
<body link="#226c30" vlink="#009900" alink="lime">
  <form method="post" name="myForm">
    <input type="hidden" name="reg1" value="%d" readonly>
    <input type="hidden" name="reg2" value="%d" readonly>
    <input type="hidden" name="reg3" value="%d" readonly>
  </form>
  <script language="JavaScript"><!--
P=new Array();
document.open();
var ireg1 = parseInt(document.myForm.reg1.value);
var ireg2 = parseInt(document.myForm.reg2.value);
var ireg3 = parseInt(document.myForm.reg3.value);
var total = ireg1 + ireg2 + ireg3;
if (total == 0) {
  total = 3; ireg1 = 1; ireg2 = 1; ireg3 = 1;
}
var pct1 = ireg1*100/total;
var pct2 = ireg2*100/total + pct1;
var pct3 = ireg3*100/total + pct2;
P[0]=new Pie(100,240,10,80,0*3.6,pct1*3.6,"#ff6060");
P[1]=new Pie(100,240,0,80,pct1*3.6,pct2*3.6,"#ffa000");
P[2]=new Pie(100,240,0,80,pct2*3.6,pct3*3.6,"#f6f600");
new Bar(200,190,280,210,"#ff6060","Register 1","#000000","","void(0)","MouseOver(0)","MouseOut(0)");
new Bar(200,230,280,250,"#ffa000","Register 2","#000000","","void(0)","MouseOver(1)","MouseOut(1)");
new Bar(200,270,280,290,"#f6f600","Register 3","#000000","","void(0)","MouseOver(2)","MouseOut(2)");
document.close();
function MouseOver(i) { P[i].MoveTo("", "",10); }
function MouseOut(i) { P[i].MoveTo("", "",0); }
-->
</script>
  <p><font face="Verdana, Arial, Helvetica, sans-serif"><a href="../index.html">Log In</a></font>
  </p>
</body>
</html>

```

Advanced JavaScript using EJSCharts

A free version of the Emprise JavaScript Charts library is now available at www.ejschart.com. This is a JavaScript library that supports more advanced graphical objects like live gauges as illustrated below. This is definitely not for the novice, but if you are up to the challenge, the combination of hidden variables, automatically refreshing iframes, and this library, can create a completely animated dash board.

The following example is a screen shot of a live page being served by a Babel Buster BB3-6101-V3SP. (Yes, the gauges move in real time.)

