eWON Family

500, 2001, 2001CD, 4001, 4001CD, 2101, 2101CD, 4101, 4101CD,
4002, 4102, 2104, 4104, 2005, 2005CD, 4005, 4005CD

Web Reference Guide

ver 1.1

COOL INTERNET TELECONTROL SOLUTION

1 User defined Web Site

1.1 Introduction

The eWON can host a web site containing user define web pages. The hosting management can be done like for common web site using the eWON FTP server.

When connecting to the eWON FTP server, the root directory contains a /usr directory. The pages from the custom Web site can be located in that directory or in subdirectories of /usr.

A file **/usr/index.htm** will be accessed using **http://10.0.0.53/usr/index.htm** (If 10.0.0.53 is the eWON IP address)

Static web pages stored in the eWON have a limited interest. That is why the eWON provides 3 ways to make its web content dynamic:
- **Build /usr files using BASIC at regular intervals or upon context change.**
- **Use the SSI (server side include) syntax.**
- **Use bASP (Basic Active Server Page).**

The SSI Stands for Server Side Include; this method is used for creating web sites in which the server updates parts of the HTML page before sending the page to the client.

- **The eWON uses that technique to allow dynamic update of the pages. The following type of data can be dynamically replaced by the eWON in the user page:**
- **Tag Value.**
- **Creation of HTML table for different types of data (historical, event, etc.).**
- **Creation of Graphs containing Real time values or Historical values.**
- **Script expression.**
- **…**

The bASP consists in putting BASIC program block in your web pages saved on the server, when the page is delivered to the WEB client. The BASIC block is executed. The BASIC block can access parameters passed to the page and it can output HTML content directly to the page delivered to the client.

The eWON also provides 3 types of FORMS to perform the following actions in user-defined pages:
- **Modify Tag values**
- **Acknowledge alarm**
- **Execute script action**

1.2 SSI Syntax

1.2.1 HTML Page extension

The normal processing for an SSI page is the following:
- **The user requests a page from the server by entering the page address, example: http://myewon/usr/index.shtm (there are many other ways to request a page of course).**
- **The server (eWON) checks that the given page is available in the file system.**
- **The server checks the page extension (here .shtm), if this extension is NOT .shtm the page is transferred as-is to the client. This applies to .htm files for example, but also .jpg, .gif, etc.**
- **If the page's extension is .shtm, then server will parse the page and replace all the special HTML Tags (see below) by there current value.**

So the SSI extension for the pages in the eWON is **.shtm** (case sensitive). This extension is recognized by most HTML editors.

1.2.2 Special eWON SSI Tags

There are 4 special HTML Tags defined to provide SSI functionality in the eWON:

```
<%#TagSSI,YYYYY%>

<%#ParamSSI,XXXXXX%>

<%#VarSSI,YYYYY%>

<%#ExeSSI,XXXXXX%>
```

-

These 4 special HTML Tags follow a generic syntax for SSI Tags, i.e. they start by **<%** and end by **%>**. The consequence of this is that HTML editor will not check the syntax inside these Tag and they will be completely ignored by the editors.

1.2.2.1   TagSSI HTML Tag

The TagSSI is used to insert the current value of a Tag into the page:

`<%#TagSSI,TagName%>`

`TagName is the name of the Tag that must be inserted.`

Notes:

- **Do not insert any space in the TagSSI:**
- **TagSSI is case sensistive:**

`<%#TagSSI,TagName%>`

Below is the example of an user webpage named "TEMP_page.shtm" which displays the current value from the Tag "TEMP":":

```
<html>
  <body>
  <p>
    Last Value of TEMP: <%#TagSSI,TEMP%>>
  </p>
  </body>
</html>
```

`<%#tagssi,TagName%>`

1.2.2.2 ParamSSI HTML Tag

The ParamSSI is used to insert the content of an export block into the page (See also chapter "Export Block Descriptor" in the General User Guide).

```
<%#ParamSSI,Parameter%>
```

"Parameter" follows the "Export Block Descriptor" syntax described in this manual. The resulting output is directly inserted into the HTML page returned. This format is useful to return HTML Tables into the page or return Script expression ($dtSE) into the page.

With this method, the exported data is inserted directly into the page. If what you need is a hyperlink to an exported data, the **ParamForm** described in chapter "Produce an export data block" on page 13 should be used:

```
2        <body>
3                <%#ParamSSI,[$dtRL$ftH$st_m0$et_m0$tnTank_Level]%>
4                <img border="0" src="/rcgi.bin/ParamForm?AST_Param=$dtRL$ftG$tnTank_Level">
5        </body>
6 </html>
```

TimeInt	TimeStr	Value
1116597012	20/05/2005 13:50:12	30.000000
1116597017	20/05/2005 13:50:17	30.000000
1116597022	20/05/2005 13:50:22	10.000000
1116597027	20/05/2005 13:50:27	8.000000
1116597032	20/05/2005 13:50:32	8.000000
1116597037	20/05/2005 13:50:37	20.000000
1116597042	20/05/2005 13:50:42	20.000000
1116597047	20/05/2005 13:50:47	50.000000
1116597052	20/05/2005 13:50:52	50.000000
1116597057	20/05/2005 13:50:57	150.000000
1116597062	20/05/2005 13:51:02	150.000000
1116597067	20/05/2005 13:51:07	3.000000
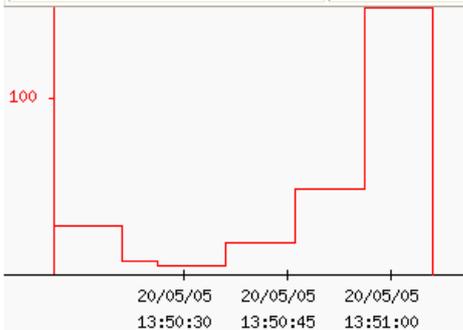
1.2.2.3 Web context indirection

With the ParamSSI and Web context variables, you can display the content of an Export Bloc Descriptor inline a web page (shtm).
See "Web context indirection" on page 10

1.2.3   VarSSI HTML Tag

See:
"Web context variable directly inserted with SSI" on page 8"

1.2.4   ExeSSI HTML Tag

See:
> *"Building page content by using bASP" on page 6"*

1.3   bASP Syntax

The bASP allows to insert BASIC blocks in the WEB page in order to build the page on the fly while the page is transmitted to the client's web browser.

The big advantages of this technique are:

- **It can generate pure HTML, compatible with any WEB client (not even javascript enabled).**
- **The BASIC code is inserted directly in the WEB page, so it can be created at the same time the rest of the page layout is designed.**
- **It is very flexible.**

1.3.1   Building page content by using bASP

The page created will contain special Tags with BASIC block embedded.

Example:

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Time:<%#ExeSSI,PRINT #0,TIME$%><BR>
  Array of data:<BR>
  <%#ExeSSI,
  for i%=1 to 3
     print #0,"TData(";i%;") ";a(i%)
  next i%
  %>
</p>
</body>
</html>
```

In this example the page actually generated by the eWON will be:

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Time:05/06/2002 14:09:57<BR><BR>
  Array of data:<BR>
TData(1) 10<BR>
TData(2) 20<BR>
TData(3) 30<BR>
</p>
</body>
</html>
```

If a(1)=10, a(2)=20, a(3)=30.

The following remarks can be made about this example:

- **The syntax for the bASP inclusion is:**

```
<%#ExeSSI,Basic_block%>
```

   (Basic block can span multiple lines)
- **PRINT #0,xxx is an extended syntax of the PRINT command that will route printed data to the client WEB page. As for the common PRINT command, if the sequence of data to print does not ends with a ";", a <BR> is added to the data to force a new line.**

- **The execution of the blocks occurs as the page is sent to the user. The blocks nearer from the top of page are executed first.**
- **The user will never see the Basic Block, he will only see the content of its execution.**

SYNTAX:

The bASP syntax is:

```
<%#ExeSSI,Basic_block%>
```

When the block contains multiple BASIC lines, the leading space are not trimmed, they are part of the line submitted to the BASIC interpreter and should be avoided.

1.3.2   WEB context variables

When the client submits a WEB request to the eWON web server:

```
GET /usr/index.shtm
```

Then the web server will create a context to send a response to the client. This context will live until whole data in the index.shtm page have been sent to the client. This context can be used by the BASIC bASP to save temporary data and the access the FORMS and REQUEST parameters.

SYNTAX:

The context variable in basic are ending with a "**!**", a context variable must always be in **lowercase**. The context variables are always STRING variables (like A$).

1.3.2.1   Web context variables for request parameters

WEB servers support the syntax where client passes parameters to the server in the URL. In this syntax, each parameter is separated by a **&**, the first parameter is separated from the file URL by a **?** and each parameter has the format **param_name=param_value** (param_name in lowercase).

`Example:`

```
http://10.0.0.53/usr/index.shtm?param1=1234&param2=ABCD
```

Requests the index.shtm page with:

```
param1=1234
param2=ABCD
```

The context variables **param1** and **param2** will automatically be created. This means that any bASP program block in the index.shtm page will have access to the 2 basic variables **param1!** and **param2!**

`Example:`

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Param1:<%#ExeSSI,PRINT #0,param1!%><BR>
  Param2:<%#ExeSSI,PRINT #0,param2!%><BR>
</p>
</body>
</html>
```

In this example we print the 2 parameters to the WEB client. The resulting output would be (for the request above):

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Param1:1234<BR><BR>
  Param2:ABCD<BR><BR>
</p>
</body>
</html>
```

1.3.2.2 Web context variable directly inserted with SSI

There is another syntax for inserting Web context variables in the client web page. A special SSI Tag called VarSSI can be used:

`<%#VarSSI,variable_name%>`

`<%#VarSSI,variable_name,default_value%>`

Where variable_name is the Web context variable (without the ending "**!**").
In the second syntax, you can have a default value (usefull in case of Form parameters).

The previous example can be written differently like follows:

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Param1:<%#VarSSI,param1%><BR>
  Param1:<%#VarSSI,param2%><BR>
</p>
</body>
</html>
```

The output is exactly the same, yet there is a strong performance issue between these two implementations:
In the first implementation using ExeSSI, we have 2 bASP blocks that are posted to the eWON basic queue. It means that the current execution of the program will have to be interrupted twice; the blocks must be inserted in the BASIC program and executed before the output is available to the client.

In the second implementation using VarSSI, the Web context variable are directly read from the context, there is no intervention of the eWON BASIC in that operation. It is much faster.

1.3.2.3 Improving WEB performance using Web context variables and bASP

The example above introduces how the WEB performance can be improved.
Suppose to following example:

```
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Tag Name:<%#ExeSSI,
SETSYS TAG,"load","Tag1"
PRINT #0,GETSYS TAG,"Name"%>
  Tag Desc:<%#ExeSSI,
SETSYS TAG,"load","Tag1"
PRINT #0,GETSYS TAG,"Description"%>
</p>
</body>
</html>
```

In this example, we print the config Name and Description fields for a Tag called "Tag1". There are 2 bASP blocks and we call SETSYS TAG,"load" twice because at the end of the first bASP block there may be a context switch with another BASIC request loading the TAG edition area with another Tag value (depending of the rest of the application, but let's take the worst case).
These 2 SETSYS TAG,"load" take some time and we have 2 BASIC context switch to generate the page.
The following implementation returns the same result with a significantly improved performance:

```
<%#ExeSSI,
  SETSYS TAG,"load","Tag1"
  tag1_name!= GETSYS TAG,"Name"
  tag1_desc!= GETSYS TAG,"Description"
%>
<html>
    <head>
        <title>SSI Demo</title>
    </head>
    <body>
<p>
  Tag Name:<%#VarSSI,tag1_name%><BR>
  Tag Desc:<%#VarSSI,tag1_desc%><BR>
</p>
</body>
</html>
```

In this case Tag1 is loaded only once in the edition area and we generate only one BASIC context switch to execute the only bASP block.

1.3.2.4 Using Web context variables with FORMS

The Web context variables provide the easiest way to use FORM fields when the FORM is posted.
As described below, there is a special form call ExeScriptForm that allows to request execution of a script command.

Example:

```
<form method="POST" action="/rcgi.bin/ExeScriptForm">
<input type="hidden" name="Command" value="goto UseForm">
</form>
```

• When the form is posted, the "goto UseForm" request is posted in the BASIC queue.
• If other fields are added in this form, there content can be accessed using Web context variables.

Example:

```
<form method="POST" action="/rcgi.bin/ExeScriptForm">
<input type="text" name="edit1" size="20">
<input type="hidden" name="Command" value="goto UseForm">
</form>
```

And the BASIC contains the following code at UseForm label:

```
UseForm:
   REM save Edit1 parameter
   A$ = edit1!
   PRINT "Edit1 value entered was:";A$
   END
```

Warning:

All Web context variables must be lowercase.

1.3.2.5   Web context indirection

The Web context variables can be used to display the content of an Export Bloc Descriptor inline a web page (shtm).

The syntax is the following:

`<%#ParamSSI,+WebParam%>`

Where "WebParam" is the Web context parameter which is passed to the shtm form.

`example:`

Place the "<%#ParamSSI,param1%>" in a page /usr/Page1.shtm on your eWON.

```
<HTML>
<BODY>
Start of Export Bloc<BR>
<%#ParamSSI,+Param1%>
<BR>End of Export Bloc
</BODY>
</HTML>
```

If you type the following request in your Internet Browser:

`http://10.0.0.53/usr/Page1.shtm?Param1=$dtEV$ftH`

will generate an output like this:

Start of Export Bloc

EventTimeInt	EventTimeStr	EventStr	ThreadStr	ThreadId	Event
1205940477	19/03/2008 15:27:57	System Booting, FWR: EW_5_3s6 (5.3), SN: 0508-0004-88 [EF0000]	elog	79300	-22602
1205940477	19/03/2008 15:27:57	Reboot reason: Script Request	wd	79308	1073762140
1205940525	19/03/2008 15:28:45	emodem-Improperly inserted card	ppp	79311	27207
1205941195	19/03/2008 15:39:55	bakfile-Could not rename file to backup	http	79307	28002

1206113080	21/03/2008 15:24:40	eftp-Close FTP session (User: adm)	ftps	79310	1073763130
1206114359	21/03/2008 15:45:59	eftp-Open FTP session (User: Adm)	ftps	79310	1073763129
1206114364	21/03/2008 15:46:04	eftp-Close FTP session (User: adm)	ftps	79310	1073763130
1206114476	21/03/2008 15:47:56	exp-The data type is invalid	http	79307	23601

End of Export Bloc

1.4   Special FORMS

Some special forms are handled by the eWON to allow:

- **Update of eWON Tag values**
- **Hyperlink to Graph, Text or HTML table**
- **Execute Script action**
- **Acknowledge Alarm**

1.4.1   Update Tag Value (and acknowledge)

The purpose of this form is to:

- **Update the value of a Tag.**
- **Acknowledge a Tag alarm.**

```
Syntax
Form name: UpdateTagForm
Form fields name:
```
        TagName
        TagValue
        ResultPageOk

Or:

        TagName1
        TagValue1
        TagName2
        TagValue2
        TagNameN
        TagValueN
        ResultPageOk

The first field name syntax can be used to update only 1 Tag per form, while the second syntax can be used if it is required to update more than one Tag at the time. With the second syntax, the number of Tags that may be updated with one FORM is not limited. The eWON will check items with an increment of 1 until not found.

> **Note:**
> If TagName1, TagName2 and TagName4 are defined, only TagName1 and TagName2 will be updated because TagName3 is missing.

TagName	Is a form field that will define the name of the Tag to update. Usually it will be a hidden field.
TagValue	Is a field that will hold the new value of the Tag. This may be a list box or a text edit field or any other type, and its initial value may be filled with a TagSSI (see examples), or maybe with a TagSSI and Java Script in case of list box.
ResultPageOk	Is a field that is also usually hidden. This field is **optional** and defines the page to show when the update has been performed.

Table 1: Tags to update fields

WARNING:

The "ResultPageOk" URL must be specified from the eWON root: i.e.: /usr/xxxx

1.4.1.1 Examples

• **Single Tag update**

```
<form method="POST" action="/rcgi.binrcgi.bin/UpdateTagForm">
<input type="hidden" name="TagName" value="Pressure">
<p>Pressure:
<input type="text" name="TagValue" size="20" value="">
</p>
</form>
```

This example shows how to update one Tag called "Pressure". There is one hidden form field for the Tag name and its value is "Pressure" which is the name of the Tag to update.

There is another form field called TagValue, it is a text edit field where the user can enter the new value of the Tag. When the form is shown, the initial value of the TagValue field is empty.

• **Single Tag update with initial value**

```
<form method="POST" action="/rcgi.bin/UpdateTagForm">
<input type="hidden" name="TagName" value="Pressure">
<p>Pressure:
<input type="text" name="TagValue" size="20" value="<%#TagSSI,Pressure%>">
</p>
</form>
```

The only difference between this example and the previous one is the initial value of the TagValue form field. In this case a TagSSI as been placed in the "value" attribute of the text field. The eWON will replace this placeholder with the current value of the pressure field when the page is displayed.

• **Single Tag update with no-default result page**

```
<form method="POST" action="/rcgi.bin/UpdateTagForm">
<input type="hidden" name="TagName" value="Pressure">>
<p>Pressure:
<input type="text" name="TagValue" size="20" value="">
</p>
<input type="hidden" name="ResultPageOk" value="/usr/x.shtm">
</form>
```

Again, this is the same example as the first one, here an additional field has be inserted to define which page to show when the FORM update has been executed correctly. The additional field is hidden, its name is `ResultPageOk` and its value is the page to display in case of update success.

• **Multiple Tags update**

```
<form method="POST" action="/rcgi.bin/UpdateTagForm">
<input type="hidden" name="TagName1" value="Pressure">
<p>Pressure:
<input type="text" name="TagValue1" size="20" value="">
</p>
<input type="hidden" name="TagName2" value="Speed">
<p>Pressure:
<input type="text" name="TagValue2" size="20" value="">
</p>
</form>
```

This example shows the syntax for updating more than one Tag at the time. For each TagNameX there must be a corresponding TagValueX. The first field must have index 1, then the next must follow with an increment of 1.

1.4.1.2 Acknowledge Tags alarms

The same FORM can be used to acknowledge a Tag alarm; the syntax is exactly the same as for Tag update, the only difference if for the content of the **TagValue**.

For Tag alarm acknowledgement, the TagValue field will contain the keyword "**ack**" optionally followed by the "**,UserName**" who will be logged in alarm history.

Example:

```
<input type="hidden" name="TagValue" value="ack,adm">
```

This example shows a hidden field used to request the acknowledgement of the Tag by the Admin user ("adm"):

```
<input type="hidden" name="TagValue" value="ack">
```

Would yield to the same result because the default user is the Administrator (if none is specified for acknowledgement).

1.4.2 Produce an export data block

Using a <%#ParamSSI,xxx>, it is possible to insert an export data in a page. But sometimes what is needed is a hyperlink to an exported block, for example:
- **Hyperlink to the event file**
- **Hyperlink to a Real time graph picture (will appear as a picture in the page).**

Inserting a picture in a page is not possible with the <%#ParamSSI,xxx>, because the binary content of the picture would be included in the page. A picture in a page is an hyperlink to the file of that picture. Instead of pointing to a file, we can point to a FORM that provides the picture data.

The syntax for the export form hyperlink source is the following:

SourceURL

"/rcgi.bin/ParamForm?AST_Param=XXXXXXXXX"

(ParamForm is case sensitive).

Where XXXXXXXXX is the Export Block Descriptor (Please refer to chapter "Export Block Descriptor" in the General User Guide). The URL source can be used anywhere an URL is required, the MIME type of the data returned will match the actual type: PNG, HTML or PLAIN TEXT.

Examples:

- **Real time graph hyperlink**

```
<img src="/rcgi.bin/ParamForm?AST_Param=$$dtRL$ftG$tnPressure"><br>
```

- **Hyperlink to a text file containing real time data**

```
<a href="/rcgi.bin/ParamForm?AST_Param=$$dtRL$tnPressure$st_s20$ftT" > Upload Text file</a><br>
```

1.4.3   Execute an eWON script

The purpose of this form is to execute an eWON script command line by posting a FORM from the web server. Using this feature, you can have a button in the Web page that starts execution of a script section.
The form allows executing a sequence of one or more script commands. Each command is executed as if it was typed and executed from the *Script Control* window. If more than one command is specified, they are executed in sequence.
The length of the command line is limited to 250 chars.

1.4.3.1   Syntax

Form name: ExeScriptForm (case sensitive)

Form fields name

> Command
> ResultPageOk

Or:

> Command1
> Command2
> ...
> CommandN
> ResultPageOk

If there is only one script command to execute, the first syntax can be used. If more than one command must be issued, the second syntax applies. The second syntax requires that the CommandX fields starts with Command1 and goes on by increment of 1.

> **Note:**
>
> *If Command1, Command2 and Command4 are defined, only Command1 and Command2 will be executed because Command3 is missing.*

ResultPageOk is a field that is also usually hidden. This field is **optional** and defines the page to show when the update has been performed.

> **WARNING:**
>
> **The page URL must be specified from the eWON root: i.e.: /usr/xxxx**

As for execution of command directly from the *Script Window*, and when the program is running, the command is executed between 2 sections executions. This means that the command will NEVER be executed between 2 lines of instructions inside a script.
For example, the commands will be executed between 2 executions of a cyclic section, or after the complete execution of an OnTimer section. In other words, the commands are always executed after a script END command of a section being executed.

When the FORM is posted, the result page returned by the eWON web server is not sent after execution of the command. The command is actually posted for execution but there is a queue of commands that can be more or less filled, and a section execution may be in progress also. This means that if your result page contains fields that should be updated by your command, it is normal that these fields are not yet updated at the time the response page is sent by the eWON web server.

Any script command can be issued with this form, including Goto commands (Do not use Gosub because the rest of the line after Gosub is not executed).

1.4.3.2   Examples
• Single command execution:

```
<form method="POST" action="/rcgi.bin/ExeScriptForm">
<input type="hidden" name="Command" value="a$='close':MyTag@=1">
</form>
```

Note:

Please note the use of the (a$='close') single quote instead of the common a$="close" syntax that would conflict with the HTML quotes.

• Single command execution with no-default result page

```
<form method="POST" action="/rcgi.bin/ExeScriptForm">
<input type="hidden" name="Command" value="a$='close':MyTag@=1">
<input type="hidden" name="ResultPageOk" value="usr/x.shtm">
</form>
```

Compared to the previous example, the page displayed after execution is the user page x.shtm.

• Multiple commands execution:

```
<form method="POST" action="/rcgi.bin/ExeScriptForm">
<input type="hidden" name="Command1" value="a$='close'">
<input type="hidden" name="Command2" value="MyTag@=1">
</form>
```

In this example the same commands are executed as in the first example. One of the main differences is that in the first example, the 2 commands (a$='close' and MyTag@=1) are executed together (no instruction can be inserted between these 2 instructions). In the second case, another command or another section may be executed between the 2 commands.