***eis system***

*didactic SmartHome electrical simulator*

**How to write a new device**

*upf, 2013-06-09, ver. 1.0*

1. **Preliminary operations**

After the successful installation of the ***eis*** system on the machine, go to the ***eis*** main directory (the same shown in the *$eis\_conf[“path”]* variable inside the *eis\_conf.php* file). Then copy the folder “*eis.device\_template.A01*” in the “*system*” directory to the “*devices*” directory renaming it in accordance with the ***eis*** system requirements:

*device\_brand\_name****.****device\_model\_name****.****device\_serial\_string*

The 3 fields can contain only alphanumeric and “\_” characters, separated by “.” character (example: *saturn****.****wash\_and\_dry****.****02B34598*). Finally move to the newly created directory (your device).

1. **Configure your device**

Edit the “*device\_conf.php*” file in the “private” folder, which contains the definition of the associative array *$eis\_dev\_conf* storing all the specific configuration parameters of this device. Change the contents of the following indexes according to your needs:

* ***version***  
  the version of the device implementation as *version.subversion.release* (string)
* ***date***  
  the date of the device implementation release (string)
* ***author***  
  the name of the device implementation author(s) (string)
* ***ifport***  
  the number (integer) of the UDP port used by the HTML realtime interface (see after). Must be different for each device installed in this host.
* ***class***  
  the class ID (string) which this device belongs to. Recognized class IDs are listed in another specific document.
* ***description***a short description (string) of this device
* ***type***the type ID (string) which this device belongs to. Recognized types are:   
   ***load*** a device that only consumes electrical energy   
   ***generator*** a device that only generates electrical energy   
   ***load&gen*** a device that can consume and generate electrical energy

To understand the other indexes, it is important to remember the electrical model of the ***eis*** simulator:

grid interface

storage

unprotected line

protected line

gen

load

gen

load

grid

When the “*grid interface*” is installed, the simulator will simulate a grid-connected system where the electrical energy can be purchased from or sold to the public grid (at the current prices). If the “*protected line*” is active, an electrical storage system must be installed. In this case a grid-connected system with storage will be simulated.

When the “*grid interface*” is not installed, the simulator will simulate an off-grid system with electrical storage. In this case no loads are connected to the “*unprotected line*” which is used only to connect an auxiliary generator (usually a diesel generator).

Following this scheme, the device must be configured for its consumption parameters and/or generation parameters.

***Consumption parameters*** (only for device type “***load***” or “***load&gen***”):

* ***cline***the electrical line which this device (as load) is connected to. Recognized types are: ***unprotected*** or ***protected***
* ***cpower1, cpower2, cpower3***the max electrical power (integer) in watts that the device can consume on a specific electrical phase (1, 2 or 3). Set a value to zero if the corresponding phase is not connected (e.g. a 1-phase device will have 2 of 3 values set to zero).

***Generation parameters*** (only for device type “***generator***” or “***load&gen***”):

* ***gline***the electrical line which this device (as generator) is connected to. Recognized types are: ***unprotected*** or ***protected***
* ***gpower1, gpower2, gpower3***the max electrical power (integer) in watts that the device can generate on a specific electrical phase (1, 2 or 3). Set a value to zero if the corresponding phase is not connected (e.g. a 1-phase device will have 2 of 3 values set to zero).

1. **Configure your device status variables**

In the ***eis*** system, devices interact using HTTP calls. Therefore the device code is executed each time a call is made by another device or an HMI, this means that all the contained variables will be re-initialized at each execution if not explicitly saved at the end of a call and loaded at its beginning.

For this reason the ***eis*** system implements a built-in “status” mechanism, which allows having persistent variables, i.e. variables that maintain their values from a call to another. Such variables need also to have known (default) values to be assumed when the device is initialized.

To declare a status variable and set its initial value, the *$eis\_dev\_conf["status"]* associative array is available into the “*device\_conf.php*” file. If the variable “*mystatusvar*” with initial value “37” has to be declared, simply add the following index to the *$eis\_dev\_conf["status"]* array:

*“mystatusvar” => 37*

**System-managed status variables:**

The following status variables are always required and managed by the system itself:

* ***timestamp***hold the current UNIX timestamp of the current simulation (default 0)
* ***masterurl***hold the URL of the current simulation master device
* ***enabled***hold the enable/disable device status

**Required status variables:**

The following status variables are always required but must be managed by your code:

* ***power (boolean)***hold the current power-on/power-off device status
* ***cpower1, cpower2, cpower3 (integer) (only for “load” or “load&gen”)***hold the current consumed electrical power (watt) on phase 1,2 or 3
* ***cenergy1, cenergy2, cenergy3 (float) (only for “load” or “load&gen”)***hold the consumed electrical energy (kWh) on phase 1,2 or 3 from the beginning of current simulation
* ***gpower1, gpower2, gpower3 (integer) (only for “generator” or “load&gen”)***hold the current generated electrical power (watt) on phase 1,2 or 3
* ***genergy1, genergy2, genergy3 (float) (only for “generator” or “load&gen”)***hold the generated electrical energy (kWh) on phase 1,2 or 3 from the beginning of current simulation

**Other private status variables:**

Other status variables can be freely added to the *$eis\_dev\_conf["status"]* array. They will have to be completely managed by your code.

1. **Predefined commands and signals**

In the ***eis*** system, your new device is now ready to accept commands and signals and to answer to them. Some commands and signals are predefined and already implemented without any additional coding.

**Main predefined commands:**

* ***ping***  
  *returns the call input parameters as return parameters   
  input parameters : any   
  output parameters : the input parameters   
  errors : none*
* ***delay***  
  *delay the answer of “duration” seconds (default 10)   
  input parameters : duration (integer, optional)   
  output parameters : none   
  errors : none*
* ***getlog***  
  *returns the last “numrow” lines (default 10) from the log   
  input parameters : numrow (integer, optional)   
  output parameters : getlog (array of log lines)   
  errors : none*
* ***getstatus***  
  *returns the value(s) of status variable(s). With no input parameters returns all the device status variables. If the “fields” parameter is present, returns only the status variables specified in it as a list of comma separated names   
  input parameters : fields (string, optional) example: varname1,varname2   
  output parameters : all the requested variables as name=>value   
  errors : none*
* ***setstatus***  
  *sets the value(s) of status variable(s). The variables to set are given as input parameters (name=>value)   
  input parameters : all the requested variables as name=>value   
  output parameters : none   
  errors : none*
* ***getconfig***  
  *returns the $eis\_dev\_conf array as output parameters   
  input parameters : none   
  output parameters : $eis\_dev\_conf array as name=>value   
  errors : none*
* ***help***  
  *returns an help string (if any) as output parameter   
  input parameters : none   
  output parameters : “help” field containing the help string   
  errors : none*
* ***init***  
  *initialize a new simulation starting at UNIX time “timestamp” passed as input parameter. The device status is reset to its initial value (stored in $eis\_dev\_conf[“status”] array), the “timestamp” status variable is then set to the value of the passed input parameter “timestamp” and the “masterurl” status variable is set to the URL of the device issuing the call. The log register is also reset and the device is enabled. This command returns the $eis\_dev\_conf array as output parameters with the “status” field set as described before.   
  input parameters : timestamp (integer, mandatory)   
  output parameters : $eis\_dev\_conf array as name=>value, with set “status” fields  
  errors : system:timestampMissing (no timestamp in call parameters)*

**Main predefined signals:**

* ***enable***  
  *set the internal status of the device to “enabled”. This means that the device will respond to any implemented command or signal (normal operation)   
  input parameters : none*
* ***disable***  
  *set the internal status of the device to “disabled”. This means that the device will not respond to any implemented command or signal (it is not available for operations) except for the “enable” signal and the “init” command. Both set the internal status to “enabled” again.   
  input parameters : none*

1. **How to write your code**

The new device MUST implement at least 1 command and 2 signals. Edit the file “*device.php*” in the “private” folder which will contain your actual implementation, remembering that 4 global variables are always available:

* ***$eis\_conf***  
  an associative array as defined in “*eis\_conf.php*” (global configuration);
* ***$eis\_dev\_conf***  
  an associative array as defined in your “*device\_conf.php*” (device configuration);
* ***$eis\_dev\_status***  
  an associative array as defined in *$eis\_dev\_conf[“status”]* containing the device current status;
* ***$calldata***  
  an associative array describing the current call. Useful indexes are:   
   ***from*** contains the URL of the caller   
   ***type*** call type (exec, dexec or signal)   
   ***cmd*** the command/signal name   
   ***param*** the call input parameters as an associative array

Inside this file, 3 functions are already defined:

* ***eis\_device\_init()***  
  put in it your device initialization code (if any). This code will be executed when the command “***init***” is sent to the device.
* ***eis\_device\_exec($calldata****)*  
  put in it the code implementing your device commands. A skeleton is already given and should be used, each case of the switch corresponds to a command implementation. The command “***simulate***” MUST always be implemented.
* ***eis\_device\_signal($calldata)***  
  put in it the code implementing your device signals. A skeleton is already given and should be used, each case of the switch corresponds to a signal implementation. The signals “***poweron***” and “***poweroff***” MUST always be implemented.

In case of error, return an appropriate message using the following code:

***return eis\_error\_msg(****"your\_brand.your\_model:your\_code ","your\_error\_message"****);***

where “*your\_brand.your\_model” should be taken from your deviceID, “your\_code*” is your specific error code and “*your\_error\_message*” is a short message describing the error.

**How to write the “simulate” command:**

This required command implements a single step of the electrical simulation. The main goal of the step is:

* update the current energy values in kWh (cenergy1, cenergy2, cenergy3 and/or genergy1, genergy2, genergy3 fields of the ***$eis\_dev\_status***) using the corresponding electrical powers in watt (cpower1, cpower2, cpower3 and/or gpower1, gpower2, gpower3) and the time passed from the last “simulate” call in seconds *($callparam["timestamp"]-$eis\_dev\_status["timestamp"]*).
* update the electrical power consumed/generated by the device at the new timestamp (cpower1, cpower2, cpower3 and/or gpower1, gpower2, gpower3 fields of the ***$eis\_dev\_status***), considering also its power status (on or off as true or false in ***$eis\_dev\_status[“power”]*** and any other required input parameters. In order to make an input parameter (e.g. “your\_parameter”) mandatory, add the following code:   
   *if (!array\_key\_exists("your\_parameter",$callparam))   
   return eis\_error\_msg(“system:parameterMissing","your\_parameter");*
* update the timestamp in the corresponding status variable with the new value:  
   *$eis\_dev\_status["timestamp"]=$callparam["timestamp"];*

**How to write the “poweron” and ”poweroff” signals:**

These required signal implements respectively the actions of electrically switching on and switching off the device. Their main goals are:

* ***poweron***  
  set the *$eis\_dev\_status["power"]* to true and update accordingly all the suitable status variables.
* ***poweroff***  
  set the *$eis\_dev\_status["power"]* to false and update accordingly all the suitable status variables (e.g. all the powers should be set to zero).

1. **Testing the device**

The new device can be tested by sending to it commands and/or signals using the page *console.php* available in any master device.

1. **Writing the device help**

It is important to write a device help, which will be available through the “***help***” command or through the web page “*help.php*” in the device folder. To do this, edit the file “*help.txt*” in the “private” folder. This is a standard ASCII file written using the following simple sintax:

* lines beginning with a "**#**" char are *comments* and will be escaped
* to start a section description use {\*\* your\_section\_name \*\*}
* to start a command description use [\*\* your\_command\_name \*\*]
* to start a signal description use (\*\* you\_signal\_name \*\*)

At least the following section should be present: {\*\*Description\*\*}, {\*\*Commands\*\*}, {\*\*Signals\*\*}. Follow the skeleton for details.