EPICS ON THE WEB

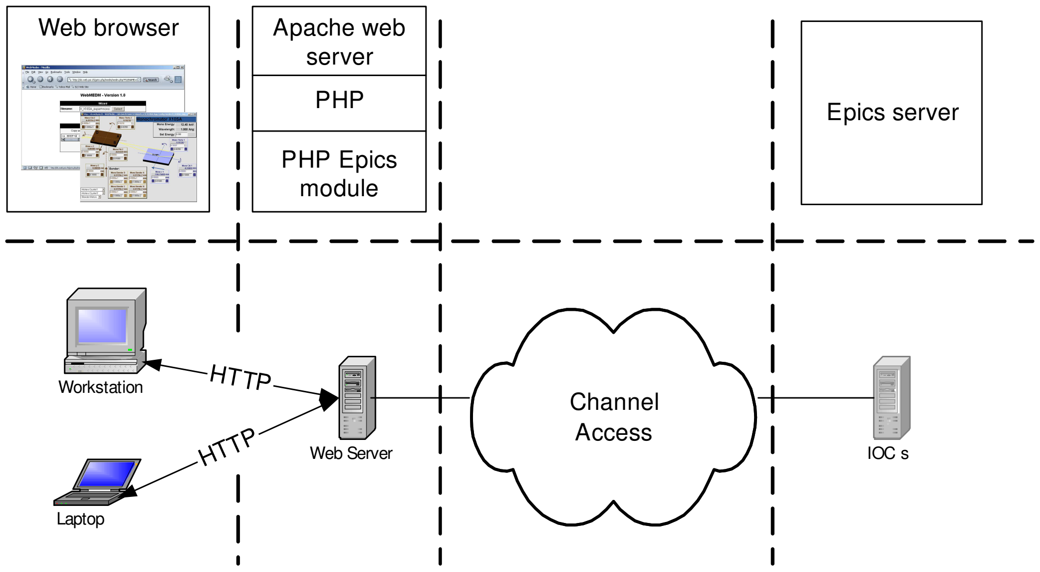
A. Bertrand, R. Krempaska
Swiss Light Source, Paul Scherrer Institute, Villigen, Switzerland

THE CONTROL SYSTEM VIA THE WEB

Most of the scientific instruments at the Swiss Light Source (SLS) at PSI are controlled using the EPICS[1] control system, which offers many solutions to easy implement applications, but they are platform dependent. For developing users' services, the platform independence and remote access are starting to be mandatory. To fulfill those needs, we have developed an interface between the EPICS channel access protocol and the PHP[2] Web scripting language. In order to speedup the development of the epics web accessibility, a MEDM parser/interpreter has been developed using the PHP Epics module. Using this software (WebMEDM), most MEDM[3] panels are directly, without any further work accessible via a web browser, allowing remote controls of the beamlines. The resulting HTML pages have the same look and feel as their MEDM counterpart and all the dynamic values contained are updated in real-time as the original.

TECHNOLOGY

One of the strong characteristics of PHP is the possibility to extend it by writing modules in C. We developed therefore a PHP extension [4], which allows to directly have access to EPICS PV (process variables) via the epics protocol Channel Access. The following diagram represents the communication between a client using a web browser and an Epics server.



PHP Epics architecture

Data between the client and the web server are exchanged using the standard HTTP protocol, which is then passed over by our module to the channel access protocol to the epics server.

FUNCTIONS IMPLEMENTED

Once the PHP-Epics module is loaded inside PHP, 5 new functions are available which covers all the needs for a full EPICS communication:

varriant ca_get(channel_name)	Get the value of the channel and return it with the correct variable type (double, int, string)
ca_put(channel_name, channel_value)	Set the value of the specified channel to the new value. Returns either OK or an error message.
ca_monitor(channels, timouout, callback)	Setup a monitor on the channels (separated by commas) for the given "timeout" number of seconds. The function "callback" will be called each time some channel values change. The callback function need to receive 2 parameters, where the first is the channel modified, and the second is the new value. If 0 is specified for the timeout, the monitor will continue forever (not recommended).
ca_type(channel)	Return the type of data.
ca_state(channel)	Return the state of the channel. For example "Valid chid, connected to server"
ca_info(channel)	Return information about the channel, like the host serving it (returned in array).

For example, to have access to a given PV and output the value to a web page, the following code would be needed:

```
<?PHP
dl('php_epics.so');                          // Load the PHP – Epics module
echo ca_get('MY-PV-NAME.VAL');               // Will connect to the PV and output the value.
?>
```

In order to implement a monitor, a call back function need to be defined:

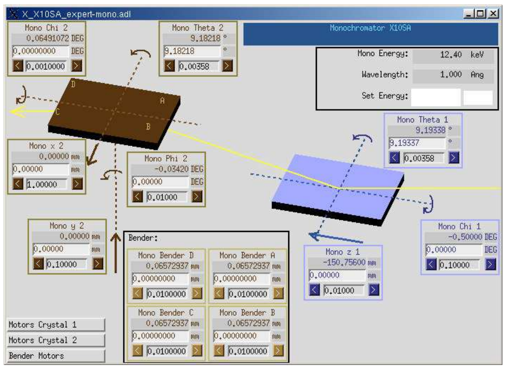
```
<?PHP
dl('php_epics.so');                                        // Load the PHP – Epics module

function my_callback($channel,$value)
{
        echo "$channel = $value<BR>\n";
}

ca_monitor("MY-PV-NAME.VAL",40,"my_callback");  // Will call the callback each time there is a change.
?>
```

TREES TO SPEED UP

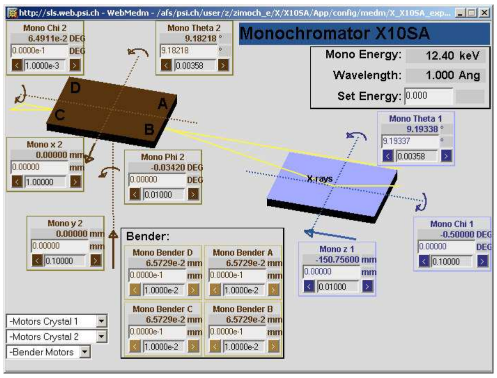
The php-epics module is implemented in such a way that when a client does an access to an EPICS PV for the first time, a connection is established and a channel identifier or chid is created. We are storing the chid in a structure so that when the next access to this PV is done, the already created chid is used. First we used a chained linked list or chain. However, with increasing number of channels, the performance was slow, which resulted to long php page load time. That’s why we started to implement a tree structure, which should speed up the search. First we used a binary tree, a structure where each node has at most two leaves - right and left. This is working for so called true binary trees. If, however the elements are in the alphabetical order, we can end up with a degenerated tree, which has one branch very long. That’s why we implemented another solution - a quaternary tree having four branches at each node. The search through quaternary tree and through linked chain list is factor three faster.

EXAMPLES OF MEDM PANELS

Using the PHP Epics module, a MEDM parser has been also developed, and allows to directly view/use most panels directly from a browser, on any platform. As a direct comparison of the look & feel of a real MEDM panel, you can see that most features are correctly displayed.



Original MEDM version



WebMEDM version

This allow to develop easily interfaces which can be then either used locally at the institute or from any other part in the world, for example for remote control, or user support.As this is a real MEDM ADL file parser, it doesn't go through some sort of translator, which means as soon as you change the ADL file and re-load the web page, the WebMEDM panel will be updated.

REFERENCES

[1] Epics.. http://www.aps.anl.gov/epics/
[2] PHP .... http://www.php.net
[3] MEDM .... http://www.aps.anl.gov/epics/extensions/medm/index.php
[4] PHP Epics .... http://www.sls.psi.ch/controls/help/howto/php_epics.html