My IoT Strategy

02/15/17 - Nicole Stevens

Device Addresses

Devices will be accessed using the HTTP protocol. They may be named devices (i.e., have a dns name or a name resolvable using the system resolver(8) libraries, or using a dotted IPv4 or IPv6 address.

URL Format

Each device will use a descriptive URL. For example to obtain temperature from a temperature sensor, the URL might look like "{device_address}/sensor/temp". A writable unit might look like "device_address}/device/lights?cmd=on". By using a consistent format, upstream services do not need to determine the minutia of obtaining or setting device information.

Data format

Data Return Format

Each device will return data in Javascript Object Notation (JSON) format. This allows multiple languages the ability to parse this data and return it in native format.

Example

```
{"time":1487212707,"temperature":69.8,"units":"f","status":"ok","humidity":
34}
```

Time values

Unless otherwise needed, times will be represented as UNIX epoch seconds. This may be an integer or a float. Some implementations such as Javascript require milliseconds which is a trivial matter to multiply by 1000.

Status Values

Each dataset returned will specify a status of 'ok' or 'fail'. It is desirable for the device, when using a status of 'fail' to put a descriptive error in a field called 'error': e.g., {"status": "fail", error="No such device or address"}

Data Input Format

Each device will accept data in HTTP urlencoded format as GET variables. Each time data is input to the device, the device must respond with at least a status.

Device and Data Abstraction

On more complex device hosts it might be desirable the device perform other functions. Being polled for data may be more resource intensive than desired. The device may cache data in anyway it needs to so long as the data is returned timely. For this reason it is important that the device return a time value so the polling host can determine if the cache is ‘stale’.

As an example, my Raspberry Pi and Banana Pi devices create a tmpfs filesystem called /sensor. On start up, a daemon process starts which polls the actual device and writes the JSON data to that directory. It is, then, a trivial matter for the web server to simply return that data to the polling host.

Code Example

```
<?php
function devUrl($sys) {
    return sprintf("http://%s.ducksfeet.com/sensor/temp",$sys);
}
if (php_sapi_name() == "cli") {
    for($i = 1; $i < $argc; $i ++) {
        if(strpos($argv[$i],'=') != -1) {
            list($vi,$vv) = explode('=',$argv[$i],2);
            $_GET[$vi] = $vv;
        }
    }
    $eol = "\n";
} else {
    $eol = "<br/>";
}
if(!isset($_GET['sys'])) {
    $dht = array("status" => "fail", "error" => "device not specified",
"mode" => php_sapi_name());
    $sys = null;
}
else {
    $sys = $_GET['sys'];
}
if($sys != null) {
    try {
        $timey = time()+60;
        unset($dht);
        $url = devUrl($sys);
        $dht = @json_decode(file_get_contents($url), true);
        if(!isset($dht)) {
            $dht["sys"] = $sys;
            $dht["url"] = $url;
            throw new Exception("Unable to get data from '${sys}': No
response.");
        }
        $timex = (int)$dht["time"];
        if($timex > $timey) {
            throw new Exception("Stale Data");
        }
        $dht["sys"] = $sys;
    } catch(Exception $e) {
        $dht["status"] = "fail";
        $dht["error"] = $e->getMessage();
        if (!isset($dht["status"])) {
            $dht["tf"] = 0;
            $dht["tc"] = 0;
            $dht["h"] = 0;
        }
    }
}
foreach($dht as $k => $v) {
    echo "${k}: $v${eol}";
}
?>
```