"The first step in a great mobile experience"

CLOUD CLIENT PERL

Support

The ScientiaMobile Enterprise Support Portal is open to all WURFL users, both commercial license holders and evaluation users. It represents the combined knowledge base for the WURFL community. Commercial licensees are invited to post questions in the forum using the account to which their licenses are associated. This may mean faster handling of those posts by ScientiaMobile's personnel.

For commercial license holders, there are tiered support levels to address a variety of business support needs. After logging into your account, commercial licensees with support options can access the Enterprise Support portal to post tickets. These tickets will receive expedited attention.

To inquire about support plans, use our License Inquiry or our General Inquiry form.

Update Notifications

If you would like to be notified of our API updates, major data updates, and other technical changes, please subscribe to our ScientiaMobile Announcements list

www.scientiamobile.com
Tel +1.703.310.6650
E-mail: sales@scientiamobile.com

ScientiaMobile WURFL Cloud Client for Perl

Introduction

The WURFL Cloud Service by ScientiaMobile, Inc., is a cloud-based mobile device detection service that can quickly and accurately detect over 500 capabilities of visiting devices. It can differentiate between portable mobile devices, desktop devices, SmartTVs and any other types of devices that have a web browser.

In order to use the WURFL Cloud client you need to obtain an API key from Scientiamobile. The WURFL Cloud API is offered in two main flavors - **Base** and **Premium**. The Base version is limited to returning only a small number of capabilities; the Premium version doesn't have an upper limit to the number of returned capabilities and also offers more caching options for improving the overall performance. Once you've properly set up an account, you can download the cloud client and receive an API key to successfully use the library. You can create an account by visiting ScientiaMobile.com.

Integration

If you use a PSGI-compatible web framework (such as Catalyst, Dancer, Mojo and others), the easiest way to use this client is to apply the Plack::Middleware::WURFL::ScientiaMobile module to your application. It will provide the device capabilities to your request handlers automatically with minimal programming effort.

Here is a quick example of how to get up and running quickly:

```
use Net::WURFL::ScientiaMobile;

my $scientiamobile = Net::WURFL::ScientiaMobile->new(
   api_key => '...',
);

# process this HTTP request
$scientiamobile->detectDevice($env);

# check if the device is mobile
if ($scientiamobile->getDeviceCapability('ux_full_desktop')) {
   print "This is a desktop browser.";
}
```

Querying The Cloud Client API

To get the requested capabilities from the WURFL Cloud for the given HTTP Request. If the second argument is not provided, all available capabilities will be fetched.

```
$scientiamobile->detectDevice($env);
$scientiamobile->detectDevice($env, ['ux_full_desktop', 'brand_name']);
```

Refer to the documentation of your web framework to learn how to access$env. For example, Catalyst provides it in $ctx->request->env, Dancer provides it inrequest->env, Mojo provides it in$self->tx->req->env.

Instead of the $env hashref you can also supply a HTTP::Headers or a Mojo::Headers object. This is handy when you're not running in a PSGI environment and your web server doesn't supply a PSGI-compatible $env hashref (for example, when running ./myapp.pl daemon in a Mojolicious::Lite application. Note that the Dancer built-in web server still provides a PSGI-compatible $env).

The sample below returns the value of the requested capability. If the capability does not exist, returns undef.

```
my $is_wireless = $scientiamobile->getDeviceCapability('is_wireless_device');
```

Flat capabilities hashref, thus containing'key' = 'value'> pairs. Since it is flattened, there are no groups in this array, just individual capabilities.

The sample below reads the user agent string from$env.

```
my $user_agent = $scientiamobile->getUserAgent($env);
```

Shown below, is how to get the date that the WURFL Cloud Server was last updated as a UNIX timestamp (seconds since Epoch). This will be undefined if there has not been a recent query to the server, or if the cached value was pushed out of memory.

```
my $date = $scientiamobile->getLoadedDate;
```

Constructors

The constructor accepts the following named arguments.

- **api_key**: Required. The full API key provided by the WURFL Cloud Service.
- **cache**: A Net::WURFL::ScientiaMobile::Cache object (or class name as string). If none is provided, no caching will occur.
- **http_timeout**: The timeout in milliseconds to wait for the WURFL Cloud request to complete. Defaults to 1000.
- **compression**: Boolean flag to enable/disable compression for querying the WURFL Cloud Service. Using compression can increase CPU usage in very high traffic environments, but will decrease network traffic and latency. Defaults to true.
- **auto_purge**: If true, the entire cache (e.g. memcache, etc.) will be cleared if the WURFL Cloud Service has been updated. This option should not be enabled for production use since it will result in a massive cache purge, which will result in higher latency lookups. Defaults to false.
- **report_interval**: The interval in seconds, after which, API will report its performance.
- **wcloud_servers**: WURFL Cloud servers to use for uncached requests. Theweight field can contain any positive number, the weights are relative to each other. Use this if you want to override the built-in server list. For example:

```
my $scientiamobile = Net::WURFL::ScientiaMobile->new(
   api_key => '...',
   wcloud_servers => {
       # nickname => [ host => weight ],
       'wurfl_cloud' => [ 'api.wurflcloud.com' => 80 ],
   },
);
```

Methods for Server Pool Management

- **addCloudServer**: Adds the specified WURFL Cloud Server. The last argument is the server's weight. It specifies the chances that this server will be chosen over the other servers in the pool. This number is relative to the other servers' weights.

```
$scientiamobile->addCloudServer('wurfl_cloud', 'api.wurflcloud.com', 80);
```

- **clearServers**: Removes the WURFL Cloud Servers.

```
$scientiamobile->clearServers;
```

- **getWeightedServer**: Uses a weighted-random algorithm to chose a server from the pool. It returns an arrayref whose first argument is the host and the second argument is the weight. You don't need to call this method usually. It is called internally when the client prepares the request to the WURFL Cloud Service.

```
my $server = $scientiamobile->getWeightedServer;
```