The better PHP API

(По-доброто PHP API)

By Vasil Rangelov a.k.a. boen_robot
boen.robot@gmail.com

What is this “API” thing?

- In programming:
  - “**A**pplication **P**rogramming **I**nterface” - A set of operations, their inputs and outputs, unified by common purpose, and available to the programmer.
- In MikroTik RouterOS:
  - **Layer7 client/server protocol** (ala HTTP, FTP, SSH, etc.)
  - Intended for machines, not for humans
  - Rigid, structured and lacking “human” conveniences
  - To avoid confusion, the protocol will be referred to as “RouterOS API”.

What can it be used for?

- Creating custom management UI
- Integration of RouterOS with external applications, e.g.
  - Billing systems
  - Self-service applications
  - Monitoring tools
- In a word: Automation.

SSH vs. RouterOS API

- Both could be used for automation
- SSH is intended for human use
  - "Cosmetic" changes from one RouterOS version to another can cause problems with automation tools
  - Renames/removal of commands/arguments/properties can cause not just <u>**hard** to detect failures</u>, but potential damages too
- RouterOS API is intended for machine use
  - Only changes to the protocol itself could possibly cause issues
  - <u>**Easy** to detect failures</u>, and *potential* recovery without damages.

Languages with general purpose RouterOS API clients

- **PHP** (3)
- Java (2)
- .NET (C# (3), VB.NET (1))
- Python (2)
- Node.js (1)
- Perl (2)
- Ruby (1)
- Delphi (2)
- C/C++ (4)
- Erlang (1)
- ActionScript (1)

Comparison of PHP RouterOS API clients

Client	API PHP class (Denis Basta)	RouterOS PHP class (ayufan)	API PHP package (boen_robot)
Requires PHP version	5.**2+**	5.**2+**	5.**3+**
Raw protocol I/O	Yes	No	Yes
General purpose conveniences	One method (comm())	No	Set of classes
CRUD and misc. conveniences	No	Yes	Yes (1 class)
Asynchronous requests	No	Callbacks only	Yes
Scripts	No	Pseudo	Yes
Persistent connections	No	No	Yes
Command line emulation	No	No	Pseudo

Preparing the target router for RouterOS API

- For RouterOS earlier than v6.0, enable the protocol with
  - /ip service enable api
- The protocol is enabled by default since RouterOS v6.0
- The RouterOS user needs to have the “api” policy
  - Default user groups all have it; Take care when using custom groups
- If using the firewall’s “input” or “output” chains, ensure **incoming** connections **to** TCP port 8728 are allowed, as are **outgoing** connections **from** TCP port 8728.

Preparing the client device for RouterOS API

- Install everything needed to create and run programs in your language of choice
  - For PHP, this means installing the PHP interpreter itself, and perhaps plug it into a web server
    - Easiest with all-in-one solutions like XAMPP: http://apachefriends.org/
    - For command line applications, PHP itself from http://php.net/ is enough
- Allow all involved executables (“php” and maybe “httpd”) to make outgoing connections to TCP port 8728
  - For Windows, to get to the firewall, press Win+R, and run “wf.msc”.

First use of the PHP RouterOS package (quick way)

- Download the PHAR file from https://pear2.github.io/Net_RouterOS/
- Create a file in Apache’s “htdocs” folder called f.e. “rostest.php”, with the following contents:

```
<?php
use PEAR2\Net\RouterOS;
require_once 'PEAR2_Net_RouterOS-1.0.0b5.phar';

try {
    $client = new RouterOS\Client('192.168.0.1', 'admin', 'password');
    echo 'OK';
} catch (Exception $e) {
    die($e);
}
```

- Adjust the path to the “.phar” file, the RouterOS IP and credentials as needed
- Open a web browser and run http://localhost/rostest.php.

Alternative installation methods

- Extract the TGZ or ZIP files from https://pear2.github.io/Net_RouterOS/
  - Include “src/PEAR2/Autoload.php”
- Using Composer:
  - composer require pear2/net_routeros
  - Include “vendor/autoload.php”
- Using PEAR:
  - pear channel-discover pear2.php.net
  - pear install -a pear2/PEAR2_Net_RouterOS-alpha
  - Include “PEAR2/Autoload.php”
- Using Pyrus:
  - pyrus install -o PEAR2_Net_RouterOS-alpha
  - Include “PEAR2/Autoload.php”.

Troubleshooting (step 1: environment issues)

- Open up the command line
  - On Windows, Win+R, and run “cmd”
- Start the PHAR file with PHP
  - Windows:
    - "D:\path\to\php.exe" "D:\path\to\PEAR2_Net_RouterOS.phar“
  - UNIX:
    - "/the/path/to/php" "/the/path/to/PEAR2_Net_RouterOS.phar“
- Check the output for any warnings or errors
- Note: For brevity, “php PEAR2_Net_RouterOS.phar” will be used in the following slides in place of full paths. Adjust accordingly.

Troubleshooting (step 2: connection issues)

- Start the PHAR file with the router's IP as an argument
  - php PEAR2_Net_RouterOS.phar 192.168.0.1
- If using a different TCP port for the RouterOS API (e.g. 443):
  - php PEAR2_Net_RouterOS.phar -p 443 192.168.0.1
- Check the output for any errors and possible solutions
  - All is OK if you get nothing, and are allowed to type
    - Enter "/quit" (without the quotes), and press Enter twice to exit.

Troubleshooting (step 3: login issues)

- Start the PHAR file with the RouterOS username and password added at the end, e.g.
  - php PEAR2_Net_RouterOS.phar 192.168.0.1 "admin" "password"
- Check the output for any errors and possible solutions
  - All is OK if you get nothing, and are allowed to type
    - Enter “/quit” (without the quotes), and press Enter twice to exit.

Example: Torch for 4 seconds

```php
<?php
use PEAR2\Net\RouterOS;
require_once 'PEAR2_Net_RouterOS-1.0.0b5.phar';

$client = new RouterOS\Client('192.168.0.1', 'admin', 'password');

header('Content-Type: text/plain');

$torchRequest = new RouterOS\Request('/tool torch duration=4');
$torchRequest->setArgument('interface', $_GET['nic']);
foreach ($client->sendSync($torchRequest) as $response) {
    foreach ($response as $name => $value) {
        echo "{$name}: {$value}\n";
    }
    echo "====\n";
}
```

Example: Print RouterOS logs

```
<?php
use PEAR2\Net\RouterOS;
require_once 'PEAR2_Net_RouterOS-1.0.0b5.phar';

$client = new RouterOS\Client('192.168.0.1', 'admin', 'password');

header('Content-Type: text/plain');

$util = new RouterOS\Util($client);
foreach ($util->setMenu('/log')->getAll() as $item) {
    echo $item('time')    . ' ' .
         $item('topics') . ' ' .
         $item('message') . "\n";
}
```

Example: Executing script with parameters

```
<?php
use PEAR2\Net\RouterOS;
require_once 'PEAR2_Net_RouterOS-1.0.0b5.phar';

$client = new RouterOS\Client('192.168.0.1', 'admin', 'password');

header('Content-Type: text/plain');

$util = new RouterOS\Util($client);
$util->exec('
    /ip dhcp-client lease
        make-static [find address=$address]
        comment [find address=$address] $name
    /log info "User $name now has the static IP $address"
    ',
    array(
        'name' => $_GET['user'],
        'address'   => $_GET['ip']
    )
);
```

Support

- Documentation and more examples
  - GitHub wiki: https://github.com/pear2/Net_RouterOS/wiki
  - MikroTik wiki: http://wiki.mikrotik.com/wiki/API_PHP_package
- Ask questions
  - MikroTik forum: http://forum.mikrotik.com/
- Report bugs or request features for the client
  - GitHub issue tracker: https://github.com/pear2/Net_RouterOS/issues
- Report bugs or request features for the protocol
  - MikroTik support: support@mikrotik.com.

Technical FAQ

- Can I run a RouterOS API application on the router itself?
  - No
- Can I log in hotspot users with the API protocol?
  - No. Not with the API protocol, not with SSH, not with Winbox even.
  - Hotspot users can ONLY be logged in when THEIR devices make an HTTP(S) request to the router with their credentials
- What if my target router is behind NAT?
  - Solutions are same as those applicable for any TCP protocol
    - VPN?
- What about encryption?
  - The API protocol has an TLS encrypted variant on port 8729
  - PEAR_Net_RouterOS supports it, but due to PHP problems, such connections are **very** unstable currently.

Social FAQ

- Why make another PHP client?
  - I don't like the other PHP clients
    - Also many of the non-PHP clients, but it's better to have something than nothing, so…
  - The protocol is easy to implement for Computer Science bachelors like myself, so ultimately "why not?"
- Why PEAR(2)?
  - PEAR is not a framework, but a collection of packages following a common coding standard
  - It's a kind of "stamp of quality", albeit no longer a widely recognized one
- Why is the current version a beta?
  - For PEAR2, "beta" does **NOT** mean "error prone"
    - It means "possible breaking changes in the next release; Review change logs carefully when upgrading"
  - Think of it as how Gmail was in beta for a long time.

Thank you

Question time

boen.robot@gmail.com