API Descriptions

https://campus.barracuda.com/doc/3539049/

service.add

This method adds a Service to the configuration database. The VIP address and port combination must be unique.

Parameter	Description
`name`	Service name
`vip`	Set this to the Virtual IP address of the Service
`protocol`	The protocol used by the Service, either TCP or UDP
`port`	The port used by the Service
`type`	The Service type. Supported values: • L4 with protocol UDP adds a Layer 4 – UDP Service • L4 with protocol TCP adds a Layer 4 – TCP Service • L7 with protocol TCP adds a Layer 7 – HTTP Service • L7TCP with protocol TCP adds a TCP Proxy Service • L7UDP with protocol UDP adds a UDP Proxy Service • FTP with protocol TCP adds a Layer 7 – FTP Service Other combinations are not valid. Adding SSL offloaded Services (e.g., Layer 7 – HTTPS) is not supported.

Example

```
my $result;

#Add a Service
$result = $xmlrpc->call('service.add'
{name=>'xml_rpc_test',vip=>'192.168.132.214',protocol=>'TCP',port=>'21',
type=>'L4'});

print Dumper ($result);
```

Result

The output of a successful call is a hash with a value and a message.

val	msg
200	Service successfully added

```
$VAR1 = {
          ’msg’ => ’Service successfully added’,
          ’val’ => ’200’
        };
```

service.delete

This method deletes an existing Service from the configuration database.

Parameter	Description
vip	Set this to the Virtual IP address of the Service
protocol	The protocol used by the Service, either TCP or UDP
port	The port used by the Service

Example

```
my $result;

$result = $xmlrpc->call('service.delete'
{vip=>'192.168.132.214',port=>'21', protocol=>'TCP'});

print Dumper ($result);
```

Result

The output of a successful call is a hash with a value and a message.

val	msg
200	Service successfully deleted

```
$VAR1 = {
          ’msg’ => ’Service successfully deleted’,
            ’val’ => ’200’
        };
```

service.show

This method displays configuration and state information for a Service and/or a Real Server. If no parameters are entered to identify one Service or Real Server, information about all existing Services and Real Servers is returned.

Parameter	Description
`vip`	Optional. Set this to the Virtual IP address of the Service that this Real Server is to be associated with, followed by a colon, followed by TCP or UDP, i.e., VIP address: `protocol`
`ip`	Optional. The IP address of the Real Server
`port`	The protocol used by the Service, either TCP or UDP
`show`	A list of what is to be shown, separated by ‘/’. For example, to show the status of a Service and the state of a Real Server, use `status/state`

Example

```
my $result;

#Show Service status and Real Server state
my $service_show = 'status/state' ;
$result = $xmlrpc->call('service.show',
{vip=>'192.168.132.214:21:TCP', port=> '21', ip=> '15.15.15.11',
show=> $service_show});

print Dumper ($result);
```

Result

The output of a successful call is a hash for each Service with a set of variables:

Variable	Description
`proto`	TCP or UDP
`ip`	VIP address of the Service
`status`	Service status – down or up

name	Service name
port	Service port

and a hash for each Real Server with a set of variables:

Variable	Description
ip	Address of the Real Server
status	Real Server status – disabled or enabled
port	Real Server port
state	Real Server state – maintenance, disabled, or enabled

server.add

This method adds a Real Server to the configuration database. It must be added to an existing Service.

Parameter	Description
vip	Set this to the Virtual IP address of the Service that this Real Server is to be associated with, followed by a colon, followed by TCP or UDP, i.e., VIP address: protocol
ip	The IP address of the Real Server
port	The port used by the Real Server

Example

```
my $result;

#Add a Real Server
$result = $xmlrpc->call('server.add',
{vip=>'192.168.132.214:21:TCP',ip => '15.15.15.11',port => '21'});

print Dumper ($result);
```

Result

The output of a successful call is a hash with a value and a message.

val	msg
200	Server successfully added

```
$VAR1 = {
          ’msg’ => ’Server successfully added’,
          ’val’ => ’200’
        };
```

server.delete

This method removes an existing Real Server from an associated Service.

Parameter	Description
vip	Set this to the Virtual IP address of the Service that this Real Server is associated with, followed by a colon, followed by TCP or UDP, i.e., VIP address: `protocol`
ip	The IP address of the Real Server
port	The port used by the Real Server

Example

```
my $result;

#Delete the Real Server
$result = $xmlrpc->call('server.delete',
{vip=>'192.168.132.214:21:TCP',ip => '15.15.15.11',port => '21'});

print Dumper ($result);
```

Result

The output of a successful call is a hash with a value and a message.

val	msg
200	Service successfully deleted

```
$VAR1 = {
          ’msg’ => ’Server successfully deleted’,
          ’val’ => ’200’
        };
```

server.change_state

This method changes the state of a Real Server to disabled, maintenance or enabled. Changing it to disabled terminates all connections immediately. When placed in maintenance mode, the Real Server keeps existing connections but does not accept any new ones. Maintenance can be performed when all active connections are closed.

Parameter	Description
`vip`	Set this to the Virtual IP address of the Service that this Real Server is associated with, followed by a colon, followed by TCP or UDP, i.e., VIP address: `protocol`
`ip`	The IP address of the Real Server
`port`	The port used by the Real Server
`action`	Set to one of the following strings: disable / enable / maintenance

Example

```
my $result;

#Change Real Server state to (disabled / enabled / maintenance mode)

$result = $xmlrpc->call('server.change_state',
{vip=>'192.168.132.214:21:TCP', port=>'21', ip => '15.15.15.11',
action=>'disable'});

print Dumper ($result);
```

Result

The output of a successful call is a hash with a value and a message.

val	msg

200	Server mode successfully changed

```
$VAR1 = {
          'msg' => 'Server mode successfully changed',
          'val' => '200'
        };
```

Related Articles

- Automation API Guide
- Sample Program - Multiple Functions
- Sample Program - Changing Real Server Status to Maintenance Mode