1. Barracuda Email Security Gateway API Guide
1.1 General APIs
1.1.1 Config.get
1.1.1.1 Config.get - Tied Variable Examples
1.1.2 Config.list
1.1.3 Config.set
1.1.4 Config.create
1.1.5 Config.delete
1.1.6 Config.add
1.1.7 Config.remove
1.1.8 Config.reload
1.1.9 Config.varlist
1.1.10 Config.var_attr
1.2 APIs for the Barracuda Email Security Gateway
1.2.1 User.create
1.2.2 User.list
1.2.3 User.remove
1.2.4 User.update_pref
1.2.5 Domain.add
1.2.6 Domain.delete
1.3 Use Case Scenarios
1.4 Error Codes

Barracuda Email Security Gateway API Guide

How the Barracuda API Works

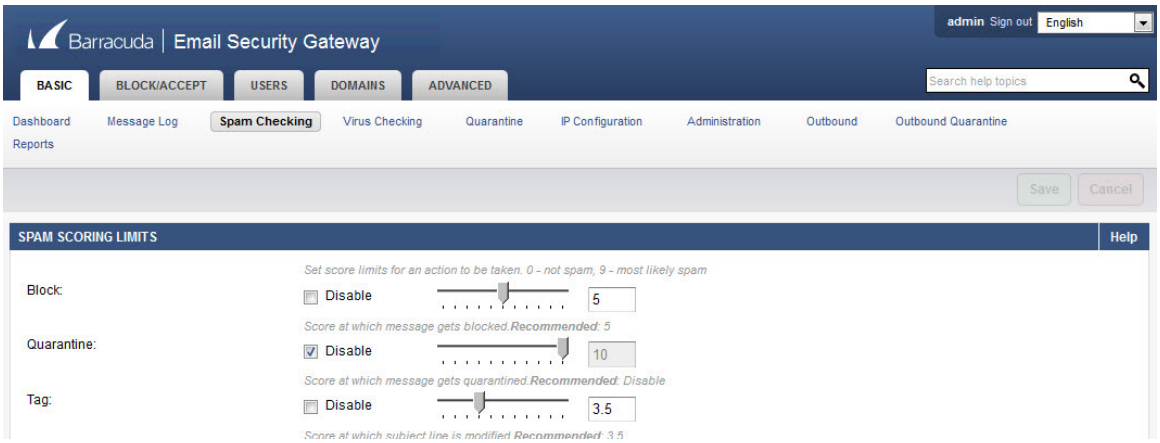
Barracuda Networks provides APIs for remote administration and configuration of the Barracuda Email Security Gateway version 4.x and above. Two sets of APIs are presented in this guide. Most of the examples shown use Perl script.

- The **General APIs** section covers "generic" APIs that may be used with all Barracuda Networks appliances that support an API. For more information, see General APIs.
- The **APIs for the Barracuda Email Security Gateway** section covers APIs that are specific only to the Barracuda Email Security Gateway. For more information, see APIs for the Barracuda Email Security Gateway.

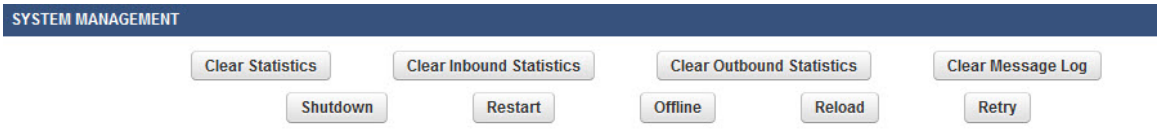
The framework of the API provides for the programmer to get or set variables inside an XML-RPC request that correspond to field values in the configuration database in the Barracuda Email Security Gateway. Some languages such as Perl, for example, provide wrappers for XML-RPC requests, providing an interface to form the request.

What Can Be Configured With the APIs

The APIs work through manipulation of variables inside of the system configuration database, and anything that can be declared in that database can be set or checked with the APIs. This includes most things that you can set by clicking the **Save** button in the Barracuda Email Security Gateway web interface. For example, from the **BASIC > Spam Checking** page, you can set global **Spam Scoring Limits** for the actions Block, Tag, or Quarantine, and then click **Save**:



Conversely, most things that correspond to "action" type buttons in the web interface cannot be configured by the APIs. For example, from the **BASIC > Administration** page, you can click a button to take the system offline, to shut it down, or to clear the message log, but you cannot execute these "actions" via the APIs. An exception to this is the **Reload** feature/button – there is an API to re-apply the system configuration.



Understanding Variables in the Configuration

The examples in this guide demonstrate getting and setting some of the variables in the configuration database. Some examples use variable names in the method calls, while other examples use explicit values, just to demonstrate both ways of making API calls.

> Make sure not to use an editor that may add special characters. Also make sure to use single quotes to surround literal values in your calls, and use double quotes to surround variables.

```
my $url    = "http://$cuda_ip:80/cgi-mod/api.cgi?password=help";
my $result = $xmlrpc->call('config.set', {
 type => 'domain',
 path => 'barracuda.com',
 mta_relay_advanced_host => '1.3.3.7'
});
```

Two of the methods in the **General APIs** section, config.varlist and config.var_attr, are utilities that provide information on scope and attributes of

configuration variables to help you understand how to access and use them. Calling these methods prior to using the other APIs will provide a good reference of the configuration variables.

Secured Access to the APIs

Access to these APIs are limited to IP addresses on a trusted IP address list configured on the **BASIC > Administration** page in the **Allowed API IP/Range** section of the Barracuda Email Security Gateway web interface. Make sure to enter the IP address(es) from which you'll access the APIs in this section of the web interface as the first thing you do. Attempts to call these APIs from any IP address that is not on the list will be denied. All calls to the APIs require the use of the API password, which is set on the same page and section in the web interface.

XML-RPC Model

In the APIs, action parameters are received as XML strings that comply with the XML-RPC specification, which can be viewed here: http://www.XMLrpc.com/spec. This requires that requests for all actions be in the form of an HTTP POST request. All actions are rolled into one CGI script (for example: api.cgi) and map to an XML-RPC method, and the parameters are those needed for the action to complete.

For example, the **get** action maps to the config.get XML-RPC method and all the parameters needed for the **get** will be sent in the XML body. The Perl module XML::RPC (note that this is not a part of the standard Perl distribution) will be used by api.cgi to retrieve the requested method and parameters. Once this is done, the action is performed and the response is sent back to the client. When an error is detected, a response that complies with the fault response of the XML-RPC specification is sent (see examples below). This response contains both a fault code and a meaningful fault string. See Error Codes for a list and explanation of fault codes.

The XML-RPC Request and Response

The XML script is called from a Perl script or other scripting language. Each API takes its own set of parameters which are submitted in the XML body of the request. Examples of the XML output follow the request example below, both for a successful request as well as for a request that returns an error. The single-value request / response involves a single variable value. Responses that contain multiple values will send the values back as an XML-RPC array. The response from the scripts is in the form of XML per the examples shown in this guide.

To make the request, use the base URL of your Barracuda Email Security Gateway that you use for connecting to the web interface and append the script name you wish to use. For example, if your script is called 'api.cgi', your URL might look something like this: **http://barracuda.mydomain.com:8000/cgi-mod/api.cgi**

Parameters used to build the request typically include some or all of the following:

- **variable** – A required parameter that tells the API which variable to return from the configuration. For example, the configuration variable 'scana_block_level' represents the global Spam Scoring Limit block level as set on the BASIC > Spam Checking page in the web interface. To get or set this variable's value, you'd put 'scana_block_level' in the XML request body specified as a variable:

```
<name>variable</name>
<value>
 <string><![CDATA[scana_block_level]]></string>
</value>
```

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by the administrator on the BASIC > Administration page in the API Password field. See the contents of 'my $url' in the Single Value Request / Response example below, which uses a password of '1234'.
- **type** – A parameter that specifies the class/scope of a variable. The "scope" of a variable would be one of either global (for global settings), domain (for per-domain settings) or user (for per-user settings).

If the variable is a "tied variable", however, one or more other variables are related to it, so multiple variables will be specified in the XML request. For example, on the **BLOCK/ACCEPT > IP Reputation** page, a custom RBL domain name or IP address is associated with, or "tied to" an "action" of Block, Quarantine or Tag. The variable names to set, which you'll see in the configuration file, are mta_rbl_custom_name and mta_rbl_custom_action respectively. In this case, the "type" would be 'mta_rbl_custom_name'.

- **path** – A parameter that typically corresponds to scope_data which refers to the particular instance of the object. For variables with global scope, the path is an empty string because there can be only one instance of global and it is the "starting point" in the same manner, for example, as the root (/) directory in Unix. So all variable and objects under global scope have type as 'global' and path as an empty string.

When setting the value of a variable or variables that have a type of 'domain', the path would be expressed as the domain name. When working with tied variables such as 'httpd_acl_ip_config_address' which relates to a value of 'httpd_acl_ip_config_netmask', for example, the path would be expressed as the actual IP address corresponding to 'httpd_acl_ip_config_address', as shown in this example:

To get the value of httpd_acl_ip_config_netmask corresponding to the httpd_acl_ip_config_address of 192.168.1.1 , the arguments would be:

```
type:      httpd_acl_ip_config_address
path:      192.168.1.1
variable:  httpd_acl_ip_config_netmask
```

Single Value Request / Response

To determine the **Spam Scoring Limit** global Block level (set in the web interface on the **BASIC > Spam Checking** page) for the Barracuda Email Security Gateway, use the config.get method to retrieve the current Block value as shown in this example.

To set the value of the global Block level, call the config.set method and set the variable scana_block_level to the desired value. Both calls deal with a single value. In the configuration, you'll see this entry for the global **Spam Scoring Limit** Block level, indicating that the current setting is '9' on the scale from 0-10:

```
# Default Block Level
scana_block_level = 9
```

Example: Perl

The config.get request would look something like this as called from a Perl script. The additional examples in further sections of this guide will only show the call from a Perl script.

```
#!/usr/bin/perl
 use strict;
 use LWP::UserAgent;
 use HTTP::Request::Common;
 # IP Address of your Barracuda
 my $cuda_ip = '192.168.126.98';
 my $url    = "http://$cuda_ip:80/cgi-mod/api.cgi?password=help" ;
 my $ua = new LWP::UserAgent;
 my $req = new HTTP::Request 'POST', $url;
 my $xml = qq|
Here's the XML:
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
  <methodName>config.get</methodName>
    <params>
     <param>
      <value>
        <struct>
         <member>
           <name>variable</name>
           <value><string><![CDATA[scana_block_level]]></string>
           </value>
         </member>
         <member>
           <name>type</name>
           <value><string><![CDATA[global]]></string>
           </value>
         </member>
        </struct>
       </value>
     </param>
    </params>
  </methodCall>
  |;
  # setup transport object with request object
  $req->content_type('text/xml');
  $req->content($xml);
# send the request over transport object
 my $res = $ua->request($req);
# show the response from the Barracuda
 print $res->as_string;
# END
```

The request is an HTTP POST to the /cgi-mod/api.cgi '. The POST data is an XML body that contains the **request** method config.get inside the <methodName> tag. The requested method is config.get since we are trying to retrieve the global Block level.
Note that the mandatory parameters needed for completing this action, "variable" (name of the configuration variable) and "password", are contained inside the <struct> tag. Each parameter is identified by the name (<name> tag) and the value (<value> tag). Possible types for each parameter are restricted by the types listed in the XML-RPC specification.

This example includes only the mandatory parameters. Optional parameters can be added to the XML body using the format mentioned and will be processed accordingly. Sample output for the request would look something like this:

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 24 Jun 2010 18:41:47 GMT
Server: BarracudaHTTP 2.0
Content-Type: text/xml; charset=UTF-8
Client-Date: Thu, 24 Jun 2010 18:42:08 GMT
Client-Peer: 192.168.126.98:80
Client-Response-Num: 1
Client-Transfer-Encoding: chunked
```

Here's the XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<methodResponse>
  <methodName>config.get</methodName>
    <params>
     <param>
      <value>
        <i4>7</i4>
       </value>
     </param>
    </params>
</methodResponse>
```

All responses will contain the 200 OK success status code. Content-type of the response will be text/XML. The actual response, i.e. the value of the requested configuration variable, will be sent inside the <value> tag.

Multi-Value Response

Responses that contain multiple values will send the values back as an XML-RPC array. The example below is a request for a list of domains configured as Accepted Email Recipient Domain(s) on the Barracuda Email Security Gateway, which can be set from the **BASIC > IP Configuration** page in the web interface and which are stored in the configuration database in the domain variable.
The response may include multiple values, returned as an array inside the <array> tag. The format of the XML response body looks like this, returning three (domain name) values:

```
OK <?XML version="1.0" encoding="UTF8"?>
   <methodResponse>
    <params>
     <param>
      <value>
        <struct>
         <member>
           <name>domain</name>
           <value>
           <array>
            <data>
             <value>
              <string>domain1.com</string>
            </value>
            <value>
             <string>domain2.com</string>
            </value>
             <value>
              <string> domain3.com</string>
             </value>
              </data>
             </array
           </value>
         </member>
        </struct>
       </value>
     </param>
    </params>
  </methodResponse>
```

Error responses use the XML-RPC faultCode and faultString formats. The error code will be the value of the faultCode member and the error string will be the valueError Response of the faultString member. See Error Codes for a list of faultCodes and descriptions of possible errors. Here's an example of an error response, showing the XML:

```
OK <?XML version="1.0"?>
   <methodResponse>
    <fault>
     <value>
      <struct>
       <member>
         <name>faultCode</name>
         <value><i4>500</4></value>
          </member>
          <member>
           <name>faultString</name>
         <value>
           <string>No such variable in configuration</string>
          </value>
         </member>
       </struct>
      </value>
    </fault>
  </methodResponse>
```

Example: PHP

This example calls the user.create API to create a new user account, which is covered in the **APIs for the Barracuda Email Security Gateway** section of this guide. The library used for this example can be found on the following sourceforge page: http://sourceforge.net/projects/phpxmlrpc/ In the code the library is included as a file. Make sure this file is readable from within your environment.

```
<?php
  include("xmlrpc.inc");
  $y = new xmlrpcval(
    array(
       "user" => new xmlrpcval("newuser@domain.com", "string")
     ), "struct");
    $m = new xmlrpcmsg('user.create');
    $m->addParam($y);
   $c = new xmlrpc_client("/cgi-mod/api.cgi?password=[APIPassword]", "[BarracudaIP]",
[BarracudaPort]);
   $r = $c->send($m);
    if (!$r->faultCode()) {
        $v = $r->value();
        print $r->serialize();
    } else {
       print "Fault <BR>";
       print "Code: " . htmlentities($r->faultCode()) . "<BR>" .
           "Reason: '" . htmlentities($r->faultString()) . "'<BR>";
   }
   ?>
```

Example: Java

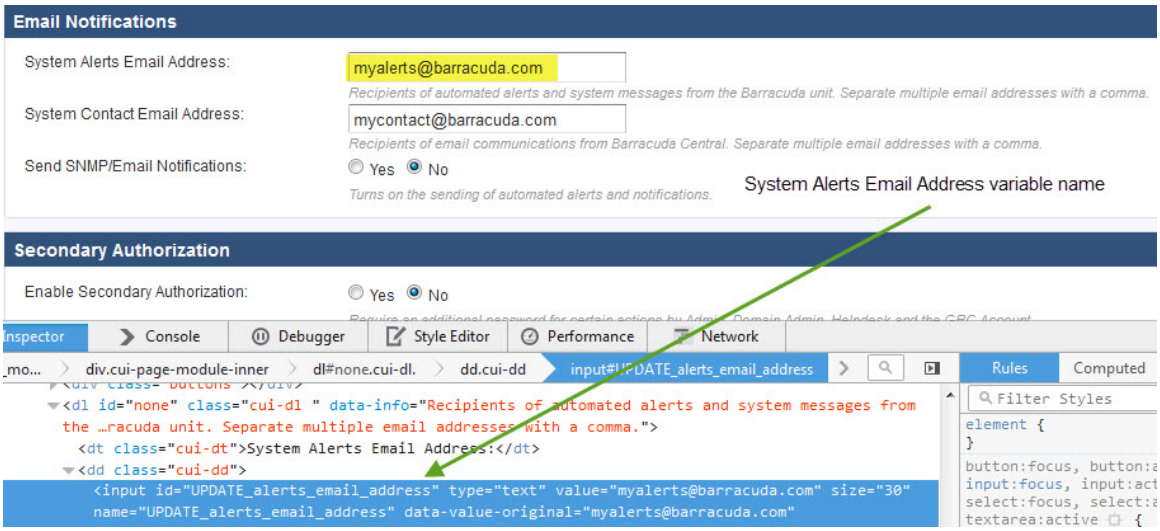
This example calls the user.create API to create a new user account, which is covered in the APIs for the Barracuda Email Security Gateway section of this guide. In the example, a key value pair is created using a standard Map class and added into a Vector list.

Required: Apache WS XML-RPC: http://ws.apache.org/xmlrpc/

```
import org.apache.xmlrpc.client.XmlRpcClient;
   import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;
   import java.net.URL;
   import java.util.Hashtable;
     import java.util.Map;
  import java.util.Vector;
  public class BarracudaAPI {
    public static void main(String[] argv) {
      try {
         XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
         config.setServerURL(new
URL("http://[BarracudaIP]:[BarracudaPort]/cgi-mod/api.cgi?password=[APIPassword]"));
        XmlRpcClient client = new XmlRpcClient();
       client.setConfig(config);
        // Create key value pair
        Map keyVals = new Hashtable();
         keyVals.put("user","newuser@domain.com");
        // Start building the parameter list
          Vector params = new Vector();
        // Add key parameter
         params.add( keyVals );
        Object result = client.execute("user.create", params);
          System.out.println(result);
               } catch( Exception ex) {
               ex.printStackTrace();
     }
              }
}
```

To determine the name of the variable you want to configure, log into the Barracuda Email Security Gateway web interface as **admin**. On the page where you configure the setting, highlight the value field, right click and select Inspect Element. The *input_id* typically contains the name of the configuration variable. See the blue highlight in the figure below: the part of the *input_id* after UPDATE_ is the variable name. In this case, it is alerts_email_address.

How to Access Variables in the Configuration



General APIs

The API interfaces presented in this section are general in that they are applicable to the Barracuda Email Security Gateway as well as to other Barracuda Networks appliances. The examples presented here are specific to the Barracuda Email Security Gateway.

Example: Config.get

Use this method to retrieve values of variables in the system configuration. If the variable requested has only a single value (Spam Tag Configuration Subject Tag level, for example), the output will be different than the output for a variable that contains a list (users, domains, etc.). This method gets the value of the variable in the object of $type named $path. The return $value is a reference to an array if it is multi-valued, i.e. a list.

Refer to the example in **Single Value Response** for getting a variable with a single value and to **Multi-value Response** for getting a variable that contains a list. Arguments to the method can be specified by just adding the parameter in the XML request.

Parameters Allowed

The following variables are used with the config.get method. These variables should be provided as part of the request XML in the HTTP POST request.

- **variable** – A required parameter that tells the API which variable to return.
- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **type** – A required parameter that specifies the class/scope of a variable.
- **path** – A required parameter that is the qualified name of an object for which the value is required. Note that the value for path is an empty string for getting a variable under global scope.

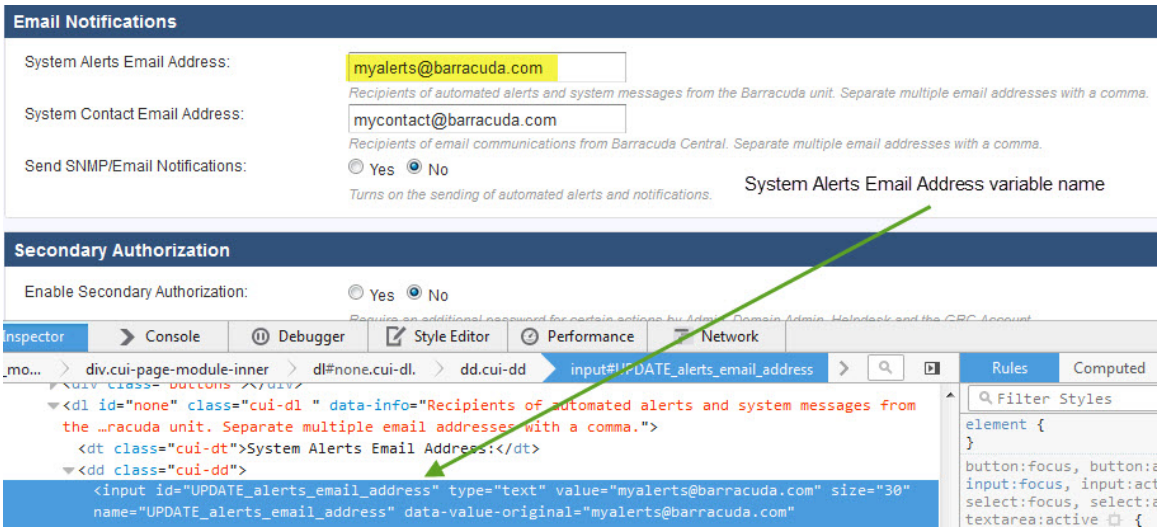
Get the Value of a Variable under Global Scope - System Alerts Email Address

Getting the current value of a system variable uses the config.get method. This example gets the value of the **System Alerts Email Address** vari able, typically set from the **BASIC > Administration** page.

Arguments

- type: 'global'
- variable: alerts_email_address

The name of the variable, **alerts_email_address**, is shown in the <input_id>, to the right of **Update**_.



XML Code for this Example

Note that the *name* tag indicates that the API applies to a single variable in the configuration. The *value* tag indicates that the expected value of that variable is a string, and takes the variable name noted above, **alerts_email_address**, as the input.

```
<?xml version="1.0" encoding="UTF8"?>
 <methodCall>
  <methodName>config.get</methodName>
    <params>
     <param>
      <value>
        <struct>
         <member>
           <name>variable</name>
           <value><string><![CDATA[alerts_email_addressl]]></string>
           </value>
         </member>
         <member>
           <name>type</name>
           <value><string><![CDATA[global]]></string>
           </value>
         </member>
        </struct>
       </value>
     </param>
    </params>
  </methodCall>
```

Perl Code for this Example

Be sure to use single quotes to surround literal values in your calls, and use double quotes to surround variables.

```
use strict;
use warnings;
use XML::RPC;

# IP Address of your Barracuda Web Filter
my $cuda_ip = "10.5.7.211";
# API Password
my $password = "1234";
my $url = "  http://$cuda_ip:8000/cgi-mod/api.cgi?password=$password  ";

#Create the XML::RPC object
my $xmlrpc = XML::RPC->new ($url);
my $result;

$result = $xmlrpc->call ('config.get',
                         {
          type => 'global',
          variable => 'alerts_email_address',
       });

# show the response from the Barracuda Web Filter
print "--- RESPONSE ---";
print $xmlrpc->xml_in();
# END
```

XML Response Returned by Perl Script

Here is the XML response returned after running the above Perl script, returning `myalerts@barracuda.com` as the **System Alerts Email Address**:

```
<methodResponse>
<params>
<param>
<value>
<string><![CDATA[myalerts@barracuda.com]]></string>
</value>
</param>
</params>
</methodResponse>
```

For more examples, see Config.get and Config.get - Tied Variable Examples.

Config.get

Use this method to retrieve values of variables in the system configuration.

Parameters Allowed

The following variables are used with the config.get method. These variables should be provided as part of the request XML in the HTTP POST request:

- **variable** – A required parameter that tells the API which variable to return.
- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **type** – A required parameter that specifies the class/scope of a variable.
- **path** – A required parameter that is the qualified name of an object for which the value is required. Note that the value for path is an empty string for getting a variable under global scope.

Example 1: Get the value of a variable under global scope - System Alerts Email Address

Getting the current value of a system variable uses the config.get method. This example gets the value of the **System Alerts Email Address** variable, typically set from the **BASIC > Administration** page.

Arguments

- type: 'global'
- variable: alerts_email_address

Sample Request

```
<?xml version="1.0" encoding="UTF8"?>
 <methodCall>
 <methodName>config.get</methodName>
   <params>
    <param>
     <value>
       <struct>
        <member>
          <name>variable</name>
          <value><string><![CDATA[scana_subject_tag]]></string>
          </value>
        </member>
        <member>
         <name>type</name>
          <value>
           <string><![CDATA[global]]></string>
         </value>
        </member>
        </struct>
       </value>
      </param>
    </params>
 </methodCall>
```

Response

```
OK <?XML version="1.0"?>
   <methodResponse>
    <params>
     <param>
      <value>
       <string><![CDATA[Block]]></string>
       </value>
      </param>
      </params>
  </methodResponse>
```

Example 2: Get the value of a variable under global scope - Subject Tag for Spam Messages

Get the value of a variable, scana_subject_tag in this case, under global scope. This example will return the **Subject Tag** string to be inserted by the Barracuda Email Security Gateway in the subject of a message determined to be spam. This setting is configured from the **BASIC > Spam Checking** page for the global setting. Note that the **path** value is an empty string and can be left out, since the scope, or type, is global.

Arguments

- type: 'global'
- variable: scana_subject_tag

Sample Request

```
<?xml version="1.0" encoding="UTF8"?>
 <methodCall>
 <methodName>config.get</methodName>
   <params>
    <param>
     <value>
       <struct>
        <member>
          <name>variable</name>
          <value><string><![CDATA[scana_subject_tag]]></string>
          </value>
        </member>
        <member>
         <name>type</name>
          <value>
           <string><![CDATA[global]]></string>
         </value>
        </member>
        </struct>
       </value>
      </param>
    </params>
  </methodCall>
```

Response

```
OK <?XML version="1.0"?>
   <methodResponse>
    <params>
     <param>
      <value>
       <string><![CDATA[Block]]></string>
       </value>
      </param>
      </params>
  </methodResponse>
```

Example 3: Get the value of a per-domain setting

This example gets the value of the **Spam Scoring Limit** block level, scana_pd_block_level, for domain `thisdomain.net`. Since this variable is in per-domain scope, the **type** is 'domain' and the **path** argument must specify a value for the domain you're working with. In the configuration, this variable is listed like this:

- # Domain Spam Block Score
- scana_pd_block_level = 5

Arguments

- type: 'domain'
- path: 'thisdomain.net'
- variable: 'scana_pd_block_level'

Sample Request

```
<?XML version="1.0" encoding="UTF8"?>
   <methodCall>
  <methodName>config.get</methodName>
    <params>
     <param>
      <value>
        <struct>
         <member>
           <name>variable</name>
           <value><string><![CDATA[mta_acl_ip_allow_comment]]></string>
           </value>
         </member>
         <member>
          <name>type</name>
           <value>
            <string><![CDATA[global]]></string>
          </value>
         </member>
         </struct>
        </value>
       </param>
    </params>
  </methodCall>
```

Response

```
OK <?XML version="1.0" encoding="UTF8"?>
   <methodResponse>
    <params>
     <param>
      <value>
       <i4>5</i4>
       </value>
      </param>
      </params>
  </methodResponse>
```

Config.get - Tied Variable Examples

The config.get method can also be used to get the values of variables that are dependent upon, or "tied to" other variables.

Example 1: Get the value of a global tied variable

This example gets the netmask value, httpd_acl_ip_config_netmask, tied to the **Allowed API IP/Range** value, httpd_acl_ip_config_address, set on the **BASIC > Administration** page. These IP addresses allow access to the Barracuda Email Security Gateway via SNMP queries to retrieve error information or to administer the system via the API. In the request, the IP address is specified in the path. These variables appear in the configuration like this:

- # API/SNMP IP Address List
- httpd_acl_ip_config_address = 192.168.1.1
- # API/SNMP IP Netmask List
- httpd_acl_ip_config_netmask = 255.255.128.0

Arguments

- type: httpd_acl_ip_config_address
- path: 192.168.1.1
- variable: httpd_acl_ip_config_netmask

Sample Request

```
<?XML version="1.0" encoding="UTF8"?>
     <methodCall>
          <methodName>config.get</methodName>
     <params>
      <param>
        <value>
         <struct>
           <member>
           <name>variable</name>
             <value><string> <![CDATA[httpd_acl_ip_config_netmask]]> </string>
         </value>
         </member>
           <member>
           <name>path</name>
           <value><string>< ![CDATA[global]]> ></string>
         </value>
         </member>
          <member>
           <name>type</name>
            <value>
            <string><![CDATA[domain]]></string>
         </value>
        </member>
       </struct>
     </param>
  </params>
   </methodCall>
```

Response

```
OK <?XML version="1.0" encoding="UTF8"?>
  <methodResponse>
   <params>
    <param>
     <value>
      <string>![CDATA[255.255.128.0]]</string>
      </value>
     </param>
     </params>
 </methodResponse>
```

Example 2: Get the value of a global tied variable

This example gets the action (Block, Tag, Quarantine) currently assigned to a custom reputation blocklist (RBL) which can be set from the **BLOCK/ACCEPT > IP Reputation** page in the web interface. The call gets the value of the mta_rbl_custom_action variable, which is set to "Block", corresponding to the mta_rbl_custom_name sbl.spamblocklist.org, which is under global scope. These variables appear in the configuration like this:

- # Custom RBL Action List
- mta_rbl_custom_action = Block
- # Custom RBL List
- mta_rbl_custom_name = sbl.spamblocklist.org

Arguments

- type: 'mta_rbl_custom_name'
- path: 'sbl.spamblocklist.org'
- variable: 'mta_rbl_custom_action'

Sample Request

```
<?XML version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.get</methodName>
  <params>
   <param>
    <value>
     <struct>
<member>
 <name>variable</name>
         <value>
          <string><![CDATA[mta_rbl_custom_action]]></string>
         </value>
</member>
<member>
 <name>path</name>
         <value>
          <string><![CDATA[sbl.spamblocklist.org]]></string>
         </value>
</member>
<member>
        <name>type</name>
         <value>
          <string><![CDATA[mta_rbl_custom_name]]></string>
         </value>
</member>
</struct>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
<?XML version="1.0" encoding="UTF8"?>
  <methodResponse>
    <params>
      <param>
<value>
  <string><![CDATA[Block]]></string>
</value>
       </param>
    </params>
  </methodResponse>
```

Example 3: Get the value of a per-domain tied variable

This example gets the netmask currently assigned to a configured IP whitelist, which can be set from the **BLOCK/ACCEPT > IP Filters** page in the web interface. The call gets the value of the mta_acl_ip_allow_netmask variable, which is set to "255.255.255.255", corresponding to the mta_acl_ip_allow_address 2.2.2.2, which is under domain scope with scope_data of `thisdomain.net`. These variables appear in the configuration like this:

- # Whitelist Netmask
- mta_acl_ip_allow_netmask = 255.255.255.255
- # Whitelist Address
- mta_acl_ip_allow_address = 2.2.2.2

Arguments

- type: 'mta_acl_ip_allow_address'
- path: 'thisdomain.net:2.2.2.2'
- variable: 'mta_acl_ip_allow_comment'

Sample Request

```
<?XML version="1.0" encoding="UTF8"?>
    <methodCall>
          <methodName>config.get</methodName>
    <params>
     <param>
       <value>
        <struct>
          <member>
          <name>variable</name>
            <value><string> <![CDATA[mta_acl_ip_allow_comment]]> </string>
        </value>
        </member>
          <member>
          <name>path</name>
          <value><string><![CDATA[thisdomain.net:2.2.2.2]]></string>
        </value>
        </member>
         <member>
          <name>type</name>
           <value>
           <string><![CDATA[mta_acl_ip_allow_address]]></string>
        </value>
       </member>
      </struct>
    </param>
 </params>
  </methodCall>
```

Response

```
OK <?XML version="1.0" encoding="UTF8"?>
  <methodResponse>
   <params>
    <param>
     <value>
      <string><![CDATA[255.255.255.255]]></string>
      </value>
     </param>
     </params>
 </methodResponse>
```

Config.list

This method lists the children of child_type ('domain', in this case) under the object parent_path of type 'parent_type'.

Parameters Allowed

The following variables are used by the config.list method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **type** – A required parameter that tells the API about the class/scope of the parent container.
- **path** – A required parameter that is the qualified name of a parent object. Note that the value for path is an empty string for getting a variable under global scope.
- **child_type** – A required parameter that specifies the child class/scope to list.

Example 1: List all valid domains

List all the children of type 'domain' under scope 'global'. This call returns a list of all domains for which the Barracuda Email Security Gateway will accept email, and which can be created and viewed from the DOMAINS page of the web interface. Each instance of the child_type (domain) appears in the configuration like this:

- #scope:<domain>::scope_data: = 'thisdomain.net'
- #scope:<domain>::scope_data = 'barracuda.com'

Arguments

- type: 'global'
- path: ''
- child_type: 'domain'

Sample Request

```
  <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.list</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
      <name>child_type</name>
       <value>
        <string>
         <![CDATA[domain]]>
        </string>
       </value>
      </member>
      <member>
       <name>path</name>
        <value>
         <string></string>
        </value>
       </member>
       <member>
        <name>type</name>
         <value>
          <string>
           <![CDATA[global]]>
          </string>
         </value>
        </member>
     </struct>
    </value>
   </param>
 </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <array>
     <data>
      <value>
       <string>
        <![CDATA[thisdomain.net]]>
       </string>
      </value>
      <value>
        <string>
         <![CDATA[barracuda.com]]>
        </string>
       </value>
       </data>
      </array>
     </value>
   </param>
 </params>
</methodResponse>
```

Example 2: List of tied objects - all custom RBLs

This example lists all values for the tied object mta_rbl_custom_name, under global scope. Custom RBLs are created from the web interface on the **BLOCK/ACCEPT > IP Reputation** page and have an associated action of Block, Tag or Quarantine. In the configuration, the three RBLs configured in this example would appear like this:

- # Custom RBL List
- mta_rbl_custom_name = sbl.spamblocklist.org
- xbl.spamblocklist.org
- sbl.org

Arguments

- type: 'global'
- path: ''
- child_type: 'mta_rbl_custom_name'

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
<methodName>config.list</methodName>
<params>
 <param>
  <value>
   <struct>
    <member>
     <name>child_type</name>
      <value>
       <string>
        <![CDATA[mta_rbl_custom_name]]>
        </string>
      </value>
     </member>
    <member>
     <name>path</name>
      <value>
       <string></string>
      </value>
    </member>
    <member>
     <name>type</name>
      <value>
       <string>
        <![CDATA[global]]>
       </string>
      </value>
    </member>
   </struct>
  </value>
 </param>
</params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <array>
     <data>
      <value>
       <string>
        <![CDATA[sbl.spamblocklist.org]]>
       </string>
      </value>
      <value>
       <string>
        <![CDATA[xbl.spamblocklist.org]]>
       </string>
      </value>
      <value>
       <string>
        <![CDATA[sbl.org]]>
       </string>
      </value>
     </data>
    </array>
   </value>
  </param>
</params>
</methodResponse>
```

Config.set

Use this method to set the values of variables in the system configuration. This method sets the variables(s) with the given values(s) for the object of type $type, identified by $path.

Parameters Allowed

The following variables are used by the config.set method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **type** – A required parameter that specifies the class/scope of an object.
- **path** – A required parameter that is the qualified name of an object for which the values are to be set.
- **variable list** – This is a required parameter that tells the API what variables are to be set and the corresponding values.

Example 1: Set the value for a scoped object under global scope

Set the value for a scoped object under global scope. This example sets the value of **Spam Score limit** block level to '4' for the xyz.com domain. In the web interface, this value would be set from the **BASIC > Spam Checking** page after clicking on the **Manage Domain** link for xyz.com on the **DOMAINS** page.

Arguments

- type: 'domain',
- path: 'xyz.com'
- variable list: scana_pd_block_level = 4

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.set</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>scana_pd_block_level</name>
        <value>
         <i4>4</i4>
       </value>
      </member>
      <member>
       <name>path</name>
        <value>
         <string>
          <![CDATA[xyz.com]]>
         </string>
        </value>
      </member>
      <member>
       <name>type</name>
        <value>
         <string>
          <![CDATA[domain]]>
         </string>
        </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <string>
         <![CDATA[200: OK]]>
        </string>
       </value>
      </member>
     </struct>
    </value>
  </param>
 </params>
</methodResponse>
```

Example 2: Set values for several variables under global scope

Set the values of https_port to '443' and mta_rate_control to '40'. The value for **Web Interface HTTPS/SS**L port can be set on the **ADVANCED > Secure Administration** page of the web interface, and the value of **Rate Control** is set on the **BLOCK/ACCEPT > Rate Control** page. These variables appear in the configuration like this (values not yet set):

- # HTTPS Web Interface Port
- https_port =
- # Maximum Connections By IP Per 30 Minutes
- mta_rate_control =

Arguments

- type: 'global'
- path: ''
- variable list: https_port => 443, mta_rate_control => 40

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
<methodName>config.set</methodName>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>https_port</name>
       <value>
        <i4>443</i4>
       </value>
      </member>
      <member>
       <name>mta_rate_control</name>
        <value>
         <i4>40</i4>
        </value>
      </member>
      <member>
       <name>path</name>
        <value>
         <string></string>
        </value>
      </member>
      <member>
       <name>type</name>
        <value>
          <string>
           <![CDATA[global]]>
          </string>
         </value>
        </member>
       </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Example 3: Set the value of a global tied variable

Set the value of httpd_acl_ip_config_netmask to 255.255.128.0 for the httpd_acl_ip_config_address of 192.168.130.222. Note that these variables are available in the configuration only if you have entered values for A**llowed API IP/Range** in the **BASIC > Administration** page, and would appear in the configuration like this:

- # API/SNMP IP Address List
- httpd_acl_ip_config_address = 192.168.130.222
- # API/SNMP IP Netmask List
- httpd_acl_ip_config_netmask =

Arguments

- type: 'httpd_acl_ip_config_address'
- path: '192.168.130.222'
- variable-value list: httpd_acl_ip_config_netmask =255.255.128.0

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.set
 </methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>
        httpd_acl_ip_config_netmask
       </name>
       <value>
        <string>
         <![CDATA[255.255.128.0]]>
        </string>
       </value>
       </member>
       <member>
        <name>path</name>
         <value>
          <string>
           <![CDATA[192.168.130.222]]>
          </string>
         </value>
        </member>
        <member>
         <name>type</name>
           <value>
             <string>
              <![CDATA[httpd_acl_ip_config_address]]>
             </string>
            </value>
         </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Config.create

This method creates an object of a given type and name under the specified parent path. Required variables will be set to their defaults if they have one; otherwise you must ensure that they have a value before a commit.

Parameters Allowed

The following variables are used by the config.create method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **parent_type** – A required parameter that tells the API about the class/scope of the parent container.
- **parent_path** – A required parameter that is the qualified name of a parent object under which a new object will be created.
- **type** – A required parameter that specifies the child's class/scope to be created.
- **name** – A required parameter that specifies the name of an object to be created.
- **variable list** – An optional parameter that tells the API which variable(s) to set in the new object.

Example 1: Create a scoped object in global scope - a new domain

Create a new domain entry of 'xyz.com' under global scope and set the value of variable scana_pd_block_level (per-domain Spam Block level) to '5'.

Arguments

- parent_type: 'global'
- parent_path: ''
- type: 'domain'
- name: 'xyz.com'
- variable list: scana_pd_block_level = '5'

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.create</methodName>
  <params>
    <param>
     <value>
       <struct>
        <member>
          <name>scana_pd_block_level
          </name>
          <value>
           <i4>5</i4>
          </value>
         </member>
         <member>
          <name>parent_type
          </name>
           <value>
            <string>
             <![CDATA[global]]>
            </string>
           </value>
          </member>
          <member>
           <name>name</name>
            <value>
             <string>
              <![CDATA[xyz.com]]>
             </string>
            </value>
          </member>
          <member>
           <name>type</name>
            <value>
             <string>
              <![CDATA[domain]]>
             </string>
            </value>
           </member>
           <member>
            <name>parent_path</name>
             <value>
              <string></string>
             </value>
           </member>
        </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Example 2: Create a tied object - custom RBL with a custom action

Create a tied object mta_rbl_custom_name of 'spamblocklist.org' with an mta_rbl_custom_action of 'Block'. The resulting entries in the configuration would look something like this:

- # Custom RBL List
- mta_rbl_custom_name = spamblocklist.org

Arguments

- parent_type:'global'
- parent_path: ''

- type: 'mta_rbl_custom_name'
- name: 'spamblocklist.org'
- variable list: mta_rbl_custom_action = Block

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
<methodName>config.create</methodName>
<params>
 <param>
  <value>
   <struct>
    <member>
     <name>parent_type</name>
      <value>
       <string><![CDATA[global]]></string>
      </value>
     </member>
     <member>
      <name>name</name>
       <value>
        <string><![CDATA[spamblocklist.org]]></string>
       </value>
     </member>
     <member>
      <name>type</name>
       <value>
        <string>
         <![CDATA[mta_rbl_custom_name]]>
        </string>
       </value>
      </member>
      <member>
       <name>parent_path</name>
        <value>
         <string></string>
        </value>
         </member>
          <member>
           <name>mta_rbl_custom_action</name>
           <value>
            <string><![CDATA[Block]]></string>
           </value>
          </member>
          </struct>
        </value>
      </param>
   </params>
</methodCall>
```

Config.delete

This method deletes an object of type $type identified by $path.

Parameters Allowed

The following variables are used by the config.delete method. These variables should be provided as part of the request XML in the HTTP POST request.

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **type** – A required parameter which specifies the class/scope of an object.
- **path** – A required parameter which is the qualified name of an object to be deleted.

Example 1: Deleting a scoped object

Delete domain 'xyz.com'.

Arguments

- type: 'domain'
- path: 'xyz.com'
- variable-value list: httpd_acl_ip_config_netmask = 255.255.128.0

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.delete</methodName>
  <params>
   <param>
    <value>
      <struct>
       <member>
        <name>path</name>
         <value>
          <string><![CDATA[xyz.com]]></string>
         </value>
       </member>
       <member>
        <name>type</name>
         <value>
        <string>
          <![CDATA[domain]]>
         </string>
         </value>
        </member>
       </struct>
      </value>
     </param>
   </params>
</methodCall>
```

Example 2: Delete a tied object and its tied variable values – global scope

Delete the global tied object mta_rbl_custom_name 'xyz.com' along with all of its tied variables. In this example, the tied variable is mta_rbl_custom_action, which stores the action (Block, Tag or Quarantine) to take with messages originating from IP addresses in custom external RBLs. These variables appear in the configuration like this:

- # Custom RBL Action List
- mta_rbl_custom_action = Block
- # Custom RBL List
- mta_rbl_custom_name = xyz.com

Arguments

- type: 'mta_rbl_custom_name'
- path: 'xyz.com'

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.delete</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>path</name>
        <value>
         <string><![CDATA[xyz.com]]></string>
        </value>
       </member>
       <member>
        <name>type</name>
         <value>
          <string>
           <![CDATA[mta_rbl_custom_name]]>
          </string>
         </value>
        </member>
       </struct>
     </value>
    </param>
 </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <string>
         <![CDATA[200: OK]]>
        </string>
       </value>
      </member>
     </struct>
    </value>
   </param>
 </params>
</methodResponse>
```

Example 3: Delete a tied object and its tied variable values – domain scope

Delete the per-domain tied variable mta_sender_allow_address along with its tied variable values. This example deletes the Allowed Email Address and Domains tied variable values 'test1.com' and 'test2.com' for the domain ‘barracuda.com’.

Arguments

- type: 'mta_sender_allow_address'

- path: 'barracuda.com'

Sample Request

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.delete</methodName>
   <params>
     <param>
      <value>
       <struct>
         <member>
           <name>path</name>
            <value>
             <string><![CDATA[barracuda.com]]></string>
            </value>
           </member>
           <member>
            <name>type</name>
             <value>
             <string>
                <![CDATA[domain]]>
             </string>
           </value>
         </member>
         <member>
           <name>variable</name>
               <value>
                  <string>
                   <![CDATA[mta_sender_allow_address]]>
                  </string>
              </value>
        </member>
        <member>
           <name>values</name>
               <value>
                <array>
                    <data>
                      <value>
                       <string>
                       <![CDATA[test1.com]]>
                       </string>
                       </value>
                     <value>
                       <string>
                       <![CDATA[test2.com]]>
                       </string>
                      </value>
                    </data>
              </array>
             </value>
        </member>
      </struct>
    </value>
   </param>
 </params>
</methodCall>
```

Config.add

This method adds the given values to the list variable. This method will not add values to tied variables, and a value added must not already exist in the list. For adding values to tied variables, use the config.create method.

Parameters Allowed

The following parameters are used by the config.add method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **parent_type** – A required parameter that tells the API about the class/scope of the parent container.
- **parent_path** – A required parameter which is the qualified name of a parent object
- **variable** – A required parameter that specifies the variable for which values will be added.
- **values** – A required parameter specifying a list of values to be added.

Example: Adding a value to a variable

Add values 192.168.128.34 and 192.168.128.2 to the mta_trusted_relay_host list.

Arguments

- parent_type: 'global'
- parent_path: ''
- variable: 'mta_trusted_relay_host'
- values: ['192.168.128.34', '192.168.128.2']

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.add</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>parent_type</name>
        <value>
         <string>
          <![CDATA[global]]>
         </string>
        </value>
      </member>
      <member>
       <name>variable</name>
        <value>
         <string>
          <![CDATA[mta_trusted_relay_host]]>
         </string>
        </value>
      </member>
      <member>
       <name>values</name>
        <value>
         <array>
          <data>
            <value>
             <string>
              <![CDATA[192.168.128.34]]>
             </string>
            </value>
            <value>
             <string>
              <![CDATA[192.168.128.2]]>
             </string>
            </value>
          </data>
         </array>
        </value>
        </member>
        <member>
         <name>parent_path</name>
          <value>
           <string></string>
          </value>
        </member>
       </struct>
      </value>
     </param>
   </params>
</methodCall>
```

Config.remove

Use this method to remove the given value(s) from the list variable. This will not remove values from tied variables. For removing values from tied variables, use the config.delete method.

Parameters Allowed

The following parameters are used by the config.remove method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **parent_type** – A required parameter that tells the API about the class/scope of the parent container.
- **parent_path** – A required parameter which is the qualified name of a parent object.
- **variable** – A required parameter that specifies the variable for which values should be removed.
- **values** – A required parameter specifying a list of values to be removed.

Example: Removing values from a variable under global scope

Removes host/domain name values 'mytrustedrelay1.com' and 'mytrustedrelay2.com from the mta_trusted_relay_host list. These **Trusted Relay Host/Domain** names are added or deleted on the **ADVANCED > Outbound** page in the web interface and represent trusted relays on the Barracuda Email Security Gateway.

Arguments

- parent_type: 'global',
- parent_path: ''
- variable: 'mta_trusted_relay_host'
- values: ['mytrustedrelay1.com', 'mytrustedrelay2.com']

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.remove</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>parent_type</name>
        <value>
         <string>
          <![CDATA[global]]>
         </string>
        </value>
       </member>
       <member>
        <name>variable</name>
         <value>
          <string>
           <![CDATA[mta_trusted_relay_host]]>
          </string>
         </value>
       </member>
       <member>
        <name>values</name>
         <value>
          <array>
           <data>
            <value>
             <string>
              <![CDATA[mytrustedrelay1.com]]>
             </string>
            </value>
            <value>
             <string>
              <![CDATA[mytrustedrelay2.com]]>
             </string>
            </value>
           </data>
          </array>
         </value>
        </member>
        <member>
         <name>parent_path</name>
          <value>
           <string></string>
          </value>
        </member>
       </struct>
      </value>
     </param>
   </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <string>
         <![CDATA[200: OK]]>
        </string>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodResponse>
```

Config.reload

Use this method to re-apply the system configuration, as can be done with the **Reload** button on the **BASIC > Administration** page of the web interface. The output of a successful call is a simple '200 OK' response - results are shown below.

Parameters Allowed

The following variable is used by the config.reload method:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
  <methodName>config.reload</methodName>
    <params>
     <param>
      <value>
       <struct>
       </struct>
      </value>
    </param>
  </params>
</methodCall>
```

Config.varlist

Use this method to list all the variables of the configuration and their attributes. This is a good method to call prior to using other APIs so you have a reference of the configuration variables.

Parameters Allowed

The following variable is used by the config.varlist method:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.varlist</methodName>
  <params>
   <param>
    <value>
     <struct/>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
      <name>new_mta_trusted_relay_netmask</name>
       <value>
        <struct>
         <member>
          <name>min</name>
           <value>
            <string></string>
           </value>
         </member>
         <member>
          <name>max</name>
           <value>
            <string></string>
           </value>
         </member>
         <member>
          <name>default</name>
            <value>
             <string></string>
            </value>
         </member>
         <member>
          <name>description</name>
            <value>
             <string>
              <![CDATA[Subnet Mask]]>
             </string>
            </value>
```

```
 </member>
  <member>
   <name>choices</name>
    <value>
     <array>
      <data/>
     </array>
    </value>
   </member>
   <member>
    <name>required</name>
     <value>
      <i4>1</i4>
     </value>
   </member>
   <member>
    <name>class</name>
     <value>
      <string>
       <![CDATA[global]]>
      </string>
     </value>
   </member>
   <member>
    <name>type</name>
     <value>
      <string>
       <![CDATA[ip_address]]>
      </string>
     </value>
   </member>
  </struct>
 </value>
</member>
<member>
 <name>mta_outbound_max_queue_lifetime
 </name>
  <value>
   <struct>
    <member>
     <name>min</name>
      <value>
       <string></string>
      </value>
     </member>
     <member>
      <name>max</name>
       <value>
        <string></string>
       </value>
     </member>
     <member>
      <name>default</name>
       <value>
        <i4>48</i4>
       </value>
    </member>
    <member>
     <name>description</name>
      <value>
       <string>
 <![CDATA[Outbound Queue Max Message Lifetime(hours):]]>
      </string>
     </value>
    </member>
```

```
    <member>
     <name>choices</name>
      <value>
       <array>
        <data/>
       </array>
      </value>
     </member>
     <member>
      <name>required</name>
       <value>
        <i4>1</i4>
       </value>
     </member>
     <member>
      <name>class</name>
       <value>
        <string>
         <![CDATA[global]]>
        </string>
       </value>
     </member>
     <member>
      <name>type</name>
       <value>
        <string>
         <![CDATA[float]]>
        </string>
       </value>
     </member>
    </struct>
   </value>
  </member>
  <member>
   <name>auth_radius_server</name>
    <value>
     <struct>
      <member>
       <name>min</name>
        <value>
         <string></string>
        </value>
       </member>
       <member>
        <name>max</name>
         <value>
          <string></string>
         </value>
        </member>
        <member>
         <name>default</name>
          <value>
           <string></string>
          </value>
         </member>
         <member>
          <name>description</name>
           <value>
            <string></string>
           </value>
          </member>
          <member>
           <name>choices</name>
            <value>
             <array>
```

```
                    <data/>
                   </array>
                  </value>
                 </member>
                 <member>
                  <name>required</name>
                   <value>
                    <string></string>
                   </value>
                 </member>
                 <member>
                  <name>class</name>
                   <value>
                    <string>
                     <![CDATA[domain]]>
                    </string>
                   </value>
                 </member>
                 <member>
                  <name>type</name>
                   <value>
                    <string>
                      <![CDATA[text]]>
                    </string>
                   </value>
               </member>
           </struct>
       </value>
     </member>
   </struct>
 </value>
```

```
    </param>
  </params>
</methodResponse>
```

Config.var_attr

Use this method to list the attributes of the specified variable.

Parameters Allowed

The following variables should be provided as part of the request XML in the HTTP POST request.

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the variable for which attributes are required.

Example: List the attributes and their values for global Block level.

This example lists the attributes of global blocking: min level, max level, current setting, etc. and returns the current value for each attribute.

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.var_attr</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>variable</name>
        <value>
         <string>
         <![CDATA[scana_block_level]]>
         </string>
        </value>
      </member>
     </struct>
    </value>
  </param>
 </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <struct>
     <member>
       <name>scana_block_level</name>
         <value>
           <struct>
             <member>
              <name>min</name>
              <value>
               <string></string>
              </value>
             </member>
             <member>
              <name>max</name>
              <value>
               <string></string>
              </value>
             </member>
```

```
        <member>
         <name>default</name>
         <value>
           <i4>7</i4>
         </value>
        </member>
        <member>
         <name>description</name>
         <value>
          <string>
           <![CDATA[Spam Block Level]]>
          </string>
         </value>
        </member>
        <member>
         <name>choices</name>
          <value>
           <array>
            <data/>
           </array>
          </value>
        </member>
        <member>
         <name>required</name>
          <value>
           <i4>1</i4>
          </value>
        </member>
        <member>
         <name>class</name>
          <value>
           <string>
            <![CDATA[global]]>
           </string>
          </value>
        </member>
        <member>
         <name>type</name>
         <value>
          <string>
           <![CDATA[float]]>
          </string>
         </value>
        </member>
       </struct>
     </value>
    </member>
   </struct
  </value>
```

```
      </param>
    </params>
</methodResponse>
```

APIs for the Barracuda Email Security Gateway

Creating a block of new user accounts or domains, deleting one or more of each, listing user accounts, using Regular Expressions and updating user-level spam score or quarantine inbox settings are some of the remote configuration capabilities presented here for the Barracuda Email Security Gateway.

In this Section:

- User.create
- User.list
- User.remove
- User.update_pref
- Domain.add
- Domain.delete

User.create

This method creates a user account for the user as specified. The output of a successful call is a simple '200 OK'.

Parameters Allowed

These variables should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the user account to be created.

Arguments

user: test@xyz.com

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>user.create</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>user</name>
        <value>
         <string>
          <![CDATA[test@xyz.com]]>
         </string>
        </value>
      </member>
     </struct>
    </value>
  </param>
 </params>
</methodCall>
```

User.list

This method simply lists all the user accounts currently on the system.

Parameters Allowed

The following variable should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.

Sample Request

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>user.list</methodName>
  <params>
   <param>
    <value>
     <struct/>
    </value>
   </param>
  </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params>
  <param>
   <value>
    <array>
     <data>
      <value>
       <string>
        <![CDATA[test@xyz.com]]>
       </string>
      </value>
     <value>
    <string>
     <![CDATA[test@thisdomain.net]]>
    </string>
   </value>
  </data>
 </array>
</value>
</param>
</params>
</methodResponse>
```

User.remove

Use this method to remove a user account for the user as specified. The output of a successful call is a simple '200 OK'.

Parameters Allowed

The following variables are used by the user.remove method and should be provided as part of the request XML in the HTTP POST request:

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the user account to be removed.

Arguments

user: test@abcd.com

Sample Request

```
 ?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>user.remove</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>user</name>
        <value>
         <string>
          <![CDATA[test@abcd]]>
         </string>
        </value>
       </member>
      </struct>
     </value>
    </param>
   </params>
</methodCall>
```

Response

```
 OK <?xml version="1.0" encoding="UTF8"?>
<methodResponse>
 <params
  <param>
   <value>
    <struct>
     <member>
      <name>Result</name>
       <value>
        <string>
         <![CDATA[200: OK]]>
        </string>
       </value>
      </member>
     </struct>
    </value>
   </param>
  </params>
</methodResponse>
```

User.update_pref

This method updates the preferences for the user account specified. The output of a successful call is a simple '200 OK'.

Parameters Allowed

The following variables are used by the user.update_pref method. These variables should be provided as part of the request XML in the HTTP POST request.

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the user account whose preference is to be updated.

First, use the config.set method to set the user- specific variables for preferences, then use this method to update the preferences.

Arguments

user: test@abcd.com

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>user.update_pref</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>user</name>
        <value>
         <string>
          <![CDATA[test@abcd]]>
         </string>
        </value>
       </member>
      </struct>
     </value>
    </param>
  </params>
</methodCall>
```

Domain.add

Use this method to add a domain, then use the config.set method to configure settings for that domain in a separate call. Use this method in a loop to add multiple domains. The output of a successful call is a simple '200 OK'.

Parameters Allowed

The following variables are used by the domain.add method. These variables should be provided as part of the request XML in the HTTP POST request.

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the domain to be created.

Arguments

domain: xyz.com

Sample Request

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>domain.add</methodName>
  <params>
    <param>
     <value>
      <struct>
       <member>
        <name>domain</name>
        <value>
         <string>
           <![CDATA[xyz.com]]>
         </string>
         </value>
       </member>
      </struct>
     </value>
    </param>
  </params>
</methodCall>
```

Domain.delete

This method deletes the specified domain. The output of a successful call is a simple '200 OK'.

Parameters Allowed

The following variables are used by the domain.delete method. These variables should be provided as part of the request XML in the HTTP POST request.

- **password** – A required parameter which the API uses to authenticate access to a page and which is set by your administrator.
- **domain** – A required parameter that specifies the domain to be deleted.

Arguments

domain: xyz.com

Sample Request

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
  <methodName>domain.delete</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>domain</name>
         <value>
          <string>
           <![CDATA[xyz.com]]>
          </string>
         </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Use Case Scenarios

These examples draw on the information presented above for using various methods to configure the Barracuda Email Security Gateway for common use cases. Some use cases address domain-level settings and some address global settings.

Use Case – Adding a Whitelist Entry to a User Account

Use the config.add method to add any email senders to the whitelist for a particular user account. This list of senders are not blocked even if the message matches spam rules. Virus scanning is still applied based on the policy set by the administrator. Whitelisting may be performed by full email address ("user@domain.com") or domain only ("domain.com").

Important: Per-User Quarantine must be enabled for the domain via the web interface BEFORE you attempt to add per-user whitelist entries. To do so, first, from the **DOMAINS > Domain Manager** page, click **Manage Domain** for the particular domain. For example, if the user account is *cuda_user@barracuda.com*, click on **Manage Domain** for barracuda.com. At the domain level, navigate to the **BASIC > Quarantine** page and set **Quarantine Type** to **Per-User**. Finally, set **Enable User Features** to *Yes*.

Arguments:

- my $value1 = 'user1@mymail.net';
- my $value2 = 'user2@mymail.net';
- my $user_account = 'cuda_user@mymail.net';

Sample Request:

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
  <methodName>config.add</methodName>
   <params>
    <param>
     <value>
      <struct>
       <member>
        <name>parent_type</name>
         <value>
          <string>
           <![CDATA[user]]>
          </string>
         </value>
       </member>
       <member>
        <name>variable</name>
         <value>
          <string>
           <![CDATA[user_scana_sender_allow]]>
          </string>
         </value>
       </member>
       <member>
        <name>values</name>
         <value>
          <array>
           <data>
             <value>
              <string>
               <![CDATA[$value1]]>
              </string>
             </value>
             <value>
              <string>
               <![CDATA[$value2]]>
              </string>
             </value>
            </data>
           </array>
          </value>
         </member>
         <member>
          <name>parent_path</name>
           <value>
            <string><![CDATA[$user_account]]></string>
           </value>
         </member>
        </struct>
       </value>
      </param>
    </params>
 </methodCall>
```

Use Case – Adding a Blocklist Entry for a Domain

Use the config.create method to add any IP addresses or networks to the blocklist for a particular domain. This example adds an IP address to the blocklist for the specified domain and adds values to the per-domain tied variables listed below. The mta_acl_ip_block_action is set to 'quarantine' for mail from the IP address added to the blocklist, and the mta_acl_ip_block_netmask is set to 255.255.255.0 since we're adding an individual IP address. A comment of 'Blocked IP address' is added as well.

- # Add values to per domain tied variable

- # Domain – xyz.mydomain.net
- # Variable – mta_acl_ip_block_address (domain scope): 10.5.36.59
- # Tied variables – mta_acl_ip_block_netmask, mta_acl_ip_block_action, mta_acl_ip_block_comment.

Arguments:

- type: mta_acl_ip_block_address
- parent_path: xyz.mydomain.net
- mta_acl_ip_block_netmask: 255.255.255.0
- mta_acl_ip_block_action: 'Quarantine'
- mta_acl_ip_block_comment: 'Blocked IP Address'

Sample Request:

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.create</methodName>
 <params>
  <param>
    <value>
      <struct>
      <member>
      <name>parent_type</name>
       <value>
        <string><![CDATA[domain]]></string>
       </value>
      </member>
    <member>
    <name>name</name>
       <value>
        <string><![CDATA[10.5.36.59]]></string>
       </value>
      </member>
      <member>
      <name>type</name>
       <value>
        <string>
         <![CDATA[mta_acl_ip_block_address]]>
        </string>
       </value>
      </member>
      <member>
       <name>parent_path</name>
        <value>
         <string><![CDATA[xyz.mydomain.net]]></string>
        </value>
      </member>
      <member>
       <name>mta_acl_ip_block_netmask</name>
        <value>
         <string><![CDATA[255.255.255.0]]></string>
        </value>
      </member>
      <member>
       <name>mta_acl_ip_block_action</name>
        <value>
         <string><![CDATA[Quarantine]]></string>
        </value>
      </member>
      <member>
       <name>mta_acl_ip_block_comment</name>
        <value>
         <string><![CDATA[Blocked IP address]]></string>
        </value>
      </member>
     </struct>
    </value>
 </param>
</params>
</methodCall>
```

Use Case – Adding a Regular Expression to a Content Filter

This example uses the config.create method described in the previous section. Using config.create you can add regular expressions to a content filter, which is a global setting. For more details about using regular expressions and content filtering, see the **BLOCK/ACCEPT > Content Filtering** page. The output of a successful call is a simple '200 OK'.

Arguments:

Regular Expression: \bvi.gra\b (see Regular Expressions)

Sample Request:

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
<methodName>config.create</methodName>
<params>
 <param>
  <value>
   <struct>
    <member>
     <name>parent_type</name>
      <value>
       <string><![CDATA[global]]></string>
      </value>
     </member>
     <member>
      <name>name</name>
       <value>
        <string><![CDATA[\\bvi.gra\\b]]></string>
       </value>
     </member>
     <member>
      <name>type</name>
       <value>
        <string>
         <![CDATA[filter_header_list]]>
        </string>
       </value>
      </member>
      <member>
       <name>parent_path</name>
        <value>
         <string></string>
        </value>
         </member>
          <member>
           <name>filter_header_list_comment</name>
           <value>
            <string><![CDATA[Filter this content]]></string>
           </value>
          </member>
          <member>
           <name>inbound_filter_header_list_action</name>
           <value>
            <string><![CDATA[Block]]></string>
           </value>
          </member>
          <member>
           <name>outbound_filter_header_list_action</name>
           <value>
            <string><![CDATA[Quarantine]]></string>
           </value>
          </member>
          <member>
           <name>apply_to_subject</name>
           <value>
            <string><![CDATA[1]]></string>
           </value>
          </member>
          <member>
```

```
      <name>apply_to_header</name>
      <value>
       <string><![CDATA[0]]></string>
      </value>
     </member>
     <member>
      <name>apply_to_body</name>
      <value>
       <string><![CDATA[1]]></string>
      </value>
     </member>
    </struct>
   </value>
```

```
      </param>
    </params>
 </methodCall>
```

Use Case – Listing Explicit Users (Valid Recipients) and Aliases at the Global Level

Supported by firmware version 5.1.3.006, 6.x and higher

Use the config.list method to list valid recipients and aliased accounts at the global level - i.e. not domain-specific. Explicit Users and aliased email accounts are added or deleted on the **ADVANCED > Explicit Users** page of the web interface. In this case, the Type, or scope, is blank (empty) to indicate global. Note that the 'variable' 'list_valid_recipient_aliases' is not actually a variable as defined in the configuration; rather, it is an indicator to the API of what is being listed by the config.list call.

Sample Request:

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
   <methodName>config.list</methodName>
    <params>
     <param>
      <value>
       <struct>
         <member>
          <name>variable</name>
            <value>
             <string>
               <![CDATA[list_valid_recipient_aliases]]>
             </string>
           </value>
         </member>
         <member>
           <name>child_type</name>
            <value>
              <string>
               <![CDATA[global]]>
             </string>
           </value>
        </member>
        <member>
         <name>path</name>
          <value>
           <string>
           </string>
         </value>
      </member>
      <member>
        <name>type</name>
           <value>
             <string>
              <![CDATA['']]>
            </string>
          </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
 </methodCall>
```

Use Case – Adding and Configuring Multiple Domains

Use the domain.add method, described in the previous section, in a loop to add multiple domains for which the Barracuda Email Security

Gateway should process email. These domains will then be listed in the **DOMAINS > Domain Manager** page of the web interface.

To configure the domains, use the config.set method for each domain. This example configures the 'Spam Score limit block level' to **4** for n domains, by setting the scana_pd_block_level variable, if you put the request in a loop. In the web interface, you'll see this value on the **BASIC > Spam Checking** page after clicking on the **Manage Domain** link for each domain.

Sample Request (for each domain):

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.set</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>scana_pd_block_level</name>
        <value>
         <i4>4</i4>
       </value>
      </member>
      <member>
       <name>path</name>
        <value>
         <string>
           <![CDATA[$domain]]>
         </string>
        </value>
      </member>
      <member>
       <name>type</name>
        <value>
         <string>
           <![CDATA[domain]]>
         </string>
        </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Use Case – Listing Valid Recipients and Aliases for a Domain

Supported by firmware version 5.1.3.006, 6.x and higher

Use the config.list method to list valid recipients and aliased accounts for a domain. Valid Recipients and aliased email accounts are added or deleted on the per-domain **USERS > Valid Recipients** page of the web interface. In this case, the Type, or scope, is ‘domain’, and this call returns a list of all valid recipients and aliased email accounts for the domain ‘mymail.net’. Note that the ‘variable’ ‘list_valid_recipient_aliases’ is not actually a variable as defined in the configuration; rather, it is an indicator to the API of what is being listed by the config.list call.

Sample Request:

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
<methodName>config.list</methodName>
<params>
<param>
  <value>
   <struct>
    <member>
     <name>variable</name>
      <value>
       <string>
        <![CDATA[list_valid_recipient_aliases]]>
        </string>
      </value>
     </member>
    <member>
     <name>child_type</name>
      <value>
       <string>
        <![CDATA[global]]>
        </string>
      </value>
     </member>
    <member>
     <name>path</name>
      <value>
       <string> <![CDATA[mymail.net]]>
       </string>
      </value>
    </member>
    <member>
     <name>type</name>
      <value>
       <string>
        <![CDATA[domain]]>
       </string>
      </value>
    </member>
   </struct>
  </value>
</param>
</params>
</methodCall>
```

Use Case – Adding Valid Recipients and Aliases for a Domain

Supported by firmware version 5.1.3.006, 6.x and higher

Use the config.set method to add valid recipients and aliases for a domain. This case adds a primary account and two email aliases for the domain ‘testqa.com'. Aliased accounts are added or deleted on the per-domain **USERS > Valid Recipients** page of the web interface and are linked to a 'primary account', which receives quarantined mail for the aliased accounts. The primary valid recipient is added first, followed by a number of aliases. See the per-domain **USERS > Valid Recipients** page of the web interface for details about alias linking.

Note that the ‘member’ name ‘new_valid_recipient_aliases’ is an indicator to the API of what is being set by the config.set call. Make sure the domain is present in the Barracuda Email Security Gateway before adding recipients and aliases.

Arguments:

- path: testqa.com
- type: domain
- child_type: global
- my $domain = "testqa.com";
- my $primary_valid_recip = 'user1@testqa.com';

- my $alias = 'user2@testqa.com'.' '.'user3@testqa.com';
- my $primary_and_alias = $primary_valid_recip." ".$alias;
- new_valid_recipient_aliases = 'user1@testqa.com user2@testqa.com user3@testqa.com';

Sample Request:

```
<?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.set</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>new_valid_recipient_aliases</name>
        <value>
         <string>$primary_and_alias</string>
        </value>
      </member>
      <member>
       <name>path</name>
        <value>
         <string>
           <![CDATA[$domain]]>
         </string>
        </value>
      </member>
      <member>
       <name>type</name>
        <value>
         <string>
            <![CDATA[domain]]>
         </string>
        </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Use Case – Deleting Aliases and Valid Recipients for a Domain

Supported by firmware version 5.1.3.006, 6.x and higher

Use the config.delete method to delete valid recipients and aliases for a domain.This example deletes the valid recipient and aliases for the domain ‘testqa.com’. Valid recipients and aliased accounts are added or deleted on the per-domain **USERS > Valid Recipients** page of the web interface. Note that the variable ‘delete_valid_recipient_aliases’ is not actually a variable as defined in the configuration; rather, it is an indicator to the API of what is being deleted by the config.delete call.

In this example, 'user2@testqa.com', 'user3@testqa.com' are the aliases to be deleted. Make sure the domain for which you are deleting aliased accounts is present in the Barracuda Email Security Gateway. The list of per-domain aliased user accounts to be deleted can be specified in the 'Values' variable in the XML request.

Arguments:

- path: testqa.com
- type: domain
- my $domain = "testqa.com";
- my $user2 = 'user2@testqa.com';
- my $user3 = 'user3@testqa.com';

Sample Request:

```
 <?xml version="1.0" encoding="UTF8"?>
<methodCall>
 <methodName>config.delete</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name>path</name>
        <value>
         <string>
           <![CDATA[$domain]]>
         </string>
        </value>
      </member>
      <member>
       <name>type</name>
        <value>
         <string>
            <![CDATA[domain]]>
         </string>
        </value>
     </member>
     <member>
       <name>variable</name>
        <value>
         <string>
            <![CDATA[delete_valid_recipient_aliases]]>
         </string>
        </value>
     </member>
     <member>
       <name>values</name>
        <value>
        <array>
        <data>
        <value>
         <string>
            <![CDATA[$user2]]>
         </string>
        </value>
        <value>
         <string>
            <![CDATA[$user3]]>
         </string>
        </value>
        </data>
        </array>
        </value>
     </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Error Codes

See the **Error Response** format under the **XML-RPC Request** and **Response** for an example of how the faultCodes (error codes), shown below, will be returned with the XML response.

Error (Fault) Codes

Fault Code	Description	Example Fault Strings
400	Required arguments are missing	Too few arguments: <error message>
401	Machine does not have access rights	Your machine does not have access rights to administer...
402	Domain name error	• Domain <domain name> already exists • Domain <domain name> is not a valid domain
403	Access error	Access denied <error message>
406	API was called with incorrect parameters	Incorrect parameters for API call
411	Account error	User account does not exist
412	Account error	User account already exists
421	Account error	Unable to validate account
425	Input object or variable is not valid	• Config: Error: Invalid variable: <variable name used in api> Config: Error: variable <variable name used in api> not recognized • Config: Error: Invalid object type: <variable name used in api> • Config: Error: <variable name used in api> is not tied to <parent type> • Config: Error: <variable name used in api> does not belong to any class • Config: Error: <variable name used in api> does not belong to <parent type> • Config: Error: <variable name used in api> is not of type <parent type>
426	Invalid operation	• Config: Error: invalid operation for variable <variable name used in api> • Config: Error: Cannot add values to tied variable <variable name used in api> • Config: Error: Cannot remove values from tied variable <variable name used in api>
427	The object does not exist in the database	• Config: Error: Could not find tied object: <parent type>, <parent path> [<parent type>] • Config: Error: Could not find scoped object: <parent type>, <parent path> [global] • Config: Error: Could not find scoped object: <parent type>, <parent path> [<old parent type>, <old parent path>]
428	Input value being set is not valid	Config: Error: Could not find values to delete in <parent path>: <list of invalid values>

429	Required variable is missing	Variable required to create object of type <parent type>
450	The method you used is unknown	Unknown method called <API method>
499	Unknown error	An unknown error has occurred
500	Unknown error	An unknown error has occurred