Barracuda Web Security Gateway API Guide

https://campus.barracuda.com/doc/9011969/

IT administrators can easily manage large blocks of usernames, create local or IP groups, and configure most global settings using the Barracuda Web Security Gateway APIs. The APIs allow remote administration to set single variables and to simplify data-intensive tasks such as:

- Quickly add, update, list or delete usernames and passwords in bulk
- Create IP Subnet/Groups
- Assign users to groups
- Get and set single global variables

This guide includes examples of the XML-RPC code to execute various tasks, along with example Perl scripts. Any API call requires a password that you securely configure on the Barracuda Web Security Gateway **BASIC > Administration** page when logged in as the administrator.

The Barracuda APIs allow remote administration and configuration of the Barracuda Web Security Gateway version 7.x and higher. Note that all API calls must use HTTP/1.1 or later.

How the Barracuda API Works

The framework of the API allows a programmer to get or set variables inside an XML-RPC request corresponding to field values in the configuration database of the Barracuda Web Security Gateway. Some languages, Perl is one example, provide wrappers for XML-RPC requests, providing an interface to form the request.

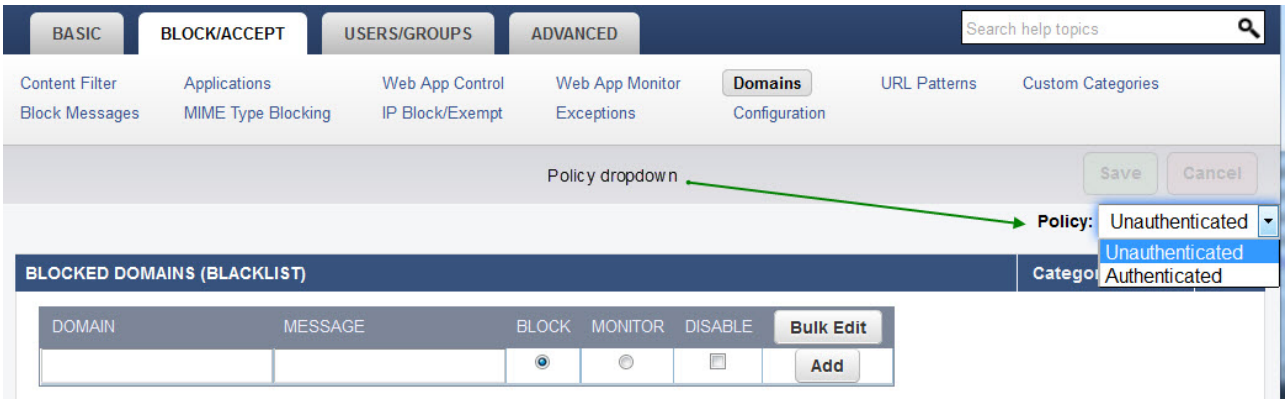
What Can Be Configured With the APIs

The APIs work through manipulation of variables inside the system configuration database. Variables that meet the following criteria can be manipulated by these APIs:

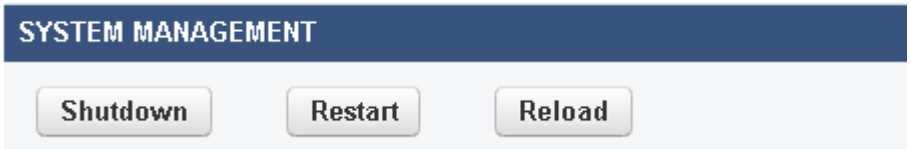
- All global variables with a simple setting that are not policy-related. This includes most settings you can set by clicking the **Save** button in the Barracuda Web Security Gateway web interface. For example, from the **BASIC > IP Configuration** page, you can enable or disable **Virus Protection** for the Barracuda Web Security Gateway and then click the **Save Changes** button:

What Cannot be Configured With the APIs

- Any variables on any page on the **BLOCK/ACCEPT** tab with the **Policy** dropdown at the top:



- Variables with a list of associated values; for example, you cannot use an api to create a custom category and add a list of related domains.
- Deleting any policy or configuration which is part of a list. For example: exceptions, custom categories.
- Most things that correspond to “action” buttons in the web interface. For example, from the **BASIC > Administration** page, you can click a button to restart the system or shut it down, but you cannot execute these “actions” via the APIs. An exception to this is the **Reload** feature/button, which has an API that re-applies the system configuration.



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 Web Security Gateway web interface. *Be sure you enter the IP address(es) where you will access the APIs in this section of the web interface before using the APIs.* Attempts to call these APIs from any IP address not listed as an allowed API IP address are denied. All calls to the APIs require you to use the API password, set on this same page and section of the web interface.

Integration with External Systems and Services - Security Considerations

The Barracuda Web Security Gateway integrates with other systems and services in your environment, like your LDAP server, FTP and SMB servers, for example. Barracuda recommends creating separate service accounts for these integration points, rather than personal accounts, and then using the principle of least privilege. This integration strategy is part of an overall security policy. For more information, see Security for Integrating with Other Systems - Best Practices.

Barracuda also strongly recommends sending API calls over HTTPS to protect passwords and other data.

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. So requests for all actions must be in the form of an HTTP POST request. All actions roll into one CGI script (for example: api.cgi) and map to an XML-RPC method, with those parameters needed for the action to complete.

For example, the **get** action maps to the `config.get` XML-RPC method and all parameters needed for the **get** are sent in the XML body. The Perl module XML::RPC (note that this is not a part of the standard Perl distribution) is used by api.cgi to retrieve the requested method and parameters. Then the action is performed and the response is sent back to the client. When there is an error, a response complying with the fault response of the XML-RPC specification is sent (see examples below). The error response contains both a fault code and a meaningful fault string. See Appendix 1 of this guide 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 possible XML output are shown below, both for a successful request and for a request that returns an error. The single-value request / response involves a single variable value. Responses containing multiple values send the values back as an XML-RPC array.

To make the request, use the base URL of the Barracuda Web Security Gateway 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

Typical parameters used to build the request 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 'alerts_email_address' represents the global **System Alerts Email Address**, set on the **BASIC > Administration** page in the web interface. To get or set this variable value, put 'alerts_email_address' in the XML request body specified as a variable:

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

- **password** :: A required parameter used to authenticate access to a page and set by the administrator on the **BASIC > Administration** page in the **API Password** field. For example:
  # API Password
  my $password = "1234";
  my $url = " http://$cuda_ip:8000/cgi-mod/api.cgi?password=$password ";

  See the contents of 'my $url' in the Perl example under *How to get the current value of a global variable* below, which uses a password of '1234'.

- **type** :: A parameter that specifies the class/scope of a variable. The "scope" of variables you can set with these APIs is always 'global'.

Success Responses

The output of a successful call where no variable is being returned is a simple '200 OK' as shown below. Otherwise, successful responses with returned values are shown with each example.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <methodResponse>
   <params>
    <param>
     <value>
      <struct>
       <member>
       <name>Result</name>
       <value><string><![CDATA[200: OK]]></string></value>
       </member>
       </value>
       </struct>
       </value>
     </param>
   </params>
 </methodResponse>
```

Error Responses

Error responses use the XML-RPC faultCode and faultString formats. The error code is the value of the faultCode member and the error string is the value of the faultString member. See Appendix 1 for a list of faultCodes and descriptions of possible errors. Here is an example of an error response, showing the XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <methodResponse>
   <fault>
    <value>
     <struct>
       <member>
       <name>faultCode</name>
       <value><i4><500></i4></value>
       </member>
       <member>
        <name>faultString</name>
       <value>
       <string>No such variable in configuration</string>
       </value>
```

```
    </member>
   </struct>
    </value>
   </fault>
  </methodResponse>
```

How to List 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. The **config.varlist** is a utility that provides information on scope of configuration variables to help you understand how to access and use them. Calling this method prior to using the other APIs will provide a good reference of the configuration variables.

Config.varlist

There are no arguments for this API.

Sample Request:

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

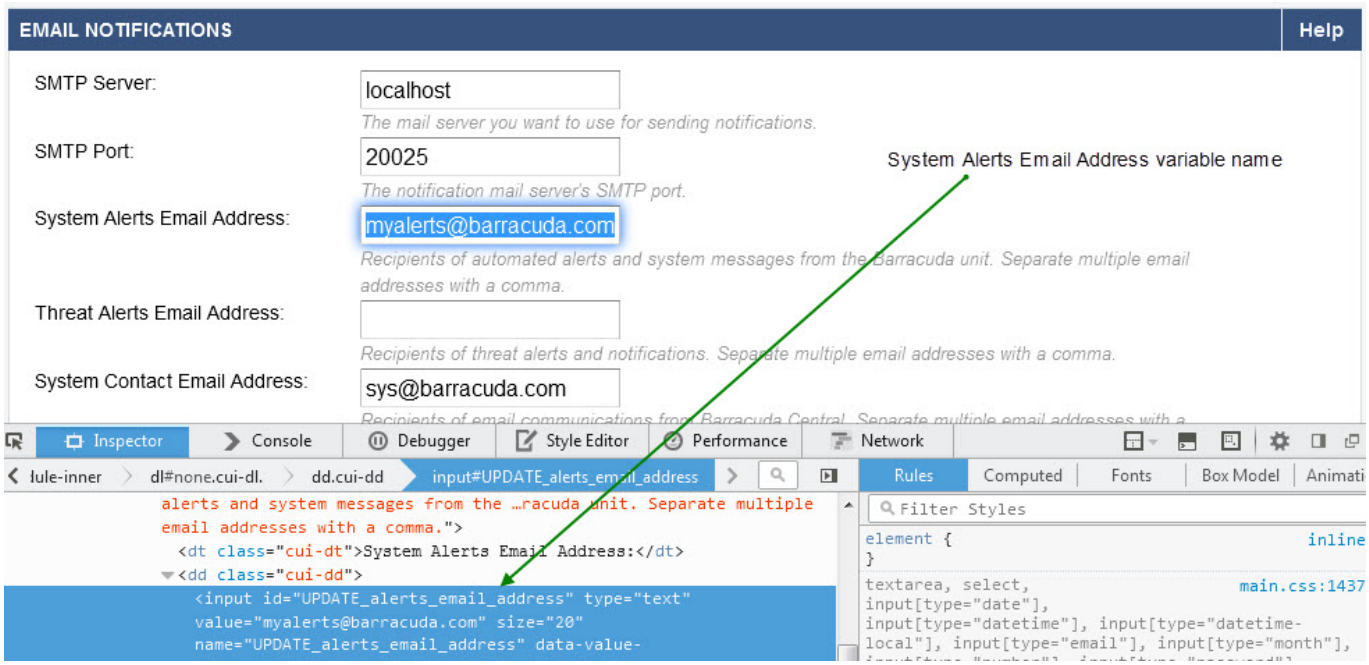
Perl code for this example:

```
use strict;
use warnings;

use XML::RPC;

# IP Address of your Barracuda Web Security Gateway
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.varlist',
                  {
     });

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

How to Access Variables in the Configuration

To determine the name of the variable you want to configure, log into the Barracuda Web 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.



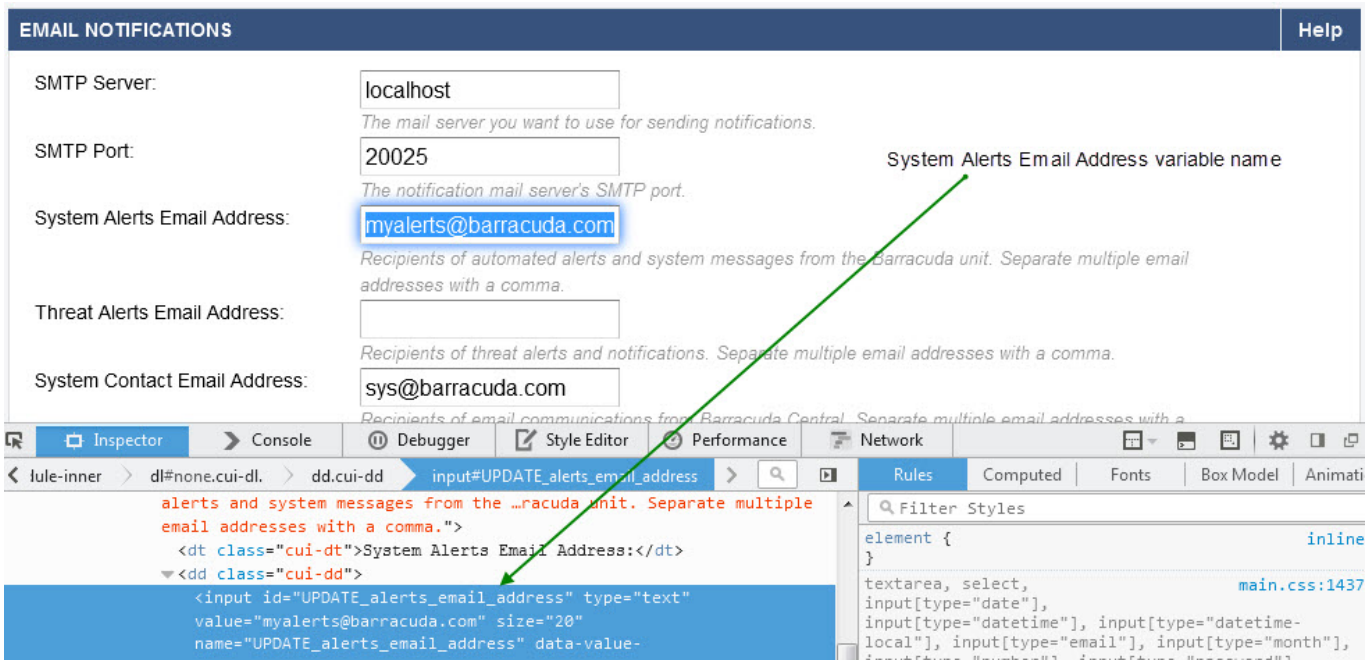
How to get the current value of a global variable

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

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 Security Gateway
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 Security Gateway
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:**

```
--- RESPONSE ---<?xml version="1.0" encoding="UTF-8" ?>
<methodResponse>
<params>
<param>
<value>
<string><![CDATA[myalerts@barracuda.com]]></string>
</value>
</param>
</params>
</methodResponse>
```

How to set the value for a single variable

Use the `config.set` method to set a value for a single variable. This example sets the **Session Expiration Length**, which specifies the elapsed time allowed before a user login expires and re-authentication is required. Minimum setting for this value is ***1*** minute. This variable is set on the **BASIC > Administration** page.

The variable name can be changed to make other configuration changes. In this example, the **Session Expiration Length** is set to 30 minutes.

Arguments

- **type** : 'global'
- **variable** : http_session_length => '30'

XML code for this example
Note that, with the `config.set` method, the <name> tag indicates the name of the single *variable* in the configuration. The <value> tag indicates that the value of that variable is an integer, and explicitly sets that value to '30' as the input.

```
<?xml version="1.0" encoding="UTF8"?>
```

```
<methodCall>
 <methodName>config.set</methodName>
  <params>
   <param>
    <value>
     <struct>
      <member>
       <name> http_session_length </name>
       <value><i4><![CDATA[30]]></i4>
       </value>
      </member>
       <member>
        <name>type</name>
         <value>
            <string> <![CDATA[global]]></string>
         </value>
       </member>
     </struct>
    </value>
   </param>
  </params>
</methodCall>
```

Perl code for this example:

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

# IP Address of your Barracuda
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 = $xmlrpc->call('config.set',

{ type => 'global', http_session_length => '30', }

);
print $xmlrpc->xml_in();
```

XML response returned by Perl script:

Here is the XML response returned after running the above Perl script indicating success.

```
--- RESPONSE ---<?xml version="1.0" encoding="UTF-8" ?>
<methodResponse>
<params>
<param>
<value>
<struct>
<member>
<name>Result</name>
<value>
<string><![CDATA[200: OK]]></string>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

How to set values for several global variables

This example modifies multiple global variables using the `config.set` method, setting the **Web Interface HTTP Port** (http_port) to 8000 and **Session Expiration Length** (http_session_length) to 20 (minutes). These variables are set on the **BASIC > Administration** page. To set several variables at once, simply list the variable names and values to set, separated by commas, as shown in the **variable list:**

Arguments:

- **type**: 'global'
- **variable list**: http_session_length => '20', http_port => '8000'

Perl code for this example:

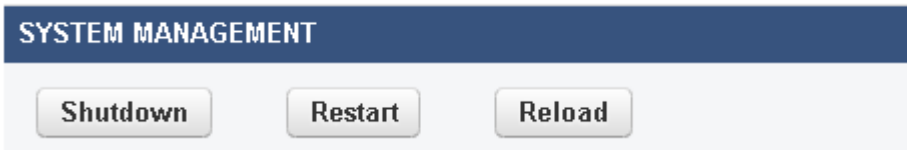
```
use strict;
use warnings;
use XML::RPC;
```

```
# IP Address of your Barracuda
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 = $xmlrpc->call('config.set',

{ type => 'global', http_session_length => '100', http_port => '8000', }

);
print $xmlrpc->xml_in();
```

Use Cases

Reloading the configuration

Use the `config.reload` method to re-apply the system configuration, corresponding to the **Reload** button on the **BASIC > Administration** page of the web interface.



Perl code for this example:

```
use strict;
use warnings;

use XML::RPC;

# IP Address of your Barracuda Web Security Gateway
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.reload',
                  {
      });

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

Response indicating success:

```
--- RESPONSE ---<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
<params>
<param>

<value>
<struct>
<member>
<name>Result</name>
<value>
<string><![CDATA[200: OK]]></string>
</value>
</member>
</struct>
</value>
</param>
</methodResponse>
```

Managing user accounts

These APIs allow the following:

- Create users
- Remove users
- Update users (change password, etc.)

Note that the `user.create` , `user.update` and `user.remove` methods do not require the **type** parameter.The output of a successful call is simply '200 OK'.

Create a local user

This example creates the user 'xyzuser' with a password of 'BWFpwd' and assigns the user to the local group 'Students', as configured on the **BLOCK/ACCEPT > New Users** page. The **Force Password Change On Next Signon** option, represented by the 'force_password_change' variable, is left out in this example since it is optional, and the default is **No**.

Arguments:

- **user:** xyzuser
- **password:** BWFPwd
- **groups:** Students

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

# IP Address of your Barracuda Web Security Gateway
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 = $xmlrpc->call('user.create',

{ user => 'xyzuser', password => 'BWFPwd', groups => 'Students', change => 'No', }

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

Remove a local user

This example removes the user 'xyzuser'.

Arguments:

- **user:** xyzuser

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

# IP Address of your Barracuda Web Security Gateway
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 = $xmlrpc->call('user.remove',

{ user => 'xyzuser', }

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

Update a local user

This example updates the password for user “xyzuser” and adds the user to two groups, 'Faculty' and 'Staff'.

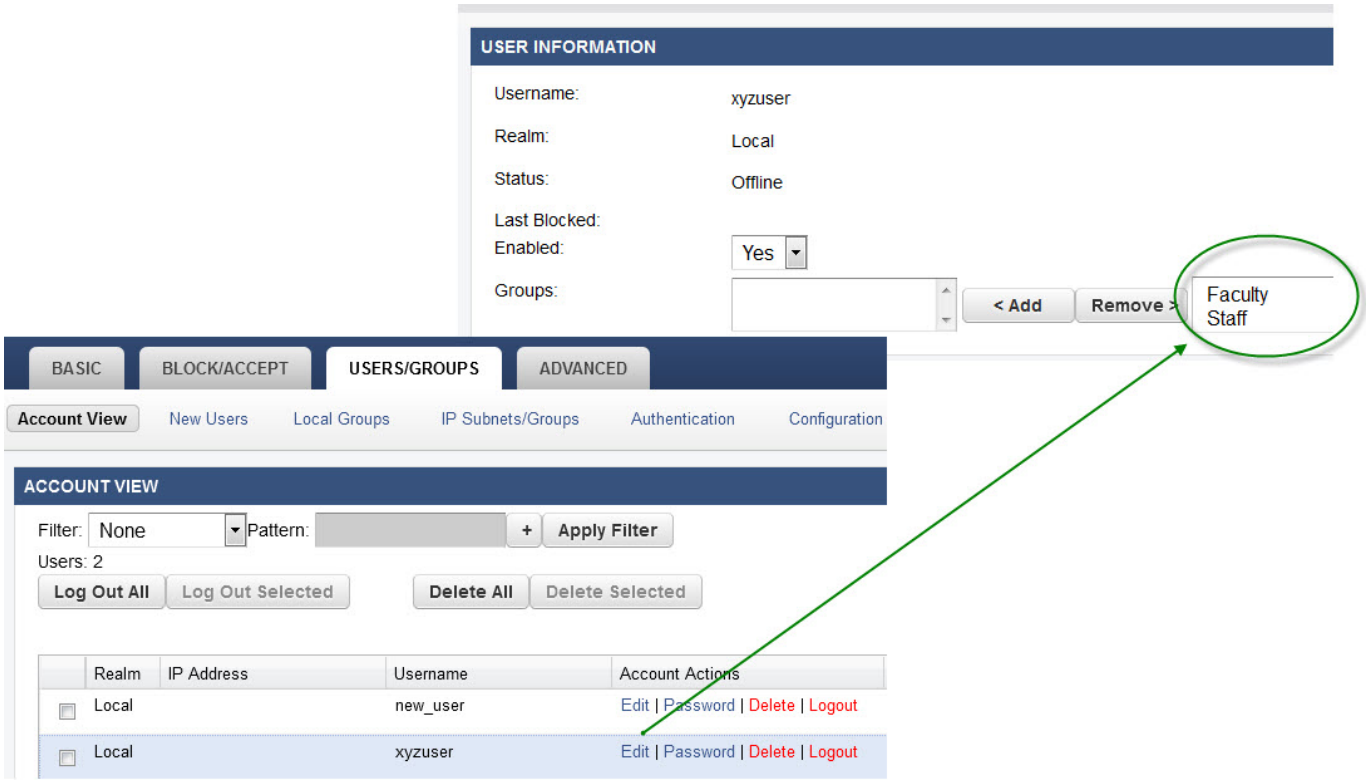
Arguments:

- **user:** xyzuser
- **password:** BWFPwd
- **groups:** Faculty\nStaff

```
use strict;
```

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

# IP Address of your Barracuda Web Security Gateway
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 = $xmlrpc->call('user.update',

{ user => 'xyzuser', password => 'BWFPwd', groups => "Faculty\nStaff", change => 'No', }

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

The following results display in the Barracuda Web Security Gateway web interface on the **USERS/GROUPS > Account View** page by clicking **Edit** for user 'xyzuser'. The **USER INFORMATION** popup shows the associated groups you added for 'xyzuser':



List all user accounts

The `user.list` method simply lists all user accounts currently on the system, as displayed on the **USERS/GROUPS > Account View** page.

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

# IP Address of your Barracuda Web Security Gateway
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 ('user.list',
{
   });
# show the response from the Barracuda Web Security Gateway
```

```
print "--- RESPONSE ---";print $xmlrpc->xml_in();
# END
```

Response

The successful response lists the two configured user accounts, 'new_user' and 'xyz_user'.

```
<?xml version="1.0" encoding="UTF-8" ?>
  <methodResponse>
   <params>
    <param>
     <value>
      <array>
       <data>
       <value>
       <string><![CDATA[new_user]]></string>
       </value>
       <value>
       <string><![CDATA[xyz_user]]></string>
       </value>
       </data>
       </array>
     < /value>
     </param>
   </params>
 </methodResponse>
```

Creating a New IP Subnet/Group

This example creates a new IP Subnet/Group called **facilities** with an IP address of 10.20.30.0 and a netmask of 255.255.255.0. This setting is configured on the **USERS/GROUPS > IP Subnets/Groups** page. The most common reason to create an IP group is to apply an exception policy to multiple users on the same IP network. Note that remote users whose web traffic is filtered via the Barracuda Web Security Agent (WSA) cannot be included in these groups.

This API is a bit more complex, with additional parameters used to build the request since this is an application of 'tied variables'. These are variables that are dependent upon, or "tied to" a key variable. In this example, the two variables `LDAP_groups_IP_netmask` and `LDAP_groups_IP_comment` are dependent upon the `LDAP_groups_IP_address`.

Arguments: The following arguments are used by the `config.create` method:

- **parent_type** :: A required parameter that tells the API about the class/scope of the parent container. In this case, the scope is 'global'.

- **parent_path** :: A required parameter that is the qualified name of a parent object under which a new object will be created. In this case, this variable is left blank.
- **type** :: A required parameter that specifies the key variable that the other variables are tied to.
- **name** :: A required parameter that specifies the explicit value of the key variable.
- **variable list** :: An optional parameter that tells the API which variable(s) to set, including explicit values.

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

# IP Address of your Barracuda Web Security Gateway
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.create',

{ parent_type=>'global',
parent_path => '',
name => '10.20.30.0',
   type => 'LDAP_groups_IP_address',
        LDAP_groups_IP_netmask => '255.255.255.0',
        LDAP_groups_IP_comment => 'faciliies',
   });
# show the response from the Barracuda Web Security Gateway
print "--- RESPONSE ---";
print $xmlrpc->xml_in();
# END
```

Response indicating success:

```
- -- RESPONSE ---<?xml version="1.0" encoding="UTF-8" ?>
<methodResponse>
<params>
```

```
<param>
<value>
<struct>
<member>
<name>Result</name>
<value>
<string><![CDATA[200: OK]]></string>
</value>
</member>
</struct>
</value>
</param>
</params>
</methodResponse>
```

Appendix 1

See the **Error Response** format under The XML-RPC Request and Response above for an example of how the faultCodes (error codes), shown below, are 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

Figures

1. Virus Scanning page.png
2. PolicyDropDownonPage.jpg
3. System Management.png
4. InspectElementExample.jpg
5. InspectElementExample.jpg
6. SessExpLen.jpg
7. System Management.png
8. UserFacultyStudents.jpg