API USERS GUIDE V6.01

13TH MARCH 2014

COPYRIGHT

Printed in Ireland.

SPAMTITAN WELCOMES YOUR COMMENTS

We want to know about any corrections or clarifications that you would find useful in our documentation, which will help us improve future versions. Include the following information:

- Version of the manual that you are using
- Section and page number
- Your suggestions about the manual

Send your comments and suggestions to us at the following email address: info@SpamTitan.com

CONTENTS

INTRODUCTION

SpamTitan exposes some of its functionality via an Application Programming Interface (API). This document is a reference for that functionality, and aims to serve as a reference for developers building tools that integrate with SpamTitan. The API currently supports a RESTful interface, that essentially means you can send an HTTP GET or POST to call exposed methods, and you'll get back an XML document in return.

When you send a request, you'll get a response in XML that looks like this:

```
<Response stat="ok" code="200">
        [Data in XML format - see individual method docs]
</Response>
```

If there's an error, SpamTitan will respond with an error message about the problem.

```
<Response code="404" stat="fail">
        <error>Error Message</error>
</Response>
```

AUTHENTICATION

Access to the SpamTitan API is limited to IPs on a trusted IP list configured on the **Settings->Access/Authentication** page, under the “API Allowed Hosts” section. Any attempt to utilize the APIs from any IP address that is not on that list will be denied.

PARAMETERS

Some API methods take optional or requisite parameters. Where applicable, we’ve documented those parameters. Unknown parameters will be silently ignored.

1. DOMAINS

DOMAIN EXISTS

Tests if the specified domain is been relayed by SpamTitan

URL: http://10.1.1.82/domain/exists?name=example.com

Parameters:

- name: Required. The name of the Domain to add

Response: If the domain exists:

```
<Response code="200" stat="ok">
        <success>Domain exists</success>
</Response>
```

If the domain does not exist or the request could not be satisfied then you will get one of the following error codes:

- 501: Insufficient rights
- 502: Required parameter not provided
- 405: Domain not found

For example:

```
<Response code="API_DOMAIN_NOTFOUND" stat="fail">
        <error>Domain not found</error>
</Response>
```

ADD DOMAIN

Adds the specified domain to the current list of domains relayed by SpamTitan to the specified destination server.

URL: http://10.1.1.82/domain/add?name=example.com&server=1.2.3.4

Parameters:

- name: Required. The name of the Domain to add. MX Lookup is automatically enabled for new domains.
- server: Required. The destination mail server for this domain. To specify a non-default port (25) append ":8025".
    - To specify a fallback server enter a list of IP addresses/FQDN's separated by spaces. SpamTitan will attempt delivery in the order listed. E.g.: "server=1.2.3.4 5.6.2.3"
    - Specify a comma (,) separated list of IP addresses and/or FQDN's. In this case SpamTitan will attempt delivery in a round-robin fashion between the listed servers. E.g.:"server=1.2.3.4,1.2.3.5"
- domaingroup: Optional. The Domain Group to add this domain to.

Response: If the domain is successfully added then you will receive a 200 success code:

```
<Response code="200" stat="ok">
        <success>Domain example.com added</success>
</Response>
```

If the domain already exists then the call will fail.

Possible failure codes include:

- 501: Insufficient rights
- 502: Required parameter not provided

  404: Domain exists

```
<Response code="404" stat="fail">
        <error>Cannot add domain: Already exists</error>
</Response>
```

SHOW DOMAIN

Returns extended information of a given domain, specified by domain name as per the required name parameter below.

URL: http://10.1.1.82/domain/show?name=example.com

Parameters:

- name: Required. The name of the Domain to add

Response: If the domain exists then you will receive a 200 success code along with the extended information regarding the specified domain:

```
<Response code="200" stat="ok">
        <Data>
                <Domain>abc.com</Domain>
                <Destination_Server>1.2.3.4:25</Destination_Server>
                <Recipient_Verification>None</Recipient_Verification>
                <Domain_Group >dg1</Domain_Group >
        </Data>
</Response>
```

If the domain does not exist then the call will fail. Possible failure codes include:

- 501: Insufficient rights
- 502: Required parameter not provided
- 405: Domain does not exist

EDIT DOMAIN

Modify the settings for a particular domain

URL: http://10.1.1.82/domain/edit?name=example.com&server=10.0.0.101

Parameters:

- name: Required. The name of the Domain to add
- server: Optional. The new destination server IP address or FQDN for this domain
- rv: Optional. The recipient verification method to use for this server. Possible values are: "none", "dynamic", "ldap", "list"
- dyn_server: Optional. The recipient verification server to use if the recipient verification is "dynamic". This setting will have no effect if the recipient verification (rv) setting is not set to "dynamic"
- ldap_server: Required.
- ldap_port: Required.
- ldap_search_dn: Optional.
- ldap_password: Optional.
- ldap_filter: Optional.
- ldap_searchbase: Optional.
- ldap_result_attribute: Optional. Defaults to 'mail'.
- email: Optional. If list based recipient verification is been used, then use this setting to add email addresses to the list. The API call needs to be called once for each email address added.
- email / delemail: add or remove entry from list based recipient verification. View page 15 for more information.
- domaingroup: Domain Group to add domain to.

Response: If the domain exists and was successfully updated, then you will receive a 200 success code along with the extended information regarding the specified domain:

```
<Response code="200" stat="ok">
        <data>
                <Domain>foobar.com</Domain>
                <Destination_Server>1.3.4.5:25</Destination_Server>
                <Destination_Port>25</Destination_Port>
                <Recipient_Verification>None</Recipient_Verification>
                <Domain_Group />
        </data>
</Response>
```

LIST DOMAINS

List all the domains relayed by SpamTitan

URL: http://10.1.1.82/domain/list

Parameters:

- server: Optional. List Domain's based on the IP or FQDN.

Response: Sample output:

```
$ wget -q -O – "http://10.1.1.82/domain/list?server=1.2.3.4"
```

```
<Response code="200" stat="ok">
  <Total>7</Total>
  <Domains>
    <Domain>test.com</Domain>
    <Domain>test31.com</Domain>
    <Domain>test30.com</Domain>
    <Domain>test32.com</Domain>
    <Domain>test33.com</Domain>
    <Domain>test3.com</Domain>
    <Domain>test2.com</Domain>
  </Domains>
</Response>
```

DELETE DOMAIN

Delete a domain. Note that this will delete all user policies associated with this domain.

URL: http://10.1.1.82/domain/delete?name=abc123.com

Parameters:

- name: Required. The name of the domain to delete.

Example: Delete policy abc123.com.

```
$ wget -q -O – " http://10.1.1.82/domain/delete?name=abc123.com"
```

```
<Response code="200" stat="ok">
        <success>Domain abc125.com deleted</success>
</Response>
```

ADD / REMOVE DOMAIN ADMIN

Add a domain admin to a domain.

URL: http://10.1.1.82/domain/edit?name=abc123.com&domainadmin=jd55@test.com

Parameters:

- name: Required. The name of the domain to add the new admin.
- Domainadmin: Required. The email address of the admin to add.
- admindel: add this cmd to end of string "&admindel" to delete specified admin.

Example 1: Add domain admin.

```
$ wget -q -O –
"http://10.1.1.82/domain/edit?name=abc123.com&domainadmin=jd55@test.com"
<Response code="200" stat="ok">
        <Data>
                <Domain>abc123.com</Domain>
                <Destination_Server>1.2.3.4</Destination_Server>
                <Recipient_Verification>None</Recipient_Verification>
                <Domain_Administrators>
                        <Allowed>jd55@test.com</Allowed>>
                </Domain_Administrators>
        </Data>
</Response>
```

Example 2: Delete domain admin.

```
$ wget -q -O –
"http://10.1.1.82/domain/edit?name=abc123.com&domainadmin=jd55@test.com&admindel"
<Response code="200" stat="ok">
        <Data>
                <Domain>abc123.com</Domain>
                <Destination_Server>1.2.3.4</Destination_Server>
                <Recipient_Verification>None</Recipient_Verification>
                <Domain_Administrators/>
        </Data>
</Response>
```

LIST DOMAIN ADMINS

List all domain admins for a domain.

URL: http://10.1.1.82/domain/list_admins?name=abc123.com

Parameters:

- name: Required. The name of the domain to add the new admin.
- domainadmin: Required. The email address of the admin to add.

Example:

```
$ wget -q -O – " http://10.1.1.82/domain/list_admins?name=abc123.com "
```

```
<Response code="200" stat="ok">
<Total>2</Total>
<abc123.com>
        <Domain_Admin>jd65@test.com</Domain_Admin>
        <Domain_Admin>jd55@test.com</Domain_Admin>
</abc123.com>
</Response>
```

CHECK FOR DOMAIN ADMIN

Search using email address to check if the user is an domain admin.

URL: http://10.1.1.82/domain/admin_check?name=abc123.com&adminemail=jd55@test.com

Parameters:

- Name: Required. The name of the domain to check.
- adminemail: Required. The email address of the admin to check.

Example:

```
$ wget -q -O – "
http://10.1.1.82/domain/admin_check?name=abc123.com&adminemail=jd55@test.com"
```

```
<Response code="200" stat="ok">
        <success>jd55@test.com is a domain admin</success>
</Response>
```

LIST WEB AUTHENTICATION

Show web authentication in use for a domain.

URL: http://10.1.1.82/domain/list_auth?name=abc123.com

Parameters:

- name: Required. The name of the domain to check.

Example:

```
$ wget -q -O – " http://10.1.1.82/domain/list_auth?name=abc123.com"
```

```
<Response code="200" stat="ok">
        <Domain>
                <auth>internal</auth>
        </Domain>
</Response>
```

SET WEB AUTHENTICATION

Set Web Authentication for a domain.

URL: http://10.1.1.82/domain/auth?name=abc123.com&webauth=internal

Parameters:

- name: Required. The name of the domain to set web auth.
- webauth: Required. The type of webauth. Internal, ldap, sql, pop3 and imap. Each has their own parameters.
- **Internal**:
  - none
- **LDAP**:
  - ldap_server: Required. The IP address of the LDAP server.
  - ldap_port: Required. The port of the LDAP host.
  - ldap_search_dn: Optional.
  - ldap_passwd: Optional.
  - ldap_filter: Optional.
  - ldap_searchbase: Optional.
- **SQL**
  - sql_database: Required.
  - sql_server: Required.
  - sql_port: Required.
  - sql_username: Optional.
  - sql_dbname: Optional.
  - sql_password: Optional.
  - sql_table: Optional.
  - sql_emailcolumn: Optional.
  - sql_passwordcolumn: Optional.
  - sql_passwordtype: Optional. Options: 'plaintext', 'md5', 'crypt'.
- **IMAP**
  - imap_server: Required.
  - imap_port: Required.
  - imap_auth: Optional. Options: 0,1
  - imap_type: Optional. Options: user = 1, user@domain = 0
- **POP3**
  - pop3_server: Required.

- pop3_port: Required.
- pop3_auth: Optional. Options: 0, 1
- pop_type: Optional. Options: user = 1, user@domain = 0

Example 1: Setting LDAP web authentication.

```
$ wget -q -O - "http://10.1.1.82/domain/auth?name=abc123.com
&webauth=ldap&ldap_server=1.2.3.4&ldap_port=80&ldap_search_dn=users&ldap_passwd=
pass1&ldap_filter=toto&ldap_searchbase=string"
<Response code="200" stat="ok">
        <Success>Web Authentication set to LDAP for abc123.com</Success>
</Response>
```

Example 2: Setting POP3 web authentication.

```
$ wget -q -O - "http://10.1.1.82/domain/auth?name=abc123.com
&webauth=pop3&pop3_server=1.2.3.4&pop3_port=82&pop3_type=1&pop3_auth=0"
<Response code="200" stat="ok">
        <Success>Web Authentication set to POP3 for abc123.com</Success>
</Response>
```

If the domain does not exist or the modification could not be performed, then the call will fail. Possible failure codes include:

- 501: Insufficient rights
- 502: Required parameter not provided
- 405: Domain does not exist

SHOW RECIPIENT VERIFICATION FOR A DOMAIN (LIST ONLY)

List all emails that are allowed to receive mail. This method will only work for internal list based RV.

URL: http://10.1.1.82/domain/list_rv?name=abc123.com

Parameters:

- Name: Required. The name of the domain to check.

Example:

```
$ wget -q -O – "http://10.1.1.82/domain/list_rv?name=abc123.com"
<Response code="200" stat="ok">
<Total>2</Total>
<Allowed>
        <auth>one@abc123.com OK</auth>
        <auth>two@abc123.com OK</auth>
</Allowed>
 </Response>
```

ADD / REMOVE USER TO DOMAIN RECIPIENT VERIFICATION (LIST ONLY)

Add or Remove email to list based Recipient Verification. The domain and email must have the same domain name.

URL: http://10.1.1.82/domain/edit?name=abc123.com&email=jd@abc123.com

Parameters:

- name: Required. The name of the domain to check.
- email: Required. The email of the users to add to list RV.
- delemail: Optional. The email to remove from list RV.

Example 1: Adding an email.

```
$ wget -q -O – "http://10.1.1.82/domain/edit?name=abc123.com&email=jd@abc123.com"
```

```
<Response code="200" stat="ok">
<Data>
<Domain>abc123.com</Domain>
<Destination_Server>1.2.3.4</Destination_Server>
<Recipient_Verification>List</Recipient_Verification>
<Recipients>
        <Allowed>one@abc123.com</Allowed>
        <Allowed>two@abc123.com</Allowed>
        <Allowed>jd@abc123.com</Allowed>
</Recipients>
</Data>
 </Response>
```

Example: Deleting an email.

```
$ wget -q -O –
"http://10.1.1.82/domain/edit?name=abc123.com&delemail=jd@abc123.com"
```

```
<Response code="200" stat="ok">
<Data>
<Domain>abc123.com</Domain>
<Destination_Server>1.2.3.4</Destination_Server>
<Recipient_Verification>List</Recipient_Verification>
<Recipients>
        <Allowed>one@abc123.com</Allowed>
        <Allowed>two@abc123.com</Allowed>
</Recipients>
</Data>
 </Response>
```

2. POLICY METHODS

POLICY EXISTS

Returns if the specified email address or domain policy exists

URL: http://10.1.1.82/policy/exists?user=sean@example.com

Parameters:

- user: Required. The email address or domain of the policy to check for

Example: Check if there exists a policy for jack@abc.com

```
$ wget -q -O – "http://10.1.1.82/policy/exists?user=jack@abc.com"
```

```
<Response code="200" stat="ok">
        <success>Policy found for jack@abc.com</success>
</Response>
```

Response Codes:

- 200: Operation successful; Policy exists
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
  601: Policy not found

SHOW POLICY

Return detailed information about the specified policy.

URL: http://10.1.1.82/policy/show?user=sean@example.com

Parameters:

• user: Required. The email address or domain of the policy to display. NOTE: If no policy exists for the specified user then a 601 code will be returned. However, even if no specific policy exists for a particular email address it will automatically inherit the policy of its parent domain.

Example: Show policy for jack@abc.com

```
$ wget -q -O – “http://10.1.1.82/policy/show?user=jack@abc.com”
```

```
<Response code="200" stat="ok">
        <Policy>
                <id>7</id>
                <policy_name>jack@abc.com</policy_name>
                <virus_lover>N</virus_lover>
                <spam_lover>N</spam_lover>
                <banned_files_lover>N</banned_files_lover>
                <bad_header_lover>N</bad_header_lover>
                <bypass_virus_checks>N</bypass_virus_checks>
                <bypass_spam_checks>N</bypass_spam_checks>
                <bypass_banned_checks>Y</bypass_banned_checks>
                <bypass_header_checks>N</bypass_header_checks>
                <spam_modifies_subj>N</spam_modifies_subj>
                <spam_subject_tag2 />
                <spam_quarantine_to>spam-quarantine</spam_quarantine_to>
                <virus_quarantine_to>virus-quarantine</virus_quarantine_to>
                <banned_quarantine_to>banned-quarantine</banned_quarantine_to>
                <bad_header_quarantine_to />
                <spam_tag_level>-999</spam_tag_level>
                <spam_tag2_level>5</spam_tag2_level>
                <spam_kill_level>5</spam_kill_level>
                <locked>N</locked>
                <digest>N</digest>
                <report_type>N</report_type>
                <report_kill_level>999</report_kill_level>
                <spam_dsn_cutoff_level>0</spam_dsn_cutoff_level>
                <spam_quarantine_cutoff_level>999</spam_quarantine_cutoff_level>
                <digest_language>en_US</digest_language>
        </Policy>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 601: Policy not found

ADD POLICY

Add user policy. The policy which is added will be the same policy as the domain policy. It can subsequently be modified using the edit API.

URL: http://10.1.1.82/policy/add?user=joe@example.com

Parameters:

- user: Required. The email address of the policy to add

Response:

If the policy is successfully added, then you will receive a 200 success code along with the extended information regarding the specified policy:

```
$ wget -q -O – “http://10.42.0.170/policy/add?user=sean@abc.com”
```

```
<Response code="200" stat="ok">
        <Policy>
                <id>4</id>
                <policy_name>sean@abc.com</policy_name>
                <virus_lover>N</virus_lover>
                <spam_lover>N</spam_lover>
                <banned_files_lover>N</banned_files_lover>
                <bad_header_lover>N</bad_header_lover>
                <bypass_virus_checks>N</bypass_virus_checks>
                <bypass_spam_checks>N</bypass_spam_checks>
                <bypass_banned_checks>Y</bypass_banned_checks>
                <bypass_header_checks>N</bypass_header_checks>
                <spam_modifies_subj>N</spam_modifies_subj>
                <spam_subject_tag2 />
                <spam_quarantine_to>spam-quarantine</spam_quarantine_to>
                <virus_quarantine_to>virus-quarantine</virus_quarantine_to>
                <banned_quarantine_to>banned-quarantine</banned_quarantine_to>
                <bad_header_quarantine_to />
                <spam_tag_level>-999</spam_tag_level>
                <spam_tag2_level>5</spam_tag2_level>
                <spam_kill_level>5</spam_kill_level>
                <locked>N</locked>
                <digest>N</digest>
                <report_type>N</report_type>
                <report_kill_level>999</report_kill_level>
                <spam_dsn_cutoff_level>0</spam_dsn_cutoff_level>
                <spam_quarantine_cutoff_level>999</spam_quarantine_cutoff_level>
                <digest_language>en_US</digest_language>
        </Policy>
</Response>
```

If the policy cannot be added, for instance, if the specified domain is a non-local domain then you will receive a 602 error code:

```
wget -q -O - "http://10.42.0.170/policy/add?user=sean@abcs.com"
```

```
<Response code="602" stat="fail">
        <error>Invalid Domain. Must be local.</error>
</Response>
```

DELETE POLICY

Deletes the specified user policy if it exists.

URL: http://10.1.1.82/policy/delete?user=sean@abc.com

Parameters:

- user: Required. The email address of the policy to delete.

Example 1: Delete policy sean@abc.com which does exist:

```
$ wget -q -O - "http://10.1.1.82/policy/delete?user=sean@abc.com"
```

```
<Response code="200" stat="ok">
        <success>Policy Deleted</success>
</Response>
```

Example 2: Delete policy simon@abc.com which does not exist:

```
$ wget -q -O - "http://10.1.1.82/policy/delete?user=simon@abc.com
```

```
<Response code="601" stat="fail">
        <error>No policy exists for simon@abc.com</error>
</Response>
```

EDIT POLICY

Modify the settings for an existing policy.

URL: http://10.1.1.82/policy/edit?user=jokes@foobar.com&attribute=digest&value=N

Parameters:

- user: Required. The email address or domain of the policy to modify
- attribute: Required. The policy attribute which you wish to modify
- value: Required. The value to set for the specified attribute

Attributes which you can edit are as follows:

- virus_lover: Specifies if virus infected files should be passed for this user (Y/N). Default N.
- spam_lover: Specifies if messages exceeding the spam threshold should be passed for this user (Y/N). Default N.
- banned_files_lover: Specifies if messages containing banned attachment should be passed for this user (Y/N). Default N.
- bad_header_lover: Specifies if messages containing banned attachment should be passed for this user (Y/N). Default N.
- bypass_virus_checks: Specifies if virus checking should be disabled for this user (Y/N). Default N.
- bypass_spam_checks: Specifies if spam checking should be disabled for this user (Y/N). Default N.
- bypass_banned_checks: Specifies if banned attachment checking should be disabled for this user (Y/N). Default N.
- bypass_header_checks: Specifies if header checks are bypassed for this user (Y/N). Default N.
- spam_modifies_subj: Specifies if the mail subject is changed when spam is detected for this user (Y/N). Default N.
- spam_tag_level; Add spam score headers to mail when score is greater than or equal. Default: 999. Type: float
- spam_tag2_level; Specifies the threshold over which messages will be considered spam. Default: 5. Type: float
- spam_kill_level; Quarantine or discard spam when score is greater than or equal to. Default: 5. Type: float
- report_kill_level; Specifies the threshold over which mail will be included in reports. Default: 999. Type: float
- spam_dsn_cutoff_level; Spam score at which not to generate delivery status notifications.. Default: 0. Type: float
- spam_quarantine_cutoff_level; Score at which not to quarantine. Default: 999. Type: float
- spam_quarantine_to: Specifies how to deal with spam mail. (Default) 'spam-quarantine' to quarantine mail, set to "" to reject. Set to "" and set *spam_lover* to 'Y' to Pass and Tag mail.
- virus_quarantine_to: Specifies how to deal with virus mail. (Default) 'virus-quarantine' to quarantine mail, set to "" to reject. Set to "" and set *virus_lover* to 'Y' to Pass and Tag mail.
- banned_quarantine_to: Specifies how to deal with banned attachment mail. (Default) 'banned-quarantine' to quarantine mail, set to "" to reject. Set to "" and set *banned_files_lover* to 'Y' to Pass and Tag mail.
- locked: Specifies if the policy is locked. If a policy is locked, changes to the domain policy will not be inherited by the locked user policy (Y/N). Default: N.
- digest: Specifies if this user should receive a quarantine report. N=Never, D=Daily, WD=Week Days, M=Monthly. Default N.
- report_type: Specifies the type of quarantine report to send the user. Possible values are N (New items since last report only) , A (all quarantine messages), X (All quarantined msgs, except viruses), Y (New items since last report, except viruses). Default N.
- digest_language: Specifies the language that the report should be generated in (cs_CZ/da_DK/de_DE/en_US/fr_FR/nl_NL/ja_JP/it_IT/pl_PL/es_ES). Default: en_US

Example 1: Modify the digest language for sean@abc.com to Japanese

```
$ wget -q -O -
"http://10.1.1.82/policy/edit?user=jd@test.com&attribute=digest_language&va
lue=ja_JP"
```

```
<Response code="200" stat="ok">
        <Policy>
                <id>5</id>
                <policy_name>sean@abc.com</policy_name>
                <virus_lover>N</virus_lover>
                <spam_lover>N</spam_lover>
                <banned_files_lover>N</banned_files_lover>
                <bad_header_lover>N</bad_header_lover>
                <bypass_virus_checks>N</bypass_virus_checks>
                <bypass_spam_checks>N</bypass_spam_checks>
                <bypass_banned_checks>Y</bypass_banned_checks>
                <bypass_header_checks>N</bypass_header_checks>
                <spam_modifies_subj>N</spam_modifies_subj>
                <spam_subject_tag2 />
                <spam_quarantine_to>spam-quarantine</spam_quarantine_to>
                <virus_quarantine_to>virus-quarantine</virus_quarantine_to>
                <banned_quarantine_to>banned-quarantine</banned_quarantine_to>
                <bad_header_quarantine_to />
                <spam_tag_level>-999</spam_tag_level>
                <spam_tag2_level>4</spam_tag2_level>
                <spam_kill_level>5.5</spam_kill_level>
                <locked>N</locked>
                <digest>N</digest>
                <report_type>N</report_type>
                <report_kill_level>999</report_kill_level>
                <spam_dsn_cutoff_level>0</spam_dsn_cutoff_level>
                <spam_quarantine_cutoff_level>999</spam_quarantine_cutoff_level>
                <digest_language>ja_JP</digest_language>
        </Policy>
</Response>
```

Example 2: If you enter an incorrect value you will get the following type response

```
$ wget -q -O -
"http://10.42.0.170/policy/edit?user=sean@abc.com&attribute=bypass_banned_chec
ks&value=F"
```

```
<Response code="602" stat="fail">
 <error>Invalid value for attribute bypass_banned_checks. Must be Y/N</error>
</Response>
```

Example 3: If you specify an invalid parameter then you will get the following response:

```
$ wget -q -O -
"http://10.42.0.170/policy/edit?user=sean@abc.com&attribute=badvar&value=123"
<Response code="405" stat="fail">
        <error>Invalid or Missing parameters</error>
</Response>
```

3. BLACKLIST METHODS

SHOW BLACKLIST

List all Blacklisted email addresses and/or domains for the specified user/domain. If no user or domain is specified then the global blacklist will be returned.

URL: http://10.1.1.82/blacklist/list

Parameters:

- user: Optional. The email address or domain name of the user/domain for which to view the blacklist

Response: Show all blacklisted items for user jack@abc.com:

```
Request: wget -q -O - "http://10.1.1.82/blacklist/list?user=jack@abc.com"
```

```
<Response code="200" stat="ok">
        <Blacklist>
                <item>@jacksbl.com</item>
                <item>jackbl@email.com</item>
        </Blacklist>
</Response>
```

CHECK BLACKLIST

Tests if the specified email address/domain is blacklisted by the specified user/domain, or globally if no user/domain is specified.

URL: http://10.1.1.82/blacklist/exists?sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to check for the existence of in a particular blacklist
- user: Optional. The email address or domain name of the blacklist to check. If no user is specified, then the global blacklist will be checked

Example 1: Check if the domain *bldomain.com* is blacklisted globally

```
$ wget -q -O - "http://10.1.1.82/blacklist/exists?sender=bldomain.com"
```

```
<Response code="200" stat="ok">
        <success>Blacklist entry exists</success>
</Response>
```

Example 2: Check if the email address joe@bloggs.com is blacklisted under the domain abc.com. In this case it is not

```
$ wget -q -O -
"http://10.1.1.82/blacklist/exists?user=abc.com&sender=joe@bloggs.com"
<Response code="501" stat="fail">
        <error>Blacklist entry not found</error>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 501: Blacklist entry not found
- 502: User/domain not found

ADD BLACKLIST ENTRY

Add an email address or domain to a specified users or domains blacklist. If no user/domain is specified then the entry is added to the global blacklist.

URL: http://10.1.1.82/blacklist/add?user=abc.com&sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to add to the blacklist
- user: Optional. The email address or domain name of the blacklist entry to add. If no user is specified, then the entry will be added to the global blacklist.

Example 1: Add joe@abcblacklist.com to the blacklist for users in domain abc.com

```
$ wget -q -O -
"http://10.1.1.82/blacklist/add?user=abc.com&sender=joe@abcblacklist.com"
<Response code="200" stat="ok">
        <success>Blacklist entry added</success>
</Response>
```

Example 2: Add domain foobar.com to the global blacklist

```
$ wget -q -O - "http://10.1.1.82/blacklist/add?sender=foobar.com"
<Response code="200" stat="ok">
        <success>Blacklist entry added</success>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 502: User/domain not found

DELETE BLACKLIST ENTRY

Delete an entry from the specified users/domains blacklist. If no user is specified, then delete the entry from the global blacklist.

URL: http://10.1.1.82/blacklist/delete?user=abc.com&sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to deleted from the blacklist
- user: Optional. The email address or domain name of the owner of the blacklist entry to delete. If no user is specified, then the entry will be deleted from the global blacklist.

Example 1: Delete *joe@abcblacklist.com* from the blacklist for users in domain abc.com

```
$ wget -q -O -
"http://10.1.1.82/blacklist/delete?user=abc.com&sender=joe@abcblacklist.com"
<Response code="200" stat="ok">
        <success>Blacklist entry deleted</success>
</Response>
```

Example 2: Attempt to delete *doesnotexist@xyz.com* from the global blacklist

```
$ wget -q -O - "http://10.1.1.82/blacklist/delete?sender=doesnotexist@xyz.com"
<Response code="502" stat="fail">
        <error>No such user or domain</error>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 502: User/domain not found

4. WHITELIST

SHOW WHITELIST

List all Whitelisted email addresses and/or domains for the specified user/domain. If no user or domain is specified then the global Whitelist will be returned.

URL: http://10.1.1.82/ whitelist/list

Parameters:

- user: Optional. The email address or domain name of the user/domain for which to view the whitelist

Response: Show all whitelisted items for user jack@abc.com:

```
Request: wget -q -O - "http://10.1.1.82/whitelist /list?user=jack@abc.com"
```

```
<Response code="200" stat="ok">
        <Whitelist>
                <item>@jacksbl.com</item>
                <item>jackbl@email.com</item>
        </Whitelist >
</Response>
```

CHECK WHITELIST

Tests if the specified email address/domain is whitelisted by the specified user/domain, or globally if no user/domain is specified.

URL: http://10.1.1.82/whitelist/exists?sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to check for the existence of in a particular whitelist
- user: Optional. The email address or domain name of the whitelist to check. If no user is specified, then the global whitelist will be checked

Example 1: Check if the domain *wldomain.com* is whitelisted globally

```
$ wget -q -O - "http://10.1.1.82/whitelist/exists?sender=wldomain.com"
```

```
<Response code="200" stat="ok">
        <success>Whitelist entry exists</success>
</Response>
```

Example 2: Check if the email *address joe@bloggs.com* is whitelisted under the domain abc.com. In this case it is not

```
$ wget -q -O -
"http://10.1.1.82/whitelist/exists?user=abc.com&sender=joe@bloggs.com"
<Response code="501" stat="fail">
        <error>Whitelist entry not found</error>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 501: Blacklist entry not found
- 502: User/domain not found

ADD WHITELIST ENTRY

Add an email address or domain to a specified users or domains whitelist. If no user/domain is specified then the entry is added to the global whitelist.

URL: http://10.1.1.82/whitelist/add?user=abc.com&sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to add to the whitelist
- user: Optional. The email address or domain name of the whitelist entry to add. If no user is specified, then the entry will be added to the global whitelist.

Example 1: Add *joe@abcblacklist.com* to the whitelist for users in domain *abc.com*

```
$ wget -q -O -
"http://10.1.1.82/whitelist/add?user=abc.com&sender=joe@abcblacklist.com"
<Response code="200" stat="ok">
        <success>Whitelist entry added</success>
</Response>
```

Example 2: Add domain foobar.com to the global blacklist

```
$ wget -q -O - "http://10.1.1.82/whitelist/add?sender=foobar.com"
<Response code="200" stat="ok">
        <success>Whitelist entry added</success>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 502: User/domain not found

DELETE WHITELIST ENTRY

Delete an entry from the specified users/domains whitelist. If no user is specified, then delete the entry from the global whitelist.

URL: http://10.1.1.82/whitelist/delete?user=abc.com&sender=jokes@foobar.com

Parameters:

- sender: Required. The email address or domain to delete from the whitelist
- user: Optional. The email address or domain name of the owner of the whitelist entry to delete. If no user is specified, then the entry will be deleted from the global whitelist.

Example 1: Delete *joe@abcblacklist.com* from the whitelist for users in domain abc.com

```
$ wget -q -O -
"http://10.1.1.82/whitelist/delete?user=abc.com&sender=joe@abcblacklist.com"
<Response code="200" stat="ok">
        <success>Whitelist entry deleted</success>
</Response>
```

Example 2: Attempt to delete *doesnotexist@xyz.com* from the global whitelist

```
$ wget -q -O - "http://10.1.1.82/whitelist/delete?sender=doesnotexist@xyz.com"
<Response code="502" stat="fail">
        <error>No such user or domain</error>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 502: User/domain not found

5. QUARANTINE

LIST ALL OF QUARANTINE

List all Quarantined mail for a given user.

URL: http://10.1.1.82/quarantine/list?user=jd@test.com

Parameters:

- user: Required. The email address to view the quarantine with.
  Customer is able to specify a wildcard recipient. E.g. List the quarantine for "*titan@coppfasten.com".
- display: Optional. The amount of entries to show.
- offset: Optional. From what number of the search results to show from. (default: 0)
- time: Optional. Display the quarantine from the time-frame specified:
  - 1hour
  - 3hour
  - 6hour
  - today
  - yesterday
  - 7day
  - lastreport (everything since the last generated quarantine report)
  - custom [Format: YearMonthDay e.g: (20140101)]
  - all (default)

Response: Show all quarantined items for user *jd@test.com*:

Request: wget -q -O - " *http://10.1.1.82/quarantine/list?user=jd@test.com*"

```
<Response code="200" stat="ok">
        <User>jd@test.com</User>
        <Total>3</Total>
        <Total_Spam>0</Total_Spam>
        <Total_Virus>2</Total_Virus>
        <Total_Banned>1</Total_Banned>
        <Spam/>
        <Virus>
                <mail>
                        <subject>VIRUS</subject>
                        <sender>10.1.0.202@iso82_esxi.com</sender>
                        <mail_id>1ZHmo8L14_8c</mail_id>
                </mail>
                <mail>
                        <subject>VIRUS</subject>
                        <sender>10.1.0.202@iso82_esxi.com</sender>
                        <mail_id>Dkzn2BxeidDu</mail_id>
                </mail>
        </Virus>
        <Banned>
                <mail>
```

```
                        <subject>banned2</subject>
                        <sender>10.1.0.202@iso82_esxi.com</sender>
                        <mail_id>lDYew3UnuCDP</mail_id>
                    </mail>
                </Banned>
            </Response>
```

Example 2:

```
Request: wget -q -O -
"http://10.1.1.82/quarantine/list?user=jd@test3.com&type=all&display=1&offset=0&time=c
ustom&startdate=20130101&enddate=20140301"
```

```
<Response code="200" stat="ok">
  <User>jd@test3.com</User>
  <User_Total>1</User_Total>
  <Showing>1</Showing>
  <Timeframe>custom</Timeframe>
  <Date_From>20130101</Date_From>
  <Date_To>20140301</Date_To>
  <Max_Per_Type>1</Max_Per_Type>
  <Offset>0</Offset>
  <Total_Spam>1</Total_Spam>
  <Total_Virus>0</Total_Virus>
  <Total_Banned>0</Total_Banned>
  <Spam>
    <mail>
      <subject>inbound_domain_spam_alias</subject>
      <sender>sender@testpc.com</sender>
      <recipient>jd@test3.com</recipient>
      <mail_id>mSVED-A2Ze1D</mail_id>
      <date>January 28, 2014, 4:25:51 pm</date>
    </mail>
  </Spam>
  <Virus />
  <Banned />
 </Response>
```

LIST QUARANTINE TYPE

List all Quarantined mail for a given user with a given type of Spam, Virus or Banned Attachments.

URL: http://10.1.1.82/quarantine/list?user=jd3@test.com&type=spam

Parameters:

- user: Required. The email address for which to view the quarantine
- type: Required. Type of mail to return: spam, virus or banned

Response: Show all quarantined items for user *jd@test.com* of type spam:

Request: wget -q -O - " *http://10.1.1.82/quarantine/list?user=jd2@test.com&type=spam*"

```
<Response code="200" stat="ok">
        <User>jd2@test.com</User>
        <Total>2</Total>
        <Total_Spam>2</Total_Spam>
        <Total_Virus>0</Total_Virus>
        <Total_Banned>0</Total_Banned>
        <Spam>
        <mail>
                <subject>spam subject</subject>
                <sender>10.1.0.202@iso82_esxi.com</sender>
                <mail_id>32duN0uHjW0w</mail_id>
        </mail>
        <mail>
                <subject>spam subject</subject>
                <sender>10.1.0.202@iso82_esxi.com</sender>
                <mail_id>XXPSnFPHmTl7</mail_id>
        </mail>
        </Spam>
        <Virus/>
        <Banned/>
</Response>
```

Example 1: Show all quarantined items for user *jd3@test.com* of type banned attachment:

Request: wget -q -O - " *http://10.1.1.82/quarantine/list?user=jd3@test.com&type=banned*"

```
<Response code="200" stat="ok">
        <User>jd3@test.com</User>
        <Total>3</Total>
        <Total_Spam>0</Total_Spam>
        <Total_Virus>0</Total_Virus>
        <Total_Banned>3</Total_Banned>
        <Spam/>
        <Virus/>
        <Banned>
                <mail>
                        <subject>banned2</subject>
                        <sender>10.1.0.202@iso82_esxi.com</sender>
```

```
                <mail_id>lDYew3UnuCDP</mail_id>
        </mail>
        <mail>
                <subject>banned2</subject>
                <sender>10.1.0.202@iso82_esxi.com</sender>
                <mail_id>qtvu8K-pCF1C</mail_id>
        </mail>
        <mail>
                <subject>banned2</subject>
                <sender>10.1.0.202@iso82_esxi.com</sender>
                <mail_id>GljdwYAMvX1G</mail_id>
        </mail>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter

DELIVER QUARANTINED MAIL

Release mail from the Quarantine and deliver it to the user.

URL: http://10.1.1.82/quarantine/deliver?user=jd@test.com&mail=lDYew3UnuCDP

Parameters:

- user: Required. The email address to get the quarantine
- mail: Required. The mail id of the mail to release. This can be found when listing quarantined mail.
- wl: Optional. True or false. Whitelist mail address to the user's email whitelist.

Example 1: Deliver mail.

```
Request: wget -q -O - "http://10.1.1.82/quarantine/
deliver?user=jd@test.com&mail=lDYew3UnuCDP "
<Response code="200" stat="ok">
        <Success>Message has been released from the quarantine</Success>
</Response>
```

Example 2: Deliver mail and whitelist address.

```
Request: wget -q -O - "http://10.1.1.82/quarantine/
deliver?user=jd@test.com&mail=lDYew3UnuCDP &wl=true"
<Response code="200" stat="ok">
        <Success>Message has been released from the quarantine and has been added to
        the users whitelist</Success>
</Response>
```

DELETE QUARANTINED MAIL

Delete mail from the Quarantine.

URL: http://10.1.1.82/quarantine/delete?user=jd@test.com&mail=lDYew3UnuCDP

Parameters:

- user: Required. The email address to get the quarantine
- mail: Required. The mail id of the mail to release. This can be found when listing quarantined mail.

Example:

```
Request: wget -q -O -
"http://10.1.1.82/quarantine/delete?user=jd@test.com&mail=lDYew3UnuCDP "
<Response code="200" stat="ok">
        <Success>Message has been deleted from the quarantine</Success>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter

6. REPORTING

SCAN SUMMARY FOR A DOMAIN

Show Domain Stats for a domain

URL: 10.1.1.82/reporting/scansummary?name=test.com

Parameters:

- name: Optional. The domain to check. Leave it out to get stats for all of SpamTitan.

Example:

Request: wget -q -O - "*10.1.1.82/reporting/scansummary?name=test.com*"

```
<Response code="200" stat="ok">
        <Domain>test.com</Domain>
        <Summary>
                <Clean_Messages>118</Clean_Messages>
                <Spam_Messages>326</Spam_Messages>
                <Virus_Messages>31</Virus_Messages>
                <Invalid_Recipients>1</Invalid_Recipients>
                <RBL_Reject>0</RBL_Reject>
                <HELO_Reject>0</HELO_Reject>
        </Summary>
</Response>
```

LICENSE USAGE FOR A DOMAIN

Show license usage for a domain, reports yesterday's license usage.

URL: 10.1.1.82/reporting/licenseusage?name=test.com

Parameters:

- name: Optional. The domain to check. Leave it out to get max daily and Average daily stats for all of SpamTitan license usage.

Example:

Request: wget -q -O - "*10.1.1.82/reporting/licenseusage?name=test.com*"

```
<Response code="200" stat="ok">
        <Domain>test.com</Domain>
        <Day_2013-05-08>
                <licensecount>1</licensecount>
        </Day_2013-05-08>
</Response>
```

LICENSE USAGE FOR A DOAMIN GROUP

Show license usage for a Domain Group

URL: 10.1.1.82/reporting/domaingroup_licenseusage?name=dg1

Parameters:

- name: Optional. The Domain Group to check.

Example:

Request: wget -q -O - "*10.1.1.82/reporting/domaingroup_licenseusage?name=dg1* "

```
<Response code="200" stat="ok">
    <Domain Group>
   <name>dg1</name>
   <date>2014-03-12</date>
   <licensecount>1</licensecount>
 </Domain Group></Response>
```

LAST SPAM RULES UPDATE

Show last spam rules update.

URL: 10.1.1.82/reporting/spamupdate

Example:

Request: wget -q -O - "*10.1.1.82/reporting/spamupdate* "

```
<Response code="200" stat="ok">
   <Last_Spam_Update>Mon Feb 03 03:06:12 2014</Last_Spam_Update>
   <Spam_Update_Interval>daily</Spam_Update_Interval>
   <Spam_Autoupdate_Enabled>1</Spam_Autoupdate_Enabled>
</Response>
```

LAST VIRUS RULES UPDATE

Show last virus rules update for Clam and Kaspersky

URL: 10.1.1.82/reporting/virusupdate

Example:

Request: wget -q -O - "*10.1.1.82/reporting/ virusupdate* "

```
<Response code="200" stat="ok">
  <Clam_Last_Update>Mon Feb 03 09:01:17 2014</Clam_Last_Update>
  <Clam_Update_Interval>hourly</Clam_Update_Interval>
  <Clam_Autoupdate_Enabled>1</Clam_Autoupdate_Enabled>
  <Kaspersky_Last_Update>Mon Feb 03 13:02:01 2014</Kaspersky_Last_Update>
  <Kaspersky_Update_Interval>hourly</Kaspersky_Update_Interval>
  <Kaspersky_Autoupdate_Enabled>1</Kaspersky_Autoupdate_Enabled>
  <Kaspersky_Version>8.0.3.100</Kaspersky_Version>
  <Kaspersky_Records>7468768</Kaspersky_Records>
</Response>
```

7. MAIL QUEUE

SHOW MAIL QUEUE

Show the mail queue for SpamTitan.

URL: http://10.1.1.82/mailqueue/all

Parameters:

- method: Required. The type of mail to show. Options incudes: all, active, deferred and incoming

Example:

Request: wget -q -O - "*http://10.1.1.82/mailqueue/all*"

```
<Response code="200" stat="ok">
        <Total>2</Total>
        <Active_Total>2</Active_Total>
        <Deferred_Total>0</Deferred_Total>
        <Incoming_Total>0</Incoming_Total>
        <Active>
                <mail>
                        <Message_ID>142962F264</Message_ID>
                        <Size>1308</Size>
                        <Time>Thu 09 May 11:37:01</Time>
                        <Sender>10.1.0.202@iso82_ESXi.com</Sender>
                        <Recipients>jd3@test.com</Recipients>
                </mail>
                <mail>
                        <Message_ID>3B0632F265</Message_ID>
                        <Size>1308</Size>
                        <Time>Thu 09 May 11:37:01</Time>
                        <Sender>10.1.0.202@iso82_ESXi.com</Sender>
                        <Recipients>jd3@test.com</Recipients>
                </mail>
        </Active>
        <Deferred/>
        <Incoming/>
</Response>
```

8. SPF

ADD SPF EXEMPTION

Add SPF record to SpamTitan.

URL: http://10.1.1.82/spf/add?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - " *http://10.1.1.82/spf/add?network= fe80::200:f8ff:fe21:67cf*"

```
<Response code="200" stat="ok">
        <Status>SPF added: 'fe80::200:f8ff:fe21:67cf'</Status>
</Response>
```

REMOVE SPF EXEMPTION

Remove SPF record from SpamTitan.

URL: http://10.1.1.82/spf/delete?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - "*http://10.1.1.82/spf/delete?network=1.2.3.4/32*"

```
<Response code="200" stat="ok">
        <Status>SPF removed: '1.2.3.4/32'</Status>
</Response>
```

CHECK SPF EXEMPTION

Check for SPF record in SpamTitan.

URL: http://10.1.1.82/spf/check?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - " http://10.1.1.82/spf/check?network=1.2.3.4/32"
```

```
<Response code="200" stat="ok">
        <Success>SPF '1.2.3.4/32' found</Success>
</Response>
```

LIST SPF EXEMPTIONS

List all SPF records in SpamTitan.

URL: http://10.1.1.82/spf/list

Example:

```
Request: wget -q -O - " http://10.1.1.82/spf/list"
```

```
Response code="200" stat="ok">
        <Total>6</Total>
        <SPF>
                <Allowed>1.2.3.6/32</Allowed>
                <Allowed>1.2.3.2/32</Allowed>
                <Allowed>1.2.3.12/32</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cf/128</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cd/128</Allowed>
                <Allowed>1.2.3.4/32</Allowed>
        </SPF>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

9. RBL

ADD RBL EXEMPTION

Add RBL record to SpamTitan.

URL: http://10.1.1.82/rbl/add?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - " *http://10.1.1.82/rbl/add?network= fe80::200:f8ff:fe21:67cf*"

```
<Response code="200" stat="ok">
        <Status>RBL added: 'fe80::200:f8ff:fe21:67cf'</Status>
</Response>
```

REMOVE RBL EXEMPTION

Remove RBL record from SpamTitan.

URL: http://10.1.1.82/rbl/delete?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - "*http://10.1.1.82/rbl/delete?network=1.2.3.4/32*"

```
<Response code="200" stat="ok">
        <Status>RBL removed: '1.2.3.4/32'</Status>
</Response>
```

CHECK RBL EXEMPTION

Check for RBL record in SpamTitan.

URL: http://10.1.1.82/rbl/check?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - " http://10.1.1.82/rbl/check?network=1.2.3.4/32"
```

```
<Response code="200" stat="ok">
        <Success>RBL '1.2.3.4/32' found</Success>
</Response>
```

LIST RBL EXEMPTIONS

List all RBL records in SpamTitan.

URL: http://10.1.1.82/rbl/list

Example:

```
Request: wget -q -O - " http://10.1.1.82/rbl/list"
```

```
Response code="200" stat="ok">
        <Total>6</Total>
        <RBL>
                <Allowed>1.2.3.6/32</Allowed>
                <Allowed>1.2.3.2/32</Allowed>
                <Allowed>1.2.3.12/32</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cf/128</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cd/128</Allowed>
                <Allowed>1.2.3.4/32</Allowed>
        </RBL>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

10. WHITELIST IP

> *Both Section 3 and 4 deals with Whitelist and Blacklist for Users and Domains,* ***these whitelisted emails are still subjected to frontline controls****. Sections 10 and 11 deal with Whitelisting and Blacklisting an IP address****. Any address whitelisted from here will bypass all Spam, Virus, Banned Attachments, RBL and SPF checks. HELO restrictions will still apply****.*

ADD WHITELIST IP EXEMPTION

Add whitelist entry to SpamTitan.

URL: http://10.1.1.82/whiteip/add?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - " *http://10.1.1.82/whiteip/add?network= fe80::200:f8ff:fe21:67cf*"

```
<Response code="200" stat="ok">
        <Status>Whitelist added: 'fe80::200:f8ff:fe21:67cf'</Status>
</Response>
```

REMOVE WHITELIST IP EXEMPTION

Remove whitelist entry from SpamTitan.

URL: http://10.1.1.82/whiteip/delete?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - "*http://10.1.1.82/whiteip/delete?network=1.2.3.4/32*"

```
<Response code="200" stat="ok">
        <Status>Whitelist removed: '1.2.3.4/32'</Status>
</Response>
```

CHECK WHITELIST IP EXEMPTION

Check for whitelist entry in SpamTitan.

URL: http://10.1.1.82/whiteip/check?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - " *http://10.1.1.82/whiteip/check?network=1.2.3.4/32*"

```
<Response code="200" stat="ok">
        <Success>Whitelist '1.2.3.4/32' found</Success>
</Response>
```

LIST WHITELIST IP EXEMPTIONS

List all whitelist entry in SpamTitan.

URL: http://10.1.1.82/whiteip/list

Example:

Request: wget -q -O - " *http://10.1.1.82/whiteip/list*"

```
Response code="200" stat="ok">
        <Total>6</Total>
        <Whitelist>
                <Allowed>1.2.3.6/32</Allowed>
                <Allowed>1.2.3.2/32</Allowed>
                <Allowed>1.2.3.12/32</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cf/128</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cd/128</Allowed>
                <Allowed>1.2.3.4/32</Allowed>
        </Whitelist>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

11. BLACKLIST IP

> *Both Section 3 and 4 deals with Whitelist and Blacklist for Users and Domains,* ***these whitelisted emails are still subjected to frontline controls****. Sections 10 and 11 deal with Whitelisting and Blacklisting an IP address****. Any address whitelisted from here will bypass all Spam, Virus, Banned Attachments, RBL and SPF checks. HELO restrictions will still apply****.*

ADD BLACKLIST EXEMPTION

Add blacklist entry to SpamTitan.

URL: http://10.1.1.82/blackip/add?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - "*http://10.1.1.82/blackip/add?network=fe80::200:f8ff:fe21:67cf*"

```
<Response code="200" stat="ok">
        <Status>Blacklist added: 'fe80::200:f8ff:fe21:67cf'</Status>
</Response>
```

REMOVE BLACKLIST EXEMPTION

Remove blacklist entry from SpamTitan.

URL: http://10.1.1.82/blackip/delete?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

Request: wget -q -O - "*http://10.1.1.82/blackip/delete?network=1.2.3.4/32*"

```
<Response code="200" stat="ok">
        <Status>SPF removed: '1.2.3.4/32'</Status>
</Response>
```

CHECK BLACKLIST EXEMPTION

Check for blacklist entry in SpamTitan.

URL: http://10.1.1.82/blackip/check?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - " http://10.1.1.82/blackip/check?network=1.2.3.4/32"
```

```
<Response code="200" stat="ok">
        <Success>SPF '1.2.3.4/32' found</Success>
</Response>
```

LIST BLACKLIST EXEMPTIONS

List all blacklist entries in SpamTitan.

URL: http://10.1.1.82/blackip/list

Example:

```
Request: wget -q -O - "http://10.1.1.82/blackip/list"
```

```
Response code="200" stat="ok">
        <Total>6</Total>
        <SPF>
                <Allowed>1.2.3.6/32</Allowed>
                <Allowed>1.2.3.2/32</Allowed>
                <Allowed>1.2.3.12/32</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cf/128</Allowed>
                <Allowed>fe80::200:f8ff:fe21:67cd/128</Allowed>
                <Allowed>1.2.3.4/32</Allowed>
        </SPF>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

12. TRUSTED NETWORKS

ADD TRUSTED NETWORK

Add trusted network entry to SpamTitan.

URL: http://10.1.1.82/trusted/add?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - "http://10.1.1.82/trusted/add?network=1.2.3.4/32"
<Response code="200" stat="ok">
        <Status>Trusted Network added: '1.2.3.4/32'</Status>
</Response>
```

REMOVE TRUSTED NETWORK

Remove trusted network entry from SpamTitan.

URL: http://10.1.1.82/trusted/delete?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - "http://10.1.1.82/trusted/delete?network=1.2.3.4/32"
<Response code="200" stat="ok">
        <Status>Trusted Network removed: '1.2.3.4/32'</Status>
</Response>
```

CHECK TRUSTED NETWORK

Check for trusted network in SpamTitan.

URL: http://10.1.1.82/trusted/check?network=1.2.3.4/32

Parameters:

- network: Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - "http://10.1.1.82/trusted/check?network=1.2.3.4/32"
```

```
<Response code="200" stat="ok">
        <Success>Trusted Network '1.2.3.4/32' found</Success>
</Response>
```

LIST TRUSTED NETWORK EXEMPTIONS

List all trusted network entries in SpamTitan.

URL: http://10.1.1.82/trusted/list

Example:

```
Request: wget -q -O - "http://10.1.1.82/trusted/list"
```

```
Response code="200" stat="ok">
        <Total>1</Total>
        <Trusted_Network>
                <Allowed>1.2.3.4/32</Allowed>
        </Trusted_Network>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

13. TLS

ADD TLS ENTRY

Add TLS entry to SpamTitan.

URL: http://10.1.1.82/tls/add?domain=abc123.com&subdomain=Y&policy=verify&protocol=SSLv3:TLSv1

Parameters:

- domain: Required. Domain name of entry to add.
- subdomain: Optional. 'Y' or (default) 'N'. Include all subdomains of specified domain.
- policy: Optional. Policy of entry, default 'may'. Options: 'may', 'verify', 'none'.
- protocol: Optional. Protocol of entry, default 'SSLv3:TLSv1'. Options: 'SSLv2', 'SSLv3', 'TLSv1', 'SSLv2:SSLv3', 'SSLv3:TLSv1, 'SSLv2:SSLv3:TLSv1'

Example:

```
Request: wget -q -O - "
http://10.1.1.82/tls/add?domain=abc123.com&subdomain=Y&policy=verify&
protocol=SSLv3:TLSv1"
```

```
<Response code="200" stat="ok">
        <Status>TLS Exception added: 'abc123.com'</Status>
</Response>
```

REMOVE TLS ENTRY

Remove trusted network entry from SpamTitan.

URL: http://10.1.1.82/tls/delete?domain=abc123.com

Parameters:

- domain: Required. Domain name of entry to remove.

Example:

```
Request: wget -q -O - "http://10.1.1.82/tls/delete?domain=abc123.com"
```

```
<Response code="200" stat="ok">
        <Status>TLS Exception removed: 'abc123.com'</Status>
</Response>
```

CHECK FOR TLS ENTRY

Check for TLS entry in SpamTitan.

URL: http://10.1.1.82/tls/check?domain=abc123.com

Parameters:

- "domain": Required. IPv4 / IPv6 address. CIDR accepted.

Example:

```
Request: wget -q -O - "http://10.1.1.82/tls/check?domain=abc123.com"
```

```
<Response code="200" stat="ok">
        <Success> TLS Exception 'abc123.com' found </Success>
</Response>
```

LIST TLS EXEMPTIONS

List all TLS entries in SpamTitan.

URL: http://10.1.1.82/tls/list

Example:

```
Request: wget -q -O - "http://10.1.1.82/tls/list"
```

```
<Response code="200" stat="ok">
        <TLS_Total>1</TLS_Total>
        <TLS>
                <Exception>
                        <domain>abc123.com</domain>
                        <comment>Added from API</comment>
                        <policy>may</policy>
                        <protocols>SSLv3:TLSv1</protocols>
                        <subdomain>N</subdomain>
                </Exception>
        </TLS>
</Response>
```

Response Codes:

- 200: Operation successful
- 201: Database Error
- 301: License not found
- 302: License Invalid
- 405: Missing sender parameter
- 601: API Entry does not exist

14. ALIASES

SHOW ALIAS

Show an Alias entry, including all its secondary addresses.

URL: http://10.1.1.93/alias/show?user=jd@test.com

Parameters:

- "user": Required. Alias to show.

Example:

```
Request: wget -q -O - "http://10.1.1.93/alias/show?user=jd@test.com"
```

```
<Response code="200" stat="ok">
  <User>jd@test.com</User>
  <Aliases>
    <Status>jd1@test.com</Status>
    <Status>jd2@test.com</Status>
    <Status>jd3@test.com</Status>
    <Status>jd4@test.com</Status>
    <Status>jd@test.com</Status>
    <Status>jd@test3.com</Status>
  </Aliases>
  <Is_Primary>true</Is_Primary>
  <Is_Secondary>true</Is_Secondary>
</Response>
```

ALIAS EXISTS

Check if email provided is a primary or a secondary Alias.

URL: http://10.1.1.93/alias/exists?user=jd@test.com

Parameters:

- "user": Required. Alias to show.

Example:

```
Request: wget -q -O - " http://10.1.1.93/alias/exists?user=jd@test.com"
```

```
<Response code="200" stat="ok">
  <User>jd@test.com</User>
  <Is_Primary>true</Is_Primary>
  <Is_Secondary>true</Is_Secondary>
</Response>
```

ADD ALIAS

Add a new Alias to SpamTitan.

URL:http://10.1.1.93/alias/add?user=jd_new@test.com&secondary=jd_new@test.com,jd2_new@test.com,info@test.com

Parameters:

- "user": Required. Alias to add.
- "secondary": Required. Aliases to add for this user. Separated by commas.
  Note: It is recommended to add the Primary email address (user) as a secondary address.

Example:

```
Request: wget -q -O - "http://10.1.1.93/alias/add?user=jd_new@test.com&secondary=jd_new@test.com,jd2_new@test.com,info@test.com"
```

```
<Response code="200" stat="ok">
  <User>jd_new@test.com</User>
  <Aliases>
    <Status>info@test.com</Status>
    <Status>jd2_new@test.com</Status>
    <Status>jd_new@test.com</Status>
  </Aliases>
  <Is_Primary>true</Is_Primary>
  <Is_Secondary>true</Is_Secondary>
</Response>
```

DELETE ALIAS

Remove an Alias and all its secondary addresses from SpamTitan. User policies are not removed.

URL: http://10.1.1.93/alias/delete?user=jd2_new@test.com

Parameters:

- "user": Required. Alias to remove.

Example:

```
Request: wget -q -O - "http://10.1.1.93/alias/delete?user=jd2_new@test.com "
```

```
<Response code="200" stat="ok">
  <Status>Success: Primary Alias &amp;quot;jd2_new@test.com&amp;quot; and all its aliases has been deleted</Status>
</Response>
```

DELETE DOMAIN GROUP ENTRY

Remove a secondary address from a primary Alias.

URL: http://10.1.1.93/alias/delete_entry?user=jd_new@test.com&secondary=jd2_new@test.com

Parameters:

- “user”: Required. Alias to add.
- “secondary”: Required. Aliases to remove for this user.

Example:

```
Request: wget -q -O - "http://10.1.1.93/alias/delete_entry?user=jd_new@test.com&secondary=jd2_new@test.com "
```

```
<Response code="200" stat="ok">
   <Status>Success: “jd2_new@test.com” for “jd_new@test.com” has been deleted</Status>
</Response>
```

15. DOMAIN GROUPS

SHOW DOMAIN GROUPS

Show a Domain Group and its Domain count.

URL: http://10.1.1.93/domaingroup/show?name=dg1

Parameters:

- "name": Required. Domain Group to show.

Example:

Request: wget -q -O - "http://10.1.1.93/domaingroup/show?name=dg1 "

```
<Response code="200" stat="ok">
 <Status>
  <id>1</id>
  <name>dg1</name>
  <timezone>Europe/Dublin</timezone>
  <description>testing</description>
  <domains_count>2</domains_count>
  <domaingroup_admins_count>0</domaingroup_admins_count>
 </Status>
</Response>
```

LIST DOMAIN GROUP ADMINS

List Domain Group Admins.

URL: http://10.1.1.93/domaingroup/list_admins?name=dg1

Parameters:

- "name": Required. Domain Group to list.

Example:

Request: wget -q -O - "http://10.1.1.93/domaingroup/list_admins?name=dg1"

```
<Response code="200" stat="ok">
 <Domain_Group_Admininstrators>
  <Domain_Group_Admin>
   <email>jd@test.com</email>
   <role>Domain Group Administrator</role>
  </Domain_Group_Admin>
 </Domain_Group_Admininstrators>
</Response>
```

LIST DOMAIN GROUP DOMAINS

List Domains in a Domain Group.

URL: http://10.1.1.93/domaingroup/list_domains?name=dg1

Parameters:

- "name": Required. Domain Group to list.

Example:

```
Request: wget -q -O – "http://10.1.1.93/domaingroup/list_domains?name=dg1"
<Response code="200" stat="ok">
 <Domain_Group_Domains>
  <Domain>test.com</Domain>
  <Domain>test2.com</Domain>
 </Domain_Group_Domains>
</Response>
```

ADD DOMAIN GROUP

Add a new Domain Group.

URL: http://10.1.1.93/domaingroup/add?name=dg4&timezone=Europe/Dublin&description=added from api&email=jd@test.com

Parameters:

- "name": Required. Domain Group to add.
- "timezone": Optional. Set the time zone for your Domain Group. e.g: Europe/Dublin. (Default: System Default)
- "description": Optional. Write a comment about your Domain Group.
- "email": Optional. Email of Domain Group Admin for this Domain Group.

Example:

```
Request: wget -q -O – "http://10.1.1.93/domaingroup/add?name=dg4&timezone=Europe/Dublin&description=added from api&email=jd@test.com"
<Response code="200" stat="ok">
   <success>Success: Domain Group added successfully. Domain Group Admin added successfully.</success>
 </Response>
```

ADD DOMAIN TO DOMAIN GROUP

To add a Domain to a Domain Group use the domain edit function or the domain add function.

REMOVE DOMAIN FROM DOMAIN GROUP

To remove a Domain from a Domain Group use the domain edit function or the domain delete function.

ADD DOMAIN GROUP ADMIN

Add a new Admin to a Domain Group.

URL: http://10.1.1.93/domaingroup/add_admin?name=dg4&email=jd@test.com

Parameters:

- "name": Required. Domain Group to add Admin to.
- "email": Required. Email of Domain Group Admin.

Example:

```
Request: wget -q -O –
"http://10.1.1.93/domaingroup/add_admin?name=dg1&email=jd@test.com "
<Response code="200" stat="ok">
  <success>Success:  Domain Group Admin added successfully.</success>
</Response>
```

REMOVE DOMAIN GROUP ADMIN

Remove an Admin from a Domain Group.

URL: http://10.1.1.93/domaingroup/delete_admin?name=dg4&email=jd@test.com

Parameters:

- "name": Required. Domain Group to add Admin to.
- "email": Required. Email of Domain Group Admin.

Example:

```
Request: wget -q -O –
"http://10.1.1.93/domaingroup/delete_admin?name=dg1&email=jd@test.com"
<Response code="200" stat="ok">
  <success>Success:  Domain Group Admin removed successfully.</success>
</Response>
```

DELETE DOMAIN GROUP

Remove a Domain Group. Note this will not remove the domain or user policies.

URL: http://10.1.1.93/domaingroup/delete_domaingroup?name=dg4

Parameters:

- "name": Required. Domain Group to add Admin to.

Example:

```
Request: wget -q -O – " http://10.1.1.93/domaingroup/ delete_domaingroup?name=dg1"
```

```
<Response code="200" stat="ok">
  <success>Success:  Domain Group removed successfully.</success>
</Response>
```

16. RATE CONTROLS

SHOW RATE CONTROL

Show the Rate Control configuration.

URL: http://10.1.1.93/ratecontrol/show?name=testcontrol

Parameters:

- "name": Required. Name of the Rate Control to show.

Example:

Request: wget -q -O - "http://10.1.1.93/ratecontrol/show?name=testcontrol "

```
<Response code="200" stat="ok">
  <Policy>
    <Status>
      <name>testcontrol</name>
      <sourcetype>any</sourcetype>
      <source>any</source>
      <desttype>any</desttype>
      <destination>any</destination>
      <track>Sender:user@domain</track>
      <trackbitmask />
      <type>MessageCount</type>
      <counterlimit>100</counterlimit>
      <period>30</period>
      <period_unit>minute</period_unit>
      <lastquota>1</lastquota>
      <verdict>DEFER</verdict>
      <notify />
      <response>Policy rejection; Message count quota exceeded</response>
      <disabled>0</disabled>
      <comment />
      <hits>0</hits>
      <priority>1</priority>
      <pid>3</pid>
      <qid>3</qid>
      <qlid>3</qlid>
    </Status>
  </Policy>
</Response>
```

RATE CONTROL EXISTS

Test if given Rate Control exists.

URL: http://10.1.1.93/ratecontrol/exists?name=testcontrol

Parameters:

- "name": Required. Name of the Rate Control to show.

Example:

```
Request: wget -q -O - "http://10.1.1.93/ratecontrol/exists?name=testcontrol"
```

```
<Response code="200" stat="ok">
  <success>Policy found for testcontrol</success>
</Response>
```

LIST RATE CONTROL

List all Rate Controls.

URL: http://10.1.1.93/ratecontrol/list

Parameters:

- "name": Required. Name of the Rate Control to show.

Example:

```
Request: wget -q -O - "http://10.1.1.93/ratecontrol/exists?name=testcontrol"
```

```
<Response code="200" stat="ok">
  <Policy>
    <Status>
      <name>Outbound: Sender quotas</name>
      <sourcetype>internal</sourcetype>
      <source>Internal</source>
      <desttype>any</desttype>
      <destination>any</destination>
      <track>Sender:user@domain</track>
      <trackbitmask />
      <type>MessageCount</type>
      <counterlimit>50</counterlimit>
      <period>30</period>
      <period_unit>minute</period_unit>
      <lastquota>1</lastquota>
      <verdict>REJECT</verdict>
      <notify />
      <response>Policy rejection; Message count quota exceeded</response>
      <disabled>1</disabled>
      <comment>Sample policy to rate limit all internal senders by email
address</comment>
      <hits>0</hits>
```

```
            <priority>1</priority>
            <pid>1</pid>
            <qid>1</qid>
            <qlid>1</qlid>
          </Status>
          <Status>
            <name>Inbound: Recipient quotas</name>
            <sourcetype>external</sourcetype>
            <source>External</source>
            <desttype>any</desttype>
            <destination>any</destination>
            <track>Recipient:@domain</track>
            <trackbitmask />
            <type>MessageCount</type>
            <counterlimit>50</counterlimit>
            <period>30</period>
            <period_unit>minute</period_unit>
            <lastquota>1</lastquota>
            <verdict>DEFER</verdict>
            <notify />
            <response>Policy rejection; Message count quota exceeded</response>
            <disabled>1</disabled>
            <comment>Sample policy to track/rate limit messages to each internal
        domain</comment>
            <hits>0</hits>
            <priority>2</priority>
            <pid>2</pid>
            <qid>2</qid>
            <qlid>2</qlid>
          </Status>
        </Policy>
      </Response>
```

ADD RATE CONTROL

Add a new Rate Control.

URL: http://10.1.1.93/ratecontrol/add?name=testcontrol

Parameters:

- name: Required. Name of the Rate Control to add.
- status: Optional. “t” or “f” (default). Enable or disable this policy.
- source: Optional. Indicates where the message originates. Source can be one of:
    - any (default)
    - internal: Originating from an internal network (as defined under System Setup -> Mail Relay -> Outbound -> Trusted Networks)
    - external: Originating from an external network
    - From a specific domain: specify the sender domain that should match this rule
    - From a specific email address: specify the sender email address that should match this rule
    - From a specific IP address/CIDR: specify the sender IP address or CIDR address that should match this rule
- destination. Optional. Destination indicates where the message is destined to. Destination can be one of:
    - any (default)
    - To a specific domain: specify the recipient domain that should match this rule
    - To a specific email address: specify the recipient email address that should match this rule
- lastquota. Optional. If set, when a policy matches, no further policy rules will be processed. Values: 1 – enable (default), 0 - disable
- priority: Optional. Priority to assign to the rule. All new policies default to the highest priority (1), with all other policies shifting down. Default “1”.
- track: Optional. Track specifies the attribute to maintain counters for. Track can be one of:
    - ‘SenderIP’: If you select to track the sender IP address, you must specify a bitmask to apply to the sending servers' IP address, for instance /24. This will track the triplet through the entire /24 block.
    - ‘Sender:user@domain’: Each sender email address will be tracked separately
    - ‘Sender:@domain': Each sender domain is tracked separately, and all email sent from these domains will be tracked and matched
    - 'Recipient:user@domain': Each recipient email address will be tracked separately
    - 'Recipient:@domain': Each recipient domain is tracked separately, and all email sent to these domains will be tracked and matched
- senderip_bitmask: Optional. Bitmask of track. “32” (255.255.255.255) to “8” (255.0.0.0) *note: for use with sender IP only*
- limit: Optional. This specifies if the policy applies to Message Counts (“MessageCount”) or Cumulative Message Size (“MessageCumulativeSize”). Default: “MessageCount”
- limitcount: Optional. (Only if limit is set to MessageCount). Message count threshold. (default: 100)
- limitsize: Optional. (Only if limit is set to MessageCumulativeSize). Message size threshold.

- period: Optional. Set time period. e.g.: 30, 40, 50.
- period_unit: Optional. Set time period unit. Values: “second”, “minute”, “hour”
- action: Required. This specifies the action to be taken when a rule matches and the rate limit threshold has been exceeded in the time interval specified. Action can be one of
    - DEFER: Defer Messages: (default) Messages matching this rule will be deferred
    - REJECT: Reject Messages: Messages matching this rule will be rejected.
- response: Optional. Response that is returned to the sender when a policy rule fires and a message is deferred or rejected. (default: “Policy rejection; Message count quota exceeded”)
- notify: Optional. Notify Administrator when policy is activated. Set this to the email of the admin to notify.
- comment: Optional. Comment for this rate control.

Example 1:

```
Request: wget -q -O -
"http://10.1.1.93/ratecontrol/add?name=testcontrol2&status=t&source=internal&destinati
on=jd@test.com&lastquota=0&priority=1&track=1.2.3.4&senderbitmask=255.255.255.255&
limit=
MessageCumulativeSize&limitcount=111&limitsize=100&period=55&period_unit=hour&acti
on=REJECT&response=my test response&comment=mytest"
```

```
<Response code="200" stat="ok">
  <Policy>
    <Status>
      <name>testcontrol2</name>
      <sourcetype>internal</sourcetype>
      <source>Internal</source>
      <desttype>email</desttype>
      <destination>jd@test.com</destination>
      <track>Sender:user@domain</track>
      <trackbitmask />
      <type>MessageCount</type>
      <counterlimit>111</counterlimit>
      <period>55</period>
      <period_unit>hour</period_unit>
      <lastquota>0</lastquota>
      <verdict>REJECT</verdict>
      <notify />
      <response>my test response</response>
      <disabled>0</disabled>
      <comment>mytest</comment>
      <hits>0</hits>
      <priority>1</priority>
      <pid>9</pid>
      <qid>9</qid>
      <qlid>9</qlid>
    </Status>
  </Policy>
</Response>
```

Example 2:

```
Request: wget -q -O - "http://10.1.1.93/ratecontrol/add?name=testcontrol"
```

```
<Response code="200" stat="ok">
  <Policy>
    <Status>
      <name>testcontrol2</name>
      <sourcetype>any</sourcetype>
      <source>any</source>
      <desttype>any</desttype>
      <destination>any</destination>
      <track>Sender:user@domain</track>
      <trackbitmask />
      <type>MessageCount</type>
      <counterlimit>100</counterlimit>
      <period>30</period>
      <period_unit>minute</period_unit>
      <lastquota>1</lastquota>
      <verdict>DEFER</verdict>
      <notify />
      <response>Policy rejection; Message count quota exceeded</response>
      <disabled>1</disabled>
      <comment />
      <hits>0</hits>
      <priority>1</priority>
      <pid>7</pid>
      <qid>7</qid>
      <qlid>7</qlid>
    </Status>
  </Policy>
</Response>
```

EDIT RATE CONTROL

Edit a Rate Control.

URL: http://10.1.1.93/ratecontrol/edit?name=testcontrol

Parameters: Refer to Rate Control Add.

```
Request: wget -q -O - "http://10.1.1.93/ratecontrol/edit?name=testcontrol"
```

```
<Response code="200" stat="ok">
  <success>Policy found for testcontrol</success>
</Response>
```

DELETE RATE CONTROL

Delete a Rate Control

URL: http://10.1.1.93/ratecontrol/delete?name=testcontrol

Parameters:

- name: Required. Name of the Rate Control to delete.

Example:

```
Request: wget -q -O - "http://10.1.1.93/ratecontrol/delete?name=testcontrol"
```

```
<Response code="200" stat="ok">
   <success>Policy deleted</success>
</Response>
```