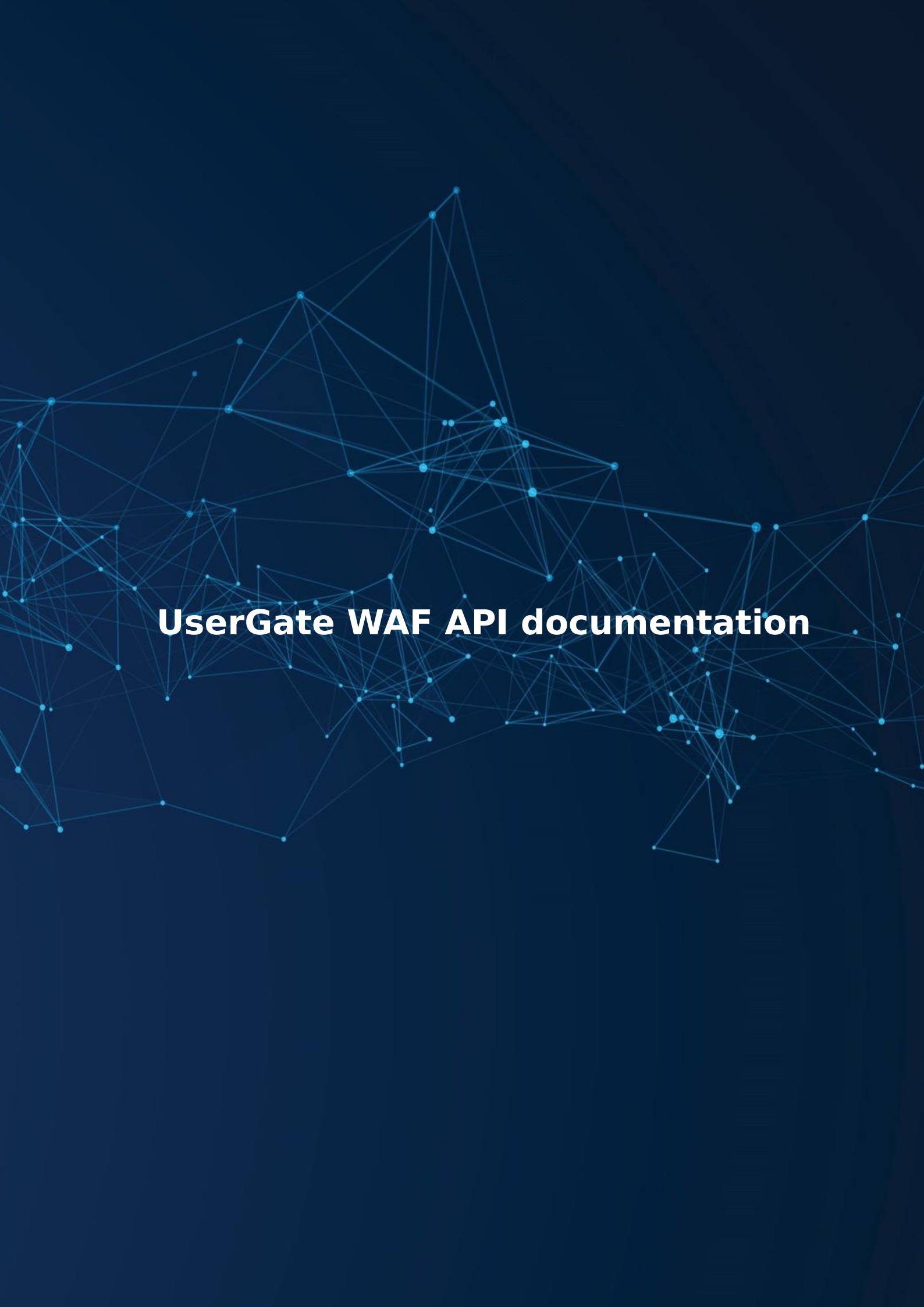


The background of the entire page is a dark blue gradient. Overlaid on this is a complex, abstract network diagram. It consists of numerous small, light blue dots (nodes) connected by thin, light blue lines (edges). The connections form a dense, interconnected web that spans across the upper and middle portions of the image, creating a sense of digital connectivity and data flow.

UserGate WAF API documentation

Content

Overview	4
Connection and authentication	5
API types	7
API errors	75

API modules

Auth	84
Certificates	99
Content filtering	102
Core	105
Device	121
diagnostics	135
Events	141
Firewall	142
High Availability	149
Http Response	154
JSON	157
Ldap	162
Libraries	165
Network manager	171
Named lists	194
Notification	202
PCAP	208
Public	215
Reverse proxy	216
Settings	225
SNMP	239
Statistics	247
Storage	251
System Status	255
UPL	266
WAF	270
Websocket	281
XML	284

Overview

This is UserGate WAF API documentation.

The only implemented API implementations (for now) are XML-RPC and JSON, see examples in [Connection and authentication](#) chapter.

Document information

Build date: 2026-10-12 03:44:20 UTC

API Version: 7.7.0

Connection and authentication

XML-RPC

Entry point for XML-RPC queries for address 192.168.1.100:

```
http://192.168.1.100:4040/rpc
```

JSON

Entry point for JSON queries for address 192.168.1.100:

```
http://192.168.1.100:4040/web_api
```

For given method `METHOD.NAME` JSON request MUST be sent as a POST HTTP query to the following URL:

```
http://192.168.1.100:4040/web_api/METHOD.NAME
```

Method arguments MUST be passed as JSON list in that query.

Errors are returned in form of the following JSON structure:

```
{
  "faultCode": 1,
  "faultString": "Undefined API method"
}
```

And some examples of JSON requests using `curl` utility:

Login with incorrect password:

```
bash$ curl -X POST http://192.168.1.100:4040/web_api/v2.core.login \
-H 'Content-Type: application/json' \
-d '["Admin", "wrong_password", {}]'

{"faultCode":109,"faultString":"Wrong login or password"}
```

Login with correct password:

```
bash$ curl -X POST http://192.168.1.100:4040/web_api/v2.core.login \
-H 'Content-Type: application/json' \
-d '["Admin", "123", {}]'

{"auth_token":"207276333621968585292439776397927341509","admin_id":1,
"admin_login":"Admin","guid":"5c3ed010-a04c-47f2-8a0a-9d258641013a",
"timezone_offset":0,"node":"wafcore@nodehostname",
"version":"7.0.0","product":"waf","cookie":"","
"logan_session_waf":"95gbh19a612745a2c0e72b8hgcd9b9"}
```

Updating response page template with ID=27 using authentication token

"207276333621968585292439776397927341509" we have obtained in previous request, Base64 data are encoded in special way described in [Types](#) chapter:

```
bash$ curl -X POST \
  http://192.168.84.71:4040/web_api/v1.libraries.response.page.template.data.update \
  -H 'Content-Type: application/json' \
  -d '["207276333621968585292439776397927341509", 27, {"__base64__": "aGVyZSBpdCBpcwo="}]'

true
```

Authentication

Almost all methods require authentication token — a special string that identifies session. Session is created by calling one of authentication methods, like, [v2.core.login](#).

Session has a lifetime, to keep it alive API client must periodically call any method to extend session lifetime. There is also a special method [v2.core.session.ping](#) that does nothing and could be used for keep alive requests.

Examples in Python

Authenticate, fetch first 100 firewall rules, print them and log out.

```
#!/usr/bin/env python3

SERVER = '192.168.84.26'
USER = 'Admin'
PASSWORD = '123'

import xmlrpc.client
from pprint import pprint

server = xmlrpc.client.ServerProxy('http://' + SERVER + ':4040/rpc', verbose=False)

res = server.v2.core.login(USER, PASSWORD, {})
auth_token = res['auth_token']

fw_rules = server.v1.firewall.rules.list(auth_token, 0, 100, {})
pprint(fw_rules)

server.v2.core.logout(auth_token)
```

API types

We use the following “generic” data types (inherited from XML-RPC):

- **Integer** — integer signed value, original XML-RPC supports only 32-bits integer, we support more than 32-bits values;
- **String** — plain string value, utf8-encoded;
- **Double** — float value, double precision;
- **Date** — datetime in ISO8601 format: `2010-03-22T19:25:00Z` ;
- **Boolean** — boolean value, literals `true` or `false` ;
- **Base64** — binary data encoded in Base64;
- **List** — plain array of elements, usually used with conjunction of single entry type, like this `List<String>` ;
- **UID** — generic identifier, integer or string value that consists of characters `[a-zA-Z0-9-]` , string identifier must be less than 60 characters long, identifier CANNOT be assigned by API user and always generated automatically;
- **Variant** — mixed-type value, can be one of any described above.
- **Null** — special type with single value `nil` , in XML-RPC it's represented by `<nil/>` in JSON RPC it's represented by `null` value.

Other structured data types are based on XML-RPC type `Struct` and they are described below.

Some fields are marked required on create, it means that this field must be set (and must be non-empty for string values) when creating a new object. When editing existing object all fields are optional. In that case not provided fields won't be changed.

In JSON interface types **Date** and **Base64** are encoded in special way. **Date** value `2010-03-22T19:25:00Z` is encoded as structure:

```
{__datetime__: "2010-03-22T19:25:00Z"}
```

Base64 value is encoded as structure:

```
{__base64__: "dGVzdAo="}
```

Fields that marked as readonly in structures are ignored when writing objects, for example, almost all identifiers are readonly.

Fields that marked as writeonly in structures are excluded when reading objects, for example, passwords are writeonly.

Fields marked as createonly in structures are ignored when object is being updated and accepted only when creating a new object in corresponding method.

ActivationInfo

Structure that is used for activation requests to license server.

Fields:

pin_code	String	product activation pin-code
-----------------	--------	-----------------------------

AdministratorInfo

Describes information about server administrator.

Fields:

id	UID	readonly unique identifier
guid	String	when type is "ldap_user" or "ldap_group" this field is required and should contain identifier of LDAP object obtained from v1.ldap.users.list or v1.ldap.groups.list
enabled	Boolean	is administrator enabled or disabled
profile_id	UID	administrator profile identifier
description	String	optional description
type	String	type of administrator, allowed values are: "local" , "ldap_user" , "ldap_group" , "auth_profile"
login	String	administrator login, unique for server; when type is "ldap_user" or "ldap_group" this field value is used for display only and ignored during login
password	String	required on create writeonly administrator password, skip this field to leave password unchanged, this value is ignored if type is not "local"
is_root	Boolean	readonly set to <code>true</code> when administrator has maximum permissions
user_auth_profile_id	UID	identifier of authentication profile, must be set to valid value if type is "auth_profile" , otherwise is ignored and should be set to integer value <code>0</code>
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC
locked	Boolean	

AdministratorPermission

List with two elements, first is a permission name, second element is one of the following string values: "none" , "read" , "readwrite" .

See method [v1.libraries.permissions.categories.list](#) for complete list of available permission names.

AdministratorProfileInfo

Structure that represents Administrator profile (roles/permissions).

Fields:

id	UID	readonly object identifier
name	String	required on create profile name
description	String	profile description
xmlrpc_permissions	List< AdministratorPermission >	
webui_permissions	List< AdministratorPermission >	
cli_permissions	List< AdministratorPermission >	

cc	Boolean	readonly contains <code>true</code> if object is fetched from MC
-----------	---------	--

AdministratorSessionInfo

Describes logged in administrator session.

Fields:

guid	String	session identifier
node_name	String	
administrator_id	UID	administrator identifier
administrator_name	String	administrator login name
administrator_user_name	String	
origin	String	origin, possible values are: <code>admin_console</code> , <code>cli_console</code>
ip	String	IP address of administrator
session_start_date	Date	

AdministratorSystemConfig

Describes global system config for system administrators.

Fields:

strong_pwd	Boolean	
n_of_invalid_auth	Integer	
block_time	Integer	block time in seconds
min_length	Integer	minimal password length
min_uppercase	Integer	minimal required number of uppercase characters
min_lowercase	Integer	minimal required number of lowercase characters
min_digit	Integer	
min_special	Integer	
max_char_repetition	Integer	

AlertRuleInfo

Structure that describes alert rule.

Fields:

id	UID	readonly unique rule identifier
name	String	rule name
description	String	rule description
enabled	Boolean	is rule on
notification_profile_id	UID	required on create notification profile
from	String	

		required on create sender address, either email or phone number (depends on notification profile type)
subject	String	email subject, required on create when notification profile type is "smtp"
events	List<String>	list of events for which WAF send notification
emails	List<TaggedValue>	list of emails list WAF should send notification to, allowed tags are: "list_id" (named list of type "emailgroup"), required on create when notification profile type is "smtp"
phones	List<TaggedValue>	list of phones list WAF should send notification to, allowed tags are: "list_id" (named list of type "phonegroup"), required on create when notification profile type is "smpp"
cc	Boolean	readonly contains true if object is fetched from MC
timeout	Integer	time to wait for next alert, in seconds

CertificateCreateInfo

Structure that represents details of generated certificate.

Fields:

name	String	required on create human readable certificate name, not included in certificate data
description	String	certificate description, not included in certificate data
cert_common_name	String	required on create component "CN", usually this is FQDN of the organization
cert_country	String	required on create country two letter code (component "C")
cert_state	String	required on create state/region (component "ST")
cert_locality	String	required on create locality/city (component "L")
cert_org_name	String	required on create organization name (component "O")
cert_org_unit	String	organization unit (component "OU" of certificate/csr DN)
cert_email	String	required on create component "emailAddress"

CertificateCSRCreateInfo

Structure that represents details of generated CSR.

Fields:

name	String	certificate name
description	String	certificate name
cert_common_name	String	required on create component "CN", usually this is FQDN of the organization
cert_country	String	required on create country two letter code (component "C")
cert_state	String	required on create state/region (component "ST")
cert_locality	String	required on create locality/city (component "L")
cert_org_name	String	required on create organization name (component "O")
cert_org_unit	String	organization unit (component "OU" of certificate/csr DN)

cert_email	String	required on create component "emailAddress"
san	List<TaggedValue>	writeln elements of Subject Alternative Name X.509 extension, allowed tags are: "email", "ip", "dns", "uri"

CertificateDetailsInfo

Structure that represents parsed certificate information.

Fields:

signatureAlgorithm	String	signature algorithm, for example, "sha256WithRSAEncryption"
publicKeyAlgorithm	String	public key algorithm, for example, "rsaEncryption"
serialNumber	String	certificate serial number, for example, "A4:79:39:6C:DD:1D:AC:ED"
notBefore	Date	date from certificate
notAfter	Date	date from certificate
issuer	CertificateDN	certificate subject fields
subject	CertificateDN	certificate issuer subject fields
altNames	List<String>	altNames data field from the certificate
keyUsage	List<String>	keyUsage data field from the certificate
chain	List<String>	

CertificateDN

Structure that represents Distinguished Name attributes.

Fields:

countryName	String
organizationName	String
localityName	String
commonName	String
emailAddress	String

CertificateStorageInfo

Structure that describes SSL certificate from the certificates storage.

Fields:

id	UID	readonly certificate identifier
name	String	required certificate name
description	String	certificate description
has_private_key	Boolean	readonly true if there is a private key for this certificate record
has_csr	Boolean	readonly true if there is a CSR for this certificate record
has_cert	Boolean	readonly true if there is a x509 certificate in this record

not_before	Date	readonly notBefore date from certificate
not_after	Date	readonly notAfter date from certificate
subject	CertificateSubjectInfo	readonly certificate subject fields
issuer	CertificateSubjectInfo	readonly certificate issuer subject fields
cert_data	Base64	writable required on create certificate data (PEM or DER base64-encoded) used only with method v2.settings.certificate.add , it may also contain DER-encoded PKCS#12 data
key_data	Base64	optional certificate private key (PEM or DER base64-encoded) used only with methods v2.settings.certificate.add and v2.settings.certificate.update
chain_data	Base64	writable optional additional chain certificates (PEM format only)
has_cert_chain	Boolean	readonly shows whether the certificate is linked
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC
role	String	certificate role, allowed values are: <code>"none"</code> , <code>"proxy_ca"</code> , <code>"webui"</code> , <code>"webui_ca"</code> , <code>"proxy_ca_chain"</code> , <code>"proxy_ca_chain_root"</code> , <code>"resp_pages"</code> , <code>"user"</code>
user_guid	UID	ID of user, used only when role == <code>"user"</code>
key_passphrase	String	writable passphrase for encrypted private key when PKCS#12 data are passed in cert_data field.
ca	Boolean	readonly
keyUsage	List<String>	readonly

Role names have the following meanings:

- `"none"`, "No role is set"
- `"proxy_ca"`, "Certificate for HTTPS/SSL decryption, must have private key. Only one certificate of such kind is allowed."
- `"webui"`, "Certificate for WebUI (HTTPS on port 8001). Only one certificate of such kind is allowed."
- `"webui_ca"`, "CA Certificate for WebUI certificate authentication mode"
- `"proxy_ca_chain"`, "Intermediate CA certificate that should be included in certificates chain during HTTPS decryption"
- `"proxy_ca_chain_root"`, "Root CA for decryption chain"
- `"resp_pages"`, "Certificate for response pages (captive portal, block page etc). Only one certificate of such kind is allowed."
- `"user"`, "Certificate that linked to specific user, identified by ID. When this role is selected field **user_id** must be set too."

CertificateSubjectInfo

Structure that represents certificate subject components (parsed certificate DN)

Fields:

common_name	String	readonly component "CN", usually this is FQDN of the organization
country	String	readonly country two letter code (component "C")
state	String	readonly state/region (component "ST")

locality	String	readonly locality/city (component "L")
org_name	String	readonly organization name (component "O")
org_unit	String	readonly organization unit (component "OU")
email	String	readonly component "emailAddress"

ClientCertificateProfileInfo

Fields:

id	UID	readonly client certificate profile identifier
name	String	required on create human readable profile name
description	String	
crl	String	CRL (Certificate Revocation List), allowed values are: "peer" , "best_effort" or Boolean (true/false)
crl_receive_timeout	Integer	duration of disable filtering, allowed values are from 1 to 100000
username_field	String	field of user name, allowed values are: "common_name" , "subject_alt_name_email" , "principal_name" , "subject_alt_name_ipv4"
ca_certificates	List<UID>	required on create certificates (UID for CertificateStorageInfo)
cc	Boolean	readonly contains true if object is fetched from MC

CustomDnsInfo

Structure that describes single custom DNS server record.

Fields:

id	UID	readonly server identifier
dns	String	required on create DNS server address
is_bad	Boolean	readonly

DashboardWidget

Structure that describes widgets of dashboard.

Fields:

id	UID	widget identifier in form uuid
type	String	required name of widgets as returned by v2.settings.dashboard.widgets.list
columnIndex	Integer	required the column number in which the widget will be located
height	Integer	required widget height
widget	Struct	required additional widget parameters depending on its type. Below is a detailed list of which fields belong to which types

For types: "license", "raid", "powersupply", "cache", "disk", "updates", "attacksbycountry", "attacksbytarget", "summary", "attacklog", "statuscodefortraffic", "methodfortraffic", "attacktypes", "cves", "attacksourcesbyip", "attacktargets", "traffic" - empty structure

For types: "requests", "performance", "diskperformance", "dnssrtt", "temperaturemetrics" :

Field	Type	Description
period	String	Defines the time interval for which information will be displayed. Available values: "hour", "day", "week", "month", "year"

For types: "throughput" :

Field	Type	Description
period	String	Defines the time interval for which information will be displayed. Available values: "hour", "day", "week", "month", "year"
iface	String	Network interface

For types: "netlink" :

Field	Type	Description
checkedList	List	List of ports from interfaces

Dashboard

Structure that describes dashboard.

Fields:

id	UID	readonly dashboard identifier
name	String	required on create dashboard name
kind	String	defines the dashboard type. The type can be either "waf" or the empty string.
columnWidths	List<Double>	the width of the widgets in each column. The first value applies to the first column, the second value applies to the second column and so on.
items	List<DashboardWidget>	list of widgets used in the dashboard
timestamp	Integer	timestamp for dashboards
syncCharts	Boolean	sync widget updates
timezoneOffset	Integer	time zone offset from UTC in seconds
autoReload	Variant	specifies the interval for automatic updating of information in widgets. Possible values: "manual" or milliseconds

DashboardWidgetInfo

Structure that describes available widget for dashboard.

Fields:

id	UID	readonly object identifier
-----------	-----	----------------------------

name	String	readonly object name
-------------	--------	----------------------

DefaultFirewallPolicyInfo

Structure that describes default firewall policy.

Fields:

enabled	Boolean	
log_session_start	Boolean	set <code>true</code> to log session start only
limit	Boolean	enable limit logging
limit_value	String	<code>"1/s"</code> , <code>"1/m"</code> , <code>"1/h"</code> , <code>"1/d"</code> , default = <code>"3/h"</code>
limit_burst	Integer	guaranteed number of events to be logged

DevicePowerSupplyInfo

Structure that describes power supply unit info.

Fields:

device_number	Integer	power supply unit number/ID
status	String	status, one of the following: <code>"on"</code> , <code>"off"</code> , <code>"unknown"</code>

DiagnosticNetworkInterfaceInfo

Network interface Info for diagnostic

Fields:

rx_bytes	String	RX bytes
rx_dropped	String	RX dropped
rx_errors	String	RX errors
rx_frame	String	RX frame
rx_overruns	String	RX overruns
rx_packets	String	RX packets
tx_bytes	String	TX bytes
tx_carrier	String	TX carrier
tx_collisions	String	TX collisions
tx_dropped	String	TX dropped
tx_errors	String	TX errors
tx_overruns	String	TX overruns
tx_packets	String	TX packets
enabled	Boolean	is enabled
ifindex	Integer	ifindex
ipv4	List<String>	IPv4 addreses list

ipv6	List<String>	IPv6 addreses list
kind	String	kind of interface
mac	String	mac address
master	String	master
mode	String	static/dhcp
mtu	Integer	mtu
name	String	interface name
peer	String	peer
qlen	Integer	qlen
running	Boolean	is running
speed	Integer	speed
xdp_info	String	xdp_info
zone	Integer	zone number

DnsQueryInfo

Structure that describes results of DNS lookup.

Fields:

txt_data	String	represents the answers that are returned from the name server(s) that were queried
-----------------	--------	--

DnsQueryParams

Structure with parameters for DNS lookup

Fields:

host	String	required the name of the resource record that is to be looked up
dns_server	String	required name or IP address of the DNS server
source_ip	String	source IP address of the query. The default value <code>0.0.0.0</code> is used if the parameter not provided
query_port	Integer	DNS server port. The value must be between <code>1</code> and <code>65535</code> . The default value <code>53</code> is used if the parameter not provided
query_type	String	the resource record type to query. Must be one of the following string values: <code>"A"</code> , <code>"AXFR"</code> , <code>"MX"</code> , <code>"NS"</code> . The default value <code>"A"</code> is used if the parameter not provided
timeout	Integer	query timeout in seconds. The value must be between <code>1</code> and <code>3600</code> . The default value <code>60</code> is used if the parameter not provided

EventComponent

String value that describes program “component” or “part” that generated the event.

- `accounts` — "Operations on users, groups, auth servers etc"
- `bandwidth` — "Operations on shaper rules/pools"
- `decryption_rules` — "Decryption rules"

- `device` — "Device operations"
- `firewall` — "Firewall, includes firewall rules"
- `console_auth` — Authentication events for admin console (user sign-in etc)
- `captiveportal` — "Captive portal events: user sign-in via captive etc"
- `content_rules` — "Content rules"
- `idps_rules` — "IDPS rules"
- `libraries` — "Operations with libraries: added, changed, deleted objects"
- `nat` , "NAT"
- `network` — "Network settings (interfaces, zones, gateways, DHCP etc)"
- `safebrowsing_rules` — "Safebrowsing rules"
- `settings` — Server settings
- `updater` — "Software updates"

EventInfo

Structure that contains information about event.

Fields:

time	Date	time when event has occurred
component	EventComponent	event component
type	EventType	event type
severity	EventSeverity	event severity
ip	String	ip address of users machine
origin	String	event origin, possible values are: <code>"admin_console"</code> , <code>"core"</code> , <code>"cpa_auth_form"</code> , <code>"cli_console"</code> , <code>"blockpage"</code>
attributes	Struct	struct with arbitrary structured data for the event

Attributes list could also include the following elements:

- `login` — login name of the user who has performed that operation from web console
- `user_id` — user identifier who has performed that operation from web console

EventSeverity

String value, possible values are:

- `"critical"` — Critical error
- `"error"` — Error
- `"warning"` — Warning
- `"info"` — Informational message

EventTrackerInfo

Structure that contains event tracker configuration.

Fields:

enabled	Boolean
----------------	---------

event_tracker_types	List<String>	list of label for event types
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

EventType

String value that describes event kind. Usually event type depends on component. See list of possible arguments in the notes to type [EventInfo](#)

- `"updated"` — "Some object has been updated, settings value for example"
- `"rule_updated"` — "Some rule has been updated"
- `"login_failed"` — "User authentication failed"
- `"login_successful"` — "Login successful"

FailoverConfigInfo

Structure that describes failover configuration options.

Fields:

enabled	Boolean	defines is connectivity checker enabled
period	Integer	checking interval, in seconds, allowed values are from <code>5</code> to <code>300</code>
percent	Integer	allowed failure rate, <code>10</code> - <code>100</code>
hosts	List<String>	IP addresses of hosts to monitor

FirewallRuleInfo

Structure that describes firewall rule.

Fields:

id	UID	readonly rule identifier
guid	UID	readonly unique rule identifier
position	Variant	object position in whole list, integer value starting with <code>1</code> , or string value <code>"top"</code> to move to the top of the list, or <code>"last"</code> to move to the bottom of the list. This field is used only for adding new objects and ignored when you update object. To change this field use one of corresponding move-methods.
position_layer	String	readonly rule position layer, possible values are: <code>"pre"</code> , <code>"post"</code> , <code>"local"</code> . <code>"pre"</code> and <code>"post"</code> rules are fetched from MC, <code>"local"</code> are rules create locally.
action	String	rule action, allowed values are: <code>"accept"</code> , <code>"drop"</code>
enabled	Boolean	is rule enabled
active	Boolean	readonly contains <code>true</code> if current rule at this moment is matched by time set in time_restrictions
name	String	required on create human readable rule name
description	String	detailed rule description, could be empty
src_zones	List<UID>	identifiers of source zone

src_zones_negate	Boolean	
dst_zones	List<UID>	identifiers of destination zone
dst_zones_negate	Boolean	
src_ips	List<TaggedValue>	list of tagged values that define source addresses, allowed tags are: "ip" , "list_id" (named list of type "network"), "geoip_code" , "urllist_id" (named list of type "url")
src_ips_negate	Boolean	
dst_ips	List<TaggedValue>	list of tagged values that define destination addresses, allowed tags are: "ip" , "list_id" (named list of type "network"), "geoip_code" , "urllist_id" (named list of type "url")
dst_ips_negate	Boolean	
log	Boolean	enable logging
log_session_start	Boolean	set <code>true</code> to log session start only
limit	Boolean	enable limit logging
limit_value	String	"1/s" , "1/m" , "1/h" , "1/d" , default = "3/h"
limit_burst	Integer	guaranteed number of events to be logged
time_restrictions	List<UID>	list of identifiers of time restrictions, if empty then rules has no time restrictions
time_created	Date	readonly rule creation time
time_updated	Date	readonly time of the last rule update
send_host_icmp	String	Defines whether the ICMP event should be sent when rule is triggered. Allowed values are: "" (empty string) — do not send ICMP event, "host-unreach" — send ICMP host unreachable, "tcp-rst" — send tcp reset, "tcp-reset-both" — send tcp reset to both parties
fragmented	String	defines packets this rule should be applied to, allowed values are: "yes" — apply only to fragmented packets, "no" — apply only to not fragmented packets, "ignore" — apply to any packets

FlowSessionIn

Substruct from FlowResult.

Fields:

src_ip	String	source IP address
src_port	Integer	source port
dst_port	Integer	destination port
proto	Integer	proto
ifindex	Integer	ifindex
dst_ip	String	destination IP address
ifname	String	interface name
vlan	Integer	vlan

FlowSessionOut

Substruct from FlowResult.

Fields:

dst_mac	String	destination MAC address
vlan	Integer	vlan
ktime	String	ktime
action	Integer	action
src_port	Integer	source port
src_ip	String	source IP address
state	Integer	state
dst_port	Integer	destination port
ifindex	Integer	ifindex
dst_ip	String	destination IP address
ifname	String	interface name
src_mac	String	source MAC address

FlowResult

Result from flow.list

Fields:

in	FlowSessionIn	input packet params
out	FlowSessionOut	output packet params

GatewayInfo

Describes information about gateway.

Fields:

id	UID	readonly gateway identifier
enabled	Boolean	is gateway enabled
name	String	required on create gateway name
description	String	gateway description
ipv4	String	required on create gateway IP address
weight	Integer	gateway weight
node_name	String	readonly node name
default	Boolean	is gateway default
multigate	Boolean	set to <code>true</code> to include gateway in balancing scheme that uses value from field weight to balance load
active	Boolean	readonly is gateway active (false if link is down)

iface	String	network interface name or special value "undefined" that indicates automatically selected interface
mac	String	readonly gateway MAC-address
vrf	String	required on create name of virtual router (not ID)
protocol	Integer	readonly gateway protocol
is_automatic	Boolean	readonly contains <code>true</code> if gateway is added automatically, for example via DHCP
_appliance_iface	String	readonly actual interface name, when field iface is set to "undefined" this field contains interface that was automatically selected by the system
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

GeoipCountry

Describes geoip country.

Fields:

id	String	object identifier
code	String	country code
name	String	country name (in english)

HAClusterEventFilter

Structure that describes HA Cluster event filter.

Fields:

start_date	Date	optional parameter: date and time from which receive events
limit	Integer	optional parameter: events limit for each node

HAClusterEvent

Structure that describes HA Cluster change state event.

Fields:

timestamp	Date	event date and time
new_status	String	new node status, allowed values are: "master", "backup", "undefined"
old_status	String	old node status, allowed values are: "master", "backup", "undefined"

HAClusterNodesEvents

Structure that describes HA Cluster change state event.

Fields:

nodeName	List< HAClusterEvent >	field name contains node name
-----------------	--	-------------------------------

HAClusterState

A structure that describes the state of the HA Cluster

Fields:

name	String	cluster name
nodes	List< HaClusterNodeState >	HA Cluster nodes status
virtual_ips	List< HAClusterVirtualIpInfo >	list of IP addresses from this HA Cluster

HaClusterNodeState

A structure that describes the state of the HA cluster node

Fields:

status	String	node status, allowed values are: "master" , "backup"
interfaces	List< HAClusterInterface >	HA Cluster interfaces info
state_change	List< HAClusterEvent >	HA Cluster change event
node	String	node name
display_name	String	display node name

HAClusterFailover

Structure that describes HA Cluster failovers.

Fields:

failover_count	Integer	failovers total
nodes	HAClusterNodesEvents	nodes failovers history

HAClusterInterface

Structure that describes HA Cluster link interface.

Fields:

ipv4	List<String>	list of IP addresses from this interface
name	String	interface name

HAClusterInfo

Structure that describes single HA Cluster.

Fields:

id	UID	readonly cluster identifier
name	String	required on create
description	String	

enabled	Boolean	
session_sync_enabled	Boolean	set <code>true</code> to enable network session synchronization including protocols TCP, SCTP and DCCP for cluster
session_sync_all	Boolean	synchronize all sessions including protocols UDP and ICMP in addition to protocols from <code>session_sync_enabled</code>
track_interfaces_enabled	Boolean	interface tracking
excluded_sync_ips	List<String>	list of ip addresses excluded from session synchronization
advertise_time	Integer	advertise time interval in seconds. It can take values from 0 to 120
master_down_timer	Integer	master down timer intervals count. It can take values from 0 to 100
preemption_delay	Integer	preemption delay interval in seconds. It can take values from 0 to 900
use_vmac	Boolean	the MAC address of the master node does not change, no GARP is required
no_preempt	Boolean	disable transition master node by priority
delivery_mode	String	delivery mode, allowed values: <code>"unicast"</code> , <code>"multicast"</code>
multicast_sync_id	Integer	can take values from 0 to 8
vrid	Integer	can take values from 1 to 255
mode	String	cluster mode, allowed values: <code>"active_active"</code> , <code>"active_passive"</code>
nodes	List<HAClusterNodeInfo>	required on create list of nodes HA cluster consists of. At least two nodes are required
virtual_ips	List<HAClusterVirtualIpInfo>	list of virtual IP address definitions
cloud	HAClusterCloud	settings for operation in a cloud environment
checkers	HAClusterCheckersInfo	the structure containing the checker settings
real_master_node	List<String>	
nodes_states	List<HAClusterNodeInfoState>	
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

HAClusterCloud

Structure that describes HA cluster settings for operation in a cloud environment.

Fields:

provider	String	cloud service provider, allowed values: <code>"none"</code> , <code>"yandex_cloud"</code>
yandex_cloud	Variant	settings for Yandex Cloud, allowed values: List or <code>false</code>

If cloud service provider is not `"none"`, corresponding settings field cannot be set to `false`.

HAClusterYandexCloud

Structure that describes Yandex Cloud specific settings for HA cluster.

Fields:

routes	List< HAClusterYandexCloudRoute >	routes to update on node status change
---------------	---	--

HAClusterYandexCloudRoute

Structure that describes an entry in [Yandex VPC route table](#) to update on node status change.

Fields:

route_table_id	String	required on create ID of Yandex VPC route table
destination_prefix	String	required on create destination prefix
next_hops	List< HAClusterYandexCloudNextHop >	required on create pair of next hops to switch between them

HAClusterYandexCloudNextHop

Fields:

node_name	String	required on create name of the node next hop address belongs to
ipv4	String	required on create next hop address

HAClusterBaseCheckerInfo

Structure that describes basic checker settings for HA cluster checkers

Fields:

enabled	Boolean	
action	String	readonly It can only take the <code>"fault"</code> value

HAClusterGatewayAvailabilityCheckerInfo

Structure that describes gateway availability checker settings for HA cluster checkers

Fields:

enabled	Boolean	
action	String	action, allowed values: <code>"fault"</code> , <code>"change_priority"</code>
interval_time	Integer	interval time in seconds. It can take values from 1 to 120
fall_count	Integer	fall count. It can take values from 1 to 10
rise_count	Integer	rise count. It can take values from 1 to 10

HAClusterResourceCheckerInfo

Structure that describes resource checker settings for HA cluster checkers

Fields:

enabled	Boolean	
action	String	action, allowed values: "fault" , "change_priority"
resources	List<String>	resources list, It can take IPv4 and fqdn values. e.g. "192.168.12.3" and "example.com"
interval_time	Integer	interval time in seconds. It can take values from 1 to 120
fall_count	Integer	fall count. It can take values from 1 to 10
rise_count	Integer	rise count. It can take values from 1 to 10

HAClusterCheckersInfo

Structure that describes all checkers for HA cluster

Fields:

check_dns	HAClusterBaseCheckerInfo	base dns checker
check_http	HAClusterBaseCheckerInfo	base http checker
check_gateway_availability	HAClusterGatewayAvailabilityCheckerInfo	gateway availability checker
check_resource	HAClusterResourceCheckerInfo	resource checker

HAClusterInterfaceInfo

Structure that describes interface mappings for specific node

Fields:

node_name	String	required on create node name
ifname	String	required on create network interface name, for example, "port1"

HAClusterAvailableNodeInfo

Structure that describes available node

Fields:

name	String	node name system node name
display_name	String	human readable node name, may be empty
is_available	Boolean	contains true if node is available to use in HA cluster

HAClusterNodeInstanceInfo

Structure that describes instance params for HA cluster node

Fields:

heartbeat	String	required on create network interface name, for example, "port1"
unicast_vrrp_src_ip	String	IPv4 Address. Use only if parameter delivery_mode is equal to "unicast"
unicast_vrrp_peer	String	IPv4 Address. Required only if parameter delivery_mode is equal to "unicast"

HAClusterSessionSyncInfo

Structure that describes session sync for HA cluster node

Fields:

sync_iface	String	required on create network interface name, for example, "port1" . Required if parameter session_sync_enabled is equal to true
unicast_sync_src_ip	String	IPv4 Address. Use only if parameter delivery_mode is equal to "unicast"
unicast_sync_peer	String	IPv4 Address. Required only if parameter delivery_mode is equal to "unicast"

HAClusterTrackInterfacesInfo

Structure that describes track interfaces for fault and change priority status for HA cluster node

Fields:

fault	List<String>	list of network interface names, for example, ["port1"]
change_priority	List<String>	list of network interface names, for example, ["port1"]

HAClusterNodeInfo

Structure that describes HA Cluster node.

Fields:

node_name	String	required on create node name
is_master	Boolean	true if this node considered as a Master Node
instance_params	List<HAClusterNodeInstanceInfo>	required on create structs list with instance params
session_sync_params	HAClusterSessionSyncInfo	session sync params structure
track_interfaces	HAClusterTrackInterfacesInfo	track interfaces structure

HAClusterNodeInfoState

Structure that describes HA cluster node state.

Fields:

enabled	Boolean
uid	UID
static_uid	UID

cluster_status_node	String
node_name	String
started	Boolean
running	Boolean

HAClusterVirtualIpInfo

Structure that describes HA Cluster virtual IP address.

Fields:

ipv4	String	required on create virtual IP address with mask, e.g. "192.168.12.3/24"
interfaces	List< HAClusterIpInterfaceInfo >	required on create list of network interfaces (for each HA Cluster node) this IP address should be assigned to

HAClusterFailedCheckInfo

Structure that describes failed check

Fields:

date	String	date of transition of the check to the fail state
check	String	check name

HardwareRAIDInfo

Structure that describes hardware RAID.

Fields:

uid	String	
device_type	String	HDD or SAS
status	String	RAID device status, one of the following values: "optimal" , "degraded" , "rebuilding"
size	String	size of RAID device, e.g. "953334 MB"
raid_level	String	RAID level, e.g. "1"
block_size	String	RAID block size, e.g. "512 Bytes"
bootable	Boolean	bootable flag
rebuild_progress	Integer	contains rebuild progress (in percents) if status is rebuilding
segments	List< HardwareRAIDSegmentInfo >	list of RAID segments (physical disks)

HardwareRAIDSegmentInfo

Structure that describes RAID segment info.

Fields:

status	String	one of the following values: "missing" , "present" , "inconsistent" , "rebuilding"
size	String	size, e.g. "953869MB"
interface	String	device interface, SAS or SATA
device_number	Integer	number of device in RAID, starting with 0
serial_number	String	physical device serial number, e.g. "W4111ABC"

HostPortPair

Fields:

host	String
port	Integer

ItemUpdateInfo

Structure that describes system element's update status.

Fields:

id	String	identifier
version	String	current element's version
date	String	the date of last successful item's update

IPProtocolInfo

Describes IP protocol.

Fields:

protocol_id	Integer
name	String

JsonSchemasJsonProfileInfo

Fields:

property	String	Default: "any". oneOf: "all", "any"
schemas	List	default: []. If you add or update it's a list of Strings. If you get list or fetch it's a list of objects #{id: UUID, name: String}

JsonProfileInfo

Structure that describes JSON profile.

Fields:

id	UID	readonly unique profile identifier
name	String	required unique profile name
description	String	profile description
log	Boolean	is rule log enabled (default: false)
content_types	List<String>	default: ["application/json", "text/json"], maxItems: 10
action	String	default: "deny", oneOf: "deny", "pass"
response_profile_id	UID	response profile identifier ResponseProfileInfo.id (Default: -1)
response_profile_name	String	if response_profile_id is not -1, only for v1.json.profiles.list
json_schemas	JsonSchemasJsonProfileInfo	
check_limits	Boolean	default: False. If it's True, then at least one of the limits (max_body_size, max_depth, max_array_elements, max_object_members, max_string_length, max_key_length) must be chosen
max_body_size	Integer	default: 0. Maximum - 10240000, minimum - 0
max_depth	Integer	default: 0. Maximum - 256, minimum - 0
max_array_elements	Integer	default: 0. Maximum - 4096, minimum - 0
max_object_members	Integer	default: 0. Maximum - 4096, minimum - 0
max_string_length	Integer	default: 0. Maximum - 4096, minimum - 0
max_key_length	Integer	default: 0. Maximum - 1024, minimum - 0

JsonSchemaInfo

Structure that describes JSON schema.

Fields:

id	UID	readonly unique schema identifier
name	String	required unique schema name
description	String	schema description
version	String	schema version
schema	String	field is required for v1.json.schema.fetch and v1.json.schema.add, field is not required for v1.json.schemas.list and v1.json.schema.update
create_date	String	readonly return only for v1.json.schemas.list and v1.json.schema.fetch
update_date	String	readonly return only for v1.json.schemas.list and v1.json.schema.fetch

XmlSchemasXmlProfileInfo

Fields:

property	String	default: "any". oneOf: "all", "any"
schemas	List	default: []. If you add or update it's a list of Strings. If you get list or fetch it's a list of objects #{id: UUID, name: String}

XmlProfileInfo

Structure that describes XML profile.

Fields:

id	UID	readonly unique profile identifier
name	String	required unique profile name
description	String	profile description
log	Boolean	is rule log enabled (Default: false)
content_types	List<String>	default: ["text/xml", "application/soap+xml", "application/xml"], maxItems: 10
action	String	default: "deny", oneOf: "deny", "pass"
response_profile_id	UID	response profile identifier ResponseProfileInfo.id (Default: -1)
response_profile_name	String	if response_profile_id is not -1, only for v1.xml.profiles.list
xml_schemas	XmlSchemasXmlProfileInfo	
check_limits	Boolean	default: False. If it's True, then at least one of the limits (max_body_size, max_depth, max_elements, max_attributes, max_tag_len, max_attr_name_len, max_attr_value_len, max_data_len) must be chosen
max_body_size	Integer	default: 0. Maximum - 10240000, minimum - 0
max_depth	Integer	default: 0. Maximum - 256, minimum - 0
max_elements	Integer	default: 0. Maximum - 4096, minimum - 0
max_attributes	Integer	default: 0. Maximum - 4096, minimum - 0
max_tag_len	Integer	default: 0. Maximum - 1024, minimum - 0
max_attr_name_len	Integer	default: 0. Maximum - 1024, minimum - 0
max_attr_value_len	Integer	default: 0. Maximum - 4096, minimum - 0
max_data_len	Integer	default: 0. Maximum - 1048576, minimum - 0

XmlSchemaInfo

Structure that describes XML schema.

Fields:

id	UID	readonly unique schema identifier
name	String	required unique schema name
description	String	schema description
schema	String	field is required for v1.xml.schema.fetch and v1.xml.schema.add, field is not required for v1.xml.schemas.list and v1.xml.schema.update
create_date	String	readonly return only for v1.xml.schemas.list and v1.xml.schema.fetch
update_date	String	readonly return only for v1.xml.schemas.list and v1.xml.schema.fetch

LdapUserAccountInfo

Structure that describes LDAP user account

Fields:

id	UID	readonly object identifier
guid	UID	readonly deprecated unique identifier
name	String	readonly account name
login	List<String>	readonly list of possible login names
ldap_dn	String	readonly user DN

LdapGroupAccountInfo

Structure that describes LDAP group account

Fields:

id	UID	readonly group identifier
guid	UID	readonly group identifier, contains the same value as in id , field is left for compatibility
name	String	group name

LDAPServerInfo

Structure that describes LDAP server.

Fields:

id	UID	readonly LDAP server identifier
name	String	required on create Human readable server name
description	String	optional description
enabled	Boolean	use or not this ldap server for user & group synchronization
ssl	Boolean	use SSL for connection
bind_dn	String	Bind DN, for example, "CN=ldapaccessuser,CN=Users,DC=example,DC=com" or "domain.org\user"
password	String	writable password for bind_dn
address	String	server address (FQDN or IP address)
domains	List<String>	list of domains
kerberos_keytab_bin	Base64	writable Keytab of the domain
keytab_exists	Boolean	readonly contains <code>true</code> if keytab file is uploaded
roots	List<String>	list of LDAP search paths
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

LDAPServerTestInfo

Structure that describes results of LDAP server testing/probing.

Fields:

authentication	Variant	readonly Specifies was authentication successful. <code>true</code> , <code>false</code> , or one of reason codes: <code>"cannot_bind"</code>
connection	Variant	readonly Specifies was connection successful. <code>true</code> , <code>false</code> or one of reason codes: <code>"empty_password"</code> , <code>"root_not_found"</code>
keytab	String	readonly Specifies keytab status. <code>true</code> if keytab is valid or one of reason codes: <code>"invalid_data"</code> if keytab data format is invalid, <code>"server_not_responding"</code> if server specified in the keytab is not responding, <code>"invalid_domains"</code> , <code>"missing"</code> if keytab is not set

LicenseInfo

Structure that contains license information.

Fields:

pin_code	String	pin code value
reg_name	String	field is ignored and left for compatibility
user_name	String	field is ignored and left for compatibility
last_name	String	field is ignored and left for compatibility
email	String	field is ignored and left for compatibility
country	String	field is ignored and left for compatibility
region	String	field is ignored and left for compatibility
company	String	field is ignored and left for compatibility
address	String	field is ignored and left for compatibility
phone	String	field is ignored and left for compatibility
fax	String	field is ignored and left for compatibility
expiry_date	Variant	license expiration date, either Date object or String value <code>"infinity"</code> , if license is unlimited
license_type	String	type of the license, possible values: <code>"normal"</code> , <code>"not_registered"</code>
modules	List< LicenseModuleInfo >	list of license modules
product	String	product name
version	String	product version
product_serial	String	serial number of hardware or virtual appliance
product_device	String	type of hardware or virtual appliance
user_limit	Integer	maximum allowed users limit
cpu_limit	Integer	get the max number of cpu cores allowed by your license. If there is no license or if the license has no limit, return <code>0</code> .

LicenseModuleInfo

Structure that describes license module.

Fields:

name	String	
expiry	Variant	module expiration date (object of type Date) or String value "infinity"
version	String	module version additional module info

LldpConfig

Structure that describes LLDP configuration.

Fields:

enabled	Boolean	
tx_interval	Integer	must be in range 1-3600, default value 30
tx_hold	Integer	must be in range 1-100, default value 4

LldpProfileInfo

Structure that describes LLDP profile.

Fields:

id	UID	object identifier
name	String	required on create human readable profile name
description	String	detailed profile description description, could be empty
status	String	port status, allowed values are: "rx-and-tx", "rx-only", "tx-only"
cc	Boolean	readonly contains true if object is fetched from MC

LoginInfo

Structure that contains various session parameters. Main parameters:

Fields:

auth_token	String	authentication token
admin_id	UID	identifier of WAF administrator object (local, LDAP user or LDAP group administrator)
admin_login	String	actual administrator login name that was used for login
guid	UID	administrator GUID, in case of LDAP Group administrator this field contains GUID of LDAP user not group
session_cookie	String	session cookie
product	String	product name, it always contains the value "waf"
version	String	product version
node	String	configuration cluster node name

Structure can also contain other fields.

MulticastRouterInfo

Structure that describes multicast router. It's used inside structure [VirtualRouterInfo](#).

Fields:

enabled	Boolean
ecmp_rebalance	Boolean
use_ecmp	Boolean
join_prune_interval	Integer
keep_alive_timer	Integer
register_suppress_time	Integer
interfaces	List< MulticastRouterInterfaceInfo >
rpmappings	List< MulticastRouterRpmappingInfo >
spt_exclusions	List<String>
ssm_address_range	List<String>

MulticastRouterInterfaceInfo

Fields:

enabled	Boolean
name	String
pim_dr_priority	Integer
pim_hello_interval	Integer
use_igmp	Boolean
use_igmp_v2	Boolean
bfd_profile	UID identifier of bfd profile or value <code>-1</code> if profile is not set

MulticastRouterRpmappingInfo

Fields:

enabled	Boolean
name	String
groups	List<String>
ipv4	String

NamedListInfo

Structure that describes Named list.

Fields:

id	UID	readonly internal list identifier
guid	UID	GUID of the list
type	NamedListType	createonly required list type
name	String	required list display name
description	String	list description
url	String	origin URL of the list, system uses this to periodically update list content, if this field is set then lists becomes readonly and items cannot be added/updated/deleted using API methods
attributes	Struct	additional attributes specific to certain list type, see list below
version	Integer	readonly list version
last_update	Date	readonly last update date of the list (for remote lists)
editable	Boolean	readonly
enabled	Boolean	
schedule	String	update schedule string in CRON-like format
list_type_update	String	type of automatic update, allowed values are <code>"static"</code> , <code>"dynamic"</code>

Allowed attributes are:

hidden_data	Boolean	readonly list contents is hidden
readonly	Boolean	readonly list data (above) is read-only
readonly_data	Boolean	readonly list contents is read-only
threat_level	Integer	list's threat level <code>1..5</code>
threshold	Variant	Morphology list threshold, non-negative integer or one of theses strings: <code>"low"</code> , <code>"normal"</code> , <code>"high"</code>
list_compile_type	String	Describes the sensitivity of the list to the case. Values: <code>"case_sensitive"</code> , <code>"case_insensitive"</code> , <code>"domain"</code> . Only for lists urls.
variant	String	readonly target system version for internal list, defined only for internal lists

NamedListItemInfo

Structure that describes item of the named list. Named list type defines which fields this structure contains. Field **id** assigned automatically when item is added.

Each list may contain regular items that described in subsections below for each list type.

Also you can add special items that reference another list, in that case structure contains these additional fields:

list	UID	nested list identifier
value	String	readonly name of the nested list

`"url"`

Fields:

id	UID	readonly item identifier
-----------	-----	--------------------------

value	String	required URL
--------------	--------	--------------

"network"

Fields:

id	UID	readonly item identifier
value	String	required must be a valid IP address with optional numeric mask (e.g. "192.168.1.0/24") or IP addresses range (e.g. "192.168.2.3-192.168.2.23")
list	UID	required must be a id of other list of type "network"

You must pass one of fields **value** or **list**, but not both.

"httpcwl"

Fields:

id	UID	readonly item identifier
value	String	required item value, cache exclusion URL or domain

"useragent"

Fields:

id	UID	readonly item identifier
value	String	required user agent

"emailgroup"

Fields:

id	UID	readonly item identifier
value	String	required proper email value

"phonegroup"

Fields:

id	UID	readonly item identifier
value	String	required proper phone number value, it must contain 11 or more digits, may starts with optional plus ("+") character

"urlcategorygroup"

Fields:

id	UID	readonly item identifier
category_id	Integer	required URL category identifier
name	String	readonly URL category name
risk	Integer	readonly URL category threat level value from range 1..5

"timerrestrictiongroup"

Fields:

id	UID	readonly item identifier
name	String	required time restriction group name
type	String	required timeset item type, allowed values are: "daily", "weekly", "monthly", "fixed", "range", "span"
days	List<Integer>	, required if current type is one of the following: "weekly", "monthly", if current type set "weekly" allowed values are in range "1-7", if current type set "monthly" allowed values are in range "1-31"
time_from	String	required time string in the form "HH:MM"
time_to	String	required time string in the form "HH:MM"
fixed_date_from	String	date string in the form "YYYY-MM-DDT00:00:00" (time part is ignored), required if current type is one of the following: "range", "span"
fixed_date_to	String	date string in the form "YYYY-MM-DDT00:00:00" (time part is ignored), required if current type is one of the following: "range", "span"
fixed_date	String	date string in the form "YYYY-MM-DDT00:00:00" (time part is ignored), required if current type is "fixed"

"websocket_protocol"

Fields:

id	UID	readonly item identifier
value	String	required name websocket protocol

"websocket_extension"

Fields:

id	UID	readonly item identifier
value	String	required name websocket extension

NamedListType

String-based value that defines named list type.

Here are the possible values:

- "url" — List of URLs/domains
- "network" — List of IP addresses
- "websocket_protocol" — List of websocket protocols
- "websocket_extension" — List of websocket extensions
- "urlcategorygroup" — List of grouped URL categories
- "timerrestrictiongroup" — Time sets
- "httpcwl" — URLs list, exceptions from HTTP cache
- "useragent" — List of User-Agent strings

- "emailgroup" — List of e-mail addresses
- "phonegroup" — List of phone numbers
- "all" — Special value for using with requests to indicate that we need to fetch all named lists

NetflowProfileInfo

Structure that describes netflow profile.

Fields:

id	UID	readonly
name	String	required on create
description	String	
host	String	required on create netflow collector IP address
port	Integer	netflow collector port port number, value in rage 1..65535
protocol	Integer	netflow protocol version, allowed values are: 5 , 9 , 10
active_timeout	Integer	required on create active flow timeout, in seconds, must be in range 1..3600
inactive_timeout	Integer	required on create inactive flow timeout, in seconds, must be in range 1..3600
inactive_timeout	Integer	inactive flow timeout, in seconds
refresh_rate	Integer	required on create template refresh rate (packets), must be in range 1..999
timeout_rate	Integer	required on create timeout to re-send old template, in seconds, must be in range 1..3600
maxflows	Integer	maximum flows required on create
natevents	Boolean	true to send NAT information
cc	Boolean	readonly contains true if object is fetched from MC

NetworkInterfaceDhcpRelayInfo

Structure that describes DHCP relay for network interface.

Fields:

enabled	Boolean	DHCP relay enabled state
host_ipv4	String	IP address that should be passed to DHCP relay
servers	List<String>	list of DHCP servers

NetworkInterfaceBondingInfo

Structure that describes bonding for network interface.

Fields:

mode	Integer	see “bonding modes” table below
miimon	Integer	required on create MII monitoring perion in milliseconds
downdelay	Integer	required on create down delay in milliseconds

updelay	Integer	required on create up delay in milliseconds
lacp_rate	Integer	LACP rate, allowed values are: <code>0</code> — slow, <code>1</code> — fast
fail_over_mac	Integer	failover MAC, allowed values are: <code>0</code> — disabled, <code>1</code> — active, <code>2</code> — follow
xmit_hash_policy	Integer	Xmit hash, allowed values are: <code>0</code> — Layer 2, <code>1</code> — Layer 3+4, <code>2</code> — Layer 2+3
slaves	List<String>	required on create list of network interface bind consists of

bonding modes

mode	description
0	Round robin
1	Active backup
2	XOR
3	Broadcast
4	IEEE 802.3ad
5	Adaptive transmit load balancing
6	Adaptive load

NetworkInterfaceBridgingInfo

Structure that describes network interface bridging

Fields:

ports	List<String>	names of interfaces that bridge in made of
forward_delay	Integer	forward delay, in seconds
max_age	Integer	max age, in seconds
stp_state	Boolean	STP state
bridge_type	String	type of bridge, allowed values: <code>"13"</code> , <code>"12"</code> , default value <code>"13"</code>
bypass_slot	Integer	bypass slot ID, <code>0</code> by default, valid only for hardware devices that support bypass on hardware level

NetworkInterfaceInfo

Structure that describes network interface (physical or VLAN).

Fields:

id	UID	readonly interface identifier, readonly
zone_id	UID	identifier of network zone (see NetworkZoneInfo), must be <code>0</code> if zone is not set
name	String	readonly interface name (e.g. <code>"port1"</code>)
description	String	interface description
node_name	String	readonly name of the node this interface belongs to
enabled	Boolean	interface status: enabled/disabled

ipv4	List<String>	list of ipv4 interfaces (in CIDR notation), defined when kind is "adapter", "bridge", "vlan", "bond", "loopback"
mac	String	MAC-address of the interface, defined when kind is "adapter", "bridge", "vlan", "loopback" or "bond"
mode	String	method of obtaining address, allowed values: "dhcp", "static", "manual"
dhcp_default_gateway	Boolean	Retrieve default gateway from server, only when mode is "dhcp" .
mtu	Integer	interface MTU, see [NetworkInterfaceMTUInfo] for correct values
mss	Integer	interface MSS always less than MTU, see [NetworkInterfaceMTUInfo] for MSS gap. Default 0 or from 4 to the entered MTU value minus 40.
tso	Boolean	TCP Segmentation Offload status: enabled/disabled, only when kind is "adapter", "bridge" or "bond", ignored for interfaces of other types
running	Boolean	readonly interface state: connected or not connected
kind	String	readonly interface type, possible values are: "adapter", "bridge", "vlan", "bond", "loopback"
master	String	readonly name of the interface that uses this one (for example, bridge or bond), by default contains boolean false
link	String	master interface name for VLAN, must be set when kind is "vlan"
vlan_id	Integer	VLAN ID, must be set when kind is "vlan"
dhcp_relay	NetworkInterfaceDhcpRelayInfo	DHCP relay data, defined when kind is "adapter", "bridge", "vlan", "bond", "loopback"
bonding	NetworkInterfaceBondingInfo	describes bond configuration, defined only when kind is "bond"
bridging	NetworkInterfaceBridgingInfo	describes bridge configuration, defined only when kind is "bridge"
tap	Boolean	only when kind is "adapter", "loopback" or "vlan", if "true", it means "mirror" mode, else "l3", default value false
netflow_profile	UID	identifier of netflow profile or value "undefined" if profile is not set
lldp_profile	UID	identifier of LLDP profile or value "undefined" if profile is not set
speed	Integer	readonly current interface speed as it reported by the system, in Mb/s/sec
exist	Boolean	existence of interface
vni	Integer	

readonly used as a numerical designation of the physical interface on the FG (null if not FG). The field is absent when **kind** is "vpn" , "ppp" , "tunnel"

ifalias	String	alias of interface
conf_mode	[InterfaceConfigMode]	Interface config mode, defined on UserGate FG only when kind is "adapter" and name in range "port2..port19" , otherwise returns empty struct

Explanation of error codes from field errors:

Code	Description
"configuration_failed"	Interface configuration is invalid. For example, value from field ipv4 conflicts with another record.
"ipv4_is_different"	Actual address of the interfaces instance on that node differs from one defined in the record.
"enabled_is_different"	Actual ** state differs from one defined in the record.

NetworkZoneFloodProtectionProfileInfo

Structure that describes DOS/flood protection profile.

Fields:

enabled	Boolean	is profile enabled
kind	String	kind of profile, possible values: "syn" , "udp" , "icmp"
alert_threshold	Integer	threshold of packets/sec that triggers alert (write to log), value must be in range 0-200000
drop_threshold	Integer	threshold of packets/sec that triggers packet drop, value must be in range 0-200000
aggregate	Boolean	
excluded_ips	List<String>	list of exclusion IP addresses

NetworkZoneServiceAccessInfo

Fields:

enabled	Boolean	is record enabled
service_id	UID	identifier of the service, all allowed values are listed below.
allowed_ips	List<String>	list of allowed IPs/subnets (each value must be in the form of valid ip address or address with mask: x.x.x.x or x.x.x.x/x)

List of allowed service identifiers:

Identifier	Description
1	Ping
2	SNMP
5	Control XML-RPC

6	Cluster
7	VRRP
8	Administrative console
14	CLI over SSH
18	Reverse proxy
23	Log analyzer
26	SNMP proxy
29	NTP service
34	HTTPS RPC

NetworkZoneInfo

Structure that describes network zone.

Fields:

id	Integer	readonly zone identifier
name	String	required on create human readable zone name, unique string among all zones
description	String	
readonly	Boolean	readonly field is ignored and left for compatibility
enable_antispoof	Boolean	true to enable spoofing control
networks	List< TaggedValue >	list of networks for spoofing control
antispoof_invert	Boolean	invert logic of checking networks for spoofing control
dos_profiles	List< NetworkZoneFloodProtectionProfileInfo >	DoS protection profiles
services_access	List< NetworkZoneServiceAccessInfo >	list of records that control access to firewall services, there must be no more than one record for each service ID, no record means no access for that service.
sessions_limit_enabled	Boolean	is sessions limit for one IP address is enabled
sessions_limit_threshold	Integer	maximum number of allowed sessions
sessions_limit_exclusions	List< TaggedValue >	list of IP addresses lists, allowed tags are: "list_id" (named list of type "network")
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

NodeDesc

Structure that describes node of network cluster.

Fields:

name	String	readonly internal name of the node, unique in the cluster
description	String	optional description
display_name	String	display name of the node
cc_node_name	String	readonly name of this node in MC if WAF device connected to MC
status	String	readonly node status, possible values: "online" , "offline"
ip	String	IP address of the node, that is used by internal resolver to build cluster or string "none"
license	Boolean	readonly possible values are: "active" , "no_license"

NotificationProfileInfo

Structure that describes notification profile

Fields:

id	UID	readonly unique identifier
name	String	required on create profile name
description	String	profile description
type	String	required on create profile type, allowed values are: "smtp" (notification via E-mail), "smpp" (notification via SMS)
host	String	
port	Integer	
cc	Boolean	readonly contains true if object is fetched from MC

Additional field when **type** is "smtp" .

security	String	connection security, allowed values are: "none" , "starttls" , "ssl"
authentication	Boolean	
login	String	
password	String	writeonly

Additional field when **type** is "smpp" .

login	String	
password	String	writeonly
ssl	Boolean	
translation_rules	List< SmppPhoneTranslationRuleInfo >	list of phone translation rules
source_addr_ton	Integer	
source_addr_npi	Integer	
dest_addr_ton	Integer	
dest_addr_npi	Integer	

NtlmServerInfo

Structure that describes NTLM server.

Fields:

id	UID	readonly object identifier
name	String	
description	String	object description
enabled	Boolean	
ip	String	
domain	String	Windows domain
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

OspfAreaInfo

Structure that describes OSPF area.

Fields:

id	UID	readonly area identifier
description	String	
enabled	Boolean	
name	String	OSPF area name, name limited to 20 octets
cost	Integer	
area_id	Variant	valid IPv4 address or corresponding positive 32-bit integer
area_type	String	OSPF area type, allowed values are: <code>"normal"</code> , <code>"stub"</code> , <code>"nssa"</code>
no_summary	Boolean	
auth_type	String	authentication type, allowed values are: <code>"none"</code> , <code>"plain"</code> , <code>"digest"</code>
interfaces	List<UID>	list of OSPF interface IDs (field id of OspfInterfaceInfo)
virtual_links	List<String>	list of IP addresses (router-id)

OspfAuthInfo

Fields:

enabled	Boolean	
auth_type	String	authentication type, allowed values are: <code>"plain"</code> , <code>"digest"</code>
md5_id	Integer	secret key used to create the message digest
auth_key	String	authentication key, maximum length is 255 characters

OspfInterfaceInfo

Structure that describes OSPF interface.

Fields:

id	UID	readonly OSPF interface identifier
description	String	
enabled	Boolean	
description	String	
network_type	String	network type, allowed values are: "broadcast" , "point-to-multipoint" , "point-to-point" , "" (empty string)
is_passive	Boolean	
iface_id	String	network interface id (e.g. "port1")
cost	Integer	
priority	Integer	must be in range 1..255
hello_interval	Integer	
dead_interval	Integer	
retransmit_interval	Integer	
transmit_delay	Integer	
bfd_profile	UID	identifier of bfd profile or value -1 if profile is not set
auth_params	OspfAuthInfo	

OspfRouterInfo

Fields:

id	UID	readonly OSPF router identifier
router_id	String	router id, must be in IP address form
enabled	Boolean	
redistribute	List<String>	list of flags, allowed values are: "connected" , "static" , Ngfw
default_originate	Boolean	
metric	Integer	
areas	List< OspfAreaInfo >	
interfaces	List< OspfInterfaceInfo >	

PagingList

Structure that represents one page of large list of elements of type `Type` .

Fields:

items	List<Type>	list fragment items, usually corresponds to one page
count	Integer	total number of elements in the list when filtering is applied
total	Integer	optional total number of all elements in the list without any filtering applied

ParsedRoutes

Structure that represents result of routing details output.

Fields:

network	String	network
selected	Boolean	is selected
directly_connected	Boolean	is directly connected
vrf	String	vrf name
interface	String	interface name
type	String	type
gateway	String	gateway
weight	String	weight

PCAPFile

Structure that describes packet capturing file.

Fields:

id	UID	readonly unique identifier of file
name	String	filename
path	String	path to file on disk (return only for downloading)
size	Integer	file size in bytes
ctime	Date	when file was last modified
capturing	Boolean	is capturing is running

PCAPFilter

Structure that describes packet capturing filter.

Fields:

id	UID	readonly unique identifier of filter
name	String	required on create filter name for humans
source	String	in form "1.1.1.1", "1.1.1.1/24" or empty string
source_port	String	in form "25", "1000-2000" or empty string
destination	String	in form "1.1.1.1", "1.1.1.1/24" or empty string
destination_port	String	in form "25", "1000-2000" or empty string
ethernet_protocol	String	"ip", "ip6", "arp", "rarp", "atalk", "decnet", "sca", "lat", "mopdl", "moprc", "iso", "stp", "ipx", "netbeui"
ipv4_protocol	Integer	0-255 or empty string

PCAPRule

Structure that describes packet capturing rule.

Fields:

id	Integer	readonly unique identifier of rule
name	String	required on create rule name for humans
interfaces	List<Integer>	for which interfaces we run capturing
filters	List<Integer>	which filters we use for capturing
byte_count	Integer	Max amount of captured traffic in MB
file	String	pcap filename
capturing	Boolean	readonly is capturing is running (can be changed with start/stop methods)
last_failure	String	readonly text of last error

PublicBlockPageParams

Structure that contains some block page params.

Fields:

user_ip	String	readonly user IP
user_name	String	readonly user name
group_names	List<String>	readonly list group names
display_node_name	String	readonly display node name

RadiusServerInfo

Structure that describes radius server.

Fields:

id	UID	readonly unique identifier
name	String	required name of radius server
description	String	description
secret	String	writable required public secret
enabled	Boolean	use or not this radius server for authentication
addresses	List<HostPortPair>	required list of host, port pair
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

RemoteAdminInfo

Structure that describes remote administration settings details.

Fields:

enabled	Boolean	is radmin enabled
----------------	---------	-------------------

token	String	readonly authentication token
port	String	readonly port

ResponsePageTemplateInfo

Structure that describes response page template.

Fields:

id	UID	readonly identifier of response page template
name	String	required on create name of the template
description	String	long template description, optional
type	String	required on create response page type, see list of all available types below
default_template_name	String	name of embedded template that should be used as default/fallback, see list of all possible values below
default	Boolean	<code>true</code> if this template is default, only one template of each type can be set as default, default template cannot be deleted
last_update	Date	readonly date and time of last template update
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

Allowed template types and default template names:

- `"blockpage"` — Blockpage
 - `"blockpage_en"` , `"blockpage_ru"`
- `"network_error_page"` — Network error page
 - `"network_error_page_en"` , `"network_error_page_ru"`

ReverseProxyLoadBalancingRuleInfo

Structure that describes reverse proxy load balancing rule.

Fields:

id	UID	readonly record identifier
enabled	Boolean	
name	String	required on create profile name
description	String	description
profiles	List<UID>	list of identifiers of ReverseProxyProfileInfo records
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

ReverseProxyProfileInfo

Structure that describes reverse proxy profile.

Fields:

id	UID	readonly profile ID
-----------	-----	---------------------

guid	UID	readonly deprecated object identifier
name	String	profile name
description	String	description
address	String	server address
port	Integer	server port
is_https	Boolean	<code>true</code> if this server is a HTTPS server
check_certificate	Boolean	
is_tproxy	Boolean	
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

ReverseProxyRuleInfo

Structure that describes reverse proxy rule.

Fields:

id	UID	readonly rule ID
guid	UID	readonly deprecated object identifier
enabled	Boolean	
position	Variant	object position in whole list, integer value starting with <code>1</code> , or string value <code>"top"</code> to move to the top of the list, or <code>"last"</code> to move to the bottom of the list. This field is used only for adding new objects and ignored when you update object. To change this field use one of corresponding move-methods.
position_layer	String	readonly rule position layer, possible values are: <code>"pre"</code> , <code>"post"</code> , <code>"local"</code> . <code>"pre"</code> and <code>"post"</code> rules are fetched from MC, <code>"local"</code> are rules create locally.
name	String	required on create profile name
description	String	description
ports	List<Integer>	required on create list of listen ports
servers	List< TaggedValue >	required on create list of servers, each element must be either <code>["profile", PROFILE_ID]</code> or <code>["lbrule", LB_RULE_ID]</code> where <code>PROFILE_ID</code> — reverse proxy profile ID (as returned by v1.reverseproxy.profiles.list), <code>LB_RULE_ID</code> — reverse proxy load balancing rule (as returned by v1.reverseproxy.loadbalancing.rules.list). Only first item of the list will be used!
is_https	Boolean	if true then rule will match SSL connections only, otherwise no-SSL connections only
certificate_id	UID	required on create required on update when <code>is_https</code> is <code>true</code> . Certificate that should be used for <code>is_https</code> . Default value for http is <code>-1</code> .
src_zones	List<UID>	required on create list of incoming zone identifiers associated with this rule
src_zones_negate	Boolean	

src_ips	List<TaggedValue>	list of tagged values that define source addresses, allowed tags are: "list_id" (named list of type "network"), "geoip_code" , "urllist_id" (named list of type "url")
src_ips_negate	Boolean	
dst_ips	List<TaggedValue>	list of tagged values that define destination addresses, allowed tags are: "list_id" (named list of type "network"), "geoip_code" , "urllist_id" (named list of type "url")
dst_ips_negate	Boolean	
allowed_web_services	List<String>	required on create list of allowed web services of the host/path type
rewrite_path_enabled	Boolean	
rewrite_paths	List<Struct>	list of web services path rewriting rules of the host/path type, each element is a structure with fields from and to
user_agents	List<TaggedValue>	list of user agents, each item must be one of the following types: ["ua", USER_AGENT_STRING] , ["list_id", ID] , where USER_AGENT_STRING is a literal User-Agent string and ID is a named list identifier that contains User-Agent strings
user_agents_negate	Boolean	
ssl_profile_id	UID	required on create required on update when is_https is true . Identifier of SSL profile that should be used for https. Default value for http is 0 .
waf_profile_id	UID	WAF profile that should be used in this rule
json_profile_id	UID	JSON profile that should be used in this rule
proxy_pass_real_ip	Boolean	if true then rule will use HTTP headers to detect real IP address of client
real_ip_header	String	header name to detect real IP of client
xff_recursive	Boolean	recursive mode. This field can only be true when real_ip_header is X-Forwarded-For .
set_real_ip_from	List<TaggedValue>	defines trusted addresses that are known to send correct real_ip_header header. Allowed tags are: "list_id" (named list of type "network")
ip_to_xff_header	Boolean	IP address of previous node will be added in tail of real_ip_header header and passed to backend. This field can only be true when real_ip_header is X-Forwarded-For .
websocket_profile_id	UID	Websocket profile that should be used in this rule
base64_url_path	Boolean	
base64_parameters	Boolean	
base64_headers	Boolean	
base64_cookie	Boolean	
base64_body	Boolean	
base64_iterations	Integer	Number of decoding iterations
x_forwarded_proto	Boolean	Add X-Forwarded-Proto header to HTTP request
x_forwarded_host	Boolean	Add X-Forwarded-Host header to HTTP request
x_forwarded_port	Boolean	Add X-Forwarded-Port header to HTTP request

technology_detection	Boolean	Enable technology detection. To enable, you need to download the package.
-----------------------------	---------	---

RipRouterInfo

Structure that describes RIP router.

Fields:

enabled	Boolean	
administrative_distance	Integer	
default_metric	Integer	
default_originate	Boolean	
interfaces	List< RipRouterInterfaceInfo >	
networks	List<Struct>	each element is a structure with exactly one field from this list: ifname , ipv4
redistribute	List< RipRouterRedistributeInfo >	
version	Integer	

RipRouterInterfaceInfo

Fields:

name	String
passive_mode	Boolean
password	String
poisoned_reverse	Boolean
receive_version	Integer
send_version	Integer
split_horizone	Boolean

RipRouterRedistributeInfo

Fields:

kind	String	one of these values: "connected", "static", "kernel", "ospf", "bgp"
metric	Integer	

RouteInfo

Structure that describes route. It's used inside structure [VirtualRouterInfo](#).

Fields:

id	UID	readonly identifier
name	String	required on create route name

enabled	Boolean	is route enabled
description	String	route description
dest	String	required on create ip of destination
gateway	String	ip of gateway
ifname	UID	identifier of network interface, "undefined" if interface is not set
metric	Integer	weight of route
kind	String	route kind, allowed values are: "unicast", "unreachable", "prohibit", "blackhole"

RoutesInfo

Structure that describes routes for update.

Fields:

mask	RouteInfo	finding filter parameter as part of RouteInfo structure
data	RouteInfo	route parameter as RouteInfo structure

RoutersInfoParams

Structure with parameters for routers info methods

Fields:

vrf	String	vrf name
option	String	option to specify routers type

RoutingInfoParams

Structure with parameters for routing info methods

Fields:

vrf	String	vrf name
routes_type	String	one of the following values: "bgp", "ospf", "rip", "kernel", "connected", "summary", "route_ip"

QueryFilterInfo

Structure with filtering options

Fields:

query	String	contains query string, for example, "name ~ bot" will find objects with name containing word "bot"
--------------	--------	--

RulesChangeCounters

Structure rules changes counters

Fields:

vectors	Integer	vectors rebuild count
ports	Integer	ports updates for rule count
lists	Integer	nlists refresh count

SamldpServerInfo

Fields:

id	UID	readonly identifier
enabled	Boolean	
name	String	
description	String	
certificate_id	UID	if the certificate is not used, set the value to <code>0</code>
sso_url	String	
sso_http_binding	String	allowed values are: <code>"post"</code> , <code>"redirect"</code>
slo_url	String	
slo_http_binding	String	allowed values are: <code>"post"</code> , <code>"redirect"</code>
metadata_url	String	contains URL that can be used to fetch SAML IDP server configuration data using method v1.auth.saml.idp.server.download.metadata
external_metadata_url	String	readonly contains external URL that can be used to fetch SAML IDP server configuration from WAF server
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

ScenarioCondition

Structure that describes single condition for Scenario rule.

Required fields:

Fields:

kind	String	condition kind, each kind have it's own fields that are listed below.
-------------	--------	---

url_category

url_categories	List	list of URL category IDs, see [ContentRuleInfo] for details
-----------------------	------	---

virus_detection

This kind has no additional options.

application

apps	List	list of L7 application IDs or application group IDs, see FirewallRuleInfo for details
max_event_count	Integer	number of triggered events
count_interval	Integer	during interval, time period in minutes

ips

ips_tl	Integer	Threat level, possible values are: 1, 2, 3, 4, 5
---------------	---------	--

net_packet_size

packet_size	Integer	size of network packet in bytes
--------------------	---------	---------------------------------

sessions_per_ip

sessions_limit	Integer	number of maximum allowed sessions per IP
-----------------------	---------	---

traffic_limit

period	String	measurement period, allowed values are: "minute", "hour", "day", "week", "month"
traffic_limit	Integer	traffic limitation value in bytes

health_check

health_check_method	String	allowed values are: "ping", "dns", "get"
health_check_address	String	IP address of the server that should be checked
health_result	String	allowed values are: "positive", "negative"
max_event_count	Integer	number of triggered events
count_interval	Integer	during interval, time period in minutes
gateway	UID	network gateway identifier or value "undefined" to use default gateway
health_request_timeout	Integer	request timeout, in seconds
health_answer_timeout	Integer	response timeout, in seconds
health_request_address	String	FQDN for request
health_type_request	String	DNS query type, allowed values are: "a", "aaaa", "cname", "ns", "ptr"

ScenarioRuleTrigger

trigger value, one of the following strings:

- "one_user" — Trigger on one user that matched by scenario and corresponding rule
- "all_users" — Trigger on all users matched by corresponding rule

ScheduleElement

Structure that describes schedule.

Fields:

name	String	name of library
schedule	String	required on create

ServerTimeInfo

Structure that describes server time settings.

Fields:

utc_time	Date	complete date and time
ntp_enabled	Boolean	<code>true</code> if NTP sync enabled
ntp_servers	List<String>	list of NTP servers to use
ntp_synced	Boolean	readonly <code>true</code> if last NTP sync attempt was successful
local_time	Date	readonly Local time
timezone	String	readonly Time zone

SettingsExportRule

Structure that describes settings export rule

Fields:

id	UID	readonly rule identifier
enabled	Boolean	rule status: enabled/disabled
name	String	required rule name
description	String	rule description
server_type	String	required remote server type, allowed values are: <code>"ftp"</code> , <code>"ssh"</code>
server_address	String	required remote server address
server_port	Integer	required remote server port
server_login	String	required remote server user login name
server_password	String	remote server user password
server_path	String	required remote server path where logs should be uploaded
schedule	String	required schedule string in CRON-like format
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

SettingsImportParams

Structure that describes settings import parameters.

Fields:

nodes_mapping	List	each element is a list <code>[NODE_NAME, IMPORT_NODE_NAME]</code>
----------------------	------	---

`NODE_NAME` is a node name from current installation, `IMPORT_NODE_NAME` is a node name that should be used as a source during backup restore.

SettingsBackupDetails

Structure that describes settings backup file.

Fields:

version	String	
date	String	date for last update or create backup file
nodes	List<Struct>	

Each item in **nodes** field contains list of records with these fields:

name	String	node name
display_name	String	node display name

SnmpEngineIdInfo

Structure that describes SNMP engine ID.

Fields:

length	String	"dynamic" or "fixed"
type	String	"text", "octets", "ip4", "ip6", "mac"
data	String	length must be 27 bytes or less

SnmpEvent

Structure that describes single SNMP event.

Fields:

name	String	required on create event name, get list of allowed value through <code>v1.snmp.rule.events</code> method
-------------	--------	--

SnmpRuleInfo

Structure that describes SNMP rule.

Fields:

id	UID	readonly rule identifier
enabled	Boolean	is rule enabled
name	String	required on create human readable rule name
description	String	
community	String	community
context	String	context

version	Integer	SNMP protocol version, allowed values: 2 and 3
enabled_query	Boolean	true to enable querying
enabled_trap	Boolean	true to enable traps
trap_host	String	address of SNMP manager
trap_port	Integer	port of SNMP manager
snmp_security_profile	UID	identifier of snmp security profile, must be set (required on create) to valid value if version is 3, otherwise is ignored and should be set to integer value 0
events	List<SnmpEvent>	list of events tracked by this rule
cc	Boolean	readonly contains true if object is fetched from MC

SnmpSecurityProfileInfo

Structure that describes SNMP security profile.

Fields:

id	UID	readonly profile identifier
name	String	required on create human readable name
username	String	user name
auth_type	String	authentication/encryption type, allowed values are: "noauth" (no authentication, no encryption), "auth_no_private" (enabled authentication, no encryption), "auth_private" (both authentication and encryption enabled)
auth_alg	String	authentication algorithm, allowed values are: "md5", "sha", "sha224", "sha256", "sha384", "sha512"
auth_password	String	authentication password
private_alg	String	encryption algorithm, allowed values are: "aes", "des"
private_password	String	session encryption key
cc	Boolean	readonly contains true if object is fetched from MC

SnmpStringInfo

Structure that describes SNMP string.

Fields:

data	String	length must be between 0 and 255
-------------	--------	----------------------------------

SmppPhoneTranslationRuleInfo

Structure that describes phone number translation for SMPP.

Fields:

from	String	expression that describes what should be replaced in phone number (regex I guess)
-------------	--------	---

to	String	expression that describes resulting string of translation
-----------	--------	---

SortingSpec

Structure that defines sorter.

Fields:

property	String	
direction	String	sorting direction, allowed values are: "ASC" , "DESC"

SslProfileInfo

Structure that describes SSL Profile.

Fields:

id	UID	readonly object identifier
name	String	required on create human readable object name
description	String	detailed object description, could be empty
ssl_min_proto	String	minimum TLS protocol version, inclusive, allowed values are: "tlsv1" , "tlsv1.1" , "tlsv1.2" , "tlsv1.3"
ssl_max_proto	String	maximum TLS protocol version, inclusive, allowed values are: "tlsv1" , "tlsv1.1" , "tlsv1.2" , "tlsv1.3"
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC
supported_groups	List<String>	supported groups list, minimum 2 of allowed values: "secp256r1" , "secp384r1" , "secp521r1" , "ffdhe2048" , "ffdhe3072" , "ffdhe4096" , "ffdhe6144" , "ffdhe8192"
ssl_ciphers	List<String>	list of ciphers to use, see list of allowed values below.

```

"TLS_RSA_WITH_3DES_EDE_CBC_SHA"
"TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256"
"TLS_AES_128_GCM_SHA256"
"TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA"
"TLS_DHE_RSA_WITH_AES_128_GCM_SHA256"
"TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
"TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384"
"TLS_RSA_WITH_AES_256_CBC_SHA256"
"TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256"
"TLS_RSA_WITH_AES_256_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384"
"TLS_RSA_WITH_AES_128_CBC_SHA256"
"TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA"
"TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA"
"TLS_AES_128_CCM_SHA256"
"TLS_RSA_WITH_AES_128_CBC_SHA"
"TLS_RSA_WITH_AES_256_GCM_SHA384"
"TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256"
"TLS_DHE_DSS_WITH_AES_256_CBC_SHA256"
"TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384"
"TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA"
"TLS_DHE_DSS_WITH_AES_256_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256"
"TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384"
"TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA"
"TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA"
"TLS_DHE_DSS_WITH_AES_128_CBC_SHA256"
"TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384"
"TLS_RSA_WITH_AES_128_GCM_SHA256"
"TLS_DHE_DSS_WITH_AES_128_CBC_SHA"
"TLS_DHE_DSS_WITH_AES_256_GCM_SHA384"
"TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256"
"TLS_DHE_RSA_WITH_AES_256_CBC_SHA256"
"TLS_DHE_RSA_WITH_AES_256_CBC_SHA"
"TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA"
"TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256"
"TLS_ECDH_RSA_WITH_AES_256_CBC_SHA"
"TLS_DHE_RSA_WITH_AES_128_CBC_SHA256"
"TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256"
"TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384"
"TLS_AES_256_GCM_SHA384"
"TLS_DHE_RSA_WITH_AES_128_CBC_SHA"
"TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA"
"TLS_DHE_DSS_WITH_AES_128_GCM_SHA256"
"TLS_DHE_RSA_WITH_AES_256_GCM_SHA384"
"TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"
"TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA"
"TLS_ECDH_RSA_WITH_AES_128_CBC_SHA"
"TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384"

```

StatisticsMonitorInfo

Structure that describes single connection.

Fields:

ip_src	String	source ip address
ip_dst	String	destination ip address
user	UID	user identifier
name	String	user name
app	UID	app identifier
app_name	String	application name
threat	String	threat level
in	Integer	transfer rate in bit per second
out	Integer	transfer rate in bit per second

StatLogExportLogFormat

Structure that describes logtype and format for export.

Fields:

type	String	Log type, one of "events", "traffic", "webaccess"
format	String	One of supported formats "cef", "cef_compact", "json", "cee_json"

StatLogExportRuleInfo

Structure that describes statistics log-export rule.

Fields:

id	UID	readonly rule identifier
enabled	Boolean	rule status: enabled/disabled
name	String	required on create rule name
description	String	rule description
logs	List< StatLogExportLogFormat >	List of structs describing log type & format to be exported.
schedule	String	required on create schedule string in CRON-like format, for ftp or ssh transport
transport	Struct	required on create Transport to deliver logs, one of StatLogExportTransportFTP , StatLogExportTransportSSH , StatLogExportTransportSyslog
mc_id	UID	readonly Management Console assigned identifier
cc	Boolean	readonly from Management Console

StatSendOnceRuleInfo

A structure describing a one-time export of statistics logs.

Fields:

send_once_options	String	options for one-time sending of logs, valid values: fresh , range .
start_export_range	String	a string in the format "YYYY-MM-DDThh:mm:ssZ", indicating the moment from which a one-time sending of logs will be carried out.

end_export_range	String	a string in the format "YYYY-MM-DDThh:mm:ssZ", indicating the moment at which one-time sending of logs will be completed.
-------------------------	--------	---

StatLogExportTransportFTP

Transport to deliver logs via FTP connection

Fields:

type	String	required on create must be a string value "ftp"
target	String	remote server address
port	Integer	remote server port
passive	Boolean	enable or disable passive mode for FTP
login	String	remote server user login name
password	String	writelnonly remote server user password
path	String	remote server path where logs should be uploaded

StatLogExportTransportSSH

Transport to deliver logs via SSH connection

Fields:

type	String	required on create must be a string value "ssh"
target	String	remote server address
port	Integer	remote server port
login	String	remote server user login name
password	String	writelnonly remote server user password
path	String	remote server path where logs should be uploaded

StatLogExportTransportSyslog

Transport to deliver logs via Syslog

Fields:

type	String	required on create must be a string value "syslog"
target	String	remote server address
port	Integer	remote server port
transport	String	Syslog transport, one of "udp" , "tcp"
protocol	String	Syslog protocol, one of rfc3164 , rfc5424
severity	Integer	Syslog message severity value 0..7
facility	Integer	Syslog message facility value 0..23
hostname	String	Syslog message hostname value. Default is logan hostname.

app_name	String	Syslog message <code>app_name</code> value. Default is <code>utm-loganalyzer</code>
-----------------	--------	---

StatServerConfig

Statistics server configuration options.

Fields:

server_address	String	readonly
secure	Boolean	readonly

StatServerStatus

Statistics server connection status.

Fields:

alarms	List<Struct>	Alarms set on remote server: <code>{system_memory_high_watermark: ""}</code> , <code>{datadir_almost_full: String dir_path}</code>
last_error	String	optional Remote server last error
state	String	Remote server state, values are <code>"undefined"</code> , <code>"ready"</code> , <code>"installing"</code> , <code>"maintenance"</code> , <code>"need_upgrade"</code> , <code>"error"</code>
version	String	Remote server version
metrics	SyslogMetricsInfo	optional syslog server metrics data

SyslogMetricsInfo

Structure that describes syslog server metrics info.

Fields:

eps	Integer	syslog server incoming events per second value
------------	---------	--

SysStatCounters

Structure that contains core counters.

Fields:

adblock	Integer	ADBlock counter
dns_block	Integer	DNS Blocked
dns_cache_hit	Integer	DNS Cache hits
dns_request	Integer	DNS Requests
dns_response	Integer	DNS Responses (any response)
dns_response_fail	Integer	DNS failed
dns_response_noerr	Integer	DNS success
dns_response_nosuch	Integer	DNS not found

http_request	Integer	HTTP requests
http_cache_valid	Integer	HTTP Cache hits
http_cache_validate	Integer	HTTP Cache Validation hits
http_req_allow	Integer	HTTP request allowed
http_req_block	Integer	HTTP request blocked
http_resp_allow	Integer	HTTP response allowed
http_resp_block	Integer	HTTP response blocked

SysStatCountersMS

Structure that contains core counters.

Fields:

protocol_mail	Integer	Field name in the format <code>protocol:mail</code> , protocol allowed values are: <code>"pop"</code> , <code>"smtp"</code> , <code>"all"</code>
protocol_spam	Integer	Field name in the format <code>protocol:spam</code> , protocol allowed values are: <code>"pop"</code> , <code>"smtp"</code> , <code>"all"</code>
protocol_virus	Integer	Field name in the format <code>protocol:virus</code> , protocol allowed values are: <code>"pop"</code> , <code>"smtp"</code> , <code>"all"</code>

SysStatHttpCache

Structure that contains HTTP Cache usage statistics.

Fields:

disk_max	Integer	Maximum Disk size available in bytes
disk_used	Integer	Disk usage in bytes
ram_max	Integer	Maximum RAM available in bytes
ram_used	Integer	RAM usage in bytes

SysStatInterfaceStatus

Structure that describes network interface status.

Fields:

ip	List<String>	list of IP addresses assigned to interface
mask	String	network mask of interface
name	String	interface name
speed	Integer	interface speed as reported by system or value <code>nil</code> if speed is not defined
rx	Integer	number of received bytes
tx	Integer	number of transmitted bytes
stat	String	interface status, possible values are: <code>"up"</code> , <code>"down"</code>

SysStatDNSSRTT

Structure that contains SRTT data series for each used DNS server.

Fields:

DNSSERVER	List<Integer>	SRTT in msec
------------------	---------------	--------------

SysStatIfaceThroughput

Structure with data series for network interface.

Fields:

kb_read	List<Double>	Total number of kilobytes received per second
kb_write	List<Double>	Total number of kilobytes transmitted per second
pck_read	List<Double>	Total number of packets received per second
pck_write	List<Double>	Total number of packets transmitted per second

SysStatThroughput

Struct with network interfaces statistics.

Fields:

INTERFACENAME	SysStatIfaceThroughput	Network interface INTERFACENAME statistics
----------------------	--	--

SysStatIntervalStatistics

Structure that contains time series core counters statistics for interval.

Fields:

concurrent_users	List<Integer>	Amount of users connected
cpu_used	List<Integer>	% of CPU usage
disk_io	List<Double>	% of I/O Disks utilization
dns_requests_ps	List<Double>	DNS Requests per second
dns_srtt	SysStatDNSSRTT	per-DNS SRTT in msec
http_requests_ps	List<Double>	HTTP Requests per second
mem_free	List<Integer>	Amount of free memory available in kilobytes (kmemfree)
mem_cached	List<Integer>	Amount of cached memory in kilobytes (kbcached)
mem_used	List<Integer>	Amount of used memory in kilobytes (kmemused)
swap_free	List<Integer>	Amount of free swap space in kilobytes (kswapfree)
swap_used	List<Integer>	Amount of used swap space in kilobytes (kswapused)
throughput	SysStatThroughput	Network per-interface statistics
timestamp	List<Integer>	Timestamps in system posix time
therm_sensors	List<Variant>	

Temperature gathered from sensors. Can be Integer, Double or Null if sensor is not available

SysStatIntervalStatisticsMS

Structure that contains time series core counters statistics for interval.

Fields:

protocol_hits	List<Integer>	Field name in the format <code>protocol:hits</code>
timestamp	List<Integer>	

SysStatMemoryInfo

Current system memory usage information.

Fields:

mem_free	Integer	Amount of free memory in kilobytes (<code>/proc/meminfo</code> <code>MemFree</code>)
mem_cached	Integer	Amount of cached memory in kilobytes (<code>/proc/meminfo</code> <code>Cached</code>)
mem_total	Integer	Amount of total available memory in kilobytes (<code>/proc/meminfo</code> <code>MemTotal</code>)
swap_free	Integer	Amount of free swap space in kilobytes (<code>/proc/meminfo</code> <code>SwapFree</code>)
swap_total	Integer	Amount of total swap space in kilobytes (<code>/proc/meminfo</code> <code>SwapTotal</code>)

SysStatDiskInfo

Current disk usage statistics.

Each field is a Struct with fields **total** and **used**.

Fields:

var	Struct
root	Struct

Description of the structure used in fields **var** and **root**:

total	Integer
used	Integer

SysStatRuleInfoByNode

Rule statistics by node.

Fields:

node	String	node name
first_time	Integer	UTC timestamp (in microseconds by default) of first rule hit
last_time	Integer	UTC timestamp (in microseconds by default) of last rule hit

count	Integer	rule hits count
--------------	---------	-----------------

SysStatRules

Rules statistics.

Fields:

UID	List< SysStatRuleInfoByNode >	UID is rule uuid associated with SysStatRuleInfoByNode list
------------	---	---

StorageFileInfo

Structure that contains file information from the storage.

Fields:

content_type	String	File MIME-type as it was provided by web browser
original_filename	String	Original file name as it was provided by web browser
size	Integer	file size in bytes
storage_file_uid	UID	storage file identifier

StorageUploadSessionCheckInfo

Structure that contains results of uploading process status check.

Fields:

total_length	Integer	total length of file (in bytes) that is being uploaded right now
uploaded_length	Integer	how much bytes has been uploaded

StorageUploadSessionInfo

Structure that contains information about storage upload session.

Fields:

upload_url	String	URL that should be used by web browser for actual file upload
check_token	String	special string, that should be using to check uploading process status and progress
storage_file_uid	UID	internal identifier of the file in storage

TacacsPlusServerInfo

Structure that describes tacacs plus server.

Fields:

id	UID	readonly object identifier
enabled	Boolean	
name	String	required

description	String	description
use_single_connection	Boolean	
timeout	Integer	
address	String	required
port	Integer	
secret	String	writelnonly required
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

TaggedValue

This data item is a list that consists of two elements: TAG and VALUE. TAG is a string that defines type of the value. Here are some examples:

- `["ip", "192.168.12.4"]` — specifies single IP address
- `["geoip_code", "RU"]` — specifies geoip country
- `["urllist_id", 1451]` — specifies URL list identifier as returned by [v2.nlists.list](#)
- `["list_id", 1451]` — named list identifier as returned by [v2.nlists.list](#), type of list depends on context
- `["mac", "01:02:03:04:05:06"]` — specifies MAC address

UpdateSchedule

Structure contains update schedule

Fields:

software_schedule	String	schedule in crontab format for system update
lists_schedule	List< ScheduleElement >	list structures with name (name of library update) and schedule (schedule in crontab format) fields
lists_schedule_apply_for_all	Boolean	if set to <code>true</code> , all elements work on same schedule
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

UplSection

String value that defines objects category. Possible values are:

- `"firewall"` — Firewall rules
- `"reverseproxy"` — Reverse proxy rules
- `"reverseproxy.balancing"` — Reverse proxy load balancing

UplListResponse

Structure that describes response from UPL-methods that contains list of objects.

Fields:

body	String
-------------	--------

count

Integer

UserAuthMethodInfo

Structure describes authentication methods.

Fields:

type	String	required authentication type, one of the following values: "ldap", "kerberos", "radius", "local_user", "policy_accept", "tacacs_plus", "ntlm", "saml_idp", "basic"
ldap_server_id	UID	required identifier of LDAP server (as returned by v1.auth.ldap.servers.list , available only when type is "ldap")
radius_server_id	UID	required identifier of RADIUS server (as returned by v1.auth.radius.servers.list), available only when type is "radius"
tacacs_plus_server_id	UID	required identifier of TACACS PLUS server (as returned by v1.auth.tacacs.plus.server.list), available only when type is "tacacs_plus"
saml_idp_server_id	UID	required identifier of SAML server (as returned by v1.auth.saml.idp.servers.list), available only when type is "saml_idp"
ntlm_server_id	UID	required identifier of NTLM server (as returned by v1.auth.ntlm.server.list), available only when type is "ntlm"

UserAuthProfile

Fields:

id	UID	readonly profile identifier
name	String	required on create profile name
description	String	profile description
idle_time	Integer	user auth session idle timer, it resets each time user sends any network request. User session will be terminated in a idle_time seconds after last network request.
expiration_time	Integer	Expiration time (in seconds), session will be terminated in a expiration_time seconds after user authentication. Usually expiration_time is much greater than idle_time .
max_auth_attempts	Integer	number of allowed authentication attempts, when limit is reached all further attempts will be rejected
auth_lockout_time	Integer	time in seconds of authentication lockout (lockout is turned on when user failed a few authentication attempts, specified in field max_auth_attempts)
allowed_auth_methods	List< UserAuthMethodInfo >	required on create list of allowed authentication methods
2fa_profile_id	UID	identifier of 2FA profile, Boolean <code>false</code> if 2FA profile isn't set
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

VirtualRouterInfo

Structure that represents virtual router.

Fields:

id	UID	readonly object identifier
name	String	name of virtual router, name <code>"default"</code> is reserved and cannot be used to create a new virtual router
description	String	
node_name	String	readonly
interfaces	List<String>	list of network interfaces attached to this virtual router, for default router this list is always empty and cannot be updated
routes	List<RouteInfo>	list of static network routes
ospf	OspfRouterInfo	
bgp	[BgpRouterInfo]	
rip	RipRouterInfo	
pimsm	MulticastRouterInfo	
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

UpstreamProxyConfigInfo

Structure that describes upstream proxy configuration.

Fields:

enabled	Boolean	is upstream proxy enabled or not
mode	String	upstream proxy mode, allowed values are: <code>"socks5"</code> , <code>"https"</code>
ip	String	IP address of upstream proxy
port	Integer	port
auth	Boolean	authentication enabled
user	String	username
pass	String	writable password

WebuiAuthMode

Value that defines webui server authentication * `"password"` — Authentication using login and password * `"cert"` — Authentication with x509 certificates * structure that describes authentication with client certificate profile:

type	String	value <code>"pki"</code>
client_certificate_profile_id	UID	client certificate profile that should be used if type is set to <code>"pki"</code>

WafProfileInfo

Structure that describes WAF profile.

Fields:

id	UID	readonly profile identifier
-----------	-----	-----------------------------

name	String	required profile name
description	String	profile description
max_body_inspection_size	Integer	max value in bytes for body analysis. Default value 200000. Maximum - 2000000000, minimum - 1 bytes
layers	List< WafProfileLayerInfo >	required list of system and custom layers selected in current profile. WafProfileLayerInfo structures with "custom_layer" type should be placed at the beginning of the list. WafProfileLayerInfo structure with "system_rule_layer" type should be placed after the structures with "custom_layer" type. WafProfileLayerInfo structures with "system_rule_layer" type that describe the layers from same package of expertise should be located in one part of the list one after the other. It is not allowed to place WafProfileLayerInfo structures with "system_rule_layer" type that describe layers from different expertise packages one after the other.
protection_technologies	List<UID>	required list of selected protection technology ids
overridden_rules	List< WAFRulesOverrides >	list of system rules that were overwritten in the profile
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

WafProfileLayerInfo

Structure that describes selected layer in [WafProfileInfo](#) structure

Fields:

id	UID	required identifier of layer returned by v1.waf.custom.layers.list for custom layer or v1.waf.system.layers.list for system layer
type	String	required layer type, allowed values are: "custom_layer" , "system_rule_layer"

WafCustomLayerInfo

Structure that describes WAF custom layer.

Fields:

id	UID	readonly layer identifier
name	String	required layer name
description	String	layer description
upl	String	layer upl
cc	Boolean	readonly contains <code>true</code> if object is fetched from MC

WAFRulesOverrides

Structure of overridden rules.

Fields:

id	UID	required rule identifier
rule_log	Boolean	is rule log enabled
enabled	Boolean	is rule enabled
action	String	allowed values are: "pass" , "deny" , "force_deny" , "force_pass" , "no_action"
response_profile_id	UID	response profile identifier. This field can accept an empty string "". An empty string means that there is no profile in this override.
exception_ids	List<UID>	list of exception identifier

WafSystemRuleInfo

Structure that describes WAF system rule.

Fields:

id	UID	readonly rule identifier
name	String	readonly rule name
description	String	readonly rule description
package_id	UID	readonly identifier of the update package in which this rule was delivered
package_name	String	readonly name of the update package in which this rule was delivered
package_version	Integer	readonly version of the update package in which this rule was delivered
reference	List<String>	readonly list of rule references
threat_level	Integer	readonly threat level, allowed values are: 1 (lowest), 2 , 3 (highest)
technology_ids	List<UID>	readonly list of technology ids
layer_id	UID	readonly system layer id
last_update	Date	readonly date and time of last system rule update
rule_log	Boolean	is rule log enabled (default value true)
enabled	Boolean	is rule enabled (default value true)
action	String	allowed values are: "pass" , "deny" , "force_deny" , "force_pass" , "no_action"
response_profile_id	UID	response profile identifier. This field can accept an empty string "". An empty string means that there is no profile in this rule.
exception_ids	List<UID>	list of exception identifier

WafSystemLayerInfo

Structure that describes WAF system rule layer.

Fields:

id	UID	readonly object identifier
name	String	readonly object name
description	String	readonly object description in selected language
package_id	UID	

		readonly identifier of the update package in which this layer was delivered
package_name	String	readonly name of the update package in which this layer was delivered
package_version	Integer	readonly version of the update package in which this layer was delivered
version	Integer	readonly
rules_count	Integer	readonly count of all rules (with id) in layer
active_rules_count	Integer	readonly count of rules in layer which corresponding specified protection technologies and overrides.
default_protection_technologies	List<UID>	readonly list of default protection technologies identifiers for profile (as returned by v1.waf.system.rules.technologies.list)
default_threat_levels	List<Integer>	readonly default threat levels for layer, possible values are: 1 (lowest), 2, 3, 4, 5 (highest)
active_threat_levels	List<Integer>	readonly active threat levels in layer which corresponding specified protection technologies and overrides.

WafSystemRuleTechnologyInfo

Structure that describes WAF system rule technology.

Fields:

id	UID	readonly technology identifier
name	String	readonly technology name
category	String	readonly name of technology category

WafExceptionInfo

Structure that describes WAF exception.

Fields:

id	UID	object identifier
name	String	required on create object name
description	String	object description
log	Boolean	is exception log enabled
conditions	List< WafExceptionCondition >	list of conditions for exception

WafExceptionCondition

Structure that describes exception condition.

Fields:

name	String	required condition name. Available values: "request.header.Host", "url.path", "http.method", "src.ip", "request.header.User-Agent", "request.header.Referer", "request.header.Accept", "qparam",
-------------	--------	--

```
"request.header.Cookie", "request.body", "request.body.nocase",
"response.body", "response.body.nocase", "request.body.ugregex",
"response.body.ugregex", "upl.condition"
```

operator	String	condition operator. Available values: "=", "!=" , "~" , "!~" . required for all name values excluding "upl.condition" .
values	List<String>	required values with which the conditions are compared. If the name field is "http.method" , then allowed values are http method names in uppercase.
key	String	optional key of cookie or query params. This field available only when name field is "qparam" or "request.header.Cookie"

WebsocketProfileInfo

Structure that describes websocket profile.

Fields:

id	UID	readonly profile identifier
name	String	required on create profile name
description	String	profile description
block	Boolean	required on create is block websocket request
check_request	Boolean	is validation request enabled
check_origin	Boolean	is check origin enabled
origin_negate	Boolean	
origin	List<TaggedValue>	list of tagged values that define source addresses, allowed tags are: "ip" , "list_id" (named list of type "network"), "urllist_id" (named list of type "url")
check_protocol	Boolean	is validation protocol enabled
protocol_negate	Boolean	
protocol	List<TaggedValue>	list of tagged values that define websocket protocol, allowed tags are: "protocol_list_id" (named list of type "websocket_protocol")
check_extension	Boolean	is validation extension enabled
extension_negate	Boolean	
extension	List<TaggedValue>	list of tagged values that define websocket extension, allowed tags are: "extension_list_id" (named list of type "websocket_extensions")
log	Boolean	required on create is log enabled

ResponseProfileInfo

Structure that describes response profile.

Fields:

id	UID	readonly profile identifier
name	String	required on create profile name
description	String	profile description

response_type	String	required on create type of response, allowed tags are: "redirect" , "tcp_rst" , "response_page"
response_code	Integer	required on create if "response_type" = "response_page" or "response_type" = "redirect" code response. For "response_type" = "redirect" use 3XX, "response_type" = "response_page" use 1xx, 2xx, 4xx, 5xx
response_text	String	text for response. Maximum number of characters = 128
custom_redirect	String	required on create if "response_type" = "redirect" . Url for redirect. Maximum number of characters = 256
blockpage_template_id	UID	required on create if "response_type" = "response_page" . Integer identifier of response page template of type "blockpage" that should be used. The default value -1

API errors

This section contains list of all possible error codes with detailed explanation. Every error has required error code — a numeric integer positive value and optional error string that informally describes an error.

Errors with the numbers above 1 000 000 are reserved and not used in this API.

General errors

Code	Typical error string	Detailed explanation
1	Undefined API method	
2	Unknown error	General unknown error
102	Access denied	You don't have enough permissions to call this method.
104	Authentication required	Request not authenticated due to missing, invalid, or expired auth token.
105	Group or user not found	
106	Failed to establish connection with master node	
107	Parameter not found	Parameter with given name is not found in the settings.
108	Users limit reached	
109	Unknown login name or password	Used in functions that perform authentication
110	Incorrect method argument	
111	Some field value in passed structure is invalid	
112	Invalid method format	Incorrect argument type or incorrect number of arguments.
301	Licensing server error: database error	
302	Licensing server error: pincode not found	
303	Licensing server error: pincode is not active	
304	Licensing server error: activations number limit is reached	
305	Licensing server error: module not found	
306	Licensing server error: source pincode not found	
307	Licensing server error: new pincode not found	
308	Licensing server error: new pincode is already activated	
309	Licensing server error: new pincode is not active	
310	Licensing server error: failed to generate license key	
311	Licensing server error: failed to send email	
312	Licensing server error: version mismatch	
313	Licensing server error: unknown user	
314	Licensing server error: source pincode is already converted to a new one	

315	Licensing server error: no more CT keys	
316	Licensing server error: old and new pincode products doesn't match	
317	Licensing server error: attempt to activate pincode without old pincode	
318	Licensing server error: incorrect variable	
319	Licensing server error: events mismatch	
320	Licensing server error: pincode doesn't match the program	
321	Licensing server error: transport_error, Reply was not received	
322	Licensing server error: transport_error, Wrong user data	
323	Licensing server error: transport_error, There is no enough data	
324	Licensing server error: transport_error, Wrong action for this type of the license	
325	Licensing server error: transport_error, Server's response validation failed	
326	Licensing server error: transport_error, There are no modules to process	
327	Licensing server error: transport_error, Wrong license type while creating module request	
351	Licensing server error: not supported method	
352	Licensing server error: license/key is expired	
353	Licensing server error: limit of cluster nodes is reached	
354	Licensing server error: License is not available for this client	
355	Licensing server error: Hardware id not found	
356	Incorrect value in "position" argument	
404	Object not found	Used as a general "Not found" error.
408	Maximum objects limit is reached	
409	Duplicate value, conflict with the existing object	Used to indicate conflict with already existing object.
501	Unknown cluster secret	
502	Object deletion failed due to constraints	Object cannot be deleted because it's being used by another object
513	Incorrect object field value	
601	Missing required field	Required field in the passed structure is missing
602	Incorrect field data in UPL data	Passed field value in UPL string contains data not allowed for that field
603	Incorrect paging parameters	Incorrect paging parameters are passed into the function, e.g. range boundaries
604	Service is starting	
606	Not implemented	Method not yet implemented

607	License server error	License server error #xxx
608	Communication with appliance error	
609	Error while processing multiple objects	
610	Temp dir is unavailable	
611	Internal service error	Communication with one of internal services (via XML-RPC) has failed
612	Invalid offline activation data	
613	Cannot connect to remote admin server	
614	Config locked	
615	Maximum allowed GEOIP records limit is reached	
616	Invalid node	
617	Blocked by active sync with MC	
618	Incorrect search query expression	
619	GeoIP not allowed for this mode of operations	

Netmanager errors

Code	Typical error string	Detailed explanation
1001	Specified master interface not found	
1002	Unknown interface	
1003	Physical interface cannot be deleted	
1005	Unknown leased record ID	
1006	You cannot update readonly zone	
1007	You cannot delete readonly zone	
1009	Failed to fetch connections list	
1010	Unknown interface	unknown interface for routing operations: OSPF, BGP etc
1011	Failed to fetch traffic records	
1012	Incorrect GUID	
1013	DHCP conflict	
1014	Slave interface cannot be updated	Slave interface in bridge, bond etc cannot be modified.
1015	Generic Virtual router error	Error message contain specific details.
1016	Virtual router interface is already assigned to another virtual router	
1017	DHCP ip range conflict	
1019	Gateway IP address conflict	
1020	OSPF interface conflicts with another virtual router	
1021	OSPF area_id conflict	More than one area with the same area_id is defined in the same virtual router
1022	Virtual router network interface conflict	One or more network interfaces defined in the virtual router conflicts with interfaces in other virtual routers

1023	Failed to generate report
1202	Bridge/Bond sub interface usage in other interface
1203	Bridge interface must have exactly two adapter sub interfaces or one bypass interface
1204	Cannot delete default virtual router
1205	IP address conflict
1206	LLDP service is not running
18014	Invalid MTU for interface

Named list errors

Code	Typical error string	Detailed explanation
2001	Item duplicate	Item with the same value already exists
2002	You can't update/delete readonly list	
2003	You can't add/update/delete item to/from readonly list	
2004	Failed to import list contents	
2005	Incorrect morpholist: too complex data	
2006	Server is busy	
2007	Unsupported list type	
2008	Bad list update URL	
2009	Cannot export empty list	
2010	List content is not available	
2011	Found cyclic dependence	
2012	You can't update field name of list	
2013	Invalid time or date range	
2014	The named list contains duplicate items	

Settings

Code	Typical error string	Detailed explanation
3001	Failed CA certificate generation	Failed to generate new CA certificate
3002	Module is not registered	
3004	Failed to set certificate as active	Detailed reason could be passed as fault string value
3005	Incorrect time value	
3006	Unknown certificate kind	
3007	Certificate and private key don't match	
3008	Incorrect certificate file format	
3009	Incorrect key file format	
3010	No proper CA certificate set for cert auth	

3011	Unsupported auth mode	
3012	Failed CSR generation	Failed to generate new CSR
3013	Incorrect port number	Incorrect port number
3014	Failed to delete logs	
3015	Port already in use	
3016	Database import failed	Broken or invalid file
3017	Incorrect certificate chain data	
3018	Database import failed	Version mismatch
3019	Passphrase doesn't match file	
3020	Invalid adapter used	
3021	Too much zones	
3022	Internal error	
3023	Incorrect certificate role for this certificate	
3050	Invalid CCC access configuration data	
3051	Link setup failed: key error	Something wrong with the response: failed to decrypt using shared key
3052	Link setup failed: unknown/invalid token	Provided link setup token is not found on the CC server
3053	Link setup failed: not handled error	Not handled link setup error
3054	Link setup failed: Connection failed	Unable to establish connection with CC server
3055	Link setup failed: Agent must be enabled	
3056	Storage upload session not found	
3057	Storage file not found	
3058	Database import failed	wrong product

Libraries

Code	Typical error string	Detailed explanation
4001	Unknown response page template type	
4002	Cannot delete default template	
4003	Cannot remove state 'default' from template object	
4004	Geoip database is not available	

Accounts

Code	Typical error string	Detailed explanation
5001	Group not found	
5002	Duplicate account login name	User with the same login name already exists
5003	Duplicate user IP	MAC address or e-mail
5004	Can't update or delete ad group	

5005	User not found
5006	Cannot add or delete user to/from ad group
5008	Cannot enable expired user

Statistics

Code	Typical error string	Detailed explanation
6000	Incompatible data	Unsupported Statistics Server version
6001	Statistics Server connection error	
6002	Statistics Server authentication error	
6003	Statistics Server database error	
6004	Statistics Server send message to connector failed	
6100	Rule test failed	Logs export: rule test failed
6101	Remote server unreachable	Logs export: remote server unreachable
6102	Remote server connection refused	Logs export: remote server refused connection
6103	Remote server authentication failed	Logs export: remote server authentication failed
6104	Remote directory not found	Logs export: remote directory not found
6105	Remote directory access denied	Logs export: remote directory access denied. Needs file overwrite(delete) permissions
6200	Reports common error	
6201	Report generation error	
6202	Report file not found	
6998	Bad crontab parameters	
6999	Other error	

Core

Code	Typical error string	Detailed explanation
7001	Can't suggest url category	
7002	Unsupported certificate output format	
7003	Cannot connect to urlf server	Cannot connect to category checking server
7004	Administrator account is locked	
7005	Administrator cannot delete own account	
7006	Administrator password doesn't match secure password requirements	
7007	Administrator password not provided	
7008	Invalid node display name	
7009	Invalid administrator login	
7010	Incorrect old password	
7011	Administrator cannot be managed	

7012	Cannot add a new slave node, cluster is busy	
7013	Cluster license is required	
7021	Update has been revoked	
7022	Update cannot be installed	Generic error
7023	You cannot delete current node	Deletion of current node from a cluster is not allowed
7024	There is no update (when the file is not uploaded to the server)	
7025	Update upload failed (when an error occurred during the checkout and unpacking process)	
7026	Master server version mismatch	Master server has version that differ from current one
7027	Cluster node name conflict	There is a node with the same CC Node name in the cluster

Content

Code	Typical error string	Detailed explanation
8001	Request to whitelist for this URL is already added	
8002	Suggest categories request failed	

Captive portal errors

Code	Typical error string	Detailed explanation
10001	Incorrect security token	Incorrect security token and new one should be generated
10002	Too much auth attempts, account is locked now	
10004	Your device is blocked by administrator	
10005	Device limit is reached	
10006	Approve is required by administrator	
10007	You haven't got rights to login anonymously	
10008	No such IP address	
10009	Invalid 2FA code	
10010	2FA code is required	
10011	First time TOTP setup	Contains local path to file with key to configure TOTP on client device or app
10012	Invalid captive portal login cookie	
10013	2FA engine error	Some unhandled error in 2FA engine, administrator assistance required

High Availability errors

Code	Typical error string	Detailed explanation
11009	Conflict HA cluster tag	

11010	Conflict virtual ip values
--------------	----------------------------

Auth errors

Code	Typical error string	Detailed explanation
12001	Invalid keytab	
12002	Cannot connect to LDAP/AD server	
12003	Cannot bind to LDAP/AD server	
12004	Cannot find root in LDAP/AD server	
12005	LDAP/AD server is disabled	
12006	Invalid domains in keytab	
12007	LDAP/AD connection error: empty password	
12008	Incorrect certificate for IDP Server: in use	
12009	Failed to fetch metadata URL	Some error (not HTTP 404) happened
12010	Failed to connect AD controller	Some other (unhandled AD error)
12011	Server unavailable	
12012	TOTP key is already initialized	

Backup

Code	Typical error string	Detailed explanation
13001	Backup is already running	

Notification

Code	Typical error string	Detailed explanation
14001	SMPP notification profile test failed	
14002	SMTP notification profile test failed	
14003	SMTP notification profile test return temporary failure	
14004	SMTP notification profile test return permanent failure	
14005	SMTP notification profile test return network failure	
14006	SMTP notification profile test return unknown error	

PCAP

Code	Typical error string	Detailed explanation
17002	Can't download PCAP file	Return if downloading PCAP file is failed

Public

Code	Typical error string	Detailed explanation
19000	Incorrect input data	

Device

Code	Typical error string	Detailed explanation
20001	Rule check error: connection failed	
20002	Rule check error: authentication failed	
20003	No hardware RAID detected	
20004	Required RAID component is missing	Missing binary required to fetch status.
20005	Required power supply unit status component is missing	Missing binary required to fetch status
20006	Failed to export settings	

Reverse proxy, load balancing, virtual server

Code	Typical error string	Detailed explanation
21000	Invalid paths	
21001	virtual server address and real servers address cannot be same	
21002	Reverse proxy path rewrite is invalid	

SNMP rules, properties, security profiles

Code	Typical error string	Detailed explanation
23000	SNMP Rule with the same name already exists	
23001	SNMP v2 Rule with the same community already exists	
23002	SNMP v3 Rule needs SNMP Security profile	
23003	SNMP Rule with the same Security profile already exists	
23004	SNMP Rule not found	
23005	SNMP Security profile not found	
23006	SNMP Security profile with the same username already exist	
23007	SNMP Security profile name already exist	
23008	SNMP Security profile needs name	
23009	SNMP Security profile is used by SNMP Rule	

Component: Auth

Module methods

v1.auth.ldap.servers.list	84	v1.auth.user.auth.profile.delete	91
v1.auth.ldap.server.fetch	84	v1.auth.tacacs.plus.server.list	91
v1.auth.ldap.server.add	85	v1.auth.tacacs.plus.server.fetch	92
v1.auth.ldap.server.update	85	v1.auth.tacacs.plus.server.add	92
v1.auth.ldap.server.delete	86	v1.auth.tacacs.plus.server.update	93
v1.auth.check.all.servers.status	86	v1.auth.tacacs.plus.server.delete	93
v1.auth.ldap.server.initSync	86	v1.auth.ntlm.server.list	93
v1.auth.ldap.server.checkConnection	87	v1.auth.ntlm.server.fetch	94
v1.auth.ldap.server.test.connection	87	v1.auth.ntlm.server.add	94
v1.auth.radius.servers.list	88	v1.auth.ntlm.server.update	95
v1.auth.radius.server.fetch	88	v1.auth.ntlm.server.delete	95
v1.auth.radius.server.add	88	v1.auth.saml.idp.server.download.metadata ..	95
v1.auth.radius.server.update	89	v1.auth.saml.idp.servers.list	96
v1.auth.radius.server.delete	89	v1.auth.saml.idp.server.fetch	96
v1.auth.user.auth.profiles.list	90	v1.auth.saml.idp.server.add	97
v1.auth.user.auth.profile.add	90	v1.auth.saml.idp.server.update	97
v1.auth.user.auth.profile.fetch	90	v1.auth.saml.idp.server.delete	97
v1.auth.user.auth.profile.update	91		

v1.auth.ldap.servers.list

```
List<LDAPServerInfo> v1.auth.ldap.servers.list(String auth_token, Struct filter)
```

Fetch list of all registered ldap servers.

Parameters:

- **auth_token** — authentication token
- **filter** — search filter, the only allowed key is **enabled** with values either `true` or `false`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of structures in the form [LDAPServerInfo](#)

v1.auth.ldap.server.fetch

```
LDAPServerInfo v1.auth.ldap.server.fetch(String auth_token, UID id)
```

Fetch registered ldap server by given id.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of ldap server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Ldap server with given id not found

Return:

Ldap server structure in the form [LDAPServerInfo](#)

v1.auth.ldap.server.add

```
UID v1.auth.ldap.server.add(String auth_token, LDAPServerInfo info)
```

Add a new ldap server.

Parameters:

- **auth_token** — authentication token
- **info** — ldap server properties

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field

Return:

identifier of just created ldap server

v1.auth.ldap.server.update

```
Boolean v1.auth.ldap.server.update(String auth_token, UID id, LDAPServerInfo info)
```

Update ldap server with given id.

Parameters:

- **auth_token** — authentication token
- **id** — id of ldap server that should be updated
- **info** — ldap server properties with fields that should be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Ldap server with given id not found
- **409** — Duplicate value, conflict with the existing objects

Return:

true

v1.auth.ldap.server.delete

```
Boolean v1.auth.ldap.server.delete(String auth_token, UID id)
```

Delete ldap server with given id.

Parameters:

- **auth_token** — authentication token
- **id** — id of ldap server that should be deleted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Ldap server with given id not found

Return:

true

v1.auth.check.all.servers.status

```
List<Struct> v1.auth.check.all.servers.status(String auth_token)
```

Fetch status of all auth servers.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of structures, each structure contains fields: **type**, **id**, **status**.

v1.auth.ldap.server.initSync

```
Boolean v1.auth.ldap.server.initSync(String auth_token, UID id)
```

Start synchronization process for given ldap server. Method returns true right away and for synchronization status you need to call method [v1.auth.check.all.servers.status](#).

Parameters:

- **auth_token** — authentication token
- **id** — id of ldap server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Ldap server with given id not found

Return:

true

v1.auth.ldap.server.checkConnection

```
Boolean v1.auth.ldap.server.checkConnection(String auth_token, LDAPServerInfo info)
```

Check connection and bind for given ldap config.

Don't use this method as it performs just basic checking, use [v1.auth.ldap.server.test.connection](#) instead.

Parameters:

- **auth_token** — authentication token
- **info** — config for ldap server (if field **id** is provided in info and you don't provide password or provide empty password, then method get password from the database)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **12001** — Invalid keytab
- **12002** — Cannot connect to LDAP/AD server
- **12003** — Cannot bind to LDAP/AD server
- **12004** — Cannot find root in LDAP/AD server
- **12006** — Invalid domains in keytab
- **12007** — Invalid auth domain address
- **12011** — Server unavailable
- **12013** — LDAP/AD connection error: empty password

Return:

Always returns `true`.

v1.auth.ldap.server.test.connection

```
LDAPServerTestInfo v1.auth.ldap.server.test.connection(String auth_token, UID id, LDAPServerInfo info)
```

Test LDAP server configuration: check connectivity, authentication and keytab.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of existing LDAP server or boolean `false`
- **info** — LDAP server configuration that needs to be checked, if **id** is `false` then **info** field MUST contain full server configuration, if **id** is existing server identifier then **info** may contain partial server configuration, the rest of it will be fetched from the database

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Structure with checking results, [LDAPServerTestInfo](#).

v1.auth.radius.servers.list

```
List<RadiusServerInfo> v1.auth.radius.servers.list(String auth_token, Struct filter)
```

Fetch list of all registered radius servers.

Parameters:

- **auth_token** — authentication token
- **filter** — search filter, the only allowed key is **enabled** with values either `true` or `false`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of structures in the form [RadiusServerInfo](#)

v1.auth.radius.server.fetch

```
RadiusServerInfo v1.auth.radius.server.fetch(String auth_token, UID id)
```

Fetch radius server by unique identifier.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of radius server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Radius server with given id not found

Return:

radius server structure in the form [RadiusServerInfo](#)

v1.auth.radius.server.add

```
UID v1.auth.radius.server.add(String auth_token, RadiusServerInfo info)
```

Add a new radius server.

Parameters:

- **auth_token** — authentication token
- **info** — radius server properties

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field

Return:

id of just created radius server

v1.auth.radius.server.update

```
Boolean v1.auth.radius.server.update(String auth_token, UID id, RadiusServerInfo info)
```

Update radius server with given id.

Parameters:

- **auth_token** — authentication token
- **id** — id of radius server that should be updated
- **info** — radius server properties with fields that should be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Radius server with given id not found
- **409** — Duplicate value, conflict with the existing object

Return:

true

v1.auth.radius.server.delete

```
Boolean v1.auth.radius.server.delete(String auth_token, UID id)
```

Delete radius server with given id.

Parameters:

- **auth_token** — authentication token
- **id** — id of radius server that should be deleted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Radius server with given id not found

Return:

true

v1.auth.user.auth.profiles.list

```
List<UserAuthProfile> v1.auth.user.auth.profiles.list(String auth_token)
```

Fetch list of all user authentication profiles.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of profiles

v1.auth.user.auth.profile.add

```
UID v1.auth.user.auth.profile.add(String auth_token, UserAuthProfile profile)
```

Create a new user authentication profile.

Parameters:

- **auth_token** — authentication token
- **profile** — authentication profile data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field

Return:

identifier of just created profile

v1.auth.user.auth.profile.fetch

```
UserAuthProfile v1.auth.user.auth.profile.fetch(String auth_token, UID profile_id)
```

Fetch single profile data.

Parameters:

- **auth_token** — authentication token
- **profile_id** — identifier of auth profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Profile not found

Return:

profile data

v1.auth.user.auth.profile.update

```
Boolean v1.auth.user.auth.profile.update(String auth_token, UID profile_id,
    UserAuthProfile profile)
```

Update profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — identifier of auth profile
- **profile** — authentication profile data that should be updated

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Profile not found
- **409** — Duplicate value, conflict with the existing object

Return:

true

v1.auth.user.auth.profile.delete

```
Boolean v1.auth.user.auth.profile.delete(String auth_token, UID profile_id)
```

Delete user authentication profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — identifier of auth profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Profile not found

Return:

true

v1.auth.tacacs.plus.server.list

```
List<TacacsPlusServerInfo> v1.auth.tacacs.plus.server.list(String auth_token, Struct
    filter)
```

List servers.

Parameters:

- **auth_token** — authentication token
- **filter** — filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of servers

v1.auth.tacacs.plus.server.fetch

```
TacacsPlusServerInfo v1.auth.tacacs.plus.server.fetch(String auth_token, UID server_id)
```

Fetch single server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth tacacs

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

server info

v1.auth.tacacs.plus.server.add

```
UID v1.auth.tacacs.plus.server.add(String auth_token, TacacsPlusServerInfo server)
```

Add a new server.

Parameters:

- **auth_token** — authentication token
- **server** — server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field

Return:

just created server identifier

v1.auth.tacacs.plus.server.update

```
Boolean v1.auth.tacacs.plus.server.update(String auth_token, UID server_id, TacacsPlusServerInfo server)
```

Update existing server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth tacacs
- **server** — server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Object not found
- **409** — Duplicate value, conflict with the existing object

Return:

true

v1.auth.tacacs.plus.server.delete

```
Boolean v1.auth.tacacs.plus.server.delete(String auth_token, UID server_id)
```

Delete server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth tacacs

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.auth.ntlm.server.list

```
List<NtlmServerInfo> v1.auth.ntlm.server.list(String auth_token, Struct filter)
```

List servers.

Parameters:

- **auth_token** — authentication token
- **filter** — filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of servers

v1.auth.ntlm.server.fetch

```
NtlmServerInfo v1.auth.ntlm.server.fetch(String auth_token, UID server_id)
```

Fetch single server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth ntlm

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

server info

v1.auth.ntlm.server.add

```
UID v1.auth.ntlm.server.add(String auth_token, NtlmServerInfo server)
```

Add a new server.

Parameters:

- **auth_token** — authentication token
- **server** — server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object

Return:

just created server identifier

v1.auth.ntlm.server.update

```
Boolean v1.auth.ntlm.server.update(String auth_token, UID server_id, NtlmServerInfo server)
```

Update existing server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth ntlm
- **server** — server

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Object not found
- **409** — Duplicate value, conflict with the existing object

Return:

true

v1.auth.ntlm.server.delete

```
Boolean v1.auth.ntlm.server.delete(String auth_token, UID server_id)
```

Delete server.

Parameters:

- **auth_token** — authentication token
- **server_id** — identifier of auth ntlm

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.auth.saml.idp.server.download.metadata

```
Struct v1.auth.saml.idp.server.download.metadata(String auth_token, String url)
```

Download SAML IDP server spec using provided URL. Structure in response data may contain the following keys: **certificate** (DER certificate as base64-encoded text), **sso_url**, **slo_url**, **sso_http_binding**, **slo_http_binding**

Parameters:

- **auth_token** — authentication token
- **url** — URL

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — URL not found
- **12002** — Cannot connect to the server
- **12009** — Failed to fetch URL

Return:

structure that contains fields we can use to create SAML IDP record.

v1.auth.saml.idp.servers.list

```
List<SamlIdpServerInfo> v1.auth.saml.idp.servers.list(String auth_token, Struct filter)
```

List all SAML IDP servers.

Parameters:

- **auth_token** — authentication token
- **filter** — search filter, the only allowed key is **enabled** with values either `true` or `false`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of servers

v1.auth.saml.idp.server.fetch

```
SamlIdpServerInfo v1.auth.saml.idp.server.fetch(String auth_token, UID id)
```

fetch server.

Parameters:

- **auth_token** — authentication token
- **id** — server identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

server info

v1.auth.saml.idp.server.add

```
UID v1.auth.saml.idp.server.add(String auth_token, SamlIdpServerInfo server)
```

Create a new SAML IDP server.

Parameters:

- **auth_token** — authentication token
- **server** — server specification

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **513** — Incorrect object field value

Return:

created server identifier

v1.auth.saml.idp.server.update

```
Boolean v1.auth.saml.idp.server.update(String auth_token, UID id, SamlIdpServerInfo server)
```

Update existing SAML IDP server.

Parameters:

- **auth_token** — authentication token
- **id** — server identifier
- **server** — server specification

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Some field value in passed structure is invalid
- **404** — Object not found
- **409** — Duplicate value, conflict with the existing object
- **513** — Incorrect object field value

Return:

true

v1.auth.saml.idp.server.delete

```
Boolean v1.auth.saml.idp.server.delete(String auth_token, UID id)
```

Delete server.

Parameters:

- **auth_token** — authentication token
- **id** — server identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

Component: Certificates

Module methods

v1.certificates.client.profiles.list	99	v1.certificates.client.profile.update	100
v1.certificates.client.profile.fetch	99	v1.certificates.client.profile.delete	101
v1.certificates.client.profile.add	100		

v1.certificates.client.profiles.list

```
PagingList<ClientCertificateProfileInfo> v1.certificates.client.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all client certificate profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options (currently only supports {"search": "name_value"})
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

Each entry is represented by structure of type [ClientCertificateProfileInfo](#).

v1.certificates.client.profile.fetch

```
ClientCertificateProfileInfo v1.certificates.client.profile.fetch(String auth_token, UUID id)
```

Fetch one client certificate profile.

Parameters:

- **auth_token** — authentication token
- **id** — object identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **404** — User certificates profile with given id not found

Return:

Object information represented by type [ClientCertificateProfileInfo](#).

v1.certificates.client.profile.add

```
UID v1.certificates.client.profile.add(String auth_token, ClientCertificateProfileInfo profile)
```

Create a new client certificate profile.

Parameters:

- **auth_token** — authentication token
- **profile** — client certificate profile object (as structure [ClientCertificateProfileInfo](#)) that should be added

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **404** — Certificate not found (referenced CA certificate does not exist)
- **409** — Name already exists
- **411** — Certificate is not CA (referenced certificate is not a CA certificate)
- **601** — Missing required field

Return:

Identifier of just created object.

v1.certificates.client.profile.update

```
Boolean v1.certificates.client.profile.update(String auth_token, UID id, ClientCertificateProfileInfo profile)
```

Update existing client certificate profile.

Parameters:

- **auth_token** — authentication token
- **id** — client certificate profile identifier
- **profile** — client certificate profile object (as structure [ClientCertificateProfileInfo](#)) that should be updated, only passed fields are updated.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **111** — Invalid field value
- **404** — Client certificate profile with given id not found
- **404** — Certificate not found (referenced CA certificate does not exist)
- **409** — Name already exists
- **411** — Certificate is not CA (referenced certificate is not a CA certificate)

Return:

Always returns `true`.

v1.certificates.client.profile.delete

```
Boolean v1.certificates.client.profile.delete(String auth_token, UID id)
```

Delete existing client certificate profile.

Parameters:

- **auth_token** — authentication token
- **id** — client certificate profile identifier to delete

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **404** — Client certificate profile with given id not found
- **502** — Client certificate profile is used by another configuration object

@return Always returns `true`.

Component: Content filtering

Module methods

v1.content.ssl.profiles.list.....	102	v1.content.ssl.profile.update	103
v1.content.ssl.profile.fetch.....	102	v1.content.ssl.profile.delete	103
v1.content.ssl.profile.add	103		

v1.content.ssl.profiles.list

```
PagingList<SslProfileInfo> v1.content.ssl.profiles.list(String auth_token, Integer start, Integer limit, String filter)
```

Fetch list of SSL profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — filter by profile name

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

ssl profile information list

v1.content.ssl.profile.fetch

```
SslProfileInfo v1.content.ssl.profile.fetch(String auth_token, UID profile_id)
```

Fetch single SSL profile record.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **404** — Profile not found

Return:

profile information

v1.content.ssl.profile.add

```
UID v1.content.ssl.profile.add(String auth_token, SslProfileInfo profile)
```

Add a new SSL profile.

Parameters:

- **auth_token** — authentication token
- **profile** — profile (as a structure [SslProfileInfo](#)) that should be added

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid.
- **409** — Duplicate name value.
- **601** — Missing required field.

Return:

identifier of just created profile

v1.content.ssl.profile.update

```
Boolean v1.content.ssl.profile.update(String auth_token, UID profile_id, SslProfileInfo profile)
```

Update existing SSL profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — identifier of the profile
- **profile** — SSL profile data, only passed structure fields will be updated

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **111** — Some field value in passed structure is invalid.
- **404** — Profile not found.
- **409** — Object with same data already exists.

Return:

true

v1.content.ssl.profile.delete

```
Boolean v1.content.ssl.profile.delete(String auth_token, UID profile_id)
```

Delete SSL profile.

Parameters:

- **auth_token** — authentication token

- **profile_id** — identifier of the profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument.
- **404** — Profile not found

Return:

true

Component: Core

Module methods

v2.core.login	105	v2.core.administrator.profile.update	112
v2.core.logout	105	v2.core.administrator.profile.delete	113
v2.core.session.permissions.list	106	v2.core.administrator.list	113
v2.core.session.ping	106	v2.core.administrator.fetch	114
v2.core.license.check	106	v2.core.administrator.add	114
v2.core.license.activation	107	v2.core.administrator.update	114
v2.core.license.offline.request	108	v2.core.administrator.delete	115
v2.core.license.offline.activation	108	v2.core.administrator.unlock	115
v2.core.node.remove	108	v2.core.administrator.sessions.list	115
v2.core.node.status	109	v2.core.administrator.session.terminate	116
v2.core.nodes.list	109	v2.core.administrator.change.password	116
v2.core.node.update	109	v2.core.system.bypass.status	117
v2.core.license.info	110	v2.core.system.set.bypass	117
v2.core.get.categories	110	v2.core.system.security.updates.status	117
v2.core.get.category.risks	110	v2.core.system.security.updates.changelog	118
v2.core.administrator.config.get	111	v2.core.system.security.updates.apply	118
v2.core.administrator.config.set	111	v2.core.system.update.info	118
v2.core.administrator.profiles.list	111	v2.core.system.update.start	119
v2.core.administrator.profile.fetch	112	v2.core.check.url	119
v2.core.administrator.profile.add	112	v2.core.ip.protocol.list	120

v2.core.login

[LoginInfo](#) **v2.core.login**(String login, String password)

Plain login to the system

Parameters:

- **login** — login name
- **password** — password

Errors:

- **604** — Service is starting, cannot log in
- **7004** — Administrator account is locked

Return:

Authentication token and other service data in the structure [LoginInfo](#)

v2.core.logout

Boolean **v2.core.logout**(String auth_token)

Logout from the system

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v2.core.session.permissions.list

```
Struct v2.core.session.permissions.list(String auth_token, String subsystem)
```

Fetch list of permissions for current administrator session.

Parameters:

- **auth_token** — authentication token
- **subsystem** — one of the following values: `"xmlrpc"`, `"webui"`, `"cli"`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Subsystem not found

Return:

Returns structure where key is a permission name and value is one of the following values: `"read"`, `"none"`, `"readwrite"`.

v2.core.session.ping

```
Boolean v2.core.session.ping(String auth_token)
```

Ping current session to keep it alive.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v2.core.license.check

```
Boolean v2.core.license.check(String auth_token)
```

Initiate license check process.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v2.core.license.activation

```
Boolean v2.core.license.activation(String auth_token, ActivationInfo activation_params)
```

Activate product license key.

Parameters:

- **auth_token** — authentication token
- **key_params** — license activation params

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **301** — DIRECT: database error
- **302** — DIRECT: pincode not found
- **303** — DIRECT: pincode is not active
- **304** — DIRECT: activations number limit is reached
- **305** — DIRECT: module not found
- **306** — DIRECT: source pincode not found
- **307** — DIRECT: new pincode not found
- **308** — DIRECT: new pincode is already activated
- **309** — DIRECT: new pincode is not active
- **310** — DIRECT: failed to generate license key
- **311** — DIRECT: failed to send email
- **312** — DIRECT: version mismatch
- **313** — DIRECT: unknown user
- **314** — DIRECT: source pincode is already converted to a new one
- **315** — DIRECT: no more CT keys
- **316** — DIRECT: old and new pincode products doesn't match
- **317** — DIRECT: attempt to activate pincode without old pincode
- **318** — DIRECT: incorrect variable
- **319** — DIRECT: events mismatch
- **320** — DIRECT: pincode doesn't match the program
- **321** — DIRECT: transport_error, Reply was not received
- **322** — DIRECT: transport_error, Wrong user data
- **323** — DIRECT: transport_error, There is no enough data
- **324** — DIRECT: transport_error, Wrong action for this type of the license
- **325** — DIRECT: transport_error, Server's response validation failed
- **326** — DIRECT: transport_error, There are no modules to process

- **327** — DIRECT: transport_error, Wrong license type while creating module request
- **351** — DIRECT: not supported method
- **352** — DIRECT: license/key is expired
- **353** — DIRECT: limit of cluster nodes is reached
- **354** — DIRECT: License is not available for this client
- **355** — DIRECT: Hardware id not found

Return:

true

v2.core.license.offline.request

```
Base64 v2.core.license.offline.request(String auth_token, ActivationInfo
activation_params)
```

Generate request for offline activation.

Parameters:

- **auth_token** — authentication token
- **key_params** — license activation params

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

String value that should be sent to sales team

v2.core.license.offline.activation

```
Boolean v2.core.license.offline.activation(String auth_token, Base64 data)
```

Perform an offline activation.

Parameters:

- **auth_token** — authentication token
- **data** — activation data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **612** — Invalid offline activation data

Return:

true

v2.core.node.remove

```
Boolean v2.core.node.remove(String auth_token, String node_name)
```

Remove erlang node from the cluster.

Parameters:

- **auth_token** — authentication token
- **node_name** — name of the node that should be removed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **7023** — You cannot delete current node

Return:

true

v2.core.node.status

```
Struct v2.core.node.status()
```

Fetch a current node status. Method does not require authentication.

Return:

Current node status, possible values are: "work" (node is working properly), "install" (node is in initial install mode), "starting" (node is starting).

v2.core.nodes.list

```
List<NodeDesc> v2.core.nodes.list(String auth_token)
```

Fetch a list of all nodes of network cluster.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of nodes as structures of type [NodeDesc](#)

v2.core.node.update

```
Boolean v2.core.node.update(String auth_token, String node_name, NodeDesc node)
```

Update node information.

Parameters:

- **auth_token** — authentication token
- **node_name** — internal node name
- **node** — node data that should be updated, you can change these fields only: **display_name**

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Node not found
- **409** — Object with the same display name already exists

Return:

Always returns `true`.

v2.core.license.info

```
LicenseInfo v2.core.license.info(String auth_token)
```

Fetch license information for the server.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

license information

v2.core.get.categories

```
List v2.core.get.categories()
```

Fetch list of all url filtering categories. Method does not require authentication.

Return:

List of all URL filtering categories, each list item is a structure with fields **id** and **name**.

v2.core.get.category.risks

```
List v2.core.get.category.risks()
```

Fetch list of all url filtering categories. Method does not require authentication.

Return:

List of all URL categories threat level, each list item is a structure with fields **id** and **risk**. **risk** may be one of the following values:

- **1** — very low
- **2** — low
- **3** — medium
- **4** — high
- **5** — very high

v2.core.administrator.config.get

```
AdministratorSystemConfig v2.core.administrator.config.get(String auth_token)
```

Fetch administrators global options.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

configuration options

v2.core.administrator.config.set

```
Boolean v2.core.administrator.config.set(String auth_token, AdministratorSystemConfig config)
```

Update administrators global options.

Parameters:

- **auth_token** — authentication token
- **config** — config

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v2.core.administrator.profiles.list

```
List<AdministratorProfileInfo> v2.core.administrator.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch one page from administrator profiles list.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied

- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of profiles

v2.core.administrator.profile.fetch

```
AdministratorProfileInfo v2.core.administrator.profile.fetch(String auth_token, UID id)
```

Fetch administrator profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Profile not found

Return:

profile

v2.core.administrator.profile.add

```
UID v2.core.administrator.profile.add(String auth_token, AdministratorProfileInfo profile)
```

Create a new administrator profile.

Parameters:

- **auth_token** — authentication token
- **profile** — profile data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Profile not found

Return:

identifier of just created profile object

v2.core.administrator.profile.update

```
Boolean v2.core.administrator.profile.update(String auth_token, UID id, AdministratorProfileInfo profile)
```

Create a new administrator profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

- **profile** — profile data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Profile not found

Return:

true

v2.core.administrator.profile.delete

```
Boolean v2.core.administrator.profile.delete(String auth_token, UID id)
```

Delete profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Profile not found

Return:

true

v2.core.administrator.list

```
List v2.core.administrator.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch one page from administrators list.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of all administrators in the form of `AdministratorInfo` structure

v2.core.administrator.fetch

```
AdministratorInfo v2.core.administrator.fetch(String auth_token, UID id)
```

Fetch information about administrator by unique identifier.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of administrator

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Administrator with given ID not found

Return:

information about administrator with given id in the form of `AdministratorInfo` structure

v2.core.administrator.add

```
UID v2.core.administrator.add(String auth_token, AdministratorInfo info)
```

Add new administrator to server.

Parameters:

- **auth_token** — authentication token
- **info** — information about new administrator in the form `AdministratorInfo`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **409** — Duplicate, user with the same login already exists
- **601** — Login name required
- **7006** — Administrator password doesn't match secure password requirements
- **7007** — Administrator password not provided

Return:

unique identifier of just created administrator

v2.core.administrator.update

```
Boolean v2.core.administrator.update(String auth_token, UID id, AdministratorInfo info)
```

Update administrator with given id.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of administrator
- **info** — information for updating in the form `AdministratorInfo`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Administrator with given id not found
- **7006** — Administrator password doesn't match secure password requirements
- **7007** — Administrator password not provided

Return:

true

v2.core.administrator.delete

```
Boolean v2.core.administrator.delete(String auth_token, UID id)
```

Delete administrator with given id.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of administrator

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Administrator with given id not found
- **500** — You can't delete administrator with root role

Return:

true

v2.core.administrator.unlock

```
Boolean v2.core.administrator.unlock(String auth_token, UID id)
```

Unlock blocked administrator account.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of administrator

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Administrator with given id not found

Return:

true

v2.core.administrator.sessions.list

```
List<AdministratorSessionInfo> v2.core.administrator.sessions.list(String auth_token)
```

Fetch list of all administrator sessions.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of sessions

v2.core.administrator.session.terminate

```
Boolean v2.core.administrator.session.terminate(String auth_token, String node_name, String guid)
```

Terminate administrator session.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **guid** — session identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Session with given id not found

Return:

true

v2.core.administrator.change.password

```
Boolean v2.core.administrator.change.password(String auth_token, String old_password, String new_password)
```

Change current administrator password.

Parameters:

- **auth_token** — authentication token
- **old_password** — old password
- **new_password** — new password

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **7006** — Administrator password doesn't match secure password requirements
- **7010** — Incorrect old password
- **7011** — Administrator cannot be managed

Return:

true

v2.core.system.bypass.status

```
Boolean v2.core.system.bypass.status(String auth_token)
```

Fetch bypass mode status.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Return `true` if bypass mode is enabled, i.e. all network traffic is passed without any intercepting.

Returns `false` if bypass mode is disabled.

v2.core.system.set.bypass

```
Boolean v2.core.system.set.bypass(String auth_token, Boolean bypass)
```

Set network traffic bypass. (will be disabled)

Parameters:

- **auth_token** — authentication token
- **bypass** — if true then all network traffic will be passed without intercepting (i.e. all firewall features)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **102** — Access denied

Return:

true

v2.core.system.security.updates.status

```
String v2.core.system.security.updates.status(String auth_token)
```

Check if we have new security updates to install previous update was failed, `"in_progress"` if updates are being installed right now

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

"new_updates_available" if there are updates ready to install, "nothing_to_update" if there are no new updates, "update_failed" if the update failed

v2.core.system.security.updates.changelog

```
String v2.core.system.security.updates.changelog(String auth_token)
```

Return changelog information for available updates.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

full changelog

v2.core.system.security.updates.apply

```
Boolean v2.core.system.security.updates.apply(String auth_token, Struct options)
```

Install downloaded security updates. Argument options may contain the following fields:

create_restore_point	Boolean	if <code>true</code> then this update will create a new system restoration point
-----------------------------	---------	--

Parameters:

- **auth_token** — authentication token
- **options** — additional installation options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **7021** — Update has been revoked
- **7022** — Update cannot be installed

Return:

Always returns `true`.

v2.core.system.update.info

```
List<ItemUpdateInfo> v2.core.system.update.info(String auth_token)
```

Return update status for WAF lists and modules. Some of elements can be missed - it mean that there were no success updates for this element.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

array of items

v2.core.system.update.start

```
Boolean v2.core.system.update.start(String auth_token)
```

Force WAF update

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v2.core.check.url

```
List<Integer> v2.core.check.url(String auth_token, String url)
```

Check url category for given URL. This method doesn't check overridden domains database, to perform complete check use method [v1.content.override.domain.check].

Parameters:

- **auth_token** — authentication token
- **url** — URL

Errors:

- **3** — Incorrect url or domain
- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **354** — No License

Return:

array of categories

v2.core.ip.protocol.list

```
List<IPProtocolInfo> v2.core.ip.protocol.list()
```

Fetch list of available/supported IP protocols list.

Method does not require authentication.

Return:

list of IP protocols

Component: Device

Module methods

v1.device.poweroff	121	v1.device.settings.download	127
v1.device.reboot	121	v1.device.network.settings.download	128
v1.device.reboot.nodes	122	v1.device.export.settings.start	128
v1.device.reboot.cluster	122	v1.device.export.network.settings.start	128
v1.device.logs.clear	122	v1.device.export.settings.state	129
v1.device.loglevel.get	123	v1.device.settings.import.details	129
v1.device.loglevel.set	123	v1.device.settings.import	130
v1.device.custom.debug.log.set	123	v1.device.network.settings.import	130
v1.device.backup.upload.storage	124	v1.device.settings.export.rules.list	131
v1.device.backup.export.create.ssh.key	124	v1.device.settings.export.rule.add	131
v1.device.backup.export.get.ssh.key	124	v1.device.settings.export.rule.test	131
v1.device.backup.export.import.ssh.key	125	v1.device.settings.export.rule.fetch	132
v1.device.backup.export.rules.list	125	v1.device.settings.export.rule.update	132
v1.device.backup.export.rule.add	125	v1.device.settings.export.rule.delete	132
v1.device.backup.export.rule.test	126	v1.device.hardware.raid.status	133
v1.device.backup.export.rule.fetch	126	v1.device.power.supply.status	133
v1.device.backup.export.rule.update	127	v1.device.dns.query	133
v1.device.backup.export.rule.delete	127	v1.device.logs.remove.archive.file	134

v1.device.poweroff

Boolean **v1.device.poweroff**(String auth_token)

Power off the server.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.reboot

Boolean **v1.device.reboot**(String auth_token)

Reboot the server.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.reboot.nodes

```
Boolean v1.device.reboot.nodes(String auth_token, List<String> nodes)
```

Reboot the all nodes in cluster.

Parameters:

- **auth_token** — authentication token
- **nodes** — list of nodes that should be rebooted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.reboot.cluster

```
Boolean v1.device.reboot.cluster(String auth_token)
```

Reboot the all nodes in cluster.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.logs.clear

```
Boolean v1.device.logs.clear(String auth_token)
```

Delete all system logs.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3014** — Failed to delete logs

Return:

Always returns `true`.

v1.device.loglevel.get

```
String v1.device.loglevel.get(String auth_token)
```

Get system logs logging level.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

One of `"off"`, `"error"`, `"warning"`, `"info"`, `"debug"`

v1.device.loglevel.set

```
Boolean v1.device.loglevel.set(String auth_token, String level)
```

Set system logging level. The change is temporary and will reset on server restart.

Parameters:

- **auth_token** — authentication token
- **level** — One of `"off"`, `"error"`, `"warning"`, `"info"`, `"debug"`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.custom.debug.log.set

```
Boolean v1.device.custom.debug.log.set(String auth_token, String component)
```

Set temporary debug log for specified component. Calling [v1.device.loglevel.set](#) will reset this configuration.

Parameters:

- **auth_token** — authentication token

- **component** — name of the component debug logs should be enabled for, allowed values are:
`"rpc"`

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.backup.upload.storage

```
Boolean v1.device.backup.upload.storage(String auth_token, String storage_file_uid)
```

Upload backup file for system restore.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — identifier of imported backup file uploaded to storage

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.backup.export.create.ssh.key

```
String v1.device.backup.export.create.ssh.key(String auth_token)
```

Creates and stores new private and public keys for remote backup by ssh.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Created public key

v1.device.backup.export.get.ssh.key

```
String v1.device.backup.export.get.ssh.key(String auth_token)
```

Returns current public key for remote backup by ssh.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

public key

v1.device.backup.export.import.ssh.key

```
String v1.device.backup.export.import.ssh.key(String auth_token, String storage_file_uid)
```

Stores imported private key for remote backup by ssh. Creates public key for it.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — identifier of imported private key file uploaded to storage

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.device.backup.export.rules.list

```
List<SettingsExportRule> v1.device.backup.export.rules.list(String auth_token, Struct pattern)
```

List all backup export rules.

Parameters:

- **auth_token** — authentication token
- **pattern** — filtering pattern

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of rules

v1.device.backup.export.rule.add

```
UID v1.device.backup.export.rule.add(String auth_token, SettingsExportRule rule)
```

Create a new backup export rule.

Parameters:

- **auth_token** — authentication token

- **rule** — rule specifications

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with same data already exists,
- **601** — Missing required field,
- **6998** — Bad crontab parameters

Return:

identifier of just created object

v1.device.backup.export.rule.test

```
Boolean v1.device.backup.export.rule.test(String auth_token, SettingsExportRule rule)
```

Test connection and write access for backup export rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule specifications

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20001** — Rule test failed: connection error
- **20002** — Rule test failed: authentication error

Return:

Always returns `true`.

v1.device.backup.export.rule.fetch

```
SettingsExportRule v1.device.backup.export.rule.fetch(String auth_token, UID rule_id)
```

Fetch backup export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

rule

v1.device.backup.export.rule.update

```
Boolean v1.device.backup.export.rule.update(String auth_token, UID rule_id, SettingsExportRule rule)
```

Update existing backup export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **rule** — rule specifications

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

Always returns `true`.

v1.device.backup.export.rule.delete

```
Boolean v1.device.backup.export.rule.delete(String auth_token, UID rule_id)
```

Fetch backup export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

Always returns `true`.

v1.device.settings.download

```
String v1.device.settings.download(String auth_token)
```

Export server settings to a file in the storage.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20006** — Failed to export settings

Return:

Identifier of exported database storage file

v1.device.network.settings.download

```
String v1.device.network.settings.download(String auth_token)
```

Export server settings (network configuration only) to a file in the storage.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20006** — Failed to export settings

Return:

Identifier of exported database storage file

v1.device.export.settings.start

```
String v1.device.export.settings.start(String auth_token)
```

Start asynchronous export of settings.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always return `true`

v1.device.export.network.settings.start

```
String v1.device.export.network.settings.start(String auth_token)
```

Start asynchronous export of settings (networking data only).

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always return `true`

v1.device.export.settings.state

Struct `v1.device.export.settings.state`(String auth_token)

Get state of export settings process.

Parameters:

- **auth_token** — authentication token

Errors:

- **2** — Unknown error
- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20006** — Failed to export settings

Return:

if process is running, then return structure:

Field	Value
status	"running"

if the process is completed, then return structure:

Field	Type	Description
file_storage_uid	String	Identifier of exported database storage file

v1.device.settings.import.details

`SettingsBackupDetails` `v1.device.settings.import.details`(String auth_token, String storage_file_uid)

Fetch information about uploaded settings file.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — identifier of imported settings file uploaded to storage

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3016** — Import failed: invalid file
- **3058** — Import failed: wrong product

Return:

settings backup file info

v1.device.settings.import

```
Boolean v1.device.settings.import(String auth_token, String storage_file_uid,  
SettingsImportParams params)
```

Import server settings from the file.

This method doesn't reboot server automatically after import completion. You should reboot it using [v1.device.reboot](#) or [v1.device.reboot.nodes](#) methods

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — identifier of imported settings file uploaded to storage
- **params** — import parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3016** — Invalid file
- **3018** — Version mismatch
- **3020** — Invalid adapter used
- **3022** — Import failed
- **3058** — Wrong product

Return:

Always returns `true`.

v1.device.network.settings.import

```
Boolean v1.device.network.settings.import(String auth_token, String storage_file_uid,  
SettingsImportParams params)
```

Import server settings (network configuration only) from the file.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — identifier of imported settings file uploaded to storage
- **params** — import parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3016** — Invalid file
- **3018** — Version mismatch
- **3020** — Invalid adapter used
- **3022** — Import failed
- **3058** — Wrong product

Return:

Always returns `true`.

v1.device.settings.export.rules.list

```
List<SettingsExportRule> v1.device.settings.export.rules.list(String auth_token, Struct pattern)
```

List all settings export rules.

Parameters:

- **auth_token** — authentication token
- **pattern** — filtering pattern

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of rules

v1.device.settings.export.rule.add

```
UID v1.device.settings.export.rule.add(String auth_token, SettingsExportRule rule)
```

Create a new settings export rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule specifications

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with same data already exists,
- **601** — Missing required field,
- **6998** — Bad crontab parameters

Return:

identifier of just created object

v1.device.settings.export.rule.test

```
Boolean v1.device.settings.export.rule.test(String auth_token, SettingsExportRule rule)
```

Test connection for settings export rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule specifications

Errors:

- **102** — Access denied

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20001** — Rule test failed: connection error
- **20002** — Rule test failed: authentication error

Return:

Always returns `true`.

v1.device.settings.export.rule.fetch

```
SettingsExportRule v1.device.settings.export.rule.fetch(String auth_token, UID rule_id)
```

Fetch settings export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

rule

v1.device.settings.export.rule.update

```
Boolean v1.device.settings.export.rule.update(String auth_token, UID rule_id, SettingsExportRule rule)
```

Update existing settings export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **rule** — rule specifications

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

Always returns `true`.

v1.device.settings.export.rule.delete

```
Boolean v1.device.settings.export.rule.delete(String auth_token, UID rule_id)
```

Fetch settings export rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Rule not found

Return:

Always returns `true`.

v1.device.hardware.raid.status

```
HardwareRAIDInfo v1.device.hardware.raid.status(String auth_token)
```

Fetch hardware RAID status information.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20003** — No hardware RAID detected
- **20004** — Required RAID component is missing

Return:

hardware RAID status information

v1.device.power.supply.status

```
List<DevicePowerSupplyInfo> v1.device.power.supply.status(String auth_token)
```

Fetch power supply status.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **20005** — Required power supply unit status component is missing

Return:

list of status objects

v1.device.dns.query

```
DnsQueryInfo v1.device.dns.query(String auth_token, DnsQueryParams params)
```

Performs DNS lookups.

Parameters:

- **auth_token** — authentication token
- **params** — DNS query parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some parameter value is invalid
- **601** — Missing required parameter

Return:

Structure `DnsQueryInfo` with DNS query result.

v1.device.logs.remove.archive.file

Boolean **v1.device.logs.remove.archive.file**(String auth_token)

Remove file with archived logs.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **2001** — Remove file error

Return:

Always return `true`.

diagnostics

Module methods

v1.diagnostic.ethtool.info	135	v1.diagnostic.pim.routers.info	138
v1.diagnostic.interface.info	135	v1.diagnostic.pim.interface.info	138
v1.diagnostic.conntrack.list	136	v1.diagnostic.routing.info	139
v1.diagnostic.flow.list	136	v1.diagnostic.interface.brief	139
v1.diagnostic.parsed.routes	136	v1.diagnostic.cli.dig	139
v1.diagnostic.igmp.routers.info	137	v1.diagnostic.multicast.info	140
v1.diagnostic.igmp.interface.info	137		

v1.diagnostic.ethtool.info

String **v1.diagnostic.ethtool.info**(String auth_token, String interface)

String **v1.diagnostic.ethtool.info**(String auth_token, String node, String interface)

Return ethtool text output for node network interface.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **interface** — name of network interface

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect node or interface name.

Return:

Interface information from ethtool

v1.diagnostic.interface.info

[DiagnosticNetworkInterfaceInfo](#) **v1.diagnostic.interface.info**(String auth_token, String node, String interface)

[DiagnosticNetworkInterfaceInfo](#) **v1.diagnostic.interface.info**(String auth_token, String interface)

Return detailed network interface information.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **interface** — name of network interface

Errors:

- **102** — Access denied

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **107** — Incorrect interface name.
- **110** — Incorrect interface name.

Return:

Detailed Interface information

v1.diagnostic.conntrack.list

```
String v1.diagnostic.conntrack.list(String auth_token, Struct filter)
```

```
String v1.diagnostic.conntrack.list(String auth_token, String node, Struct filter)
```

Return conntrack -L output filtered with "filter" struct.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **filter** — Key-value struct with filtering params. Supported filter keys: src_ip, dst_ip, protocol.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

conntrack -L output

v1.diagnostic.flow.list

```
List<FlowResult> v1.diagnostic.flow.list(String auth_token, Struct filter)
```

```
List<FlowResult> v1.diagnostic.flow.list(String auth_token, String node, Struct filter)
```

Return session from ebpf on node

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **filter** — Key-value struct with filtering params. Supported all keys from FlowResult.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

sessions list

v1.diagnostic.parsed.routes

```
List<ParsedRoutes> v1.diagnostic.parsed.routes(String auth_token, RoutingInfoParams params)
```

```
List<ParsedRoutes> v1.diagnostic.parsed.routes(String auth_token, String node, RoutingInfoParams params)
```

Return routes information about routes.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — filtering params

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

ParsedRoutes structures list

v1.diagnostic.igmp.routers.info

```
String v1.diagnostic.igmp.routers.info(String auth_token, RoutersInfoParams params)
```

```
String v1.diagnostic.igmp.routers.info(String auth_token, String node, RoutersInfoParams params)
```

Detailed information for igmp routers

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — supported option: "groups ", "join ", "sources ", "statistics "

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.igmp.interface.info

```
String v1.diagnostic.igmp.interface.info(String auth_token, RoutersInfoParams params)
```

```
String v1.diagnostic.igmp.interface.info(String auth_token, String node, RoutersInfoParams params)
```

Detailed information for igmp interface

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — supported option: "interface_name ", "traffic ", "detail "

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.pim.routers.info

```
String v1.diagnostic.pim.routers.info(String auth_token, RoutersInfoParams params)
```

```
String v1.diagnostic.pim.routers.info(String auth_token, String node, RoutersInfoParams params)
```

Detailed information for pim routers

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — supported option: "group-type", "join", "neighbor", "rp-info", "secondary", "state", "statistics", "nexthop", "vxlan-groups"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.pim.interface.info

```
String v1.diagnostic.pim.interface.info(String auth_token, RoutersInfoParams params)
```

```
String v1.diagnostic.pim.interface.info(String auth_token, String node, RoutersInfoParams params)
```

Detailed information for pim interface

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — supported option: "interface_name", "traffic", "detail"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.routing.info

```
String v1.diagnostic.routing.info(String auth_token, RoutingInfoParams params)
```

```
String v1.diagnostic.routing.info(String auth_token, String node, RoutingInfoParams params)
```

Summary information about network routing

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — filtering params

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.interface.brief

```
String v1.diagnostic.interface.brief(String auth_token, Struct params)
```

```
String v1.diagnostic.interface.brief(String auth_token, String node, Struct params)
```

Summary information about network interfaces

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — optional key "vrf" with vrf name

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

v1.diagnostic.cli.dig

```
String v1.diagnostic.cli.dig(String auth_token, Struct params)
```

```
String v1.diagnostic.cli.dig(String auth_token, String node, Struct params)
```

Dig command for cli

Parameters:

- **auth_token** — authentication token
- **node** — name of node

- **params** — supported keys: "dns ", "nocmd ", "host ", "reverse ", "noall ", "answer ", "tcp "

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

String with dig output

v1.diagnostic.multicast.info

String **v1.diagnostic.multicast.info**(String auth_token, Struct params)

String **v1.diagnostic.multicast.info**(String auth_token, String node, Struct params)

Return detailed multicast table.

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **params** — supported keys: "vrf ", "fill ", "option "; supported option: "count ", "summary "

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Detailed routing information.

Component: Events

Module methods

v1.events.add..... 141

v1.events.add

Boolean **v1.events.add**(String auth_token, [EventInfo](#) event)

Add record to system event log

Parameters:

- **auth_token** — authentication token
- **event** — information about event

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

Component: Firewall

Module methods

v1.firewall.rules.list	142	v1.firewall.rule.move.bottom.....	145
v1.firewall.rule.fetch	142	v1.firewall.rule.move.up	146
v1.firewall.rule.add.....	143	v1.firewall.rule.move.down	146
v1.firewall.rule.add.lite	143	v1.firewall.rule.move.position	147
v1.firewall.rule.update	144	v1.firewall.sessions.clear.....	147
v1.firewall.rule.delete	144	v1.firewall.policylog.module.info.....	147
v1.firewall.rule.delete.lite.....	145	v1.firewall.policylog.module.update.....	148
v1.firewall.rule.move.top.....	145		

v1.firewall.rules.list

```
PagingList<FirewallRuleInfo> v1.firewall.rules.list(String auth_token, Integer start, Integer limit, QueryFilterInfo filter)
```

Fetch page of firewall rules.

Parameters:

- **auth_token** — authentication token
- **start** — offset
- **limit** — size of page
- **filter** — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters
- **618** — Incorrect field query value in search filter

Return:

page of firewall rules

v1.firewall.rule.fetch

```
FirewallRuleInfo v1.firewall.rule.fetch(String auth_token, UID id)
```

Fetch firewall rule with given identifier.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **110** — Invalid method argument
- **404** — Firewall rule with given id not found

Return:

firewall rule with given id

v1.firewall.rule.add

```
UID v1.firewall.rule.add(String auth_token, FirewallRuleInfo info)
```

```
Boolean v1.firewall.rule.add(String auth_token, List<FirewallRuleInfo> items)
```

Add firewall rule.

Parameters:

- **auth_token** — authentication token
- **info** — firewall rule
- **items** — list of firewall items to add (only for MC >= 7.2.0)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **112** — Empty list of items
- **113** — Too many items (only when the second argument is an array)
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field
- **614** — Config locked
- **615** — Too much geoip
- **621** — DB key already exists (only when the second argument is an array)

Return:

identifier of just created rule or true in case of multiple items

v1.firewall.rule.add.lite

```
UID v1.firewall.rule.add.lite(String auth_token, FirewallRuleInfo info)
```

Add firewall rule without validate positioning and no sync event. **To apply changes need restart node(s).**

Parameters:

- **auth_token** — authentication token
- **info** — firewall rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **601** — Missing required field

- **614** — Config locked
- **615** — Too much geoip

Return:

identifier of just created rule

v1.firewall.rule.update

```
Boolean v1.firewall.rule.update(String auth_token, UID id, FirewallRuleInfo info)
```

Update firewall rule with given identifier. Field **position** is ignored, to move rule you must use one of corresponding move-methods.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule
- **info** — firewall rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **111** — Some field value in passed structure is invalid
- **404** — Firewall rule with given id not found
- **409** — Duplicate value, conflict with the existing object
- **614** — Config locked
- **615** — Too much geoip

Return:

true

v1.firewall.rule.delete

```
Boolean v1.firewall.rule.delete(String auth_token, UID id)
```

```
Boolean v1.firewall.rule.delete(String auth_token, List<UID> ids)
```

Delete firewall rule with given identifier.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule
- **ids** — list of rule identifiers to delete

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **112** — Empty list of items
- **113** — Too many items (only when the second argument is an array)

- **404** — Firewall rule with given id not found
- **614** — Config locked
- **620** — Transaction aborted

Return:

true

v1.firewall.rule.delete-lite

Boolean **v1.firewall.rule.delete-lite**(String auth_token, UID id)

Delete firewall rule with given identifier without validate positioning and no sync event. **To apply changes need restart node(s).**

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule
- **ids** — list of rule identifiers to delete

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Firewall rule with given id not found
- **614** — Config locked

Return:

true

v1.firewall.rule.move.top

Boolean **v1.firewall.rule.move.top**(String auth_token, UID id)

Move firewall rule to the top of rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **404** — Firewall rule with given id not found

Return:

true

v1.firewall.rule.move.bottom

Boolean **v1.firewall.rule.move.bottom**(String auth_token, UID id)

Move firewall rule to the bottom of rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **404** — Firewall rule with given id not found

Return:

true

v1.firewall.rule.move.up

```
Boolean v1.firewall.rule.move.up(String auth_token, UID id)
```

Move firewall rule up one position in rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **404** — Firewall rule with given id not found

Return:

true

v1.firewall.rule.move.down

```
Boolean v1.firewall.rule.move.down(String auth_token, UID id)
```

Move firewall rule down one position in rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **404** — Firewall rule with given id not found

Return:

true

v1.firewall.rule.move.position

```
Boolean v1.firewall.rule.move.position(String auth_token, UID rule_id, Variant position)
```

Move rule to given position in the list. will be moved to the top, if position is more than last list element position, the rule will be moved to the end of list

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **position** — new rule position, integer value starting with `1`, or string value `"top"` to move to the top of the list, or `"last"` to move to the bottom of the list

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid argument
- **356** — Incorrect value in "position" argument
- **404** — rule not found

Return:

Always returns `true`.

v1.firewall.sessions.clear

```
Boolean v1.firewall.sessions.clear(String auth_token)
```

Clear firewall sessions on all nodes.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.firewall.policylog.module.info

```
DefaultFirewallPolicyInfo v1.firewall.policylog.module.info(String auth_token)
```

Fetch parameters of default firewall policy that triggered if no other policy is matched when processing network traffic.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Default firewall policy as a structure of type [DefaultFirewallPolicyInfo](#).

v1.firewall.policylog.module.update

```
Boolean v1.firewall.policylog.module.update(String auth_token, DefaultFirewallPolicyInfos)
```

Update parameters of default firewall policy that triggered if no other policy is matched when processing network traffic.

Parameters:

- **auth_token** — authentication token
- **s** — default firewall policy information

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid

Return:

Always returns `true`.

Component: High Availability

Module methods

v1.ha.cluster.state	149	v1.ha.cluster.update	151
v1.ha.cluster.state	149	v1.ha.cluster.fetch	151
v1.ha.cluster.failover	150	v1.ha.cluster.delete	152
v1.ha.cluster.failover	150	v1.ha.available.nodes.list	152
v1.ha.clusters.list	150	v1.ha.cluster.failed.checks	152
v1.ha.cluster.add	151	v1.ha.cluster.force.disable	153

v1.ha.cluster.state

`HAClusterState v1.ha.cluster.state(String auth_token)`

Get current HA cluster state.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Current cluster not found

Return:

HAClusterState structure

v1.ha.cluster.state

`HAClusterState v1.ha.cluster.state(String auth_token, UID cluster_id)`

Get HA cluster state by cluster id.

Parameters:

- **auth_token** — authentication token
- **cluster_id** — cluster id

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found

Return:

HAClusterState structure

v1.ha.cluster.failover

```
HAClusterFailover v1.ha.cluster.failover(String auth_token, HAClusterEventFilter filter)
```

Get current HA cluster state.

Parameters:

- **auth_token** — authentication token
- **filter** — event filter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found

Return:

HAClusterFailover structure

v1.ha.cluster.failover

```
HAClusterFailover v1.ha.cluster.failover(String auth_token, UID cluster_id, HAClusterEventFilter filter)
```

Get current HA cluster state.

Parameters:

- **auth_token** — authentication token
- **cluster_id** — cluster id
- **filter** — event filter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found

Return:

HAClusterFailover structure

v1.ha.clusters.list

```
List<HAClusterInfo> v1.ha.clusters.list(String auth_token)
```

Fetch list of call HA clusters.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of [HAClusterInfo](#) structures

v1.ha.cluster.add

```
UID v1.ha.cluster.add(String auth_token, HAClusterInfo cluster)
```

Add a new HA cluster. You need to correctly specify master and backup instances.

Parameters:

- **auth_token** — authentication token
- **cluster** — cluster spec

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **11009** — Conflict HA cluster tag
- **11010** — Conflict virtual ip values

Return:

identifier of just created cluster

v1.ha.cluster.update

```
Boolean v1.ha.cluster.update(String auth_token, UID cluster_id, HAClusterInfo cluster)
```

Update cluster info.

Parameters:

- **auth_token** — authentication token
- **cluster_id** — cluster identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found
- **11009** — Conflict HA cluster tag
- **11010** — Conflict virtual ip values

Return:

Always returns `true`.

v1.ha.cluster.fetch

```
HAClusterInfo v1.ha.cluster.fetch(String auth_token, UID cluster_id)
```

Fetch cluster info.

Parameters:

- **auth_token** — authentication token

- **cluster_id** — cluster identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found

Return:

cluster spec as a structure [HAClusterInfo](#)

v1.ha.cluster.delete

```
Boolean v1.ha.cluster.delete(String auth_token, UID cluster_id)
```

Delete cluster.

Parameters:

- **auth_token** — authentication token
- **cluster_id** — cluster identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Cluster not found

Return:

true

v1.ha.available.nodes.list

```
List<HAClusterAvailableNodeInfo> v1.ha.available.nodes.list(String auth_token, UID cluster_id)
```

List available nodes for this cluster.

Parameters:

- **auth_token** — authentication token
- **cluster_id** — cluster identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of structures of type [HAClusterAvailableNodeInfo](#)

v1.ha.cluster.failed.checks

```
List<HAClusterFailedCheckInfo> v1.ha.cluster.failed.checks(String auth_token)
```

```
List<HAClusterFailedCheckInfo> v1.ha.cluster.failed.checks(String auth_token, String node)
```

List of node failed checks.

Parameters:

- **auth_token** — authentication token
- **node** — name of node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of structures of type [HAClusterFailedCheckInfo](#)

v1.ha.cluster.force.disable

```
Boolean v1.ha.cluster.force.disable(String auth_token, String node_name, Boolean mode)
```

```
Boolean v1.ha.cluster.force.disable(String auth_token, Boolean mode)
```

Force disable a dedicated node from a cluster.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name
- **mode** — force disable mode

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

@return true

Component: Http Response

Module methods

v1.httpresponse.profiles.list.....	154	v1.httpresponse.profile.update	155
v1.httpresponse.profile.add	154	v1.httpresponse.profile.delete	155
v1.httpresponse.profile.fetch.....	155		

v1.httpresponse.profiles.list

```
List<ResponseProfileInfo> v1.httpresponse.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all response profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of response profiles

v1.httpresponse.profile.add

```
UID v1.httpresponse.profile.add(String auth_token, ResponseProfileInfo profile)
```

Create a new response profile.

Parameters:

- **auth_token** — authentication token
- **profile** — a new profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added profile

v1.httpresponse.profile.fetch

```
ResponseProfileInfo v1.httpresponse.profile.fetch(String auth_token, UID id)
```

Fetch a single response profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found

Return:

response profile

v1.httpresponse.profile.update

```
Boolean v1.httpresponse.profile.update(String auth_token, UID id, ResponseProfileInfo profile)
```

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier
- **profile** — profile parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Profile with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.httpresponse.profile.delete

```
Boolean v1.httpresponse.profile.delete(String auth_token, UID id)
```

Delete response profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true

Component: JSON

Module methods

v1.json.profiles.list.....	157	v1.json.schema.add	159
v1.json.profile.add	157	v1.json.schema.valid.....	160
v1.json.profile.fetch.....	158	v1.json.schema.fetch	160
v1.json.profile.update	158	v1.json.schema.update.....	160
v1.json.profile.delete	159	v1.json.schema.delete	161
v1.json.schemas.list.....	159		

v1.json.profiles.list

`PagingList<JsonProfileInfo> v1.json.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)`

Fetch JSON profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

List of JSON profiles

v1.json.profile.add

`UID v1.json.profile.add(String auth_token, JsonProfileInfo profile)`

Create a new JSON profile.

Parameters:

- **auth_token** — authentication token
- **profile** — a new profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added profile

v1.json.profile.fetch

```
JsonProfileInfo v1.json.profile.fetch(String auth_token, UID id)
```

Fetch a single JSON profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found

Return:

JSON profile

v1.json.profile.update

```
Boolean v1.json.profile.update(String auth_token, UID id, JsonProfileInfo profile)
```

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier
- **profile** — profile parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Profile with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.json.profile.delete

```
Boolean v1.json.profile.delete(String auth_token, UID id)
```

Delete JSON profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Object with given identifier not found
- **502** — Profile cannot be deleted because it's being used by another object

Return:

true

v1.json.schemas.list

```
PagingList<JsonSchemaInfo> v1.json.schemas.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch JSON schemas.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by schema name and description)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

List of JSON schemas

v1.json.schema.add

```
UID v1.json.schema.add(String auth_token, JsonSchemaInfo schema)
```

Create a new JSON schema.

Parameters:

- **auth_token** — authentication token
- **schema** — a new schema

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added schema

v1.json.schema.valid

```
JsonSchemaInfo v1.json.schema.valid(String auth_token, String schema)
```

Schema validation

Parameters:

- **auth_token** — authentication token
- **schema** — json schema

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

object {success: true}

v1.json.schema.fetch

```
JsonSchemaInfo v1.json.schema.fetch(String auth_token, UID id)
```

Fetch a single JSON schema.

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Schema with given identifier not found

Return:

JSON schema

v1.json.schema.update

```
Boolean v1.json.schema.update(String auth_token, UID id, JsonSchemaInfo schema)
```

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier
- **schema** — schema parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Schema with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.json.schema.delete

Boolean **v1.json.schema.delete**(String auth_token, UID id)

Delete JSON schema.

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Schema with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

@return true

Component: Ldap

Module methods

v1.ldap.users.list.....	162	v1.ldap.user.fetch	163
v1.ldap.groups.list	162	v1.ldap.group.fetch	163
v1.ldap.clear.cache	163		

v1.ldap.users.list

```
List<LdapUserAccountInfo> v1.ldap.users.list(String auth_token, UID  
ldap_server_id, String pattern)
```

Fetch list of ldap users by pattern in givenName, sn, sAMAccountName", cn.

Parameters:

- **auth_token** — authentication token
- **ldap_server_id** — unique identifier of ldap server
- **pattern** — filter by this substring

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Ldap server with given id not found
- **12002** — Can't connect to Ad server
- **12003** — Can't bind to Ad server
- **12005** — LDAP/AD server is disabled

Return:

List of ldap users

v1.ldap.groups.list

```
List<LdapGroupAccountInfo> v1.ldap.groups.list(String auth_token, UID ldap_server_id,  
String pattern)
```

Fetch list of ldap groups by pattern.

Parameters:

- **auth_token** — authentication token
- **ldap_server_id** — unique identifier of ldap server
- **pattern** — filter by this substring

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Ldap server with given id not found
- **12002** — Can't connect to Ad server

- **12003** — Can't bind to Ad server
- **12005** — LDAP/AD server is disabled

Return:

List of ldap groups

v1.ldap.clear.cache

```
Boolean v1.ldap.clear.cache(String auth_token)
```

Clear LDAP users cache.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v1.ldap.user.fetch

```
Struct v1.ldap.user.fetch(String auth_token, String guid)
```

Fetch LDAP user by its GUID.

Parameters:

- **auth_token** — authentication token
- **guid** — User GUID

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — User not found

Return:

Structure with fields: **guid**, **name**, **ldap_dn**

v1.ldap.group.fetch

```
Struct v1.ldap.group.fetch(String auth_token, String guid)
```

Fetch LDAP group by its GUID.

Parameters:

- **auth_token** — authentication token
- **guid** — Group GUID

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Group not found

Return:

Structure with fields: **guid**, **name**

Component: Libraries

Module methods	
v1.libraries.geoip.countries.list	165
v1.libraries.permissions.categories.list	165
v1.libraries.tls.ciphers.list	166
v1.libraries.application.level.gateways.list....	166
v1.libraries.response.page.templates.list	166
v1.libraries.response.page.template.add	167
v1.libraries.response.page.template.update	167
v1.libraries.response.page.template.data.update	168
v1.libraries.response.page.template.delete..	168
v1.libraries.response.page.template.fetch	169
v1.libraries.response.page.template.public.fetch	169
v1.libraries.response.page.template.public.data.fetch	169
v1.libraries.response.page.default.templates.list	170

v1.libraries.geoip.countries.list

```
List<GeoipCountry> v1.libraries.geoip.countries.list(String auth_token)
```

Fetch list of all available countries for geoip.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **4004** — Geoip database is not available

Return:

true

v1.libraries.permissions.categories.list

```
List<String> v1.libraries.permissions.categories.list(String auth_token, String component)
```

Fetch list of all available WebUI permission categories.

Parameters:

- **auth_token** — authentication token
- **component** — component permissions you want obtain permission categories for, allowed values are: "rpc" , "webui" , "cli"

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Unknown component

Return:

permissions list

v1.libraries.tls.ciphers.list

```
List<Struct> v1.libraries.tls.ciphers.list(String auth_token)
```

Fetch list of all supported TLS ciphers.

Parameters:

- **auth_token** — authentication token

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of the following structures: **id**, **security**, **protocols**

Fields:

id	String	cipher identifier as IANA name
security	String	cipher security, one of the following values: "secure" , "weak"
protocols	List<String>	list of TLS protocols this cipher is applicable to, possible values are: "ssl3" , "tlsv1" , "tlsv1.1" , "tlsv1.2" , "tlsv1.3"

v1.libraries.application.level.gateways.list

```
List<Struct> v1.libraries.application.level.gateways.list(String auth_token)
```

Returns list of available service application level gateways (ALG) modules.

Parameters:

- **auth_token** — authentication token

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **102** — Access denied

Return:

Return list of structures with following fields:

Fields:

id	String	service identifier
protocols	List<String>	list of supported protocols: "tcp" , "udp" .

v1.libraries.response.page.templates.list

```
List<ResponsePageTemplateInfo> v1.libraries.response.page.templates.list(String auth_token)
```

Fetch a list of all response page templates.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of response page templates

v1.libraries.response.page.template.add

```
UID v1.libraries.response.page.template.add(String auth_token, ResponsePageTemplateInfo template)
```

Create a new response page template. On create template is initialized by default template HTML data for given type.

Parameters:

- **auth_token** — authentication token
- **template** — template information

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **112** — Invalid method format
- **409** — Name already exists
- **601** — Missing required field

Return:

identifier of just created template

v1.libraries.response.page.template.update

```
Boolean v1.libraries.response.page.template.update(String auth_token, UID template_id, ResponsePageTemplateInfo template)
```

Update template info. You cannot update template HTML data using this method, instead you should call [v1.libraries.response.page.template.data.update](#).

Parameters:

- **auth_token** — authentication token
- **template_id** — template identifier
- **template** — template information

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **111** — Some field value in passed structure is invalid

- **409** — Object with same data already exists
- **404** — Template not found
- **4003** — You cannot turn off default state from template

Return:

true

v1.libraries.response.page.template.data.update

```
Boolean v1.libraries.response.page.template.data.update(String auth_token, UID
template_id, Base64 data)
```

Update template HTML data.

Parameters:

- **auth_token** — authentication token
- **template_id** — template identifier
- **data** — template data, usually a text/html encoded as base64 value, max size is 5 MB

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **112** — Invalid method format
- **404** — Template not found
- **4006** — The template page size is too large

Return:

true

v1.libraries.response.page.template.delete

```
Boolean v1.libraries.response.page.template.delete(String auth_token, UID template_id)
```

Delete response page template. Default template cannot be deleted.

Parameters:

- **auth_token** — authentication token
- **template_id** — template identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **112** — Invalid method format
- **404** — Template not found
- **502** — Object cannot be deleted because it's being used by another object
- **4002** — Cannot delete default template

Return:

true

v1.libraries.response.page.template.fetch

```
ResponsePageTemplateInfo v1.libraries.response.page.template.fetch(String auth_token, UID template_id)
```

Fetch information about template.

Parameters:

- **auth_token** — authentication token
- **template_id** — template identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **112** — Invalid method format
- **404** — Template not found

Return:

information about template

v1.libraries.response.page.template.public.fetch

```
ResponsePageTemplateInfo v1.libraries.response.page.template.public.fetch(String template_type, UID template_id)
```

Fetch information about template. This method does not require authentication. If system cannot find template by specified type and ID it will return default template HTML data for that type.

Parameters:

- **template_type** — template type
- **template_id** — template identifier, special numeric value `-1` could be used to force default template data request

Errors:

- **4001** — Unknown template type

Return:

information about template

v1.libraries.response.page.template.public.data.fetch

```
String v1.libraries.response.page.template.public.data.fetch(String template_type, String template_id)
```

Fetch template data. This method does not require authentication. If system cannot find template by specified type and ID it will return default template HTML data for that type.

Parameters:

- **template_type** — template type
- **template_id** — template identifier, special numeric value `-1` could be used to force default template data request

Errors:

- **4001** — Unknown template type

Return:

Template HTML data as a plain string

v1.libraries.response.page.default.templates.list

```
List<Struct> v1.libraries.response.page.default.templates.list(String auth_token)
```

List of all supported default response page templates.

Parameters:

- **auth_token** — authentication token

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **112** — Invalid method format

Return:

List of the following structures:

Fields:

id	String	template code name
response_page_type	String	template type
name	String	template name

Component: Network manager

Module methods

v1.netmanager.zones.list	171	v1.netmanager.failover.config.set	183
v1.netmanager.zones.sync	172	v1.netmanager.virtualrouters.list	184
v1.netmanager.zone.add	172	v1.netmanager.virtualrouters.list.routes.list ..	184
v1.netmanager.zone.fetch	172	v1.netmanager.virtualrouter.fetch	184
v1.netmanager.zone.update	173	v1.netmanager.virtualrouter.add	185
v1.netmanager.zone.delete	173	v1.netmanager.virtualrouter.update	185
v1.netmanager.public.interfaces.list	173	v1.netmanager.virtualrouter.update.route.add	186
v1.netmanager.interfaces.list	174	v1.netmanager.virtualrouter.update.route.update	186
v1.netmanager.interface.fetch	175	v1.netmanager.virtualrouter.update.route.delete	187
v1.netmanager.interface.add.bond	175	v1.netmanager.virtualrouter.delete	187
v1.netmanager.interface.add.bridge	176	v1.netmanager.netflow.profiles.list	188
v1.netmanager.interface.add.vlan	176	v1.netmanager.netflow.profile.fetch	188
v1.netmanager.interface.add.loopback	177	v1.netmanager.netflow.profile.add	188
v1.netmanager.public.interface.update	177	v1.netmanager.netflow.profile.update	189
v1.netmanager.interface.update	178	v1.netmanager.netflow.profile.delete	189
v1.netmanager.bulk.interface.update	178	v1.netmanager.flow.monitor.stat	189
v1.netmanager.interface.delete	179	v1.netmanager.ldap.profiles.list	190
v1.netmanager.gateway.list	179	v1.netmanager.ldap.profile.fetch	190
v1.netmanager.gateway.ping	180	v1.netmanager.ldap.profile.add	190
v1.netmanager.gateway.fetch	180	v1.netmanager.ldap.profile.update	191
v1.netmanager.gateway.add	181	v1.netmanager.ldap.profile.delete	191
v1.netmanager.gateway.update	181	v1.netmanager.ldap.neighbors	192
v1.netmanager.gateway.delete	182	v1.netmanager.ldap.statistics	192
v1.netmanager.public.route.add	182	v1.netmanager.geoip.lookup.ip	192
v1.netmanager.routes.preview	182		
v1.netmanager.failover.config.get	183		

v1.netmanager.zones.list

```
List<NetworkZoneInfo> v1.netmanager.zones.list(String auth_token)
```

List of all security zones.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

All security zones for current cluster.

v1.netmanager.zones.sync

```
Boolean v1.netmanager.zones.sync(String auth_token)
```

Sync zones.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v1.netmanager.zone.add

```
UID v1.netmanager.zone.add(String auth_token, NetworkZoneInfo zone)
```

Add new security zone.

Parameters:

- **auth_token** — authentication token
- **zone** — zone data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **408** — Maximum number of zones limit (254) is reached
- **409** — Zone with such parameters already exists.
- **601** — Missing required field

Return:

identifier of just created zone

v1.netmanager.zone.fetch

```
NetworkZoneInfo v1.netmanager.zone.fetch(String auth_token, UID zone_id)
```

Fetch zone info by zone identifier

Parameters:

- **auth_token** — authentication token
- **zone_id** — zone identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument

- **404** — Zone not found

Return:

Structure [NetworkZoneInfo](#) with zone information.

v1.netmanager.zone.update

```
Boolean v1.netmanager.zone.update(String auth_token, UID zone_id, NetworkZoneInfo zone)
```

Update existing network zone.

Parameters:

- **auth_token** — authentication token
- **zone_id** — zone identifier
- **zone** — zone data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **111** — Some field value is invalid
- **404** — Zone not found.
- **409** — Zone with such parameters already exists
- **601** — Missing required field

Return:

true

v1.netmanager.zone.delete

```
Boolean v1.netmanager.zone.delete(String auth_token, UID zone_id)
```

Delete existing security zone.

Parameters:

- **auth_token** — authentication token
- **zone_id** — identifier of zone that should be deleted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **404** — Zone not found.

Return:

true

v1.netmanager.public.interfaces.list

```
List<NetworkInterfaceInfo> v1.netmanager.public.interfaces.list()
```

List of all network interfaces for current node.

Works only for not-initialized node, on the initialized node throws error 102.

Errors:

- **102** — Access denied (node)
- **616** — Invalid node

Return:

list of all local network interfaces

v1.netmanager.interfaces.list

```
PagingList<NetworkInterfaceInfo> v1.netmanager.interfaces.list(String auth_token, String node_name, Integer start, Integer limit, Struct pattern)
```

```
PagingList<NetworkInterfaceInfo> v1.netmanager.interfaces.list(String auth_token, Integer start, Integer limit, Struct pattern)
```

Fetch one page from list of all network interfaces for selected node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if called method without this argument then interfaces for all available nodes are returned
- **start** — offset in complete list
- **limit** — max length of returned list
- **pattern** — structure that defines filtering options.

Filtering options structure in argument pattern may contain the following fields:

Field	Type	Description
"enabled"	Boolean	if <code>true</code> include enabled interfaces only, if <code>false</code> return disabled interfaces only, if omitted return all interfaces
"node"	String	if specified include interfaces only from the specified node, parameter is ignored if node name is passed as method argument
"include_vpn_tunnels"	Boolean	if <code>true</code> include node VPN interfaces, by default is <code>false</code>
"include_cluster_interfaces"	Boolean	if <code>true</code> include cluster interfaces, by default is <code>true</code>
"kind"	String	kind of included interface, all other kinds are excluded, by default is not set so all kinds are included, allowed values are: <code>"vpn"</code> , <code>"vlan"</code> , <code>"adapter"</code> , <code>"bridge"</code> , <code>"bond"</code> , <code>"ppp"</code> , <code>"gre"</code>

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters
- **616** — Invalid node

Return:

list of all interfaces for the node as a list of structures [NetworkInterfaceInfo](#)

v1.netmanager.interface.fetch

```
NetworkInterfaceInfo v1.netmanager.interface.fetch(String auth_token, String ifname)
```

```
NetworkInterfaceInfo v1.netmanager.interface.fetch(String auth_token, String node_name,  
String ifname)
```

Fetch interface details.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **ifname** — interface name, e.g.: "port2"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

interface details

v1.netmanager.interface.add.bond

```
String v1.netmanager.interface.add.bond(String auth_token, String name,  
NetworkInterfaceInfo iface)
```

```
String v1.netmanager.interface.add.bond(String auth_token, String node_name, String name,  
NetworkInterfaceInfo iface)
```

Create a new bond.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **ifname** — interface name, it must have the following format: "bondX" where X is a positive number
- **iface** — interface (bond) details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **409** — Duplicate name, conflict with the existing object
- **601** — Missing required field
- **616** — Invalid node
- **1001** — Specified master interface not found
- **1202** — Sub interface usage in other interface
- **1205** — IP address conflict

Return:

Name of just created interface.

v1.netmanager.interface.add.bridge

```
String v1.netmanager.interface.add.bridge(String auth_token, String ifname,  
NetworkInterfaceInfo iface)
```

```
String v1.netmanager.interface.add.bridge(String auth_token, String node_name, String  
ifname, NetworkInterfaceInfo iface)
```

Create a new bridge.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **ifname** — interface name, it must have the following format: "bridgeX" where X is a positive number
- **iface** — interface (bridge) details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **409** — Duplicate name, conflict with the existing object
- **601** — Missing required field
- **616** — Invalid node
- **1001** — Specified master interface not found
- **1202** — Sub interface usage in other interface
- **1203** — Bridge interface must have exactly two adapter sub interfaces or one bypass interface
- **1205** — IP address conflict

Return:

Name of just created interface.

v1.netmanager.interface.add.vlan

```
String v1.netmanager.interface.add.vlan(String auth_token, String ifname,  
NetworkInterfaceInfo iface)
```

```
String v1.netmanager.interface.add.vlan(String auth_token, String node_name, String  
ifname, NetworkInterfaceInfo iface)
```

Add new VLAN interface.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **ifname** — this value is ignored and interface name constructed from base interface name and VLAN tag
- **iface** — interface (vlan) details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **409** — Duplicate name, conflict with the existing object
- **601** — Missing required field
- **616** — Invalid node
- **1001** — Specified master interface not found
- **1205** — IP address conflict

Return:

Name of just created interface.

v1.netmanager.interface.add.loopback

```
String v1.netmanager.interface.add.loopback(String auth_token, String ifname,  
NetworkInterfaceInfo iface)
```

```
String v1.netmanager.interface.add.loopback(String auth_token, String node_name, String  
ifname, NetworkInterfaceInfo iface)
```

Add new Loopback interface.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **ifname** — interface name, it must have the following format: "loopbackX" where X is a positive number and the total name length is 15 characters or less
- **iface** — interface (loopback) details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **409** — Duplicate name, conflict with the existing object
- **601** — Missing required field
- **616** — Invalid node
- **1001** — Specified master interface not found
- **1205** — IP address conflict

Return:

Name of just created interface.

v1.netmanager.public.interface.update

```
Boolean v1.netmanager.public.interface.update(String ifname, NetworkInterfaceInfo iface)
```

Configure network interface. Works only for not yet initialized node, on the initialized node throws error 102.

Parameters:

- **ifname** — interface name
- **iface** — parameters of the interface that need to be changed, readonly fields will be ignored

Errors:

- **102** — Access Denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v1.netmanager.interface.update

```
Boolean v1.netmanager.interface.update(String auth_token, String node_name, String ifname, NetworkInterfaceInfo iface)
```

Edit network interface. Not all interface parameters are mutable.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name
- **ifname** — interface name
- **iface** — parameters of the interface that need to be changed, readonly fields will be ignored

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node
- **1014** — Slave interface cannot be updated
- **1201** — Cannot edit local VPN tunnel interface

Return:

true

v1.netmanager.bulk.interface.update

```
Boolean v1.netmanager.bulk.interface.update(String auth_token, List<String> iface_id, NetworkInterfaceInfo iface)
```

Edit network interfaces. Not all interface parameters are mutable. If used only one interface, work like a [v1.netmanager.interface.update](#). Otherwise, only parameter "enabled" is allowed.

In case of error with code 111, error message is json-encoded format string, where keys are nodenames, values are other json struct, where key is iface name and value is error message.

Parameters:

- **auth_token** — authentication token
- **iface_id** — full name of interface. Like "iface_name:node_name"
- **iface** — parameters of the interface that need to be changed, readonly fields will be ignored

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Error text contains JSON-encoded string that provides error details for each affected interface grouped by node name
- **404** — Interface with given identifier not found
- **616** — Invalid node
- **1001** — Unknown setting name
- **1014** — Slave interface cannot be updated
- **1203** — Bridge interface must have exactly two adapter sub interfaces or one bypass interface

Return:

Always returns `true`.

v1.netmanager.interface.delete

```
Boolean v1.netmanager.interface.delete(String auth_token, String node_name, String ifname)
```

```
Boolean v1.netmanager.interface.delete(String auth_token, List<List> ifaces)
```

Delete VLAN interface. Physical interface cannot be deleted using this method.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name
- **ifname** — interface name
- **ifaces** — list of interface identifiers that should be deleted, each item is an array
[`NODE_NAME`, `IFACE_ID`]

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Interface not found
- **502** — Object cannot be deleted because it's being used by another object
- **616** — Invalid node
- **1003** — Physical interface cannot be deleted

Return:

`true`

v1.netmanager.gateways.list

```
List<GatewayInfo> v1.netmanager.gateways.list(String auth_token, Struct pattern)
```

```
List<GatewayInfo> v1.netmanager.gateways.list(String auth_token, String node_name, Struct pattern)
```

Fetch list of gateways on given node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **pattern** — support only key **enabled** for finding only enabled or disabled gateways

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

list of gateways in the form `GatewayInfo` structure

v1.netmanager.gateways.ping

```
Boolean v1.netmanager.gateways.ping(String auth_token)
```

```
Boolean v1.netmanager.gateways.ping(String auth_token, String node_name)
```

Ping gateways.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if parameter is omitted use current node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Always returns `true`.

v1.netmanager.gateway.fetch

```
GatewayInfo v1.netmanager.gateway.fetch(String auth_token, UID id)
```

```
GatewayInfo v1.netmanager.gateway.fetch(String auth_token, String node_name, UID id)
```

Fetch gateway information by unique identifier.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **id** — unique identifier of gateway

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Gateway not found
- **616** — Invalid node

Return:

gateway information in the form `GatewayInfo` structure

v1.netmanager.gateway.add

```
UID v1.netmanager.gateway.add(String auth_token, GatewayInfo info)
```

```
UID v1.netmanager.gateway.add(String auth_token, String node_name, GatewayInfo info)
```

Add new gateway to given node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **info** — information about gateway in the form `GatewayInfo` structure

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value is invalid
- **409** — Conflict with existing gateway
- **616** — Invalid node
- **1002** — Interface with given `"ifname"` not found
- **1019** — Gateway IP address conflict

Return:

id of just created gateway

v1.netmanager.gateway.update

```
Boolean v1.netmanager.gateway.update(String auth_token, UID id, GatewayInfo info)
```

```
Boolean v1.netmanager.gateway.update(String auth_token, String node_name, UID id, GatewayInfo info)
```

Update gateway with given id on given node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **id** — unique identifier of gateway
- **info** — updated information about gateway in the form `GatewayInfo` structure

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value is invalid
- **404** — Gateway not found
- **409** — Conflict with existing gateway
- **616** — Invalid node

- **1002** — Interface with given `"ifname"` not found
- **1019** — Gateway IP address conflict

Return:

true

v1.netmanager.gateway.delete

```
Boolean v1.netmanager.gateway.delete(String auth_token, UID id)
```

```
Boolean v1.netmanager.gateway.delete(String auth_token, String node_name, UID id)
```

Delete gateway with given id on given node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **id** — unique identifier of gateway

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Gateway not found
- **616** — Invalid node

Return:

true

v1.netmanager.public.route.add

```
UID v1.netmanager.public.route.add(RouteInfo route, Boolean flush)
```

Add new route. Works only for not yet initialized node, on the initialized node throws error 102.

Parameters:

- **route** — information about route as [RouteInfo](#) structure
- **flush** — set to `true` to flush routing table

Errors:

- **102** — Access Denied

Return:

identifier of just created route

v1.netmanager.routes.preview

```
String v1.netmanager.routes.preview(String auth_token, List<String> kinds, String vrf_name)
```

```
String v1.netmanager.routes.preview(String auth_token, String node_name, List<String> kinds, String vrf_name)
```

Preview routes table.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name, if omitted current node is assumed
- **kinds** — what route kinds to include in preview, possible values are: `"connected"`, `"kernel"`
- **vrf_name** — existing VRF name, empty string matches default VRF

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Routes list preview, contains limited list of routes.

v1.netmanager.failover.config.get

```
FailoverConfigInfo v1.netmanager.failover.config.get(String auth_token)
```

Fetch gateways connectivity checker configuration.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

FailoverConfigInfo

v1.netmanager.failover.config.set

```
Boolean v1.netmanager.failover.config.set(String auth_token, FailoverConfigInfo config)
```

Update gateways connectivity checker configuration.

Parameters:

- **auth_token** — authentication token
- **config** — connectivity checker configuration

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.netmanager.virtualrouters.list

```
List<VirtualRouterInfo> v1.netmanager.virtualrouters.list(String auth_token)
```

Fetch list of virtual routers for all nodes, the list always contains a router with id "NODE:default" for every cluster node with name NODE. For example, "utm@vbnmp:default".

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of virtual router records.

v1.netmanager.virtualrouters.list.routes.list

```
PagingList<RouteInfo> v1.netmanager.virtualrouters.list.routes.list(String auth_token,  
String id, Integer start, Integer limit, RouteInfo pattern, Struct filter,  
List<SortingSpec> sorting_options)
```

Fetch page of static routes

Parameters:

- **auth_token** — authentication token
- **id** — router identifier, is written in the form - "NODE_ID:OBJECT_ID"
- **start** — offset
- **limit** — size of page
- **pattern** — structure with filtering options as [RouteInfo](#) structure
- **filter** — structure with filtering options
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found
- **603** — Incorrect paging parameters

Return:

page of static routes

v1.netmanager.virtualrouter.fetch

```
VirtualRouterInfo v1.netmanager.virtualrouter.fetch(String auth_token, String id)
```

Fetch virtual router info.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Virtual router properties structure.

v1.netmanager.virtualrouter.add

```
UID v1.netmanager.virtualrouter.add(String auth_token, VirtualRouterInfo router)
```

Add a new virtual router, router name "default" is reserved and cannot be used.

Parameters:

- **auth_token** — authentication token
- **router** — virtual router parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **408** — Maximum number of virtual routers limit (255) is reached

Return:

Identifier of just created virtual router.

v1.netmanager.virtualrouter.update

```
Boolean v1.netmanager.virtualrouter.update(String auth_token, String id, VirtualRouterInfo router)
```

Update virtual router.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier
- **router** — virtual router parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Always returns `true`.

v1.netmanager.virtualrouter.update.route.add

```
Boolean v1.netmanager.virtualrouter.update.route.add(String auth_token, String id, RouteInfo route)
```

```
Boolean v1.netmanager.virtualrouter.update.route.add(String auth_token, String id, List<RouteInfo> routes)
```

Updates virtual router with adding one or more of static routes.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier, written in the form - "NODE_ID:OBJECT_ID"
- **route** — route parameter as [RouteInfo](#) structure
- **routes** — routes parameters as list of [RouteInfo](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **111** — Incorrect field data
- **404** — Object not found
- **409** — Duplicate name value
- **601** — Missing required field

Return:

Always returns `true`.

v1.netmanager.virtualrouter.update.route.update

```
Boolean v1.netmanager.virtualrouter.update.route.update(String auth_token, String id, RouteInfo mask, RouteInfo route)
```

```
Boolean v1.netmanager.virtualrouter.update.route.update(String auth_token, String id, List<RoutesInfo> routes)
```

Updates virtual router with updating one or more of static routes. Routes, being matched with the `mask`, will be replaced with the `route` data. If there're no matches, it returns `422 Mask not applied`.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier, is written in the form - "NODE_ID:OBJECT_ID"
- **mask** — finding filter parameter as part of [RouteInfo](#) structure
- **route** — route parameter as [RouteInfo](#) structure
- **routes** — routes parameters as list of [RoutesInfo](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **111** — Incorrect field data

- **404** — Object not found
- **409** — Duplicate name value
- **422** — Mask not applied

Return:

Always returns `true`.

v1.netmanager.virtualrouter.update.route.delete

```
Boolean v1.netmanager.virtualrouter.update.route.delete(String auth_token, String id, RouteInfo mask)
```

Updates virtual router with removing one or more of static routes. Routes, being matched with the `mask`, will be removed. If there're no matches, it returns `422 Mask not applied`.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier, is written in the form - `"NODE_ID:OBJECT_ID"`
- **mask** — finding filter parameters as part of `RouteInfo` structure

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid method argument
- **404** — Object not found
- **422** — Mask not applied

Return:

Always returns `true`.

v1.netmanager.virtualrouter.delete

```
Boolean v1.netmanager.virtualrouter.delete(String auth_token, String id)
```

Delete virtual router, router with id `"default"` cannot be deleted.

Parameters:

- **auth_token** — authentication token
- **id** — router identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found
- **502** — Object cannot be deleted because it's being used by another object
- **1204** — Cannot delete default virtual router

Return:

Always returns `true`.

v1.netmanager.netflow.profiles.list

```
PagingList<NetflowProfileInfo> v1.netmanager.netflow.profiles.list(String auth_token,  
Integer start, Integer limit, Struct filter)
```

List netflow profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

On page from complete list of profiles.

v1.netmanager.netflow.profile.fetch

```
NetflowProfileInfo v1.netmanager.netflow.profile.fetch(String auth_token, UID profile_id)
```

Fetch netflow profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Profile properties structure.

v1.netmanager.netflow.profile.add

```
UID v1.netmanager.netflow.profile.add(String auth_token, NetflowProfileInfo profile)
```

Create a new netflow profile.

Parameters:

- **auth_token** — authentication token
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Identifier of just created object.

v1.netmanager.netflow.profile.update

```
UID v1.netmanager.netflow.profile.update(String auth_token, UID profile_id,  
NetflowProfileInfo profile)
```

Update existing netflow profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.netmanager.netflow.profile.delete

```
Boolean v1.netmanager.netflow.profile.delete(String auth_token, UID profile_id)
```

Delete netflow profile

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.netmanager.flow.monitor.stat

```
List<StatisticsMonitorInfo> v1.netmanager.flow.monitor.stat(String auth_token, Integer  
limit, Struct sorter, String query)
```

Fetch traffic flow details.

Parameters:

- **auth_token** — authentication token
- **limit** — max length of returned list
- **sorter** — structure with information about sorting: property & direction

- **query** — data query

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Returns list of structures.

v1.netmanagerlldp.profiles.list

```
PagingList<LldpProfileInfo> v1.netmanager.lldp.profiles.list(String auth_token, Integer start, Integer limit, Struct filter)
```

List lldp profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of LLDP profiles.

v1.netmanagerlldp.profile.fetch

```
LldpProfileInfo v1.netmanager.lldp.profile.fetch(String auth_token, UID profile_id)
```

Fetch lldp profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Requested profile data.

v1.netmanagerlldp.profile.add

```
UID v1.netmanager.lldp.profile.add(String auth_token, LldpProfileInfo profile)
```

Create a new lldp profile.

Parameters:

- **auth_token** — authentication token
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Identified of just created LLDP profile.

v1.netmanager.lldp.profile.update

```
Boolean v1.netmanager.lldp.profile.update(String auth_token, UID profile_id,
LldpProfileInfo profile)
```

Update existing lldp profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Always returns `true`.

v1.netmanager.lldp.profile.delete

```
Boolean v1.netmanager.lldp.profile.delete(String auth_token, UID profile_id)
```

Delete lldp profile

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Always returns `true`.

v1.netmanager.lldp.neighbors

```
String v1.netmanager.lldp.neighbors(String auth_token, String node_name)
```

```
String v1.netmanager.lldp.neighbors(String auth_token)
```

Fetch information about LLDP neighbors for specified node.

If node name is omitted then fetch from current node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Requested node not found
- **1206** — LLDP service is not running

v1.netmanager.lldp.statistics

```
String v1.netmanager.lldp.statistics(String auth_token, String node_name)
```

```
String v1.netmanager.lldp.statistics(String auth_token)
```

Fetch LLDP usage statistics for specified node.

If node name is omitted then fetch from current node.

Parameters:

- **auth_token** — authentication token
- **node_name** — cluster node name

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Requested node not found
- **1206** — LLDP service is not running

v1.netmanager.geoip.lookup.ip

```
Struct v1.netmanager.geoip.lookup.ip(String auth_token, String address)
```

Lookup IP address location.

Parameters:

- **auth_token** — authentication token
- **address** — IP address, only IPv4 addresses are supported now

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **110** — Incorrect format of IP address
- **404** — IP not found in the database

Return:

Method returns structure that contains the following fields:

Field	Type	Description
country_code	String	two letter country code as defined in ISO 3166-1 alpha-2

Component: Named lists

Module methods

v2.nlists.list.....	194	v2.nlists.list.add.items	198
v2.nlists.add	194	v2.nlists.list.update	198
v2.nlists.lists.fetch.....	195	v2.nlists.list.delete	199
v2.nlists.fetch.....	195	v2.nlists.list.delete.items	199
v2.nlists.update	196	v2.nlists.list.check.updates.....	200
v2.nlists.delete.....	196	v2.nlists.list.export.....	200
v2.nlists.list.list	196	v2.nlists.list.import.....	201
v2.nlists.list.add.....	197	v2.nlists.list.clear.....	201

v2.nlists.list

```
PagingList<NamedListInfo> v2.nlists.list(String auth_token, NamedListType type, Integer start, Integer limit, Struct filter)
```

Fetch list of named lists of given type.

Parameters:

- **auth_token** — authentication token
- **type** — defines what type of lists we need to fetch
- **start** — start item index for the page
- **limit** — how much items return
- **filter** — structure with filtering options, see description below

Filtering options structure may contain the following fields (all fields are optional):

Field	Type	Description
scope	String	Search scope, allowed values are: <code>list</code> (default) search nlist items only; <code>contents</code> search through list contents. Default value is <code>"list"</code> .
search	String	Search/filter string, if omitted or empty, then all other options are ignored and function will return complete list of named lists of given type.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters
- **2007** — Unsupported list type

Return:

List of named lists

v2.nlists.add

```
UID v2.nlists.add(String auth_token, NamedListInfo list)
```

Create new local named list.

Parameters:

- **auth_token** — authentication token
- **list** — list parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with same data already exists
- **601** — Missing required field

Return:

identifier of just created named list

v2.nlists.lists.fetch

```
List<NamedListInfo> v2.nlists.lists.fetch(String auth_token, List<UID> list_ids)
```

Fetch multiple named lists at once.

Parameters:

- **auth_token** — authentication token
- **list_ids** — list of nlists identifiers, if any identifier doesn't exist, method throws error 404

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

List of found named lists

v2.nlists.fetch

```
NamedListInfo v2.nlists.fetch(String auth_token, UID id)
```

Fetch single named list properties.

Parameters:

- **auth_token** — authentication token
- **id** — list identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

list properties as [NamedListInfo](#) struct

v2.nlists.update

```
Boolean v2.nlists.update(String auth_token, UID id, NamedListInfo list)
```

Update existing named list. Only locally created lists can be edited.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of list that should be updated
- **list** — list details that should be altered, only provided fields will be modified

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Object not found
- **409** — Object with same data already exists

Return:

true

v2.nlists.delete

```
Boolean v2.nlists.delete(String auth_token, UID id)
```

Delete named list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of list that should be deleted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found
- **502** — Cannot delete the list because it's in use
- **2002** — List is readonly and cannot be modified

Return:

true

v2.nlists.list.list

```
PagingList<NamedListItemInfo> v2.nlists.list.list(String auth_token, UID id, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Get a portion (page) of named list items.

To filter results you may pass structure with the filtering options in argument `filter`. Its format depends on list type.

```
"timerrestrictiongroup"
```

Filtering not defined.

```
"url", "network", "httpcwl", "useragent", "emailgroup", "phonegroup", "urlcategorygroup",  
"websocket_protocol", "websocket_extension"
```

Allowed filtering options are:

search	String	string value matched against list item value
--------	--------	--

Parameters:

- **auth_token** — authentication token
- **id** — list identifier
- **start** — start item index for the page
- **limit** — how much items return
- **filter** — structure with filtering options
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found
- **603** — Incorrect paging parameters
- **2010** — List content is not available

Return:

List of items on single page.

v2.nlists.list.add

```
UID v2.nlists.list.add(String auth_token, UID id, NamedListItemInfo item)
```

Add a new item to the list. Only locally created lists can be modified.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **id** — list identifier where new item should be added
- **item** — new item properties in the form of structure [NamedListItemInfo](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **111** — Invalid field value
- **404** — Object not found
- **2001** — Item duplicate
- **2003** — List is readonly and cannot be modified
- **2011** — Found cyclic dependence
- **2013** — Invalid time or date range

Return:

identifier of just created list item

v2.nlists.list.add.items

```
Integer v2.nlists.list.add.items(String auth_token, UID id, List<NamedListItemInfo> items)
```

Add multiple items to the list in one pass.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **id** — list identifier where new item should be added
- **items** — list of structures of type [NamedListItemInfo](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **404** — Object not found
- **2001** — Item duplicate (no objects was added)
- **2003** — List is readonly and cannot be modified
- **2011** — Found cyclic dependence
- **2013** — Invalid time or date range
- **2014** — The named list contains duplicate items (new non-duplicate objects have been added)

Return:

number of successfully added items

v2.nlists.list.update

```
Boolean v2.nlists.list.update(String auth_token, UID list_id, UID item_id,
NamedListItemInfo item)
```

Update existing element in the list.

Important! When updating items, you need to pass all available fields to the [NamedListItemInfo](#) structure.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token

- **list_id** — list identifier where item should be updated
- **item_id** — list item identifier that should be updated
- **item** — item properties in the form of structure [NamedListItemInfo](#) that should be updated

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Object not found
- **2001** — Item duplicate
- **2003** — List is readonly and cannot be modified
- **2011** — Found cyclic dependence
- **2012** — You can't update field name of list
- **2013** — Invalid time or date range

Return:

true

v2.nlists.list.delete

```
Boolean v2.nlists.list.delete(String auth_token, UID list_id, UID item_id)
```

Delete element from the named list.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **list_id** — list identifier
- **item_id** — list item identifier that should be deleted

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found
- **2003** — List is readonly and cannot be modified

Return:

true

v2.nlists.list.delete.items

```
Integer v2.nlists.list.delete.items(String auth_token, UID list_id, List<UID> item_ids)
```

Delete multiple items from the list at once.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **id** — list identifier
- **item_ids** — list of item identifiers

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found
- **2003** — List is readonly and cannot be modified

Return:

number of successfully deleted items

v2.nlists.list.check.updates

```
Boolean v2.nlists.list.check.updates(String auth_token, UID list_id)
```

Initiates list update.

Parameters:

- **auth_token** — authentication token
- **list_id** — list identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

true

v2.nlists.list.export

```
String v2.nlists.list.export(String auth_token, UID list_id, Struct filter)
```

Export named list contents.

Parameters:

- **auth_token** — authentication token
- **list_id** — list identifier
- **filter** — filtering options, at this moment only String field **search** is supported, if this field is omitted or empty all items are exported

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found
- **2009** — Cannot export empty list

Return:

UID of file in the storage, use method [v3.storage.file.genExternalUrl](#) to generate download URL.

v2.nlists.list.import

```
Boolean v2.nlists.list.import(String auth_token, UID list_id, String import_type, String storage_file_uid)
```

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **list_id** — list identifier
- **import_type** — type of import, allowed values are: replace, append
- **storage_file_uid** — identifier of file from the storage that should be imported

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found
- **2003** — List is readonly and cannot be modified
- **2004** — Failed to import list contents
- **2005** — Incorrect morpholist: too complex data

Return:

Always returns `true`.

v2.nlists.list.clear

```
Boolean v2.nlists.list.clear(String auth_token, UID list_id, Struct filter)
```

Delete list items that match given filter.

If list has field **url** set to non-empty value this method will throw error 2003.

Parameters:

- **auth_token** — authentication token
- **list_id** — list identifier
- **filter** — filtering options, at this moment only String field **search** is supported, if this field is omitted or empty all items are removed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **2003** — List is readonly and cannot be modified

Return:

Always returns `true`.

Component: Notification

Module methods

v1.notification.profiles.list	202	v1.notification.alert.rules.list	205
v1.notification.profile.fetch	202	v1.notification.alert.rule.fetch	205
v1.notification.profile.add	203	v1.notification.alert.rule.add	205
v1.notification.profile.update	203	v1.notification.alert.rule.update	206
v1.notification.profile.delete	203	v1.notification.alert.rule.delete	206
v1.notification.profile.smpp.test	204	v1.notification.alert.rule.events	207
v1.notification.profile.smtp.test	204		

v1.notification.profiles.list

```
List<NotificationProfileInfo> v1.notification.profiles.list(String auth_token)
```

Fetch list of notification profiles.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of smpp profiles

v1.notification.profile.fetch

```
NotificationProfileInfo v1.notification.profile.fetch(String auth_token, UID profile_id)
```

Fetch SMPP profile with given id.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Profile not found

Return:

notification profile

v1.notification.profile.add

```
UID v1.notification.profile.add(String auth_token, NotificationProfileInfo profile)
```

Add new SMPP profile.

Parameters:

- **auth_token** — authentication token
- **profile** — notification profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with same data already exists
- **601** — Missing required field

Return:

identifier of just created profile

v1.notification.profile.update

```
Boolean v1.notification.profile.update(String auth_token, UID profile_id, NotificationProfileInfo profile)
```

Update notification profile with given id.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier
- **profile** — notification profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Profile not found
- **409** — Object with same data already exists

Return:

true

v1.notification.profile.delete

```
Boolean v1.notification.profile.delete(String auth_token, UID profile_id)
```

Delete notification profile with given id.

Parameters:

- **auth_token** — authentication token

- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Profile not found
- **502** — Object deletion failed due to constraints

Return:

true

v1.notification.profile.smpp.test

```
Boolean v1.notification.profile.smpp.test(String auth_token, NotificationProfileInfo profile)
```

Test given notification profile.

Parameters:

- **auth_token** — authentication token
- **profile** — notification profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **14001** — error while test smpp profile

Return:

true

v1.notification.profile.smtp.test

```
Boolean v1.notification.profile.smtp.test(String auth_token, NotificationProfileInfo profile, String sender, String recipient, String subject, String body)
```

Test given notification profile.

Parameters:

- **auth_token** — authentication token
- **profile** — notification profile
- **sender** — Email address for the sender for the email.
- **recipient** — Email address for the recipient of the email.
- **subject** — Subject for the email message.
- **body** — Message body for the email.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **14004** — SMTP notification profile test return permanent failure
- **14005** — SMTP notification profile test return network failure

Return:

true

v1.notification.alert.rules.list

```
PagingList<AlertRuleInfo> v1.notification.alert.rules.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch list of alert rules.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of alert rules

v1.notification.alert.rule.fetch

```
AlertRuleInfo v1.notification.alert.rule.fetch(String auth_token, UID id)
```

Fetch alert rule with given id.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Alert rule with given id not found

Return:

alert rule

v1.notification.alert.rule.add

```
UID v1.notification.alert.rule.add(String auth_token, AlertRuleInfo info)
```

Add new alert rule.

Parameters:

- **auth_token** — authentication token
- **info** — alert rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with same data already exists
- **404** — Profile not found
- **513** — Invalid events
- **601** — Missing required field

Return:

identifier of just created rule

v1.notification.alert.rule.update

```
Boolean v1.notification.alert.rule.update(String auth_token, UID id, AlertRuleInfo info)
```

Update alert rule with given id.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier
- **info** — alert rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Profile not found
- **409** — Object with same data already exists
- **513** — Invalid events

Return:

true

v1.notification.alert.rule.delete

```
Boolean v1.notification.alert.rule.delete(String auth_token, UID id)
```

Delete alert rule with given id.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Alert rule with given id not found

Return:

true

v1.notification.alert.rule.events

```
List<String> v1.notification.alert.rule.events(String auth_token)
```

Get alert rule events.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of available events for which WAF send notification

Component: PCAP

Module methods

v1.pcap.file.list.....	208	v1.pcap.rule.list	211
v1.pcap.file.download	208	v1.pcap.rule.fetch.....	211
v1.pcap.file.delete	209	v1.pcap.rule.add	212
v1.pcap.file.flush	209	v1.pcap.rule.update	212
v1.pcap.filter.list	209	v1.pcap.rule.delete	213
v1.pcap.filter.fetch	210	v1.pcap.start	213
v1.pcap.filter.add	210	v1.pcap.stop	213
v1.pcap.filter.update	210	v1.pcap.status.....	214
v1.pcap.filter.delete	211		

v1.pcap.file.list

```
List<PCAPFile> v1.pcap.file.list(String auth_token)
```

Fetch list of files

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of pcap files

v1.pcap.file.download

```
PCAPFile v1.pcap.file.download(String auth_token, UID id)
```

Fetch information about file for downloading

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of file

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Can't find pcap file
- **17002** — Can't download file

Return:

information about pcap file

v1.pcap.file.delete

```
Boolean v1.pcap.file.delete(String auth_token, UID id)
```

Delete given pcap file

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of file

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Can't find pcap file

Return:

true

v1.pcap.file.flush

```
Boolean v1.pcap.file.flush(String auth_token)
```

Delete all pcap files

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v1.pcap.filter.list

```
List<PCAPFilter> v1.pcap.filter.list(String auth_token)
```

Fetch list of packet capturing filters

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of pcap filters

v1.pcap.filter.fetch

```
PCAPFilter v1.pcap.filter.fetch(String auth_token, UID id)
```

Fetch given pcap filter

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of filter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap filter

Return:

pcap filter

v1.pcap.filter.add

```
UID v1.pcap.filter.add(String auth_token, PCAPFilter filter)
```

Add new pcap filter

Parameters:

- **auth_token** — authentication token
- **filter** — information about pcap filter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **601** — Missing required field

Return:

unique identifier of filter

v1.pcap.filter.update

```
Boolean v1.pcap.filter.update(String auth_token, UID id, PCAPFilter filter)
```

Update given pcap filter

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of filter
- **filter** — information about pcap filter

Errors:

- **102** — Access denied

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Invalid field value
- **404** — Can't find pcap filter

Return:

true

v1.pcap.filter.delete

```
Boolean v1.pcap.filter.delete(String auth_token, UID id)
```

Delete given pcap filter

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of filter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap filter

Return:

true

v1.pcap.rule.list

```
List<PCAPRule> v1.pcap.rule.list(String auth_token)
```

Fetch list of packet capturing rules

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of pcap rules

v1.pcap.rule.fetch

```
PCAPRule v1.pcap.rule.fetch(String auth_token, UID id)
```

Fetch given pcap rule

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap rule

Return:

pcap rule

v1.pcap.rule.add

```
UID v1.pcap.rule.add(String auth_token, PCAPRule rule)
```

Add new pcap rule

Parameters:

- **auth_token** — authentication token
- **rule** — information about pcap rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **408** — Filter per rule limit reached
- **601** — Missing required field

Return:

unique identifier of rule

v1.pcap.rule.update

```
Boolean v1.pcap.rule.update(String auth_token, UID id, PCAPRule rule)
```

Update given pcap rule

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule
- **rule** — information about pcap rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Invalid field value
- **404** — Can't find pcap rule

- **408** — Filter per rule limit reached

Return:

true

v1.pcap.rule.delete

```
Boolean v1.pcap.rule.delete(String auth_token, UID id)
```

Delete given pcap rule

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap rule

Return:

true

v1.pcap.start

```
Boolean v1.pcap.start(String auth_token, UID id)
```

Start pcap rule.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap rule

Return:

true

v1.pcap.stop

```
Boolean v1.pcap.stop(String auth_token, UID id)
```

Stop pcap rule.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap rule

Return:

true

v1.pcap.status

```
String v1.pcap.status(String auth_token, UID id)
```

Fetch rule status.

Parameters:

- **auth_token** — authentication token
- **id** — unique identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Can't find pcap rule

Return:

One of the following strings: "idle", "capturing"

Component: Public

Component: Reverse proxy

Module methods

v1.reverseproxy.profiles.list	216	v1.reverseproxy.rule.move.top	220
v1.reverseproxy.profile.add	216	v1.reverseproxy.rule.move.bottom	220
v1.reverseproxy.profile.fetch	217	v1.reverseproxy.rule.move.up	221
v1.reverseproxy.profile.update	217	v1.reverseproxy.rule.move.down	221
v1.reverseproxy.profile.delete	218	v1.reverseproxy.rule.move.position	222
v1.reverseproxy.rules.list	218	v1.reverseproxy.loadbalancing.rules.list	222
v1.reverseproxy.rule.add	218	v1.reverseproxy.loadbalancing.rule.add	222
v1.reverseproxy.rule.fetch	219	v1.reverseproxy.loadbalancing.rule.fetch	223
v1.reverseproxy.rule.update	219	v1.reverseproxy.loadbalancing.rule.update ..	223
v1.reverseproxy.rule.delete	220	v1.reverseproxy.loadbalancing.rule.delete	224

v1.reverseproxy.profiles.list

```
PagingList<ReverseProxyProfileInfo> v1.reverseproxy.profiles.list(String auth_token,  
Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Get one page from list of reverse proxy profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of all reverse proxy profiles

v1.reverseproxy.profile.add

```
UID v1.reverseproxy.profile.add(String auth_token, ReverseProxyProfileInfo profile)
```

Create a new reverse proxy profile

Parameters:

- **auth_token** — authentication token
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value

Return:

ID of just created profile

v1.reverseproxy.profile.fetch

```
ReverseProxyProfileInfo v1.reverseproxy.profile.fetch(String auth_token, UID profile_id)
```

Fetch reverse proxy profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

profile info

v1.reverseproxy.profile.update

```
Boolean v1.reverseproxy.profile.update(String auth_token, UID profile_id,  
ReverseProxyProfileInfo profile)
```

Update reverse proxy profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier
- **profile** — profile details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Incorrect field data
- **404** — Object not found
- **409** — Duplicate name value

Return:

true

v1.reverseproxy.profile.delete

```
Boolean v1.reverseproxy.profile.delete(String auth_token, UID profile_id)
```

Delete reverse proxy profile.

Parameters:

- **auth_token** — authentication token
- **profile_id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rules.list

```
PagingList<ReverseProxyRuleInfo> v1.reverseproxy.rules.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch list of reverse proxy rules.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters
- **618** — Incorrect field query value in search filter

Return:

list of all reverse proxy rules

v1.reverseproxy.rule.add

```
UID v1.reverseproxy.rule.add(String auth_token, ReverseProxyRuleInfo rule)
```

Create a new reverse proxy rule

Parameters:

- **auth_token** — authentication token
- **rule** — rule details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **404** — Object not found
- **409** — Duplicate name value
- **601** — Missing required field
- **615** — Maximum allowed GEOIP records limit is reached

Return:

ID of just created rule

v1.reverseproxy.rule.fetch

```
ReverseProxyRuleInfo v1.reverseproxy.rule.fetch(String auth_token, UID rule_id)
```

Fetch reverse proxy rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

rule info

v1.reverseproxy.rule.update

```
Boolean v1.reverseproxy.rule.update(String auth_token, UID rule_id, ReverseProxyRuleInfo rule)
```

Update reverse proxy rule. Field **position** is ignored, to move rule you must use one of corresponding move-methods.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **rule** — rule details

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Incorrect field data
- **404** — Object not found

- **409** — Duplicate name value
- **601** — Missing required field
- **615** — Maximum allowed GEOIP records limit is reached

Return:

true

v1.reverseproxy.rule.delete

```
Boolean v1.reverseproxy.rule.delete(String auth_token, UID rule_id)
```

Delete reverse proxy rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rule.move.top

```
Boolean v1.reverseproxy.rule.move.top(String auth_token, UID id)
```

Move rule to the top of rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rule.move.bottom

```
Boolean v1.reverseproxy.rule.move.bottom(String auth_token, UID id)
```

Move rule to the bottom of rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rule.move.up

Boolean **v1.reverseproxy.rule.move.up**(String auth_token, UID id)

Move rule up one position in rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rule.move.down

Boolean **v1.reverseproxy.rule.move.down**(String auth_token, UID id)

Move rule down one position in rules list.

Parameters:

- **auth_token** — authentication token
- **id** — identifier of rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

v1.reverseproxy.rule.move.position

```
Boolean v1.reverseproxy.rule.move.position(String auth_token, UID rule_id, Variant position)
```

Move rule to given position in the list. will be moved to the top, if position is more than last list element position, the rule will be moved to the end of list

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **position** — new rule position, integer value starting with `1`, or string value `"top"` to move to the top of the list, or `"last"` to move to the bottom of the list

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found
- **356** — Incorrect value in "position" argument

Return:

true

v1.reverseproxy.loadbalancing.rules.list

```
List<ReverseProxyLoadBalancingRuleInfo> v1.reverseproxy.loadbalancing.rules.list(String auth_token)
```

Fetch list of load balancing rules.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **618** — Incorrect field query value in search filter

Return:

list of all reverse proxy load balancing rules

v1.reverseproxy.loadbalancing.rule.add

```
UID v1.reverseproxy.loadbalancing.rule.add(String auth_token, ReverseProxyLoadBalancingRuleInfo rule)
```

Create a new load balancing rule.

Parameters:

- **auth_token** — authentication token

- **rule** — rule info

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field
- **21005** — Reverse proxy balancing rule with the same proxy servers already exists

Return:

identifier of just created rule

v1.reverseproxy.loadbalancing.rule.fetch

```
ReverseProxyLoadBalancingRuleInfo v1.reverseproxy.loadbalancing.rule.fetch(String auth_token, UID id)
```

Fetch a single load balancing rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

rule info

v1.reverseproxy.loadbalancing.rule.update

```
Boolean v1.reverseproxy.loadbalancing.rule.update(String auth_token, UID id, ReverseProxyLoadBalancingRuleInfo rule)
```

Update load balancing rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier
- **rule** — rule info

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Incorrect field data
- **404** — Object not found

- **409** — Duplicate name value
- **21005** — Reverse proxy balancing rule with the same proxy servers already exists

Return:

true

v1.reverseproxy.loadbalancing.rule.delete

```
Boolean v1.reverseproxy.loadbalancing.rule.delete(String auth_token, UID id)
```

Delete a single load balancing rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

true

Component: Settings

Module methods

v2.settings.get.param.....	225	v2.settings.certificate.getData	233
v2.settings.set.param	226	v2.settings.certificate.getCertWithChainData	234
v2.settings.get.params	227	v2.settings.certificate.getCsr.....	234
v2.settings.custom.dnses.list	227	v2.settings.certificate.details	235
v2.settings.custom.dns.add	228	v2.settings.webui.auth.mode.get	235
v2.settings.custom.dns.update	228	v2.settings.webui.auth.mode.set	235
v2.settings.custom.dns.delete	228	v2.settings.time.get	236
v2.settings.current.account.get.string	229	v2.settings.time.set.....	236
v2.settings.current.account.set.string	229	v2.settings.remote.admin.info	236
v2.settings.generate.certificate	229	v2.settings.remote.admin.enabled	237
v2.settings.generate.certificateRequest	230	v2.settings.stat.server.config.get.....	237
v2.settings.certificate.copy	230	v2.settings.stat.server.config.set	237
v2.settings.certificate.add	231	v2.settings.upstream.update.proxy.config.get	238
v2.settings.certificates.list	231	v2.settings.upstream.update.proxy.config.set	238
v2.settings.certificate.fetch	232		
v2.settings.certificate.update	232		
v2.settings.certificate.delete	233		
v2.settings.certificate.delete.from.storage ...	233		

v2.settings.get.param

Variant **v2.settings.get.param**(String auth_token, String Name)

Fetch one parameter value, see list of available parameters below.

Parameters:

- **auth_token** — authentication token
- **name** — name of the parameter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **107** — Parameter not found

Return:

parameter value, type of returned data depends on specific parameter

List of available parameters:

Name	Type	Description
"recursive_enabled"	Boolean	Specifies if recursive DNS queries are enabled.
"dns_max_ttl"	Integer	Max TTL for DNS records in seconds.
"dns_max_queries_per_user"	Integer	Limit DNS requests per second for user.

"api_session_lifetime"	Integer	Admin session idle timeout in seconds, allowed values are greater (or equal) than 5
"ui_language"	String	Default interface language, allowed values are: "ru" , "en"
"ui_timezone"	String	Server time zone, allowed, contains time zone name from IANA zones list .
"enable_dns_filtering"	Boolean	Specifies if DNS Filtering feature is enabled.
"only_a_for_unknown"	Boolean	Specifies if only A and AAAA DNS-records are allowed for unknown users (prohibit VPN over DNS).
"saml_server_port"	Integer	SAML server port
"updates_channel"	String	Updates channel, allowed values are: "stable" , "beta" .
"proxy_host_rfc"	String	Allow HTTP proxy 1.1 without host, allowed values are: "strict" , "relaxed"
"web_console_ssl_profile_id"	UID	Identifier of SSL Profile to use for Web console.
"add_via_enabled"	Boolean	Set to true to add header "Via" to proxy requests
"add_forwarded_enabled"	Boolean	Set to true to add header "Forwarded" to proxy requests
"add_xforwarded_enabled"	Boolean	Set to true to add header "X-Forwarded-For" to proxy requests
"smode_enabled"	Boolean	Set to true to enable SYN-proxy.
"legacy_ssl_enabled"	Boolean	Specifies if legacy SSL modes are enabled.
"module_sip_enabled"	Boolean	Enable/disable SIP module
"module_h323_enabled"	Boolean	Enable/disable h323 module
"module_sunrpc_enabled"	Boolean	Enable/disable sunrpc module
"module_ftp_alg_enabled"	Boolean	Enable/disable FTP-ALG module
"module_tftp_enabled"	Boolean	Enable/disable TFTP module
"http_connection_timeout"	Integer	Proxy connection timeout, in seconds.
"http_loading_timeout"	Integer	Proxy content loading timeout, in seconds.
"dns_receive_timeout"	Integer	DNS timeout, in milliseconds.
"dns_max_attempts"	Integer	Maximum DNS retries.
"lldp_config"	LldpConfig	LLDP configuration
"event_tracker"	EventTrackerInfo	Contains configuration of event tracker, value must be a structure of type EventTrackerInfo .
"utm_update_schedule"	UpdateSchedule	Structure that contains update schedule

v2.settings.set.param

Boolean **v2.settings.set.param**(String auth_token, String Name, Variant value)

Set parameter value, see list of available parameters in description of method [v2.settings.get.param](#).

Parameters:

- **auth_token** — authentication token
- **name** — name of the parameter
- **value** — value of the parameter

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **107** — Parameter not found
- **110** — Invalid parameter value
- **601** — Missing required field(s)

Return:

true

v2.settings.get.params

```
Struct v2.settings.get.params(String auth_token, List<String> keys)
```

Fetch a few parameters in single request.

Parameters:

- **auth_token** — authentication token
- **keys** — list of parameter names you want to fetch, see [v2.settings.get.param](#) for list of available parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **107** — Parameter not found

Return:

structure, key is one of keys, if some setting value couldn't be found corresponding key will be omitted from the response

v2.settings.custom.dnses.list

```
List<CustomDnsInfo> v2.settings.custom.dnses.list(String auth_token)
```

Get custom DNS servers list.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

list of custom DNS servers

v2.settings.custom.dns.add

```
UID v2.settings.custom.dns.add(String auth_token, CustomDnsInfo dns)
```

Create new custom DNS server record.

Parameters:

- **auth_token** — authentication token
- **dns** — DNS server structure

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **601** — Missing required field(s)
- **18004** — Duplicate dns

Return:

identifier of just created custom DNS server record

v2.settings.custom.dns.update

```
Boolean v2.settings.custom.dns.update(String auth_token, UID id, CustomDnsInfo dns)
```

Updated existing custom DNS server record.

Parameters:

- **auth_token** — authentication token
- **id** — DNS server record identifier
- **dns** — DNS server structure

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid parameter value
- **404** — Object not found
- **601** — Missing required field(s)
- **18004** — Duplicate dns

Return:

identifier of just created custom DNS server record

v2.settings.custom.dns.delete

```
Boolean v2.settings.custom.dns.delete(String auth_token, UID id)
```

Delete existing custom DNS server record.

Parameters:

- **auth_token** — authentication token
- **id** — DNS server record identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

identifier of just created custom DNS server record

v2.settings.current.account.get.string

```
String v2.settings.current.account.get.string(String auth_token, String name)
```

Get settings parameter value from currently logged in account settings storage space.

This method and [v2.settings.current.account.set.string](#) are recommended way to store any parameter values that should be bound to current logged in administrator.

Parameters:

- **auth_token** — authentication token
- **name** — settings value name

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **107** — Parameter not found

Return:

String value.

v2.settings.current.account.set.string

```
Boolean v2.settings.current.account.set.string(String auth_token, String name, String value)
```

Set settings parameter value visible to currently logged in administrator only.

Parameters:

- **auth_token** — authentication token
- **name** — settings value name, any name could be used
- **value** — settings value

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Invalid parameter value

Return:

Always returns `true`.

v2.settings.generate.certificate

```
UID v2.settings.generate.certificate(String auth_token, CertificateCreateInfo cert)
```

Generate a new self signed CA certificate.

Parameters:

- **auth_token** — authentication token
- **cert** — CA certificate parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field(s)
- **3001** — Failed to generate CA certificate

Return:

Identifier of just generated certificate.

v2.settings.generate.certificateRequest

```
UID v2.settings.generate.certificateRequest(String auth_token, CertificateCSRCreateInfo cert)
```

Generate new certificate record with private key and CSR only, but without actual certificate.

Parameters:

- **auth_token** — authentication token
- **cert** — CSR parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field(s)
- **3012** — Failed to generate CSR

Return:

Identifier of just generated certificate.

v2.settings.certificate.copy

```
UID v2.settings.certificate.copy(String auth_token, UID id, CertificateStorageInfo cert)
```

Copy (clone) certificate, all fields are copied except of **role** which is set to `"none"`.

Parameters:

- **auth_token** — authentication token
- **id** — certificate identifier
- **cert** — new certificate data, only **name** and **description** fields are accepted, all others are ignored.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Certificate not found
- **409** — Duplicate value, conflict with the existing object
- **3008** — Incorrect certificate file format
- **3009** — Incorrect key file format
- **3017** — Incorrect certificate chain data

Return:

Identifier of just copied certificate.

v2.settings.certificate.add

```
UID v2.settings.certificate.add(String auth_token, CertificateStorageInfo cert)
```

Add new certificate to the certificates storage.

Parameters:

- **auth_token** — authentication token
- **cert** — structure that describes a new certificate

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some field value in passed structure is invalid
- **409** — Duplicate value, conflict with the existing object
- **601** — Missing required field(s)
- **3007** — Certificate and private key don't match
- **3008** — Incorrect certificate file format
- **3009** — Incorrect key file format
- **3017** — Incorrect certificate chain data
- **3019** — Data decryption failed
- **3023** — Incorrect certificate role for this certificate

Return:

identifier of just added certificate

v2.settings.certificates.list

```
PagingList<CertificateStorageInfo> v2.settings.certificates.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch list of all certificates from the storage.

Parameters:

- **auth_token** — authentication token
- **start** — — offset in complete list

- **limit** — — max length of returned list
- **filter** — — structure with filtering options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of certificates.

v2.settings.certificate.fetch

```
CertificateStorageInfo v2.settings.certificate.fetch(String auth_token, UID cert_id)
```

Fetch certificate metadata.

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found

Return:

Certificate storage info

v2.settings.certificate.update

```
Boolean v2.settings.certificate.update(String auth_token, UID cert_id, CertificateStorageInfo cert)
```

Update certificate metadata. This method updates only "display" data, not certificate itself.

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier
- **cert** — certificate data, only passed values will be modified. If the value `"user"` is specified for the property **role**, the property **user_guid** must be specified.

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — certificate not found
- **409** — Duplicate value, conflict with the existing object
- **3008** — Incorrect certificate file format

- **3009** — Incorrect key file format
- **3017** — Incorrect certificate chain data
- **3023** — Incorrect certificate role for this certificate

Return:

true

v2.settings.certificate.delete

```
Boolean v2.settings.certificate.delete(String auth_token, UID cert_id)
```

Delete certificate from the storage.

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found
- **502** — certificate in use

Return:

true

v2.settings.certificate.delete.from.storage

```
Boolean v2.settings.certificate.delete.from.storage(String auth_token, String domain_name)
```

Delete cached HTTPS certificate from the certificate storage.

Parameters:

- **auth_token** — authentication token
- **domain_name** — certificate domain used in the CN (Common Name) or SAN (Subject Alternative Name)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found

Return:

true

v2.settings.certificate.getData

```
Base64 v2.settings.certificate.getData(String auth_token, UID cert_id)
```

Fetch certificate itself (in DER format).

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found

Return:

certificate binary data

v2.settings.certificate.getCertWithChainData

```
Base64 v2.settings.certificate.getCertWithChainData(String auth_token, UID cert_id)
```

Fetch certificate and all related certificates from its chain in single PEM file.

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Object not found

Return:

Base64-encoded data.

v2.settings.certificate.getCsr

```
Base64 v2.settings.certificate.getCsr(String auth_token, UID cert_id)
```

Fetch CSR (in DER format).

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found

Return:

CSR binary data

v2.settings.certificate.details

```
CertificateDetailsInfo v2.settings.certificate.details(String auth_token, UID cert_id)
```

Fetch detailed certificate information.

Parameters:

- **auth_token** — authentication token
- **cert_id** — certificate identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — certificate not found

Return:

structured certificate information

v2.settings.webui.auth.mode.get

```
WebuiAuthMode v2.settings.webui.auth.mode.get(String auth_token)
```

Get webui authentication mode.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

authentication mode, see [WebuiAuthMode](#)

v2.settings.webui.auth.mode.set

```
Boolean v2.settings.webui.auth.mode.set(String auth_token, WebuiAuthMode mode)
```

Change webui authentication mode. Webui server will be restarted after method call.

Parameters:

- **auth_token** — authentication token
- **mode** — new authentication mode

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3011** — Unsupported auth mode

Return:

true

v2.settings.time.get

```
ServerTimeInfo v2.settings.time.get(String auth_token)
```

Fetch server time settings.

Parameters:

- **auth_token** — authentication token

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

server time settings

v2.settings.time.set

```
Boolean v2.settings.time.set(String auth_token, ServerTimeInfo time)
```

Update server time settings.

Parameters:

- **auth_token** — authentication token
- **time** — time settings, only passed values are applied

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3005** — Incorrect time value

Return:

v2.settings.remote.admin.info

```
RemoteAdminInfo v2.settings.remote.admin.info(String auth_token)
```

Fetch remote admin configuration.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Remote admin configuration.

v2.settings.remote.admin.enabled

```
RemoteAdminInfo v2.settings.remote.admin.enabled(String auth_token, Boolean enabled)
```

Turn remote administration on or off.

Parameters:

- **auth_token** — authentication token
- **enabled** — `true` to enable, `false` to disable

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **613** — Connect failed to remote admin server

Return:

remote administration parameters in the form of structure [RemoteAdminInfo](#)

v2.settings.stat.server.config.get

```
StatServerConfig v2.settings.stat.server.config.get(String auth_token)
```

Fetch stat server configuration options.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

server configuration

v2.settings.stat.server.config.set

```
Boolean v2.settings.stat.server.config.set(String auth_token, Variant config)
```

Resets statistics server configuration to default value.

Parameters:

- **auth_token** — authentication token
- **config** — this parameter is ignored

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v2.settings.upstream.update.proxy.config.get

```
UpstreamProxyConfigInfo v2.settings.upstream.update.proxy.config.get(String auth_token)
```

Fetch upstream proxy for update and license activation configuration options.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Structure [UpstreamProxyConfigInfo](#).

v2.settings.upstream.update.proxy.config.set

```
Boolean v2.settings.upstream.update.proxy.config.set(String auth_token,  
UpstreamProxyConfigInfo config)
```

Configure upstream proxy for update and license activation.

Parameters:

- **auth_token** — authentication token
- **config** — upstream proxy configuration options

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

Component: SNMP

Module methods

v1.snmp.rules.list	239	v1.snmp.sys.name.set	243
v1.snmp.rule.fetch	239	v1.snmp.sys.location.get	243
v1.snmp.rule.add	240	v1.snmp.sys.location.set	243
v1.snmp.rule.update	240	v1.snmp.sys.description.get	244
v1.snmp.rule.delete	241	v1.snmp.sys.description.set	244
v1.snmp.rule.events	241	v1.snmp.security.profiles.list	244
v1.snmp.mibs.fetch	241	v1.snmp.security.profile.fetch	245
v1.snmp.engine.id.get	242	v1.snmp.security.profile.add	245
v1.snmp.engine.id.set	242	v1.snmp.security.profile.update	246
v1.snmp.sys.name.get	242	v1.snmp.security.profile.delete	246

v1.snmp.rules.list

```
PagingList<SnmpRuleInfo> v1.snmp.rules.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch one page from list of all SNMP rules.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, filtering by enabled only

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of SNMP rules

v1.snmp.rule.fetch

```
SnmpRuleInfo v1.snmp.rule.fetch(String auth_token, UID rule_id)
```

Fetch single SNMP rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **110** — Incorrect method argument
- **404** — Rule not found

Return:

v1.snmp.rule.add

```
UID v1.snmp.rule.add(String auth_token, SnmpRuleInfo rule)
```

Create a new SNMP rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Invalid field value
- **409** — Object with the same name already exists.
- **409** — Snmp rule with the same community already exists.
- **513** — Invalid events.
- **601** — Missing required field

Return:

identifier of just created rule

v1.snmp.rule.update

```
Boolean v1.snmp.rule.update(String auth_token, UID rule_id, SnmpRuleInfo rule)
```

Update existing SNMP rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **rule** — rule parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **111** — Invalid field value
- **404** — Rule not found
- **409** — Object with the same name already exists.
- **409** — Snmp rule with the same community already exists.
- **513** — Invalid events.

Return:

Always returns `true`.

v1.snmp.rule.delete

```
Boolean v1.snmp.rule.delete(String auth_token, UID rule_id)
```

Delete SNMP rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Rule not found

Return:

Always returns `true`.

v1.snmp.rule.events

```
List<String> v1.snmp.rule.events(String auth_token)
```

Get SNMP rule events.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of available SNMP events

v1.snmp.mibs.fetch

```
Base64 v1.snmp.mibs.fetch(String auth_token)
```

Fetch MIBs as Base64 data.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Base64-encoded data

v1.snmp.engine.id.get

```
SnmpEngineIdInfo v1.snmp.engine.id.get(String auth_token, String node)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Value represented by structure of type [SnmpEngineIdInfo](#)

v1.snmp.engine.id.set

```
Boolean v1.snmp.engine.id.set(String auth_token, String node, SnmpEngineIdInfo engine_id)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **engine_id** — SNMP engine id parameters

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — value_error
- **616** — Invalid node

Return:

Always returns `true`.

v1.snmp.sys.name.get

```
SnmpStringInfo v1.snmp.sys.name.get(String auth_token, String node)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Value represented by structure of type [SnmpStringInfo](#)

v1.snmp.sys.name.set

```
Boolean v1.snmp.sys.name.set(String auth_token, String node, SnmpStringInfo sys_name)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **sys_name** — encapsulated system name string

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — value_error
- **616** — Invalid node

Return:

Always returns `true`.

v1.snmp.sys.location.get

```
SnmpStringInfo v1.snmp.sys.location.get(String auth_token, String node)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Value represented by structure of type [SnmpStringInfo](#)

v1.snmp.sys.location.set

```
Boolean v1.snmp.sys.location.set(String auth_token, String node, SnmpStringInfo sys_location)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **sys_location** — encapsulated system location string

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — value_error
- **616** — Invalid node

Return:

Always returns `true`.

v1.snmp.sys.description.get

```
SnmpStringInfo v1.snmp.sys.description.get(String auth_token, String node)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **616** — Invalid node

Return:

Value represented by structure of type [SnmpStringInfo](#)

v1.snmp.sys.description.set

```
Boolean v1.snmp.sys.description.set(String auth_token, String node, SnmpStringInfo sys_description)
```

Parameters:

- **auth_token** — authentication token
- **node** — name of node
- **sys_description** — encapsulated system description string

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — value_error
- **616** — Invalid node

Return:

Always returns `true`.

v1.snmp.security.profiles.list

```
PagingList<SnmpSecurityProfileInfo> v1.snmp.security.profiles.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch one page from list of all SNMP security profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list

- **filter** — structure with filtering options, filtering by enabled only

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of SNMP security profiles.

v1.snmp.security.profile.fetch

```
SnmpSecurityProfileInfo v1.snmp.security.profile.fetch(String auth_token, UID
snmp_security_profile_id)
```

Fetch single SNMP security profile.

Parameters:

- **auth_token** — authentication token
- **snmp_security_profile_id** — SNMP security profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — SNMP security profile not found

Return:

SNMP security profiles

v1.snmp.security.profile.add

```
UID v1.snmp.security.profile.add(String auth_token, SnmpSecurityProfileInfo
snmp_security_profile)
```

Create a new SNMP security profile.

Parameters:

- **auth_token** — authentication token
- **snmp_security_profile** — SNMP security profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

identifier of just created profile

v1.snmp.security.profile.update

```
Boolean v1.snmp.security.profile.update(String auth_token, UID snmp_security_profile_id, SnmpSecurityProfileInfo snmp_security_profile)
```

Update existing SNMP security profile.

Parameters:

- **auth_token** — authentication token
- **snmp_security_profile_id** — SNMP security profile identifier
- **snmp_security_profile** — SNMP security profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — SNMP security profile not found

Return:

Always returns `true`.

v1.snmp.security.profile.delete

```
Boolean v1.snmp.security.profile.delete(String auth_token, UID snmp_security_profile_id)
```

Delete SNMP security profile.

Parameters:

- **auth_token** — authentication token
- **snmp_security_profile_id** — SNMP security profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — SNMP security profile not found

Return:

Always returns `true`.

Component: Statistics

Module methods

v1.statistics.status	247	v1.statistics.logexport.rule.test.....	248
v1.statistics.logexport.rule.add	247	v1.statistics.logexport.rule.update	249
v1.statistics.logexport.rule.delete	248	v1.statistics.logexport.rules.list	249
v1.statistics.logexport.rule.fetch.....	248	v1.statistics.logexport.rule.sendonce.....	250

v1.statistics.status

[StatServerStatus](#) **v1.statistics.status**(String auth_token)

Fetch Statistics Server status.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[StatServerStatus](#)

Logs export

v1.statistics.logexport.rule.add

UID **v1.statistics.logexport.rule.add**(String auth_token, [StatLogExportRuleInfo](#) rule)

Create a new log export rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect argument
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

identifier of just created rule

v1.statistics.logexport.rule.delete

```
Boolean v1.statistics.logexport.rule.delete(String auth_token, UID id)
```

Delete log export rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Rule not found
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

true

v1.statistics.logexport.rule.fetch

```
StatLogExportRuleInfo v1.statistics.logexport.rule.fetch(String auth_token, UID id)
```

Fetch single log export rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method argument
- **404** — Rule not found
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

rule info

v1.statistics.logexport.rule.test

```
Boolean v1.statistics.logexport.rule.test(String auth_token, StatLogExportRuleInfo rule)
```

Test log export rule.

Parameters:

- **auth_token** — authentication token
- **rule** — rule data

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect argument
- **6001** — Statistics server connection error
- **6002** — Statistics server authentication error
- **6100** — Rule test failed
- **6101** — Remote server unreachable
- **6102** — Remote server refused connection
- **6103** — Remote server authentication failed
- **6104** — Remote directory not found
- **6105** — Remote directory access denied
- **6999** — Statistics server internal error

Return:

true

v1.statistics.logexport.rule.update

```
Boolean v1.statistics.logexport.rule.update(String auth_token, UID id,  
StatLogExportRuleInfo rule)
```

Update log export rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier
- **rule** — rule data to update

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect argument
- **404** — Rule not found
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

true

v1.statistics.logexport.rules.list

```
List<StatLogExportRuleInfo> v1.statistics.logexport.rules.list(String auth_token)
```

Fetch list of all log export rules.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

list of all log export rules

v1.statistics.logexport.rule.sendonce

```
Boolean v1.statistics.logexport.rule.sendonce(String auth_token, UID id,  
StatSendOnceRuleInfo info)
```

Send export report once.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier
- **info** — information about the type and periods of one-time log sending

Errors:

- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **102** — Access denied
- **404** — Rule not found
- **111** — Incorrect argument
- **6001** Statistics server connection error
- **6002** Statistics server authentication error
- **6999** Statistics server internal error

Return:

true

Component: Storage

Module methods			
v3.storage.init.upload.session	252	v3.storage.file.fetchInfo.....	253
v3.storage.check.upload.session	253	v3.storage.file.genExternalUrl	253

In this section files storage access methods are described.

These methods should be used for uploading files and downloading large files using special URL. Here is a brief explanation of proposed file upload procedure.

Upload file to the server

1. First you must call method `v3.storage.init.upload.session` to obtain upload URL and upload session check token.
2. Then you should start upload process using URL obtained from previous step, until it's finished, you can also periodically call `v3.storage.check.upload.session` to check upload process status from different thread. You must send request using Content-type `multipart/form-data` and send file data in form field data.
3. When upload is finished, you receive just uploaded file storage identifier in the response of POST query.

Here is the example in python using module requests:

```
#!/usr/bin/env python3
import xmlrpc.client
import requests
import os

address = '192.168.84.72'
port = 4040
username = 'Admin'
password = '123'

server = xmlrpc.client.ServerProxy('http://%s:%s/rpc' % (address, port))
session_info = server.v2.core.login(username, password)
auth_token = session_info['auth_token']

upload_session = server.v3.storage.init.upload.session(auth_token)

# upload script itself
file_name = os.path.realpath(__file__)
files = {'data': open('tests', 'rb')}
req = requests.post('http://%s:%s%s' % (address, port, upload_session['upload_url']),
files=files)
uploaded_data = req.json()

try:
    # then pass uploaded_data['storage_file_uid'] to another method
    # or check that file was uploaded successfully:
    x = server.v3.storage.file.fetchInfo(auth_token, uploaded_data['storage_file_uid'])
    print(x)
except Exception as e:
    print('Upload failed, error: ', e)
    pass

server.v2.core.logout(auth_token)
```

Download file from the server

If you have identifier of file in the storage you can use method [v3.storage.file.genExternalUrl](#) to obtain relative URL of that file and download it using script or web browser.

v3.storage.init.upload.session

[StorageUploadSessionInfo](#) **v3.storage.init.upload.session**(String auth_token)

Create new upload session.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

object that contains information about upload session: upload URL, check token etc.

v3.storage.check.upload.session

```
StorageUploadSessionCheckInfo v3.storage.check.upload.session(String auth_token, String check_token)
```

Check uploading process status.

Parameters:

- **auth_token** — authentication token
- **check_token** — token created by method [v3.storage.init.upload.session](#) call

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3056** — Upload session not found

Return:

uploading process status in the form of structure [StorageUploadSessionCheckInfo](#)

v3.storage.file.fetchInfo

```
StorageFileInfo v3.storage.file.fetchInfo(String auth_token, UID storage_file_uid)
```

Get information about file in the storage.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — file identifier in the storage

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **3057** — Storage file not found

Return:

File information as a structure [StorageFileInfo](#)

v3.storage.file.genExternalUrl

```
String v3.storage.file.genExternalUrl(String auth_token, UID storage_file_uid)
```

Generate external URL for storage file, file could be downloaded using that URL directly from the server. Returned value is a path string of complete URL that must be appended to RPC server root. For example, if RPC entry point is `"http://10.0.0.1:4040/rpc"` and function returned value `"/export_api.xapi?id=0d75e176-94ac-41e6-9bbb-4bbc2b7a5438"`, then resulting URL would be: `"http://10.0.0.1:4040/export_api.xapi?id=0d75e176-94ac-41e6-9bbb-4bbc2b7a5438"`.

Parameters:

- **auth_token** — authentication token
- **storage_file_uid** — file identifier in the storage

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

File URL.

Component: System Status

Module methods

v1.sysstat.rulestat.get	255	v1.sysstat.public.last.5.minutes	259
v1.sysstat.rulestat.reset	255	v1.sysstat.public.last	260
v1.sysstat.active.ips	256	v1.sysstat.linkstat	260
v1.sysstat.active.ips.list	256	v1.sysstat.memory	260
v1.sysstat.counters	256	v1.sysstat.disk	260
v1.sysstat.public.counters	257	v1.sysstat.uptime	261
v1.sysstat.http.cache	257	v1.sysstat.uptime.system	261
v1.sysstat.last.year	257	v1.sysstat_ms.counters	261
v1.sysstat.last.month	257	v1.sysstat_ms.last.year	262
v1.sysstat.last.week	258	v1.sysstat_ms.last.month	262
v1.sysstat.last.day	258	v1.sysstat_ms.last.week	263
v1.sysstat.last.hour	258	v1.sysstat_ms.last.day	263
v1.sysstat.last.minute	259	v1.sysstat_ms.last.hour	264
v1.sysstat.public.last.hour	259	v1.sysstat_ms.last.minute	264
v1.sysstat.public.last.minute	259	v1.sysstat.flow.last.hour	265

v1.sysstat.rulestat.get

[SysStatRules](#) **v1.sysstat.rulestat.get**(String auth_token, List<String> guids)

Get rules statistics by GUIDs.

Parameters:

- **auth_token** — authentication token
- **guids** — rule GUIDs list

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some parameters value is invalid

Return:

[SysStatRules](#)

v1.sysstat.rulestat.reset

Boolean **v1.sysstat.rulestat.reset**(String auth_token, List<String> guids)

Reset rule counter for specified GUIDs.

Parameters:

- **auth_token** — authentication token
- **guids** — rule GUIDs list

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Some parameters value is invalid

Return:

true

v1.sysstat.active.ips

```
Integer v1.sysstat.active.ips(String auth_token)
```

Fetches number of active ips.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Number of active ips

v1.sysstat.active.ips.list

```
List<String> v1.sysstat.active.ips.list(String auth_token)
```

Fetches list of active ips.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of ip addresses.

v1.sysstat.counters

```
SysStatCounters v1.sysstat.counters(String auth_token)
```

Fetches various system counters.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied

- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatCounters](#)

v1.sysstat.public.counters

```
SysStatCounters v1.sysstat.public.counters()
```

Fetches various system counters.

Return:

[SysStatCounters](#)

v1.sysstat.http.cache

```
SysStatHttpCache v1.sysstat.http.cache(String auth_token)
```

Fetches HTTP Cache utilization.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatHttpCache](#)

v1.sysstat.last.year

```
SysStatIntervalStatistics v1.sysstat.last.year(String auth_token)
```

Fetches system usage statistics for last year.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#)

v1.sysstat.last.month

```
SysStatIntervalStatistics v1.sysstat.last.month(String auth_token)
```

Fetches system usage statistics for last month.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#)

v1.sysstat.last.week

```
SysStatIntervalStatistics v1.sysstat.last.week(String auth_token)
```

Fetches system usage statistics for last week.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#)

v1.sysstat.last.day

```
SysStatIntervalStatistics v1.sysstat.last.day(String auth_token)
```

Fetches system usage statistics for last day.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#) where each field contains 24 items

v1.sysstat.last.hour

```
SysStatIntervalStatistics v1.sysstat.last.hour(String auth_token)
```

Fetches system usage statistics for last hour.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#) where each field contains 360 items

v1.sysstat.last.minute

```
SysStatIntervalStatistics v1.sysstat.last.minute(String auth_token)
```

Fetches system usage statistics for last minute.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatistics](#) where each field contains 12 items

v1.sysstat.public.last.hour

```
SysStatIntervalStatistics v1.sysstat.public.last.hour()
```

Fetches system usage statistics for last hour.

Return:

[SysStatIntervalStatistics](#) where each field contains 360 items

v1.sysstat.public.last.minute

```
SysStatIntervalStatistics v1.sysstat.public.last.minute()
```

Fetches system usage statistics for last minute.

Return:

[SysStatIntervalStatistics](#) where each field contains 12 items

v1.sysstat.public.last.5.minutes

```
SysStatIntervalStatistics v1.sysstat.public.last.5.minutes()
```

Fetches system usage statistics for last 5 minutes.

Return:

[SysStatIntervalStatistics](#) where each field contains 60 items

v1.sysstat.public.last

```
SysStatIntervalStatistics v1.sysstat.public.last()
```

Fetches system usage statistics last values.

Return:

[SysStatIntervalStatistics](#) where each field contains 12 items

v1.sysstat.linkstat

```
List<SysStatInterfaceStatus> v1.sysstat.linkstat(String auth_token)
```

Fetches network interfaces status.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of structures [SysStatInterfaceStatus](#) for all system interfaces.

v1.sysstat.memory

```
SysStatMemoryInfo v1.sysstat.memory(String auth_token)
```

Fetches memory usage statistics.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatMemoryInfo](#)

v1.sysstat.disk

```
SysStatDiskInfo v1.sysstat.disk(String auth_token)
```

Fetches disk usage statistics.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatDiskInfo](#)

v1.sysstat.uptime

```
Integer v1.sysstat.uptime(String auth_token)
```

Fetches application uptime in seconds.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Application uptime

v1.sysstat.uptime.system

```
Integer v1.sysstat.uptime.system(String auth_token)
```

Fetches system uptime in seconds.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

System uptime in seconds

v1.sysstat_ms.counters

```
SysStatCountersMS v1.sysstat_ms.counters(String auth_token, String protocol)
```

```
SysStatCountersMS v1.sysstat_ms.counters(String auth_token, String node, String protocol)
```

Parameters:

- **auth_token** — authentication token
- **node** — node name

- **protocol** — allowed values are: "pop", "smtp", "all"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatCountersMS](#)

v1.sysstat_ms.last.year

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.year(String auth_token, String protocol, String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.year(String auth_token, String node, String protocol, String hits)
```

Fetches system usage statistics for last year.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat_ms.last.month

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.month(String auth_token, String protocol, String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.month(String auth_token, String node, String protocol, String hits)
```

Fetches system usage statistics for last month.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat_ms.last.week

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.week(String auth_token, String protocol,  
String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.week(String auth_token, String  
node, String protocol, String hits)
```

Fetches system usage statistics for last week.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat_ms.last.day

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.day(String auth_token, String protocol,  
String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.day(String auth_token, String node, String  
protocol, String hits)
```

Fetches system usage statistics for last day.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat_ms.last.hour

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.hour(String auth_token, String protocol, String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.hour(String auth_token, String node, String protocol, String hits)
```

Fetches system usage statistics for last hour.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat_ms.last.minute

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.minute(String auth_token, String protocol, String hits)
```

```
SysStatIntervalStatisticsMS v1.sysstat_ms.last.minute(String auth_token, String node, String protocol, String hits)
```

Fetches system usage statistics for last minute.

Parameters:

- **auth_token** — authentication token
- **node** — node name
- **protocol** — allowed values are: "pop", "smtp", "all"
- **hits** — list of values represented as JSON-encoded array: "mark", "pass", "batv", "mail", "spam", "clean", "dnsbl", "virus", "no_action", "connection", "drop_with_error", "drop_without_error"

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

[SysStatIntervalStatisticsMS](#)

v1.sysstat.flow.last.hour

List **v1.sysstat.flow.last.hour**(String auth_token, String node, String query)

List **v1.sysstat.flow.last.hour**(String auth_token, String query)

Fetches flow statistics for last hour.

Parameters:

- **auth_token** — authentication token
- **node** — node name, if omitted current node is assumed
- **query** — data query

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Returns List of structures with the following fields:

timestamp	Integer	Timestamps in system posix time
value	Variant	Data identifier, should be one of the following types: user identifier, application identifier, IP address
in	Double	
out	Double	
name	String	

Component: UPL

Module methods

v1.upl.rules.list.....	266	v1.upl.rule.update	267
v1.upl.rule.get.....	266	v1.upl.rules.clear	268
v1.upl.rule.add	267	v1.upl.rules.import	268
v1.upl.rule.delete.....	267	v1.upl.rule.move.position	268

v1.upl.rules.list

```
UplListResponse v1.upl.rules.list(String auth_token, String section, Integer start, Integer limit)
```

Fetch list of objects from given section.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **start** — offset in complete list
- **limit** — max length of returned list

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

List of objects in terms of UPL.

v1.upl.rule.get

```
String v1.upl.rule.get(String auth_token, String section, UID id)
```

Fetch single object definition represented in terms of UPL.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **id** — object identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Single object definition.

v1.upl.rule.add

```
Boolean v1.upl.rule.add(String auth_token, String section, String body, Integer position)
```

Create a new object.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **body** — object definition in terms of UPL
- **position** — object insert position

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.upl.rule.delete

```
Boolean v1.upl.rule.delete(String auth_token, String section, Integer position)
```

Delete existing object.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **position** — object position to delete

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Always returns `true`.

v1.upl.rule.update

```
Boolean v1.upl.rule.update(String auth_token, String section, String body, Integer position)
```

Update object in given position.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **body** — object definition in terms of UPL
- **position** — object position

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **404** — Object not found

Return:

Always returns `true`.

v1.upl.rules.clear

```
Boolean v1.upl.rules.clear(String auth_token, String section)
```

Delete all objects in given section

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

v1.upl.rules.import

```
Boolean v1.upl.rules.import(String auth_token, String section, String body)
```

Import objects.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **body** — object definition in terms of UPL

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **602** — Incorrect UPL data

Return:

Always returns `true`.

v1.upl.rule.move.position

```
Boolean v1.upl.rule.move.position(String auth_token, String section, Integer position,  
Integer new_position)
```

Move upl rule to new position.

Parameters:

- **auth_token** — authentication token
- **section** — objects section, see full list of sections in [UplSection](#)
- **position** — object position
- **new_position** — new object position

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Always returns `true`.

Component: WAF

Module methods

v1.waf.profiles.list.....	270	v1.waf.system.rule.update.....	275
v1.waf.profile.add.....	270	v1.waf.system.rule.fetch.....	276
v1.waf.profile.fetch.....	271	v1.waf.system.rule.restore.default.....	276
v1.waf.profile.update.....	271	v1.waf.system.layers.list.....	277
v1.waf.profile.delete.....	272	v1.waf.system.rules.technologies.list.....	277
v1.waf.custom.layers.list.....	272	v1.waf.exceptions.list.....	278
v1.waf.custom.layer.add.....	273	v1.waf.exception.add.....	278
v1.waf.custom.layer.fetch.....	273	v1.waf.exception.fetch.....	279
v1.waf.custom.layer.update.....	273	v1.waf.exception.update.....	279
v1.waf.custom.layer.delete.....	274	v1.waf.exception.delete.....	279
v1.waf.custom.layer.validate.....	274	v1.waf.system.security.package.is.loaded.....	280
v1.waf.system.rules.list.....	275		

v1.waf.profiles.list

```
List<WafProfileInfo> v1.waf.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all WAF profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of WAF profiles

v1.waf.profile.add

```
UID v1.waf.profile.add(String auth_token, WafProfileInfo profile)
```

Create a new WAF profile. The structures in the `layers` field must be arranged in a special order. For details see [WafProfileInfo](#). If the order is not followed, the fault `601 Incorrect layers order` will be returned.

Parameters:

- **auth_token** — authentication token
- **profile** — a new profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **404** — Layer with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)
- **601** — Incorrect layers order

Return:

identifier of just added profile

v1.waf.profile.fetch

```
WafProfileInfo v1.waf.profile.fetch(String auth_token, UID id)
```

Fetch a single WAF profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found

Return:

profile

v1.waf.profile.update

```
Boolean v1.waf.profile.update(String auth_token, UID id, WafProfileInfo profile)
```

The structures in the layers field must be arranged in a special order. For details see [WafProfileInfo](#). If the order is not followed, the fault `601 Incorrect layers order` will be returned.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier
- **profile** — profile parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Profile with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)
- **601** — Incorrect layers order

Return:

true

v1.waf.profile.delete

```
Boolean v1.waf.profile.delete(String auth_token, UID id)
```

Delete WAF profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true

v1.waf.custom.layers.list

```
List<WafCustomLayerInfo> v1.waf.custom.layers.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by custom layer name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parametrs

Return:

list of WAF custom layers

v1.waf.custom.layer.add

```
UID v1.waf.custom.layer.add(String auth_token, WafCustomLayerInfo layer)
```

Create a new WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **layer** — a new rule

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Conflict, WAF custom layer with such parameters already exists
- **601** — Missing required field(s)
- **602** — UPL validation error

Return:

identifier of just added rule

v1.waf.custom.layer.fetch

```
WafCustomLayerInfo v1.waf.custom.layer.fetch(String auth_token, UID id)
```

Fetch a single WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **id** — layer identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Layer with given identifier not found

Return:

rule

v1.waf.custom.layer.update

```
Boolean v1.waf.custom.layer.update(String auth_token, UID rule_id, WafCustomLayerInfo layer)
```

Update existing WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **layer** — custom layer parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Rule with given identifier not found
- **409** — Conflict, Waf custom layer with such parameters already exists
- **602** — UPL validation error

Return:

true

v1.waf.custom.layer.delete

```
Boolean v1.waf.custom.layer.delete(String auth_token, UID id)
```

Delete WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **id** — layer identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Custom layer with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true

v1.waf.custom.layer.validate

```
Boolean v1.waf.custom.layer.validate(String auth_token, String upl)
```

Validate WAF custom layer.

Parameters:

- **auth_token** — authentication token
- **upl** — upl layer

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

Structure that contains `"check_result"`, on of the following values: `"success"`, `"error"`, and `"description"`, error description

v1.waf.system.rules.list

```
PagingList<WafSystemRuleInfo> v1.waf.system.rules.list(String auth_token, Integer start, Integer limit, String language, Struct filter, List<SortingSpec> sorting_options)
```

```
PagingList<WafSystemRuleInfo> v1.waf.system.rules.list(String auth_token, Integer start, Integer limit, String language, List<WAFRulesOverrides> rules, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all WAF system rules.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **language** — selects the system rule description language (e.g., "en", "ru")
- **rules** — WAF lists structures of overridden rules
- **filter** — structure with filtering options, allowed fields are: **query**
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parametrs
- **618** — Incorrect field query value in search filter

Return:

list of WAF system rules

v1.waf.system.rule.update

```
Boolean v1.waf.system.rule.update(String auth_token, WafSystemRuleInfo rule)
```

```
Boolean v1.waf.system.rule.update(String auth_token, WafSystemRuleInfo rule, Struct filter)
```

```
Boolean v1.waf.system.rule.update(String auth_token, UID rule_id, WafSystemRuleInfo rule)
```

```
Boolean v1.waf.system.rule.update(String auth_token, List<UID> rule_id, WafSystemRuleInfo rule)
```

Update WAF system rule.

Parameters:

- **auth_token** — authentication token
- **rule_id** — rule identifier
- **filter** — structure with filtering options, allowed fields are: **query**

- **rule** — rule parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Rule with given identifier not found
- **618** — Incorrect field query value in search filter

Return:

true

v1.waf.system.rule.fetch

```
WafSystemRuleInfo v1.waf.system.rule.fetch(String auth_token, UID id)
```

Fetch a single WAF system rule.

Parameters:

- **auth_token** — authentication token
- **id** — rule identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Rule with given identifier not found

Return:

rule

v1.waf.system.rule.restore.default

```
Boolean v1.waf.system.rule.restore.default(String auth_token)
```

```
Boolean v1.waf.system.rule.restore.default(String auth_token, List<UID> rule_ids)
```

Set the default value for system rules for fields: "action", "rule_log", "enabled", "response_profile_id", "exception_ids". If `rule_ids` has not been passed then all system rules will be restored.

Parameters:

- **auth_token** — authentication token
- **rule_ids** — ids of system rules that will be restored

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

true

v1.waf.system.layers.list

```
PagingList<WafSystemLayerInfo> v1.waf.system.layers.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

```
PagingList<WafSystemLayerInfo> v1.waf.system.layers.list(String auth_token, Integer start, Integer limit, List<WAFRulesOverrides> waf_overrides, List<UID> technology_ids, Struct filter, List<SortingSpec> sorting_options)
```

```
PagingList<WafSystemLayerInfo> v1.waf.system.layers.list(String auth_token, Integer start, Integer limit, String language, List<WAFRulesOverrides> waf_overrides, List<UID> technology_ids, Struct filter, List<SortingSpec> sorting_options)
```

Get waf system layers. If `waf_overrides` and `technology_ids` fields has been passed, the values in fields `active_rules_count`, `active_threat_levels` will be counted. If `language` field is missing, the description field will be in English.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **language** — selects the system layer description language (e.g., "en", "ru")
- **filter** — structure with filtering options, allowed fields are: **search** (search by system layer name and description) and **profile_filters** (list of [WafProfileLayerInfo](#) for system layer only)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)
- **waf_overrides** — list of [WAFRulesOverrides](#) structures
- **technology_ids** — list of technology IDs

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **603** — Incorrect paging parameters

Return:

list of WAF system rule layers

v1.waf.system.rules.technologies.list

```
PagingList<WafSystemRuleTechnologyInfo> v1.waf.system.rules.technologies.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by technology name)

- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of WAF system rule technologies

v1.waf.exceptions.list

```
List<WafExceptionInfo> v1.waf.exceptions.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all WAF exceptions.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **query** (search by exception name, description and loggingStatus)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of WAF exceptions

v1.waf.exception.add

```
UID v1.waf.exception.add(String auth_token, WafExceptionInfo exception)
```

Create a new WAF exception.

Parameters:

- **auth_token** — authentication token
- **exception** — a new exception object

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added exception

v1.waf.exception.fetch

```
WafExceptionInfo v1.waf.exception.fetch(String auth_token, UID id)
```

Fetch a single WAF exception.

Parameters:

- **auth_token** — authentication token
- **id** — exception identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Exception with given identifier not found

Return:

exception

v1.waf.exception.update

```
Boolean v1.waf.exception.update(String auth_token, UID id, WafExceptionInfo exception)
```

Parameters:

- **auth_token** — authentication token
- **id** — exception identifier
- **exception** — exception parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Exception with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.waf.exception.delete

```
Boolean v1.waf.exception.delete(String auth_token, UID id)
```

Delete WAF exception.

Parameters:

- **auth_token** — authentication token
- **id** — exception identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Exception with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true

v1.waf.system.security.package.is.loaded

Boolean **v1.waf.system.security.package.is.loaded**(String auth_token)

Returns true if the OWASP TOP 10 or Anti-Bot package is loaded. Otherwise, returns false.

Parameters:

- **auth_token** — authentication token

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

@return true or false

Component: WebSocket

Module methods

v1.websocket.profiles.list	281	v1.websocket.profile.update	282
v1.websocket.profile.add	281	v1.websocket.profile.delete	282
v1.websocket.profile.fetch	282		

v1.websocket.profiles.list

```
List<WebSocketProfileInfo> v1.websocket.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch all websocket profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

list of websocket profiles

v1.websocket.profile.add

```
UID v1.websocket.profile.add(String auth_token, WebSocketProfileInfo profile)
```

Create a new websocket profile.

Parameters:

- **auth_token** — authentication token
- **profile** — a new profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added profile

v1.websocket.profile.fetch

```
WebsocketProfileInfo v1.websocket.profile.fetch(String auth_token, UID id)
```

Fetch a single websocket profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found

Return:

websocket profile

v1.websocket.profile.update

```
Boolean v1.websocket.profile.update(String auth_token, UID id, WebsocketProfileInfo profile)
```

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier
- **profile** — profile parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Profile with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.websocket.profile.delete

```
Boolean v1.websocket.profile.delete(String auth_token, UID id)
```

Delete websocket profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true

Component: XML

Module methods

v1.xml.profiles.list	284	v1.xml.schema.add	286
v1.xml.profile.add	284	v1.xml.schema.valid	287
v1.xml.profile.fetch	285	v1.xml.schema.fetch	287
v1.xml.profile.update	285	v1.xml.schema.update	287
v1.xml.profile.delete	286	v1.xml.schema.delete	288
v1.xml.schemas.list	286		

v1.xml.profiles.list

```
PagingList<XmlProfileInfo> v1.xml.profiles.list(String auth_token, Integer start, Integer limit, Struct filter, List<SortingSpec> sorting_options)
```

Fetch XML profiles.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by profile name and description)
- **sorting_options** — sorting options, may be empty to use default sorting or contain single element of type [SortingSpec](#)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

List of XML profiles

v1.xml.profile.add

```
UID v1.xml.profile.add(String auth_token, XmlProfileInfo profile)
```

Create a new XML profile.

Parameters:

- **auth_token** — authentication token
- **profile** — a new profile

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added profile

v1.xml.profile.fetch

```
XmlProfileInfo v1.xml.profile.fetch(String auth_token, UID id)
```

Fetch a single XML profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Profile with given identifier not found

Return:

XML profile

v1.xml.profile.update

```
Boolean v1.xml.profile.update(String auth_token, UID id, XmlProfileInfo profile)
```

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier
- **profile** — profile parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Profile with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.xml.profile.delete

```
Boolean v1.xml.profile.delete(String auth_token, UID id)
```

Delete XML profile.

Parameters:

- **auth_token** — authentication token
- **id** — profile identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Object with given identifier not found
- **502** — Profile cannot be deleted because it's being used by another object

Return:

true

v1.xml.schemas.list

```
PagingList<XmlSchemaInfo> v1.xml.schemas.list(String auth_token, Integer start, Integer limit, Struct filter)
```

Fetch XML schemas.

Parameters:

- **auth_token** — authentication token
- **start** — offset in complete list
- **limit** — max length of returned list
- **filter** — structure with filtering options, allowed fields are: **search** (search by schema name and description)

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **603** — Incorrect paging parameters

Return:

List of XML schemas

v1.xml.schema.add

```
UID v1.xml.schema.add(String auth_token, XmlSchemaInfo schema)
```

Create a new XML schema.

Parameters:

- **auth_token** — authentication token
- **schema** — a new schema

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **111** — Incorrect field data
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

identifier of just added schema

v1.xml.schema.valid

```
XmlSchemaInfo v1.xml.schema.valid(String auth_token, String schema)
```

Schema validation

Parameters:

- **auth_token** — authentication token
- **schema** — xml schema

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token

Return:

object: - {success: True} - {success: False, description: }

v1.xml.schema.fetch

```
XmlSchemaInfo v1.xml.schema.fetch(String auth_token, UID id)
```

Fetch a single XML schema.

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Schema with given identifier not found

Return:

XML schema

v1.xml.schema.update

```
Boolean v1.xml.schema.update(String auth_token, UID id, XmlSchemaInfo schema)
```

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier
- **schema** — schema parameters that need to be changed

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **111** — Incorrect field data
- **404** — Schema with given identifier not found
- **409** — Duplicate name value
- **601** — Missing required field(s)

Return:

true

v1.xml.schema.delete

```
Boolean v1.xml.schema.delete(String auth_token, UID id)
```

Delete XML schema.

Parameters:

- **auth_token** — authentication token
- **id** — schema identifier

Errors:

- **102** — Access denied
- **104** — Request not authenticated due to missing, invalid, or expired auth token
- **110** — Incorrect method arguments
- **404** — Schema with given identifier not found
- **502** — Object cannot be deleted because it's being used by another object

Return:

true