



Rbeapi Documentation

Release 0.5.1

Arista Networks - EOS+ Consulting Services

September 21, 2016

1	Overview	3
1.1	Requirements	3
2	Getting Started	5
2.1	Example eapi.conf File	5
2.2	Configuring [connection:localhost]	6
2.3	Using rbeapi	6
2.4	Creating a connection and sending commands	6
2.5	Using the API	7
3	Installation	9
4	Upgrading	11
5	Testing Modules	13
6	Contributing	15
7	Release Notes	17
8	License	19

Contents:

Overview

- *Requirements*

The Ruby Client for eAPI provides a native Ruby implementation for programming Arista EOS network devices using Ruby. The Ruby client provides the ability to build native applications in Ruby that can communicate with EOS either locally via Unix domain sockets (on-box) or remotely over a HTTP/S transport (off-box). It uses a standard INI-style configuration file to specify one or more connection profiles.

The rbeapi implementation also provides an API layer for building native Ruby objects that allow for configuration and state extraction of EOS nodes. The API layer provides a consistent implementation for working with EOS configuration resources. The implementation of the API layer is highly extensible and can be used as a foundation for building custom data models.

The library is freely provided to the open source community for building robust applications using Arista EOS eAPI. Support is provided as best effort through Github issues.

1.1 Requirements

- Arista EOS 4.12 or later
- Arista eAPI enabled for at least one transport (see official EOS Config Guide at arista.com for details)
- Ruby 1.9.3 or later

Getting Started

- *Example eapi.conf File*
- *Configuring [connection:localhost]*
- *Using rbeapi*
- *Creating a connection and sending commands*
- *Using the API*

In order to use rbeapi, the EOS command API must be enabled using management api http-commands configuration mode. This library supports eAPI calls over both HTTP and UNIX Domain Sockets. Once the command API is enabled on the destination node, create a configuration file with the node properties.

Note: The default search path for the conf file is ~/.eapi.conf followed by /mnt/flash/eapi.conf. This can be overridden by setting EAPI_CONF=<path file conf file> in your environment.

2.1 Example eapi.conf File

Below is an example of an eAPI conf file. The conf file can contain more than one node. Each node section must be prefaced by connection:<name> where <name> is the name of the connection.

The following configuration options are available for defining node entries:

host - The IP address or FQDN of the remote device. If the host parameter is omitted then the connection name is used
 username - The eAPI username to use for authentication (only required for http or https connections)
 password - The eAPI password to use for authentication (only required for http or https connections)
 enablepwd - The enable mode password if required by the destination node
 transport - Configures the type of transport connection to use. The default value is https. Valid values are:

socket (available in EOS 4.14.5 or later) http_local (available in EOS 4.14.5 or later) http https

port - Configures the port to use for the eAPI connection. A default port is used if this parameter is absent, based on the transport:

transport: http, default port: 80 transport: https, default port: 443 transport: https_local, default port: 8080 transport: socket, default port: n/a

open_timeout - The default number of seconds to wait for the eAPI connection to open. Any number may be used, including Floats for fractional seconds. Default value is 10 seconds.
 read_timeout - The default number of seconds to wait for one block of eAPI results to be read (via one read(2) call). Any number may be used, including Floats for fractional seconds. Default value is 10 seconds.

Note: See the EOS User Manual found at arista.com for more details on configuring eAPI values.

All configuration values are optional.

```
[connection:veos01] username: eapi password: password transport: http
```

```
[connection:veos02] transport: http
```

```
[connection:veos03] transport: socket
```

```
[connection:veos04] host: 172.16.10.1 username: eapi password: password enablepwd: itsasecret port: 1234 transport: https
```

```
[connection:localhost] transport: http_local
```

The above example shows different ways to define EOS node connections. All configuration options will attempt to use default values if not explicitly defined. If the host parameter is not set for a given entry, then the connection name will be used as the host address.

2.2 Configuring [connection:localhost]

The rbeapi library automatically installs a single default configuration entry for connecting to localhost host using a transport of sockets. If using the rbeapi library locally on an EOS node, simply enable the command API to use sockets and no further configuration is needed for rbeapi to function. If you specify an entry in a conf file with the name [connection:localhost], the values in the conf file will overwrite the default.

2.3 Using rbeapi

The Ruby Client for eAPI was designed to be easy to use and implement for writing tools and applications that interface with the Arista EOS management plane.

2.4 Creating a connection and sending commands

Once EOS is configured properly and the config file created, getting started with a connection to EOS is simple. Below demonstrates a basic connection using rbeapi. For more examples, please see the examples folder.

```
# start by importing the library require 'rbeapi/client'

# create a node object by specifying the node to work with node = Rbeapi::Client.connect_to('veos01')

# send one or more commands to the node node.enable('show hostname') node.enable('show hostname') => [{:command=>"show hostname", :result=>{"fqdn"=>"veos01.arista.com", "hostname"=>"veos01"}, :encoding=>"json"}]

# use the config method to send configuration commands node.config('hostname veos01') => [{}]

# multiple commands can be sent by using a list (works for both enable or config)

node.config(['interface Ethernet1', 'description foo']) => [{}], [{}]

# return the running or startup configuration from the node (output omitted for brevity)

node.running_config

node.startup_config
```

2.5 Using the API

The rbeapi library provides both a client for send and receiving commands over eAPI as well as an API for working directly with EOS resources. The API is designed to be easy and straightforward to use yet also extensible. Below is an example of working with the vlans API

```
# create a connection to the node require 'rbeapi/client' node = Rbeapi::Client.connect_to('veos01')
# get the instance of the API (in this case vlans) vlans = node.api('vlans')
# return all vlans from the node vlans.getall => {"1"=>{:name=>"tester", :state=>"active", :trunk_groups=>[]},
        "4"=>{:name=>"VLAN0004",           :state=>"active",           :trunk_groups=>[]},
        "100"=>{:name=>"TEST_VLAN_100",     :state=>"active",           :trunk_groups=>[]},
        "300"=>{:name=>"VLAN0300", :state=>"active", :trunk_groups=>[]}}
# return a specific vlan from the node vlans.get(1) => {:name=>"tester", :state=>"active", :trunk_groups=>[]}
# add a new vlan to the node vlans.create(400) => true
# set the new vlan name vlans.set_name(100, value: 'foo') => true
```

All API implementations developed by Arista EOS+ CS are found in the rbeapi/api folder. See the examples folder for additional examples.

Installation

The source code for rbeapi is provided on Github at <http://github.com/arista-eosplus/rbeapi>. All current development is done in the develop branch. Stable released versions are tagged in the master branch and uploaded to RubyGems.

To install the latest stable version of rbeapi, simply run `gem install rbeapi`

To install the latest development version from Github, simply clone the develop branch and run

```
$ rake build $ rake install
```

To create an RPM, run `rake rpm`

To generate a SWIX file for EOS with necessary dependencies, run `rake all_rpms` then follow the swix create instructions, provided by the build. NOTE: PuppetLabs provides a puppet agent SWIX which includes Ruby 1.9.3 in `/opt/puppet/bin/` which is different from where you might otherwise install Ruby. If you have installed the puppet-enterprise 3.x SWIX, then you should build and use the `rbeapi-puppet3` swix, below. If you have installed the puppet-enterprise 2015.x SWIX, then you should build and use the `rbeapi-puppet-aio` swix, below. Otherwise, if you have installed at least Ruby 1.9.3 in the standard system location, then the `rbeapi` SWIX may be used.

`$ bundle install --path .bundle/gems/ $ bundle exec rake all_rpms ...` RPMs are available in `rpms/noarch/` Copy the RPMs to an EOS device then run the 'swix create' command. Examples:

Puppet Open Source: `cd /mnt/flash; swix create rbeapi-0.4.0-1.swix rubygem-rbeapi-0.4.0-1.eos4.noarch.rpm rubygem-inifile-3.0.0-3.eos4.noarch.rpm rubygem-netaddr-1.5.0-2.eos4.noarch.rpm rubygem-net_http_unix-0.2.1-3.eos4.noarch.rpm`

Puppet-enterprise agent (3.x): `cd/mnt/flash; swix create rbeapi-puppet3-0.4.0-1.swix rubygem-rbeapi-puppet3-0.4.0-1.eos4.noarch.rpm rubygem-inifile-puppet3-3.0.0-3.eos4.noarch.rpm rubygem-netaddr-puppet3-1.5.0-2.eos4.noarch.rpm`

Puppet-All-in-one agent (2015.x/4.x): `cd/mnt/flash; swix create rbeapi-puppet-aio-0.4.0-1.swix rubygem-rbeapi-puppet-aio-0.4.0-1.eos4.noarch.rpm rubygem-inifile-puppet-aio-3.0.0-3.eos4.noarch.rpm rubygem-netaddr-puppet-aio-1.5.0-2.eos4.noarch.rpm rubygem-net_http_unix-puppet-aio-0.2.1-3.eos4.noarch.rpm`

On EOS:

```
Arista# copy <URI-to-RPMs> flash: Arista# bash -bash-4.1# cd /mnt/flash/ -bash-4.1# swix create rbeapi-puppet3-0.4.0-1.swix
```

```
rubygem-rbeapi-puppet3-0.4.0-1.eos4.noarch.rpm          rubygem-inifile-puppet3-3.0.0-1.eos4.noarch.rpm
rubygem-netaddr-puppet3-1.5.0-1.eos4.noarch.rpm
```

```
-bash-4.1# exit Arista# copy flash:rbeapi-puppet3-0.4.0-1.swix extension: Arista# extension rbeapi-puppet3-0.4.0-1.swix Arista# copy installed-extensions boot-extensions
```

Upgrading

On EOS:

```
Arista# no extension pe-rbeapi-0.3.0-1.swix Arista# extension rbeapi-puppet3-0.4.0-1.swix Arista# copy  
installed-extensions boot-extensions
```

Testing Modules

The rbeapi library provides spec tests. To run the spec tests, you will need to update the `dut.conf` file found in `spec/fixtures`. The switch used for testing should have at least interfaces Ethernet1-7.

To run the spec tests, run `bundle exec rspec spec` from the root of the rbeapi source folder.

Contributing

Contributing pull requests are gladly welcomed for this repository. Please note that all contributions that modify the library behavior require corresponding test cases otherwise the pull request will be rejected.

Release Notes

License

Copyright (c) 2015, 2016, Arista Networks, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Arista Networks nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL ARISTA NETWORKS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.