Arista eAPI

Introduction

Arista EOS offers multiple programmable interfaces for applications. These interfaces can be leveraged by applications running on the switch, or external to EOS. Arista's newest interface, EOS API (eAPI), allows applications and scripts to have complete programmatic control over EOS, with a stable and easy to use syntax. Once the API is enabled, the switch accepts commands using Arista's CLI syntax, and responds with machine-readable output and errors serialized in JSON, served over HTTP.

The EOS API has three major advantages:

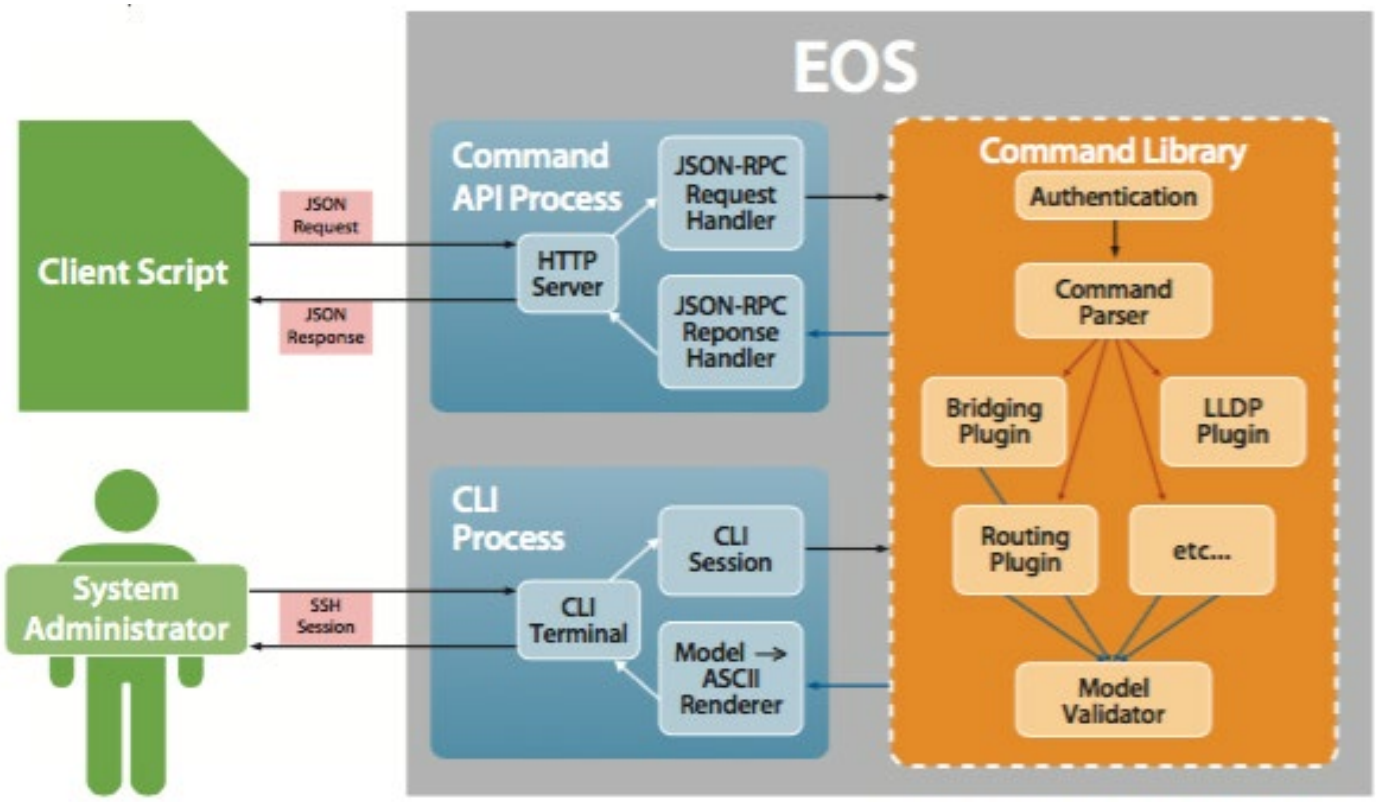
1. Comprehensiveness. With Arista's eAPI, customers can access any state and configure any properties on the switch that they could otherwise do over the CLI.

2. Ease-of-use and flexibility. The simplicity of this protocol and the availability of third party JSON clients means that eAPI is language agnostic and can be easily integrated into any existing infrastructure and workflows. Additionally, on-box, interactive documentation for the API and return values makes writing new programs simple. (To view, enable the API and visit "http://<your-switch's-ip-address>/ explorer.html" in a web browser).

3. Stability. Arista ensures that a command's structured output will remain compatible for multiple future versions of EOS. This allows end users to confidently develop critical applications without compromising their ability to upgrade to newer EOS releases and access new features. Furthermore, this affords scripts the ability to operate cleanly in datacenters running multiple versions of EOS, without compromising eAPI's simplicity.

EOS Architecture Overview

The eAPI architecture is very straightforward spacing, as diagrammed below:



- Clients send an HTTP POST request to the server, using the lightweight JSON-RPC 2.0 protocol. Requests specify
  - » The "method" to use (at this time, always "runCmds").
  - » A list of commands to run, for example ['show interfaces'], or ['configure', 'interface Ethernet 1', 'shutdown']
  - » A "version" number, specifying which revision of the model output your script expects (at this time, always "1")
- The server processes the request, and collects a structured data model for each command, which is serialized back into JSON. If a command returns an error, the JSON-RPC 2.0 'errors' field will be appropriately set, otherwise the response is placed in the 'result' field. Further documentation on response formats can be viewed at http://www.jsonrpc.org/speci cation and the on-box documentation at http://<your-switch's-ip-address>/ overview.html and http://<your switch's-ip- address>/documentation.html.

EOS Architecture Overview

eAPI's completeness, stability, and ease of use makes it well suited for a variety of customer applications. For example, a persistent application that manages Hadoop clusters may wish to monitor metrics like packet drops so it can dynamically adjust

Arista eAPI

eAPI provides access to all switch state. The command line interface (CLI) will be built leveraging the eAPI, thus exposing all CLI commands to a programmatic API interface. eAPI implementation separates the state model (such as MAC address table) from the protocol (JSON over HTTP), so that when a protocol gets implemented it gets all existing state models for free. When a new state model is implemented on the switch, it is fully accessible via all protocols.

running parameters such as number of mappers/reducers between runs, thus keeping the cluster running at maximum ef ciency. Meanwhile, latency sensitive applications can monitor and configure LANZ programmatically based off of external factors such as the time of day, current load type, etc., to optimize changing workflows.

eAPI also excels at automating manual processes. For example, suppose a system administrator wishes to hitlessly shutdown a node that is advertising a shared network address. The sysadmin could write a script to increase routing protocol metrics on the connected switch for the network address they wish to drain traffic from, thereby causing traffic to be gracefully routed to other switches announcing that address. Once egress packet counters on the connected switch have stabilized, the script can safely shutdown the node or port without impacting existing traffic flows. Once the process is automated, the possibility of human error, one of the most common source of outages, is significantly reduced.

Another use case is for the network administrator who wishes to configure many switches simultaneously. For example, if a network operator needs to create an ACL for all ports in a datacenter that are bridging traffic on VLAN 3, a simple 10 line script, written in a language of the user's choice, makes this once complicated task trivial and safe.

Summary

The above examples are a small subset of possible use-cases for eAPI. Because all CLI commands are accessible over an API that uses a widely accessible protocol, any task that previously required manual input can now be automated with incredible ease. Furthermore, eAPI's guaranteed stability ensures that even network-critical processes can be programmatically controlled. In this age of software defined networks, eAPI gives an unprecedented ability to automate and control your data center.

Santa Clara—Corporate Headquarters
5453 Great America Parkway,
Santa Clara, CA 95054

Phone: +1-408-547-5500
Fax: +1-408-538-8920
Email: info@arista.com

Ireland—International Headquarters
3130 Atlantic Avenue
Westpark Business Campus
Shannon, Co. Clare
Ireland

Vancouver—R&D Office
9200 Glenlyon Pkwy, Unit 300
Burnaby, British Columbia
Canada V5J 5J8

San Francisco—R&D and Sales Office
1390 Market Street, Suite 800
San Francisco, CA 94102

India—R&D Office
Global Tech Park, Tower A & B, 11th Floor
Marathahalli Outer Ring Road
Devarabeesanahalli Village, Varthur Hobli
Bangalore, India 560103

Singapore—APAC Administrative Office
9 Temasek Boulevard
#29-01, Suntec Tower Two
Singapore 038989

Nashua—R&D Office
10 Tara Boulevard
Nashua, NH 03062