

Is the CLI dead?

Peter Lundqvist
peter@arista.com

Agenda

- Advantages and disadvantages of CLI
- API
- RESTful
- OpenConfig
- JSON RPC over HTTP/HTTPS
- Complex stuff need complex tools

Advantages and disadvantages of CLI

The difference between stupidity and genius is that genius has its limits
- Albert Einstein

Advantages and disadvantages of CLI

- CLI offers a fast and simple way to configure network devices
- Easy for “simple” and small scale automation, i.e. cut and paste from “Notepad”, screen scrape in scripts etc.
- Does not scale to perform repetitive tasks in large networks
- When performing large scale and massive automation something else than screen scraping or scripts building cut and paste config chunks is needed
- Solution is to use APIs or other protocols intended to simplify configuration tasks. APIs can be wrapped with good GUIs

Example advantages of “smart” CLI

- One of the major difficulties with CLI is how to handle possible user related errors...
- This is a constant headache for vendors when creating features and associated commands
- Example: Remove a specific RPKI server. Config for the specific server is removed, however the cache is preserved with expiry timer which is good in case it was a “mistake”

```
' arista(config-router-bgp)# no rpkI cache fake-roa
arista(config-router-bgp)#exit
arista(config)#sh bgp rpkI ca
fake-roa:
This cache's config has been removed; ROAs retained until expiry at 2022-09-12 15:23:58
Host: 192.168.0.220 port 3324
VRF: default
Preference: 5
Last update sync: never
Last full sync: 0:00:53 ago
Last serial query: never
Last reset query: 0:01:28 ago
Entries: 379565
(...)
```

Example of warning banners in the CLI

- What is an “error” and what is intended ?
- Shutdown the whole BGP process need some thought before executing, it deserves a warning banner (and coffee break) !!!

```
arista(config)#router bgp 65000
arista(config-router-bgp)#shut
You are attempting to shutdown BGP. Are you sure you want to shutdown? [confirm]
n
arista(config-router-bgp)#
```

Example disadvantages of “smart” CLI...

- However sometimes even the sun have spots...
- In this scenario **no** “shutdown” exist in the submenu, thereby CLI jumps up one level and execute the command (if type return)

```
arista(config)#router bgp 65000
arista(config-router-bgp)#rpki cache fake-roa
arista(config-rpki-cache)#shutdown
You are attempting to shutdown BGP. Are you sure you want to shutdown? [confirm]

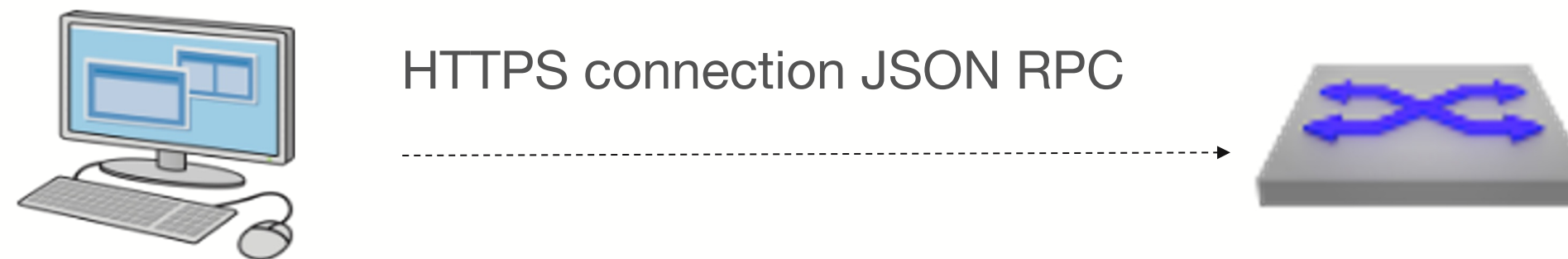
arista(config-router-bgp)#exit
arista(config)#sh ip bgp
BGP is disabled for VRF default
```

- Now it's not that funny any longer (RIB/FIB gone by the wind)...
- Bug vs “shit happens”...

API

Application Programming Interface (API)

- *APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.*
- A way with software to make use of function(s) in another software.
- Simplifies transfer, visualisation and modification of data in another software from your own software.
 - Example 1: Creating a weather app for SmartPhone that display raw data from SMHI (Swedish governmental weather bureau).
 - **Example 2: A script that configures network devices instead of using CLI to configure them.**



RESTful

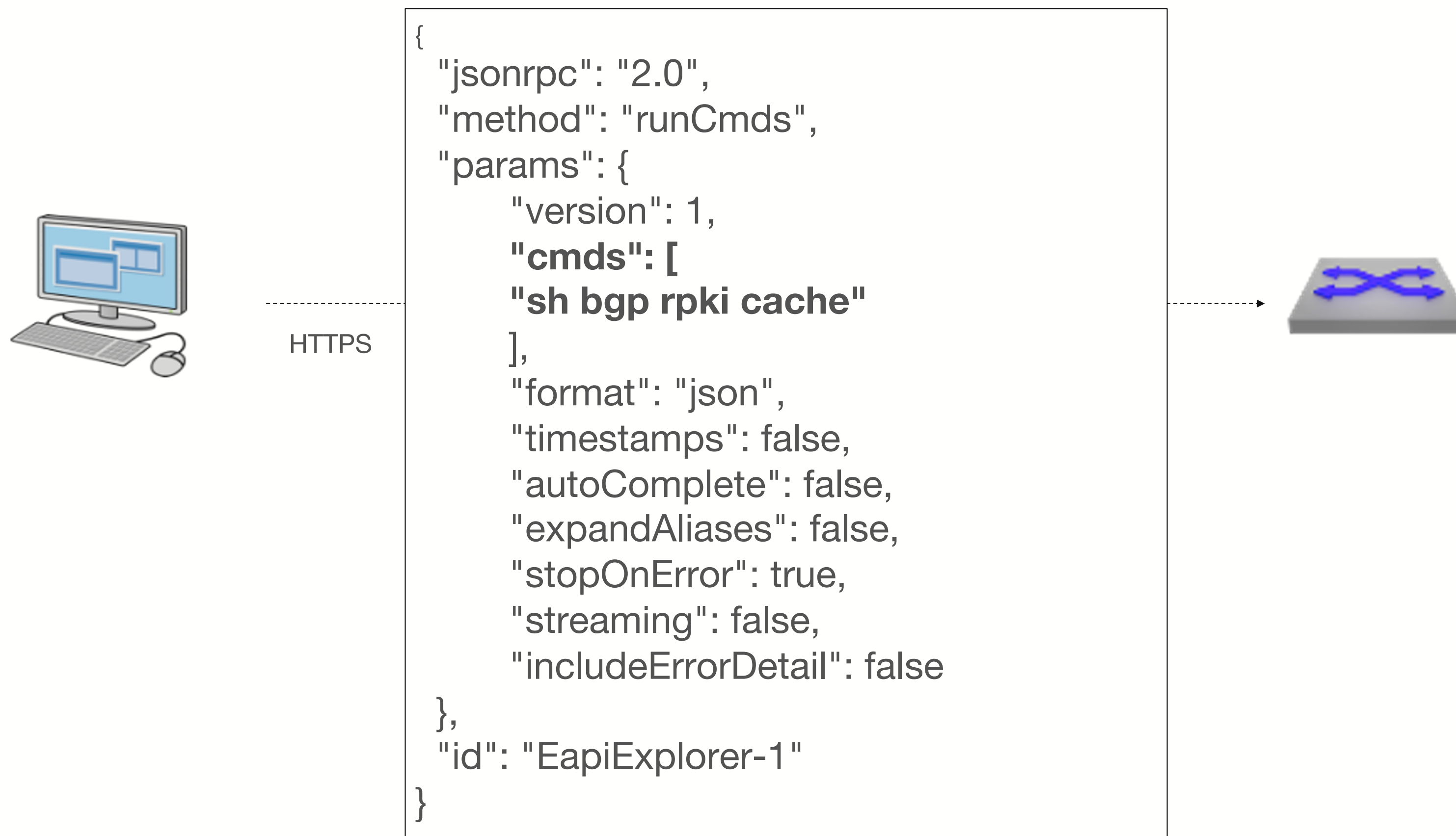
RESTful

- *A Web API (or Web Service) conforming to the REST architectural style is a REST API.*
 - Intended to have a uniform identification of resources
 - Intended to have an exchange of data through representations, i.e. data models
 - Messages are to be self descriptive on how to process the message
 - Messages are to be exchanged using URI/URL
 - Solve the Energy crises
- Even though all of the good intentions above, there is not one RESTful API model that works for *.* software. Manufacturers have adapted them to their (own) needs.
- Examples of RESTful APIs are
 - SOAP/XML driven APIs
 - NetConfig XML driven APIs
 - JSON RPC driven APIs
 - OpenConfig YAML using NetConfig as transport
 - RFC 1149

JSON RPC over HTTP/HTTPS

JSON RPC over HTTP/HTTPS

- An example of a RESTful API is JSON RPC over HTTP/HTTPS



OpenConfig

OpenConfig

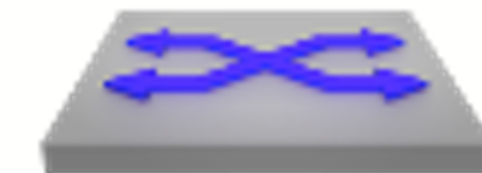
- Openconfig has been driven originally by the Cloud folks
- Intended to have a single model to perform configuration tasks platform independent, driven by an IETF group defining models
- IETF groups governing dominated by Cloud has resulted in very few models, lots of important configuration tasks are “missing” for non-Cloud...
- Network vendors have added their own model but also adapted the actual exchange of OpenConfig communication
- The initiative to create a “standard” has failed, its not “one” OpenConfig. Its vendor x OpenConfig, vendor y OpenConfig, vendor z OpenConfig etc...
- Heard about this before, SNMP Private MIBs?

OpenConfig

- An example of OpenConfig - configure a BGP neighbor



```
gnmi -addr 10.0.0.1:6030 -username admin -password admin update  
'/network-instances/network-  
instance[name=default]/protocols/protocol[name=BGP]/bgp/neighbors/neighb  
or[neighbor-address=10.10.100.43]'  
'{"config": {"neighbor-address": "10.10.100.43", "peer-as": 123}}'
```



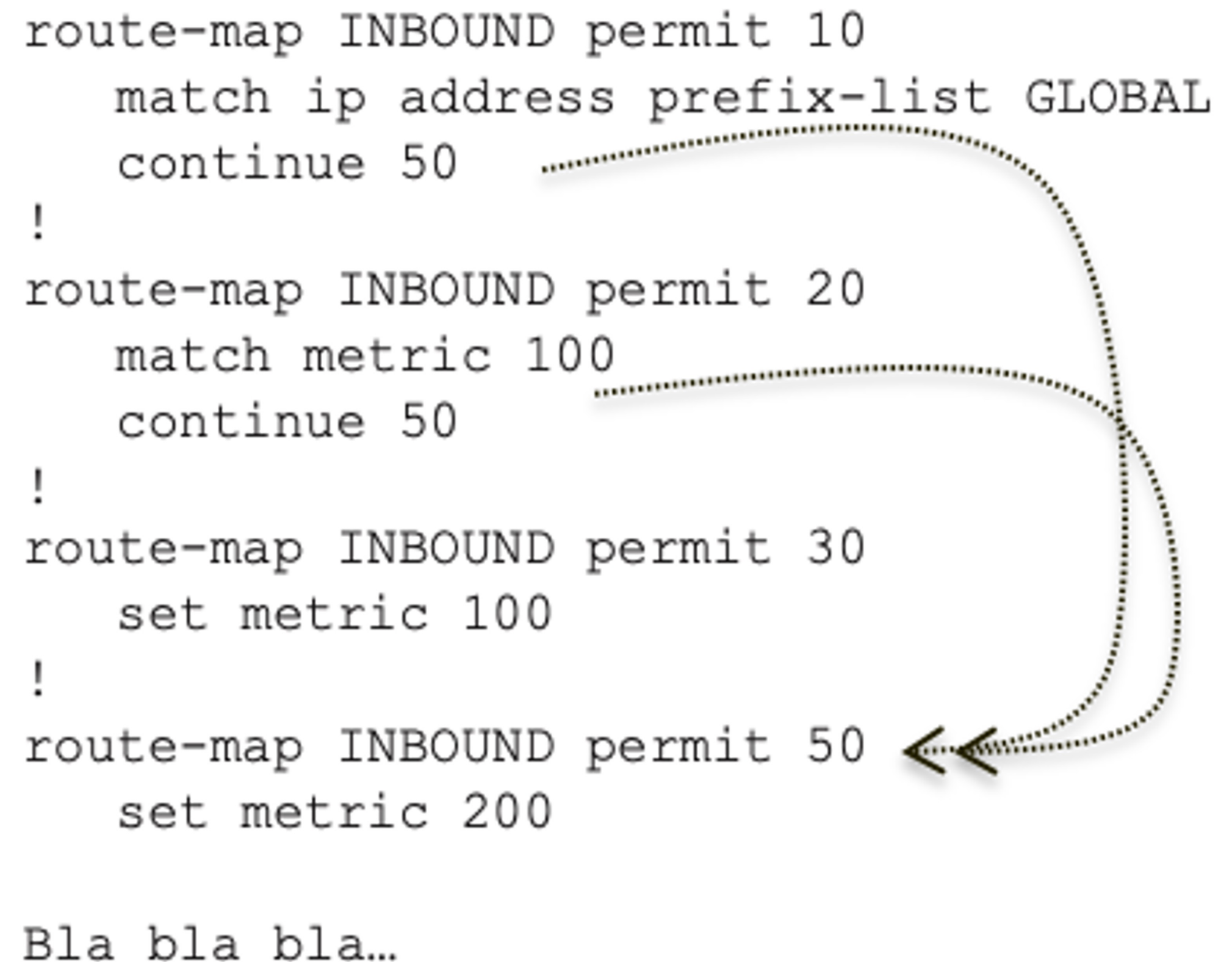
Complex stuff needs advanced tools

Sequence approach

- Policy-tools for building routing policies have been much of a sequence story
 - Read line 1 = action x
 - Read line 2 = action y
 - ...
- Results in huge configurations that breaks the whole idea with human-friendly syntax (that was the original idea)
- The policy steers the focus on what's doable rather than what policies suppose to solve
- Probably seen by most people as a "CLI"

```
route-map INBOUND permit 10
  match ip address prefix-list GLOBAL
  continue 50
!
route-map INBOUND permit 20
  match metric 100
  continue 50
!
route-map INBOUND permit 30
  set metric 100
!
route-map INBOUND permit 50
  set metric 200

Bla bla bla...
```



Programmatic approach

- The programmatic approach adds the whole arsenal of attribute and functions
 - If,then,else,not,and,or...
 - Operators: +/-,&,><...
 - States: true, false...
- The results both compressed and efficient configuration, since pretty much the very same tools are used as with programming
- Since build from scratch offline and applied as a whole (thereby avoids the line by line drama)
- The focus shift to solve the policy as efficient as possible, instead of what can actually be achieved
- Perhaps not seen as “Human friendly CLI”
?

```
function INBOUND {
    if (prefix match prefix_list_v4 GLOBAL or
        med is 100) {
        med = 200;
    } else {
        med = 100;
    }
    return true;
}
```

Validation & Debugging

- Programmatic still needs to be married with CLI.. (show commands)

```
(...)  
function RPKI_CHECK() {  
    if rpki.origin_as_validity is ROA_VALID {  
        local_preference = 200;  
        return true;  
        # Return = V and accept with LOCAL_PREF 200  
    }  
    if rpki.origin_as_validity is ROA_NOT_FOUND {  
        local_preference = 99;  
        return true;  
        # Return = U and accept with LOCAL_PREF 99  
    }  
    if rpki.origin_as_validity is ROA_INVALID {  
        return false;  
        # Return = I and drop  
    }  
    return true;  
}  
(...)
```

```
router(config)#sh bgp debug policy inbound neighbor  
192.168.230.1 ipv4 unicast rcf RPKI_CHECK() 1.6.73.0/24  
  
Path with NLRI 1.6.73.0/24, received from 192.168.230.1  
The path was allowed by the evaluation of RPKI_CHECK().  
  
2    function RPKI_CHECK() {  
3        T- T-----  
4        if rpki.origin_as_validity is ROA_VALID {  
5            -----  
            local_preference = 200;  
            T----- T---  
            return true;  
        }  
    }
```

Conclusion (Subjective)

- No, CLI is not dead... however, its usage needs to change in some scenarios
 - But it was a cool headline 😊
- CLI is still crucial for advanced troubleshooting
 - If you know the error example via Telemetry, then it's not that hard to fix "the issue" example interface congestion or burst !
 - However, how common is it that the error is obvious ?
- Add line by line to active/running configuration is slow and risky
 - since it's based on human interaction and perception
- Powerful tools like programmatic tools hard to make in a "CLI" fashion

Conclusion...

- Building configuration offline takes away “dumb” mistakes
 - Could be that it demands more preparation and not just “login and jump the gun”
- It may be seen as a good idea to add configs partially
 - However then you have to deal with possible banners that force a halt (screen scrape scripts is a no no with banners)
 - Most auto-provisioning tools adds the whole configuration when apply configuration changes
- Many things can be automated, not just the configuration. One such example is the upgrade process:
 - Baseline statistics and load (Telemetry) and take snapshots
 - Gracefully move traffic away
 - Do the upgrade
 - Enable signaling and gracefully add traffic back and compare the Telemetry data

Thank You

arista.com