From Scripting to Intent

From how to what

James Henderson
Automation Solutions Architect
Objective

• Describe what scripting and intent really are, without resorting to buzzwords
• Give a real-world example, using VLANs
• Define what a service really is, and how that relates to intent
• Show where workflows and templates fit into all of this
• Discuss what source of truth really means
• Show a service model for SDWAN
Scripting vs Intent

• **Scripting** is automation at its simplest level
• A **script** takes in some variables, it runs, and it performs the logic it is coded for
• An **intent** orientated system uses services to describe where you want the system to be, and the system will get itself to that state

Are these things really the same?  
- Take variables
- Do stuff
- Return
A little computer science theory

**Imperative**
- Specify how to do something

**Declarative**
- Specify what the end state should look like, not how to get there
- Using:
  - Data
  - Models
  - Functions
Real World Application

Let’s consider a simple example for managing customer VLANs (Virtual Local Area Networks).

• VLAN = network segmentation technique that divides a physical network into multiple logical networks, allowing devices in separate VLANs to communicate as if they were on different physical networks.

• It enhances network security and performance by isolating broadcast domains, controlling traffic flow, and enabling policy enforcement based on logical groupings rather than physical locations.
Simplified VLAN Service

- Need to construct VLANs that cross multiple devices, each device has one client port
- Each edge device has some number of trunk ports, which are directly connected to other edge devices
- To configure a VLAN, the client port needs to be configured, and the VLAN needs to be added to all necessary trunk ports
A little (applied) computer science theory

**Imperative**
- Create a VLAN
  - For each edge device
    - Add VLAN to client port
    - Add VLAN to applicable trunk ports
- Document
  - Which devices have which VLANs
  - Which VLANs are in use

**Declarative**
- State:
  - List of existing VLANs
  - Physical Map of network
- Intent
  - VLAN between ports
Problems, regardless

• What about after the VLAN is created?
• When should assurance checks be run?
• How to ensure we don’t end up using the same ID twice?
• How do we know if we got the ports right?
Model it!

- Model a database with:
  - Intended physical network
  - Shared resources
  - Intended VLANs

- Intent:
  - Each VLAN should exist and traffic should be able to flow between all connected ports
Now for the models

Service Model
- vlan
  - id
  - name
  - description
  - devices
    - name
    - client-port

Resource Model
- device
  - ip-address
  - type
  - client ports
  - trunk ports
- network
  - vlans
  - connections
    - device/port a
    - device/port z
VLAN from an API

```bash
curl -X POST https://api.network-device.com/v1/vlans \
-H "Content-Type: application/json" \
-H "Authorization: Bearer YOUR_API_TOKEN" \
-d '{
  "id": 100,
  "name": "Marketing_VLAN",
  "description": "VLAN for the marketing department",
  "devices": [
    {
      "name": "Switch1",
      "client": "MarketingDept",
      "port": "GigabitEthernet0/1"
    }
  ]
}'
```
So far...

We have learned
• The difference between imperative and declarative
• example model

Now what?
• Generic automation architecture
• Services
• Templates
• Workflows
• Sources of truths
• SDWAN Example
Generic Service Oriented Intent Driven Automation

- Users
- Other Services
- Workflows
  - Templates
  - Workflows
  - Scripts

- DB
- APIs
  - CRUD-V Logic
  - Devices
- API
What is a service? How does it relate to intent?

- API
- Database
- Hierarchical model
- Service model defines intended state, not process to get there
- Service is a noun, not:
  - CreateVlan service
  - DeviceRemove service

- Instead
  - vlan
  - device
- Services can be treated as resources themselves by other services
Templates

- Fundamentally declarative
  - Write what you want to see in terms of config
- May need a bridge between the declarative template and the device
- At least need some way to effectively run the template and apply its results to a device
VLAN Template

! Create the VLAN
vlan {{ id }}
  name {{ description }}
! Configure interface
interface {{ client_port }}
  description Interface for {{ description }}
  switchport mode access
  switchport access vlan {{ id }}
  no shutdown
!
end
Workflows

- Usually imperative, state machine, a verb (CreateVlan workflow)
- We say how to do it, not what is has to be
- Allows you to track and control multi-stage changes

- Can be useful to glue resources together, but is not itself a service
- Could be called by a service, could call services
Workflow example
Sources of Truths

- Physical state of network
  - Intended physical state of network
- Behavior of network
  - Configuration of network
  - Intended configuration of network
  - Intended network services
- History of mapping between all of the above
SDWAN

- SDWAN allows multiple customer sites to be connected across disparate links using a single overlay network
SDWAN Service

- A managed SDWAN service
- Want to have some way to provide default and easy to use QOS policies, as well as specific overrides for customer-specific applications
- Let’s look at how we can use a service to allow us to show intent
Architecture

Customer Portal

Customer Service

Resource Service

SD-WAN Provider
Traffic classification and QoS

```plaintext
---rw traffic-classification
  +=-rw policy* [id]
    +=-rw id string
    +=-rw rule* [id]
    +=-rw id string
    +=-rw src-ip? inet:ipv4-prefix
    +=-rw dest-ip? inet:ipv4-prefix
    +=-rw protocol? string
    +=-rw src-port? uint16
    +=-rw dest-port? uint16
    +=-rw dscp? uint8

---rw qos-policies
  +=-rw policy* [id]
    +=-rw id string
    +=-rw qos-class? string
    +=-rw bandwidth-percentage? decimal64
    +=-rw priority? uint8
    +=-rw traffic-classification-policy-id? -> /traffic-classification/policy/id

---rw qos-profiles
  +=-rw profile* [id]
    +=-rw id string
    +=-rw name? string
    +=-rw policy* [policy-id]
    +=-rw policy-id -> /qos-policies/policy/id
```

© 2024 Ductus
Example QOS Profile

```json
qos-profiles: {
    profile: [
        {
            id: healthcare-profile,
            name: Healthcare Provider Profile,
            policy: [
                {
                    policy-id: ehr-policy
                },
                {
                    policy-id: medical-imaging-policy
                },
                {
                    policy-id: telemedicine-policy
                }
            ]
        }
    ]
}
```
Example QOS Policy

```json
qos-policies: {
    policy: [
    {
        id: ehr-policy,
        qos-class: gold,
        bandwidth-percentage: 30.00,
        priority: 7,
        traffic-classification-policy-id: ehr-traffic-classification
    },
    ...
    }
}
```
Example QOS Traffic Classification

```
traffic-classification: {
  policy: [
    {
      id: ehr-traffic-classification,
      rule: [
        {
          id: ehr-rule1,
          protocol: TCP,
          dest-port: 443,
          dscp: 34
        },
        {
          id: ehr-rule2,
          protocol: TCP,
          dest-port: 8443,
          dscp: 34
        }
      ]
    },
    {
      id: medical-imaging-traffic-classification,
      rule: [
      ]
    }
  ]
}```
SDWAN Service
Adding QOS Profile to a Customer

```
sdwan-service: {
  customer: [
    {
      customer-id: customer1,
      customer-name: Healthcare Inc.,
      vpn: {
        qos: {
          profile: healthcare-profile,
          policy: [
            {
              id: additional-policy1,
              policy-id: critical-apps-policy,
              traffic-classification: {
                rule: [
                  {
                    id: ca-rule3,
                    protocol: TCP,
                    dest-port: 8080,
                    dscp: 46
                  }
                ]
              }
            },
            {
              id: additional-policy2,
            }
          ]
        }
      }
    }
  ]
}
```
Now what?

• Provide NETCONF or gNMIc interface based on model
• Create a webui and allow customers to manage their own settings based on the model
• Add telemetry support to model to enable customer-visible graphs of performance
• Integrate billing
So what?

- Customer wants to change, they can change their connections in minutes, not days waiting for change-request
- Customer can easily see what services they have
- Provider knows what the intent of their customers is, can connect that intent to reality (telemetry), and plan for capacity improvements
- Reduced risk for human error
Final Summary

• Scripting and intent are not in conflict, but they are not the same
• Imperative says how, declarative says what
• Resource hiding is the key to scalability
Q&A

• In what areas of network management do you see potential advantages of adopting an intent-driven approach?

• What hurdles or complexities might we anticipate when transitioning to an intent-driven automation model?

• What factors would influence your decision to move towards or away from an intent-driven automation strategy?
Thank you

James Henderson
Automation Solutions Architect