Automation in a Service Provider Brownfield Network

A technical paper prepared for presentation at SCTE TechExpo24

Mark Kayser
Sr. Communications/Network Engineer
Cox Communications
mark.kayser@cox.com

Table of Contents

List of Figures

List of Tables

1. Introduction

Service Providers face ever increasing levels of complexity in their networks to accommodate more advanced services for their customers. The management of once fairly simple networks with just a few hundred or maybe even a few thousand routers and switches has become a real challenge in complex networks with tens of thousands of network devices.

Configuration errors and security are also major factors in network reliability. In the "Annual outages analysis 2023" from Uptime Institute (Lawrence & Simon, 2023), the leading cause of network outages was configuration / change management failures and many of those were due to human error. Making configuration changes to thousands of devices manually is highly inefficient, taking a large commitment of man-hours to complete. Network Automation is really the only way to perform this kind of change activity. In addition to increased efficiency, automation removes direct human interaction with the network devices thus reducing the chance for human error.

Another significant cause of configuration errors is deviation from the standard (aka golden) config. Even with automation, if a device has a configuration that is different than what is intended based on company standards, it can lead to a configuration change that causes an unexpected impact on the device. A configuration compliance solution is important to report any deviations.

In this paper, we consider four functional areas needed to create a complete automation solution:

- Network Design – to create the intended network configuration
- Configuration Deployment – to apply changes to network device configurations
- Network Status – to provide live network status and an inventory of active network devices
- Configuration Compliance – to verify actual device configurations vs the intended configurations

Most vendors offer automation solutions, but not all these solutions support legacy hardware or legacy firmware. This paper will look at an approach to creating a complete automation solution from a mixture of open source solutions and in-house developed tools, with connectivity to existing vendor tools that provides the flexibility to be customized for whatever equipment makes up the network.

2. Issues Configuring Large Complex Networks

In the early days of Multiple Systems Operator (MSO) data networks, the number of routers and switches was pretty small in relation to the number of customers. Usually, some cable modem termination systems (CMTS) with upstream aggregation devices and backbone routers to connect the regional aggregation networks together was sufficient. This would usually amount to a few hundred devices depending on the size of the provider. In those days the data network was typically just used to provide Internet service via cable modems. There usually weren't many changes required to the network except expanding the number of supported cable modems (new IP addresses, new card configurations, new CMTS upstream connections). These simpler networks could easily be managed with manual configuration of the network devices with maybe some configuration spreadsheet tools. There were enterprising engineers that would use some scripts (Expect, Perl, Python, etc.) to do the configuration.

Then came additional services, like business data services and voice. These services introduced new customer premise access equipment like routers, switches or integrated access devices (IADs). In addition, head-end or data center equipment was needed to enable these services as well as more advanced configurations like quality of service (QoS), multi-protocol access control lists (ACLs), and more advanced routing (route-reflectors or confederations in BGP). The number of network devices started reaching several hundred to a few thousand. The growth rates were usually only a few new devices a day in a given region, but more advanced spreadsheet or configuration generation tools were needed for initial device configurations.

Today's data networks support multiple types of services such as Internet, voice, video, business customer metro ethernet, SD-WAN, cloud services and more. The number of devices requiring configuration is now in the tens of thousands or higher. Additionally, the types of devices needed to support these service types aren't your typical old routers and switches; specialized hardware is required for several of them. Even the equipment used to aggregate cable modems isn't simply an old school CMTS, it is now an R-PHY device with several layers of aggregation switches and routers. Business customer networks are highly transactional, adding hundreds of devices a day in some service provider networks. With all the complexity added by the multitude of new service types and the volume of devices being added to the network, it is extremely difficult to configure new devices and services manually with just spreadsheet configurators. Each time a new feature is added to a service, or a new service is introduced, a large amount of configuration changes to the supporting network can be required. Even just adding a new service to an existing customer often requires changes to the transport network. This quantity of configuration work presents challenges.

2.1. Change Management Man-hours

The most obvious of these challenges is time. Depending on the scope of the configuration change, all the devices of a given type may have to be updated. In this example, just consider adding one configuration change to 10,000 devices. This change activity may look like this if done manually for a potentially service impacting update (All numbers used are rough approximations based on personally performing these kinds of tasks):

Table 1 – Time to Complete Large Network Maintenances

Maintenance Activities	**Scope**	**Time (in minutes)**
Create change management tickets	10 minutes per ticket for 25 regions	250
Pre-maintenance status checks	1 minute per device	10,000
Apply configuration changes	1 line of code, 10 seconds per device	1667
Post-maintenance status checks	1 minute per device	10,000
Analyze pre and post status checks	15 minutes per maintenance window*	1380
	Total Minutes	23,297
	Total Man-hours (rounded off)	**388 (hours)**

* See Below:

1) We will assume a 6 hour maintenance window (midnight to 6AM).
2) Time needs to be allowed for fixing any issues that occur or backing out the change, so on the safe side 4 hours per maintenance window to complete the work. (**NOTE:** We will include the 15 minutes for analyzing pre and post checks from the 2 hours left for corrective action)
3) Adding up the first four rows of the table gives us ***21,917 minutes*** or ***366 hours*** (rounded up)
4) The number of maintenance windows using 4 hours per window comes out to 92 (rounded up)
5) Using those 92 maintenance windows for the "Analyze pre and post status checks" activity we get ***1380 minutes*** of work added.

Now because networks have been growing over time most network operators have come up with some form of automation to try and reduce these numbers. This example was just used to show that trying to do manual configuration on a large network really isn't practical.

2.2. Network Outages Due to Human Error

Another major challenge with performing manual changes is the potential for configuration errors. As mentioned in the Introduction, a 2023 report by Uptime Institute stated the number one cause of network outages was configuration / change management failures. The report further broke down outages to human error, or not. It found that 39% were human error, 51% were not, and 11% unknown. Correlating between the two metrics indicates that network outages due to human error is significant. My experience in network operations has proven this to be true. Whether it was a cut and paste error, not properly checking the current status of the network before applying a change, not following standards when preparing the maintenance configurations, and the list goes on. I've been guilty of some of these errors myself.

2.3. Problems Created by Non-standard Device Configurations

In a similar vein as human error, non-standard device configurations can cause a number of problems. Service Provider networks and the departments that support those networks often change over time. To quote the translated words of Greek philosopher Heraclitus, "The only constant in life is change." He must have been a network engineer too. The technology used to build networks changes frequently. There are also company priority changes and organization changes aimed at optimizing productivity. Many times different regions of a company will have different ways of configuring devices, especially if the service provider was structured in a decentralized manner at some point. Not to mention mergers and acquisitions of companies, which can lead to integrating the two companies' networks. It would be great if after each such change the network could be rebuilt from scratch with the latest technology and the new network standards. Unfortunately, reality gets in the way. The network must keep supporting the current customer bases, new equipment takes lots of time and money to roll-out, and these changes don't usually come with a bucket of new resources.

In larger networks it can be a major task to try combining all these changes into a single standard. It doesn't happen overnight. It's quite possible different parts of the network use equipment from different vendors, especially when a merger was involved. From what I've seen, it usually takes until the next technology change cycle for these disparate networks to be unified into a single network standard. Even after unification, it takes time to migrate every service off the legacy networks. Rolling out a service that touches both the new and legacy networks definitely complicates the process. This makes it important to have each part of the network configured to set standards. There may be different standards for each, but ideally these standards will align as closely as possible. Non-standard device configuration can lead to security issues that were addressed as part of the standards. It just takes one opening to expose a network to attack. Additionally, deploying configuration changes to the network with the assumption that all devices are currently configured to the standard can lead to an outage. Consider adding a device to a layer 2 network with a specific loop prevention strategy where one of the devices isn't configured to implement that strategy or inserting an access policy statement to a policy that isn't using the standard entry order. This can lead to unexpected issues.

3. Network Automation Can Mitigate Configuration Issues

These challenges presented by modern service provider networks can seem overwhelming, but a good automation strategy can help. It isn't the cure-all to network problems, but with good processes and proper testing of the automation system; these kinds of errors can be greatly reduced. The idea is to reduce the number of times someone must copy and modify the updated configuration before the changes are applied to the devices. Fewer touchpoints mean fewer opportunities for mistakes. This does not imply that the normal groups that would have previously done the configuration work would be removed from the process. In fact, these groups become more important in reviewing the configurations and making sure the changes are thoroughly tested prior to allowing the automation to send the updates to the network. As my colleagues and I often discuss, one of the downsides of network automation is if done poorly it can break the network faster than a human doing manual configuration. The more eyes on the overall process before using automation to make changes the better.

3.1. Building a Complete Network Automation Solution

An effective automation solution will consist of multiple components, each doing their specific task. The solution should aim to not duplicate effort in these components and the parts should provide checks against each other.

The concepts we are going to discuss here are just one approach to network automation. There is no single "right way" to do it. This is an area that is quickly advancing and there are lots of off the shelf solutions. Also, most network equipment vendors have solutions of their own. This specific solution tries to take legacy network devices into account along with the newest equipment. As with all engineering projects, there are trade-offs to different approaches. The in-house development one discussed requires developer resources and a strong partnership between the developers and the network subject matter experts. There isn't always an external company to lean on if problems with the platform arise. On the plus side, the solution is highly tailored to the service provider's network. The idea is to use open source software and in-house developed applications whenever possible to tie together existing back office and off the shelf systems. Approaching the solution in this way allows changes to the solution to be implemented without external vendor negotiations on cost and deliverables for the in-house developed parts. Looking at this approach from a high level, four major areas are considered: Network Design, Configuration Deployment, Network Status, and Configuration Compliance.

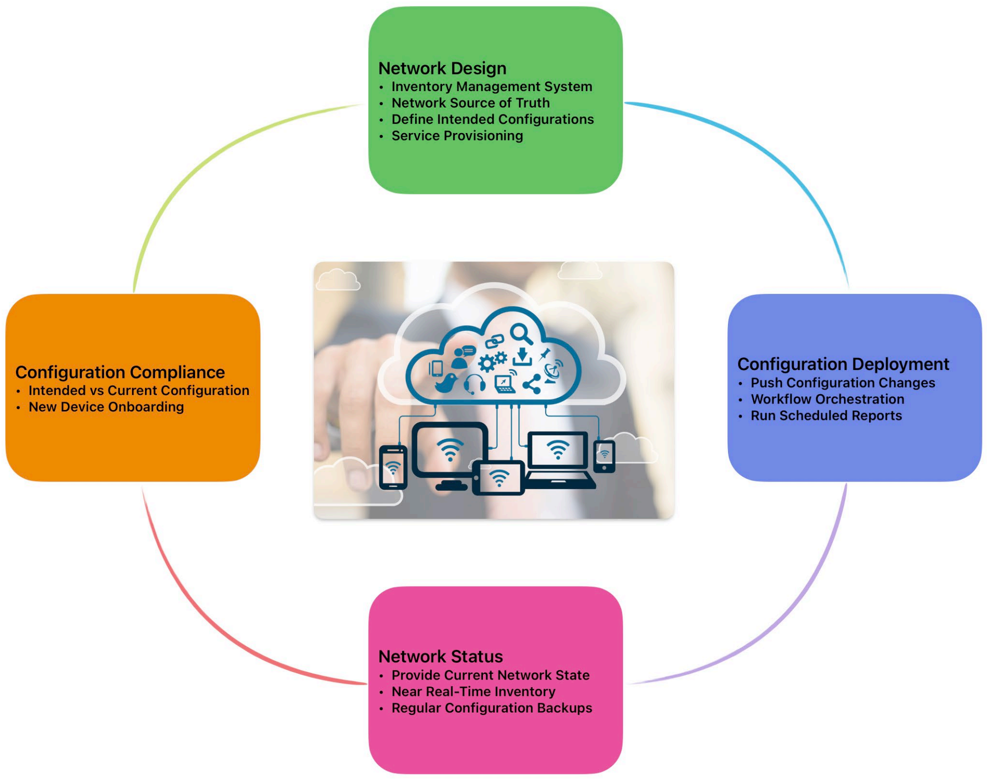
Network Design
• Inventory Management System
• Network Source of Truth
• Define Intended Configurations
• Service Provisioning
Configuration Compliance
• Intended vs Current Configuration
• New Device Onboarding
Configuration Deployment
• Push Configuration Changes
• Workflow Orchestration
• Run Scheduled Reports
Network Status
• Provide Current Network State
• Near Real-Time Inventory
• Regular Configuration Backups

Figure 1 – Diagram of Complete Network Automation Solution

"Devices in the Cloud - Technology" by perspec_photo88 is licensed under CC BY-SA 2.0.

3.1.1. Network Design

The role of network design is to define how the network should be built. It creates the intent that will be used to configure all the network elements. One part of the network design solution should be the Source of Truth (SoT) for the network, including a complete inventory of devices, both active and planned. For the design information to be highly useful in this system, it needs to contain all the data needed to create an intended configuration for each network device. In our approach, the Network Design tools does not interact directly with the network devices. If information is needed from the network, it is provided by either Network Status from stored information, or Configuration Deployment via scripts that gather the specific data on request.

Another important component of network design is an IP Address Management (IPAM) tool. I point this out specifically as it may be part of the Source of Truth (SoT) or a separate application. Several tools on the market either are a purpose built IPAM or contain one as part of a larger solution. Ideally any IP address information used in the SoT will be programmatically imported from the IPAM without need to cut and paste (or swivel chair) between the tools.

It may be beneficial to have separate tools for parts of the configuration that are highly transactional in nature or contain a customer facing component. A good example of this would be a service provisioning tool. Each device in the network is likely to have a completely different set of services configured. The data required for each type of service is also heavily variable and it is advantageous to have a data entry frontend that can be customized for the differences.

3.1.2. Configuration Deployment

This solution element represents actually getting the configuration changes to the network elements. Workflow orchestration is very useful for this area as the changes being made will probably have multiple stages. Moving from one stage to the next will likely be gated based on success or failure of the previous stage(s). External resources like the Network Status can be used to provide information as part of the process, like the current state of the network being targeted for change or even the inventory of the elements to update. The SoT from Network Design is another potential source for inventory information. Connectivity to the production data network is required for Configuration Deployment, so it is one of the two parts of this solution that provides connectivity. Some industry security certifications like NIST and SOC 2 require production network connectivity to be limited. By providing network connectivity to other parts of the automation solution this helps fill those security requirements.

The reason not to make this the only connectivity option is that Network Status has very heavy network usage requirements and funneling all communication through the Configuration Deployment solution would reduce performance of other tasks being performed by the parts of the Configuration Deployment solution. That doesn't mean it can't be used to get some network status information, like running information gathering tasks as part of a targeted reporting solution for data that isn't tracked by Network Status. Some good resources to include in developing an in-house Configuration Deployment system are Ansible, FastAPI, or any open source API framework written in the language your development team prefers. In fact, it may be composed of multiple such solutions as each have their strengths in different situations.

3.1.3. Network Status

As discussed in Configuration Deployment, Network Status is the other part of this solution that provides direct network access to the production network devices. It is used to poll and collect data about the state of the network. You may be thinking this is just a Network Management System (NMS) and part of it can be, but most NMSs provide a specific set of data, and only monitor certain aspects of the network. Based on the types of equipment in the network and the services offered, a traditional NMS probably doesn't gather state information or statistics for the unique aspects of the network. There are open source monitoring solutions that can be extended, but they often can be very heavy applications with a steep learning curve to be able to effectively modify them. These can offer the basis of a strong Network Status solution if you are willing to invest the time in getting familiar with these NMSs. Another good approach is to create a series of purpose-built monitoring applications, targeted at your specific needs. With a good data storage solution and a strong API framework to expose the data in the way your environment can best use it, this offers the highest level of customization. There is nothing preventing combining the two approaches if each have enough value to make the effort worthwhile.

Be careful not to add too much data polling to a single application unless you have a clear understanding of how it scales. This can quickly lead to performance issues with the pollers, and data being missed if the pollers can't keep up. The data needs to be gathered at a set interval for it to be meaningful for trending.

Another gotcha aspect that Network Status components can face is good data storage solutions. If the storage runs out this also causes loss of data. An effective storage solution for these large amounts of data can aggregate the data at fixed intervals (week, month, quarter, etc.) so that trending can be maintained but granularity in the older data is lost. Aggregate data is better than losing the data, so make sure you are monitoring the data usage as part of your overall management strategy.

A very nice feature to have in any monitoring solution is the ability to on-demand poll specific parts of the network. The standard polling interval may be 15 minutes or more and may take most of that 15 minutes to complete based on the scope of what is being polled. That means you could wait up to 30 minutes before you see if the status has changed. When doing maintenance on a device or tracking an outage you may want to check the status much sooner. Obviously, you can on demand poll the entire network, or why wouldn't you do that constantly. But you could poll a device or series of devices on demand as the situation calls for it.

3.1.4. Configuration Compliance

Probably the hardest part of this solution is configuration compliance, especially if you want to do compliance for the entire configuration of each device. To do a complete configuration compliance check that verifies what is missing as well as any extra configuration lines, you need to be able to fully templatize the standard configuration for each device type and device role. Not to mention that configurations often have syntax changes between versions of device firmware. This means the templates must account for these differences, since it is very likely that the network will have both versions running at the same time during upgrade cycles. Having all these templates is only the first part, next you must have the variables to fill-out the templates stored for all devices. With those two parts you can generate an intended configuration for each device that can be used to compare with its current configuration. It is not very feasible to get the configuration off each device at the time of compliance checks. This would require logging into each device to get the running configuration. In a network with 30,000 plus devices this would take several hours. It is reasonable to expect a single router could be checked in an on-demand use case this way, but to do daily compliance checks of the entire network this isn't the best approach.

Every service provider should be backing up their network equipment configurations daily. This allows for rapid restoration in the event there is a complete failure of the equipment. An efficient system would note the last time the device configuration was changed and only download a new backup if that change timestamp is different than the currently saved one. This would reduce the bandwidth load on the network when the backup process is running and should also be significantly faster since only changed devices are initiating a backup. Since we've stated that only Network Status and Configuration Deployment should have direct network access, one of those solutions needs to perform these backups. The choice of which solution to choose really depends on the individual components that make up each solution and if one of those components is best suited for the task.

With regular backups already being stored, Configuration Compliance can more efficiently run compliance checks against these backups by comparing the backup to the intended configuration it generates. Differences between the backup and intended configurations should be reported for review. Some systems have mechanisms to initiate configuration remediation. While a nice concept, this can be dangerous. There may be a change that was made to correct an issue, but the team responsible for maintaining the configuration standards may not have had an opportunity to see if this change is something that needs to be incorporated into the standard. Another case is when a new feature is being tested in a limited subset of the network before being released as part of the configuration standards. In both cases automatic remediation would cause issues. A good solution to this would be to allow these types of situations to be noted in the Configuration Compliance tool so that everyone is aware why there is a variance and why they should not try to correct it.

3.1.5. Integrating the Tools

The key to making this a holistic solution is the ability to integrate all the parts. It is important that the solutions selected or built in-house have APIs. The most common API design styles (Gavrilenko, 2023):

- REST (Representational State Transfer)
- GraphQL
- WebSocket
- Webhook
- RPC (Remote Procedure Call)
- SOAP (Simple Object Access Protocol)

These APIs are fairly common in open source and vendor-based solutions alike, which will help greatly when trying to get all of the components to work together. The following are a few examples where having these parts work together can be advantageous.

3.1.5.1. Checking Network Status During Network Design

Let's look at a case where an engineer is looking to add a new router at a customer premise for Internet access. The customer has requested a /27 prefix. The engineer checks the IP Address Management (IPAM) tool and is given 172.20.145.0/27 as the next available prefix for that area. What the IPAM didn't show was that prefix was assigned to a customer that was recently disconnected, but during the disconnect process the provisioner forgot to remove the static route from the router service the old customer. Hopefully the provisioner of the new router will check before configuring that prefix on the equipment, but, if not, now that prefix is routed in two places which can cause all kinds of unusual traffic problems for the new customer. Integrated automation tools could have helped in this case in a few ways. The Network Status tools could provide routed prefix data to the IPAM (part of Network Design) on a regular schedule. Proactively the IPAM could use that data to mark prefixes unavailable even if it was manually set as returned from a previous service account. Also, the IPAM could make an active request to Network Status checking the availability of a prefix before allowing it to be assigned.

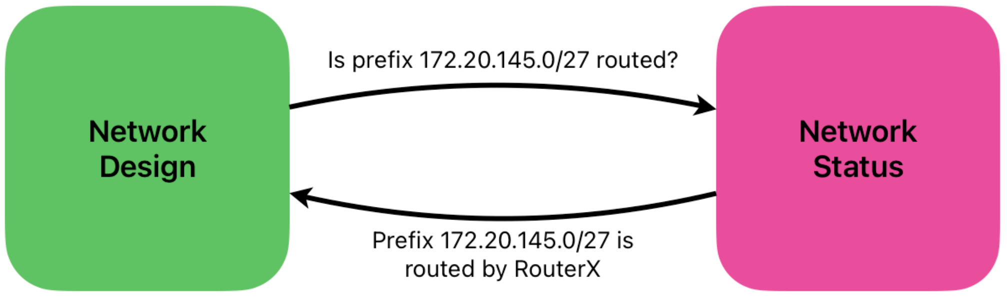
Network Design
Is prefix 172.20.145.0/27 routed?
Network Status
Prefix 172.20.145.0/27 is routed by RouterX

Figure 2 – Check Prefix Availability Prior to Assignment

This integration would also allow the IPAM to run reports on assigned prefixes on a scheduled basis, noting if an assigned prefix is not routed and when it was last seen as routed. Having that kind of information is very useful when trying to check on overall prefix availability, especially when it comes to IPv4 address space.

3.1.5.2. Checking Network Status Before Configuration Deployment

I've unfortunately seen many times where an engineer or a technician makes a network change, only to cause an outage they didn't expect. They follow the procedures that tell them to check the status of the device prior making the change. All the checks show clean on that router, they go to make the change, but an outage occurs because there was an issue somewhere else on the network. Hopefully the procedures are detailed enough to cover checking most possible issues and hopefully the technician is experienced enough to detect an issue prior to making the change, but that's not always the case.

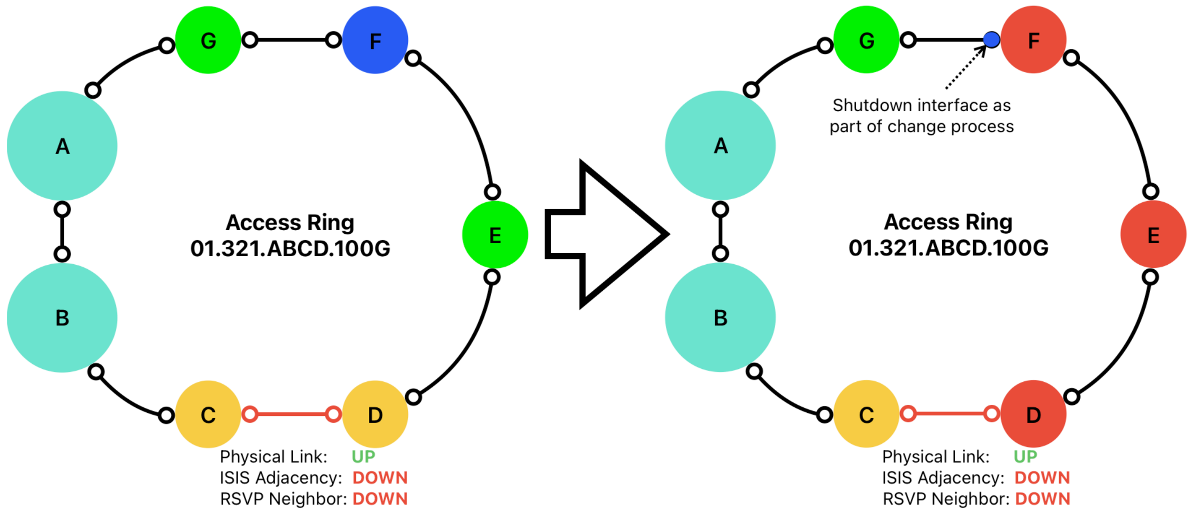

Figure 3 – Network Change Created Outage

In this example, a Network Management System (NMS) should see this issue, but many times the link status is based on just physical connectivity. Higher level protocol issues are just displayed as entries in the NMS log, so even if the technician did a quick check of the NMS, they possibly wouldn't have noticed it.

With network automation, this update could have gone differently. The technician would use the Configuration Deployment tool, to initiate the change. Part of the deployment script would be to query the Network Status tool(s) for a status of the Access Ring that the device being worked on is a member. Only after getting a positive confirmation that the ring is good, would it proceed to doing the local checks on the router followed by applying the configuration update. If a negative response was received from Network Status the change activity would be stopped before attempting the change.

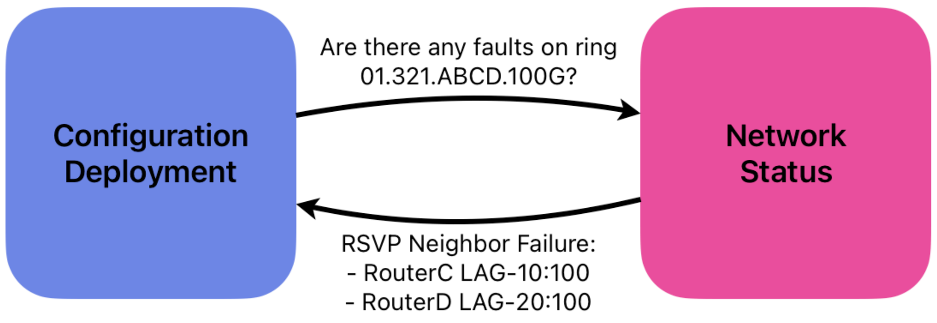

Figure 4 – Check with Network Status Before Change

There may be checks that weren't thought of which could lead to issues even using automation, however, once discovered they can be added to the scripts preventing future failures. Everyone who uses the Configuration Deployment tool will benefit from the update not only for this change but for any similar ones as well.

3.1.5.3. New Device Onboarding Process

On a happier note, integrating the automation tools can make processes run more efficiently, not just prevent issues. We are looking at a new device onboarding process. With respect to onboarding, we are talking about the process of a new device comes online and is checked to make sure everything is as is should be before considering it "In-Service".

As part of Network Design, we define what type of device is to be used and fill out the configuration variables. This information is used to generate the initial configuration that is staged onto the device prior to being installed. Once the device is put into a "Pending" state by Network Design, the tool will send notifications via API calls to Configuration Compliance detailing the intended configuration. At the same time a notification will be sent to Network Status, adding the device's management IP address to a watcher process that frequently scans the network for all devices on the watcher list. When the management IP address is reachable, the Network Status tool will perform an initial scan to gather all the tracked parameters. It will back-up the current configuration to the network back-up repository and notify, via another API call, Configuration Compliance to run a compliance check. This is to make sure no changes were made to the standards or the device parameters from the time it was staged to the time it comes online. Configuration Compliance will notify the workflow engine in Configuration Deployment of the status of the compliance check. If the check finds that the configuration does not match intended, or the firmware is not the latest version, Configuration Deployment will initiate scripts to make corrections. Once the corrections are made or if there were no issues reported by the compliance check, a notification is sent to Network Design to indicate the new device is now “In-Service”. Network Design will update all the appropriate status fields to the “In-Service” state.

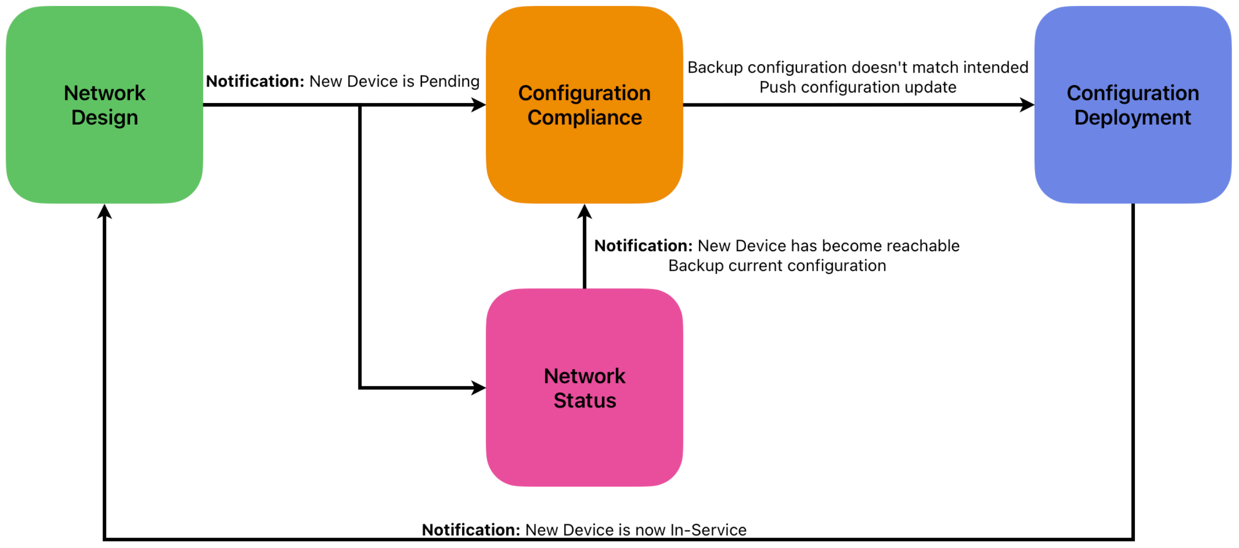
Network Design
Notification: New Device is Pending
Configuration Compliance
Backup configuration doesn't match intended
Push configuration update
Configuration Deployment
Notification: New Device has become reachable
Backup current configuration
Network Status
Notification: New Device is now In-Service

Figure 5 – New Device Onboarding Process

These are just a few examples of how getting the parts working together will make a much stronger system. I’m sure everyone can imagine even more possibilities. Having a flexible system will help make those possibilities into reality.

4. Conclusion

Over the course of this document, we have looked at some of the challenges service providers face operating large complex networks with tens of thousands of network devices. Making configuration changes manually for large scale changes is no longer feasible. We also explored how using network automation to do configuration changes can reduce errors that can cause outages. Another potential problem is having devices running with non-standard configurations. Those devices could provide bad actors a vector to attack the network, or the deviations from the standard config could create problems when rolling out new features on the network that are based on the current standards.

There are solutions on the market to address these problems individually, but we saw how creating a complete network automation solution can handle these problems more efficiently. The pillars of this complete solution are Network Design, Configuration Deployment, Network Status, and Configuration Compliance. Each of these can be composed of multiple tools ranging from commercially available solutions to open source ones mixed with in-house developed applications and integrations. How these solutions are built is very dependent on what tools already exist and can be incorporated with a more wholistic approach to automation. Having resources to include in-house development only strengthens the solution by allowing the solution to be more tailored to the specific needs of the given network.

Abbreviations

ACL	access control list
API	application programming interface
BGP	border gateway protocol
CMTS	cable modem termination system
IAD	integrated access device
IPAM	IP address management
MSO	multiple system operator
NIST	National Institute of Standards and Technology
NMS	network management system
QoS	quality of service
R-PHY	remote PHY
REST	representational state transfer
RPC	remote procedure call
SCTE	Society of Cable Telecommunications Engineers
SD-WAN	software-defined wide area network
SOAP	simple object access protocol
SOC 2	Service Organization Control 2
SoT	source of truth

Bibliography & References

Gavrilenko, A. (2023, October 24). *The System Design Cheat Sheet: API Styles - REST, GraphQL, WebSocket, Webhook, RPC/gRPC, SOAP*. Retrieved from Hackernoon: https://hackernoon.com/the-system-design-cheat-sheet-api-styles-rest-graphql-websocket-webhook-rpcgrpc-soap

Lawrence, A., & Simon, L. (2023, March 20). *Annual Outage Analysis 2023.* Retrieved from Uptime Institue: https://uptimeinstitute.com/resources/research-and-reports/annual-outage-analysis-2023