Automated Configuration Management

David Schmidt, Vijay Gill
Global Networking Services Team, Global Foundation Services
Microsoft Corporation

Microsoft

Why the need for Configuration Management Automation?

- As a Network Scales:
  - So does complexity
  - Manual configuration becomes impractical and unreliable
  - Increasing number of devices and roles
  - Networking staff become specialized in a limited space
- Automation forces consistency
- Consistency reduces complexity
- Humans make mistakes or may choose to not follow rules

Network at Microsoft

- Networking Environment at Microsoft:
  - Multiple hardware vendors
  - 77 device roles
  - Tens of thousands of network devices
  - Over 30 million lines of configurations
- This scale would not be possible without automation.
- Most tools and process developed for deployments or monitoring.

Why is Service Management important?

- Improves reliability of our services and avoids data loss
  - Great for users!
- Reduces cost
  - Allows us to scale headcount sub-linearly with network growth
- Improves time to market
  - Better customer experience and faster turnaround

Introducing Standard Enforcement Scripts

- Service Management system used by Microsoft's Global Networking Services
- Initial deployment
  - Core and backend networks
- Inspired by others in Microsoft
  - Windows Azure Fabric and Autopilot
- Follows "Lights Out" management principles

Improvements we've seen with SES

	Before	After
Router Rollout	2-3 days	30 seconds
	2-3 engineers	1 engineer
	10-20% Errors	0%
Bill of Wrongs	~4-6/mon	~0/month
Change Velocity	~50 a month	~1000/month

Principles: deployment

- Deployment is declarative
  - Describe the desired state and SES will figure out how to get there
- Everything on the Network should come through SES
  - No manual changes or other systems
- There should never be any end-user downtime during deployments

Principles: repair

- Bake repair into the application design
  - Hardware failure is guaranteed and should never require urgent human intervention
- Repair should remove urgency from issues
  - You shouldn’t have to completely understand all issues to keep the system running
  - Reporting is key
- Keep repair simple – use as few hammers as possible
  - Example: minimal set of transformations to bring system into compliance with design rule

Principles: alarming

- Alarming is required in some scenarios
  - Drive for non-urgent reporting when possible
- The same instrumentation and brain should be used for alarming and repair
  - Triggers for alarm are aware of repairs
  - Can move to repair in some cases

Principles: development lifecycle

- Management is baked into the software development lifecycle (SDLC)
  - All assets live in the source control system
  - All environments from dev box, through labs to production are deployed, monitored and repaired in the same way
- Code and test deployment, repair, alarming as part of coding cycle

Principles: safety

- SES provides a safety net
  - Human error should not be able to affect site availability
  - The system won’t let buggy code go out
  - The system won’t allow an operator to harm the system

Principles: standardization

- Standardize on small list of SKUs, network configurations, etc
  - Hardware is fungible
- Don’t micro-optimize

Standard Enforcement Scripts (SES)

Motivation

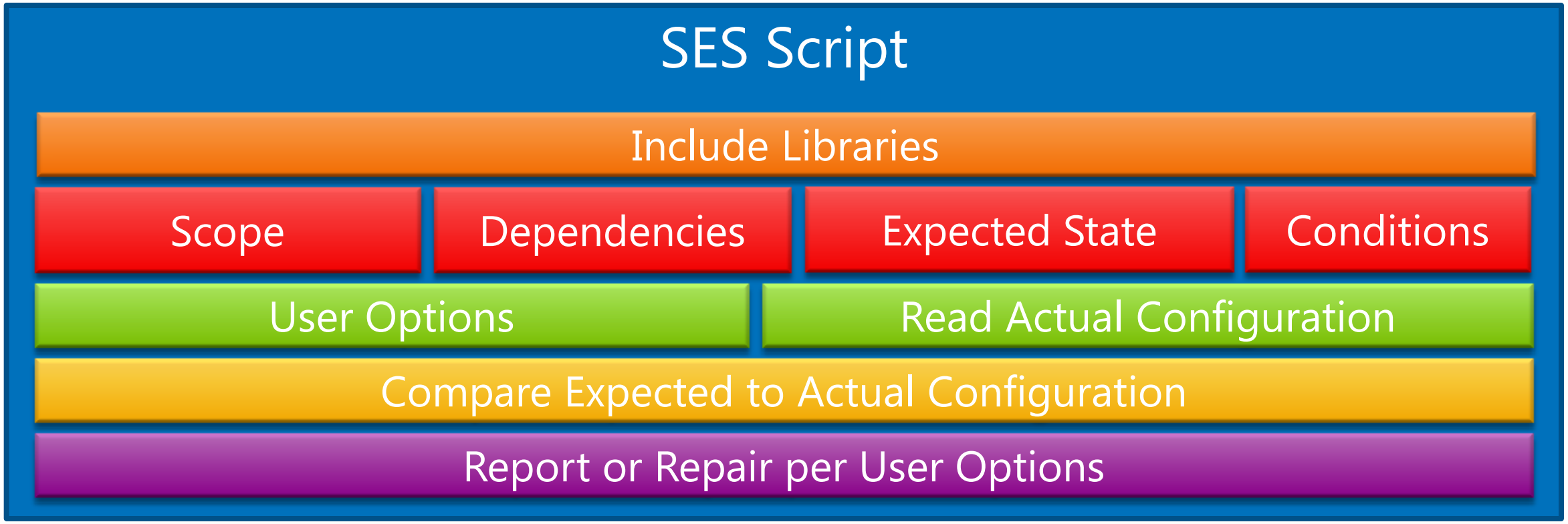
- Minimize Cruft
- Design Updates
- Post-Maintenance Verification
- Operational Simplicity
  - Limit Variations
- Deployment Efficiency/Accuracy

Approach

- Device configurations are a series of configlets
- Comparing a configlet against a standard is easy
- When they do not match; fix it

```
set system name-server a.b.c.d
set system tacplus-server w.x.y.z
set system host-name router
set firewall policer icmp-policer if-exceeding bandwidth-limit 1m
set firewall policer icmp-policer if-exceeding burst-size-limit 10k
set firewall policer icmp-policer then discard
set system syslog archive size 1m
set system syslog archive files 20
set system syslog user * any emergency
set system syslog host c.d.e.f any info
set system syslog host c.d.e.f kernel notice
set system syslog host c.d.e.f facility-override local1
set system syslog host c.d.e.f explicit-priority
```

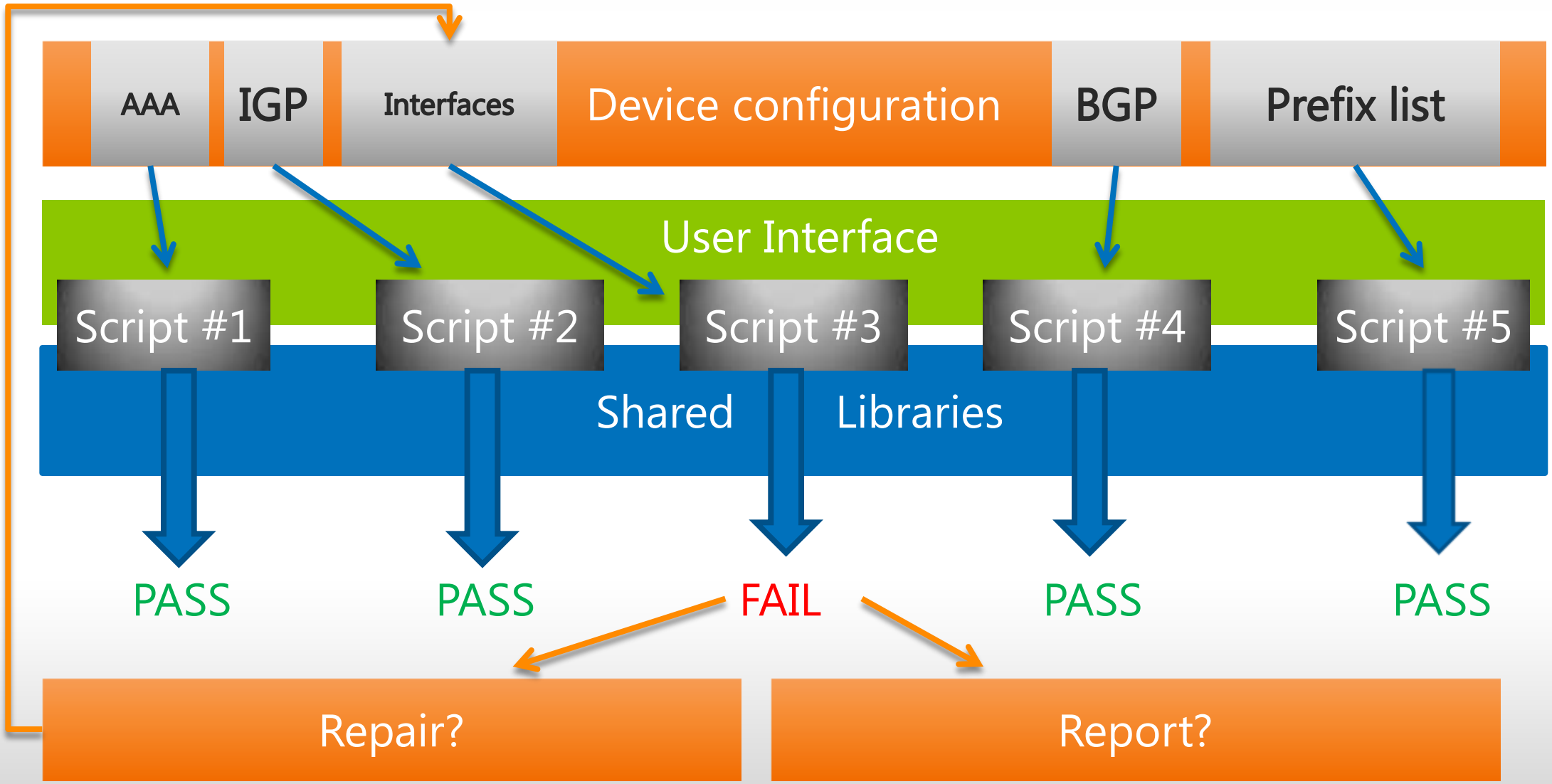
Logical Design



Libraries

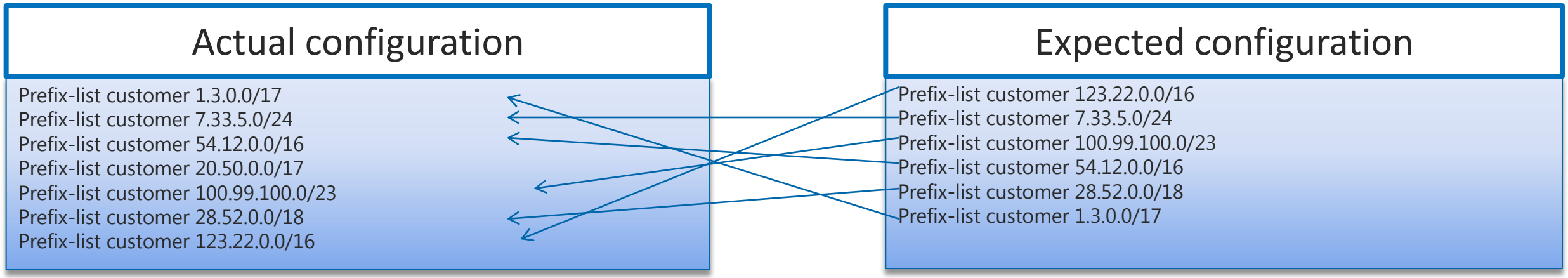
- Functions used by all scripts
- Keeps the scripts lightweight

Work flow


AAA
IGP
Interfaces
Device configuration
BGP
Prefix list
User Interface
Script #1
Script #2
Script #3
Script #4
Script #5
Shared Libraries
PASS
PASS
FAIL
PASS
PASS
Repair?
Report?

Condition Examples

RULE:
LOOSE MATCH



PASS

RULE:
STRICT MATCH

Actual configuration	
Prefix-list customer 1.3.0.0/17	√
Prefix-list customer 7.33.5.0/24	√
Prefix-list customer 54.12.0.0/16	√
Prefix-list customer 20.50.0.0/17	✖
Prefix-list customer 100.99.100.0/23	√
Prefix-list customer 28.52.0.0/18	√
Prefix-list customer 133.22.0.0/16	√

Expected configuration
Prefix-list customer 133.22.0.0/16
Prefix-list customer 7.33.5.0/24
Prefix-list customer 100.99.100.0/23
Prefix-list customer 54.12.0.0/16
Prefix-list customer 28.52.0.0/18
Prefix-list customer 1.3.0.0/17

FAIL

Remove:
Prefix-list customer 20.50.0.0/17

RULE:
STRICT ORDER MATCH

Actual configuration	Expected configuration
Prefix-list customer 1.3.0.0/17 Prefix-list customer 7.33.5.0/24 Prefix-list customer 54.12.0.0/16 Prefix-list customer 200.50.0.0/17 Prefix-list customer 100.99.100.0/23 Prefix-list customer 28.52.0.0/18 Prefix-list customer 133.22.0.0/16	Prefix-list customer 133.22.0.0/16 Prefix-list customer 7.33.5.0/24 Prefix-list customer 100.99.100.0/23 Prefix-list customer 54.12.0.0/16 Prefix-list customer 28.52.0.0/18 Prefix-list customer 1.3.0.0/17

FAIL

Remove:
Prefix-list customer
Add:
Prefix-list customer 133.22.0.0/16
Prefix-list customer 7.33.5.0/24
Prefix-list customer 100.99.100.0/23
Prefix-list customer 54.12.0.0/16
Prefix-list customer 28.52.0.0/18
Prefix-list customer 1.3.0.0/17

Dependencies

- Multi-part configurations separated into different scripts.
- Example:
  - Script 1: peer group
    - Script 2: BGP policy
      - Script 3: prefix-list
      - Script 4: as-path

Initial deployment

- Initially deployed to audit 4 device roles
- 230 scripts created
  - average 60 lines of code each
- 4000+ variations were discovered.
  - Variations ranged from cosmetic to situations that could contribute to service impacting events

Challenges

- Vendors
  - Different configuration formats
  - Behavior / Functionality
- Initial Cleanup
  - Standards can conflict with the current operating state
- Script conflicts
  - 2 or more scripts with different outcomes for the same configuration
- Script Coverage
  - Must account for all configurations to achieve maximum benefit

Other Use Cases

- Config Generator
  - By combining the output of all the template scripts, it is possible to create a base configuration for a new device
- Framework leveraged for incremental changes

Questions?

Thank you

- Vgill@microsoft.com
- Dschmidt@microsoft.com

Microsoft®

© 2012 Microsoft Corporation. All rights reserved. Microsoft, Windows, Windows Vista and other product names are or may be registered trademarks and/or trademarks in the U.S. and/or other countries.
The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation.
MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.