
BLOX FEST
Infoblox

Automating Network Change & Enforcing Configuration Policies: NetMRI Examples & Best Practices

Lou Nardo - Director, Product Marketing, Infoblox

Marty Adkins – Co-Founder Netcraftsmen

Greg Whitehead – Sr. Network Engineer, M&T Bank

BLOX FEST

Agenda

This is about customers talking with each other

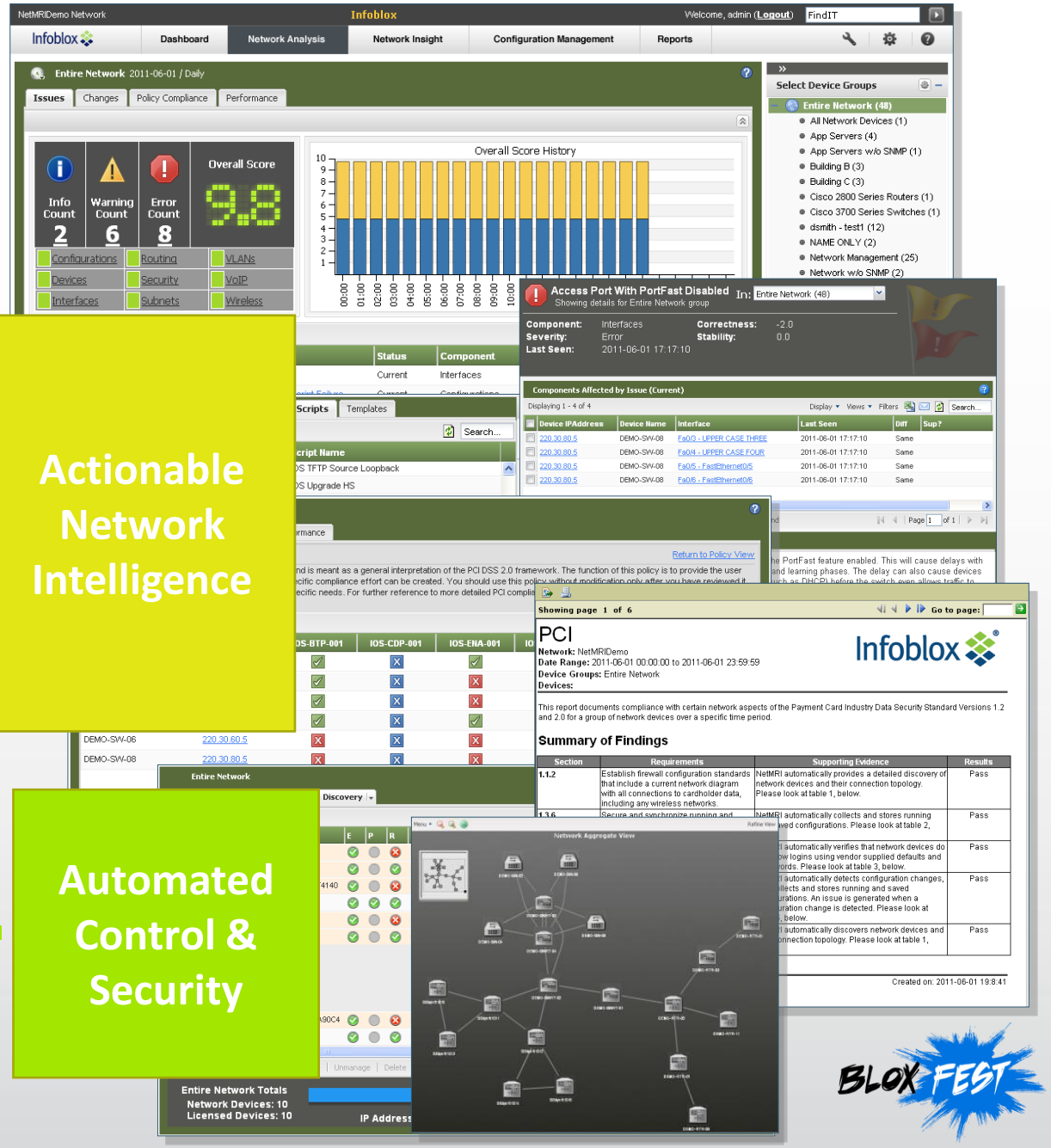
- Poll of Audience – NetMRI experience?
- Quick NetMRI overview
- Examples and tips from customers, old and new
- Q&A Forum

NetMRI Overview

1. Network Discovery and Inventory
   - Network friendly visibility
   - Network constructs, not just devices (routes, VLANs, VRRP Pairs, etc.)
   - Multi-layer topology & auto correlation
2. Network Configuration Analysis
   - Proactive
   - ID hidden problems with no fault or perf symptoms
   - Port duplex mismatch, VRRP pairs, etc.
3. Config Security Policy Enforcement
   - Bundled content
   - Easy customization + unique capabilities
   - Auto analysis, auditing, and reporting
4. Change Automation & Config Mgmt
   - Change Detection, Audit, Config B/U, etc.
   - Advanced / unique change automation

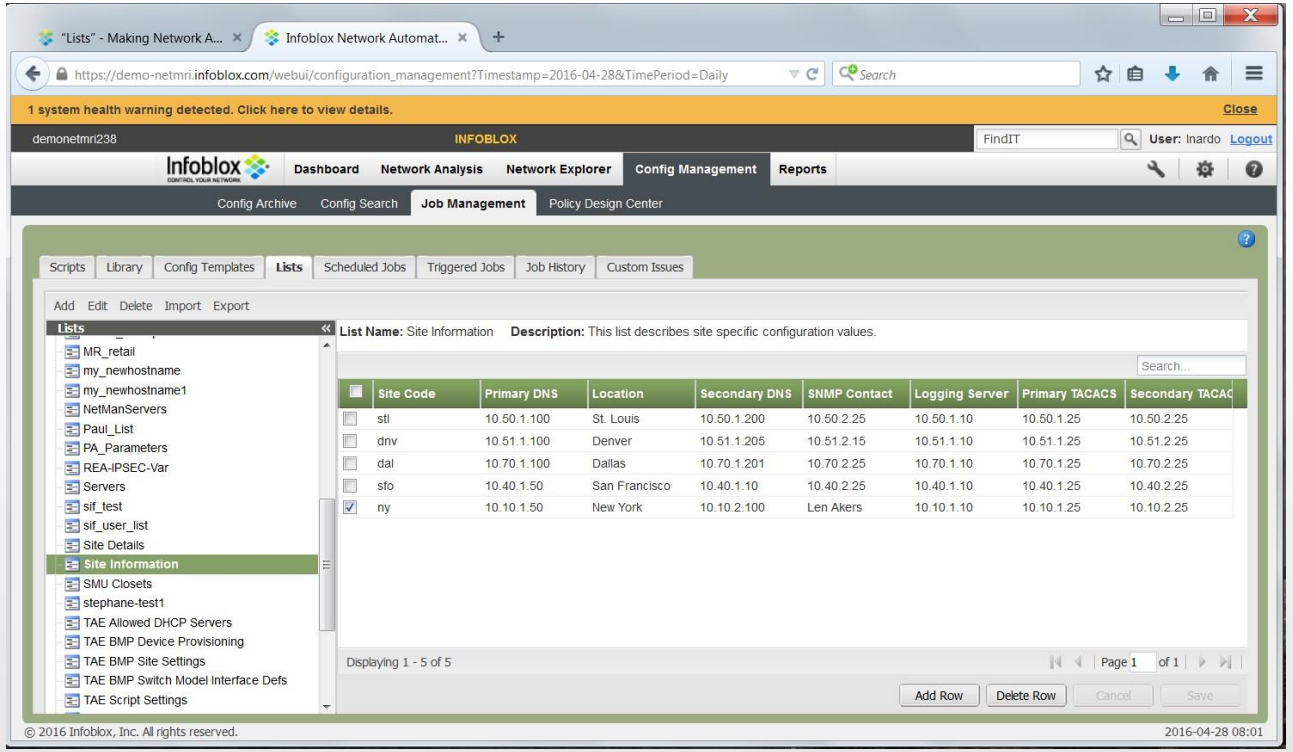
Actionable Network Intelligence

Automated Control & Security



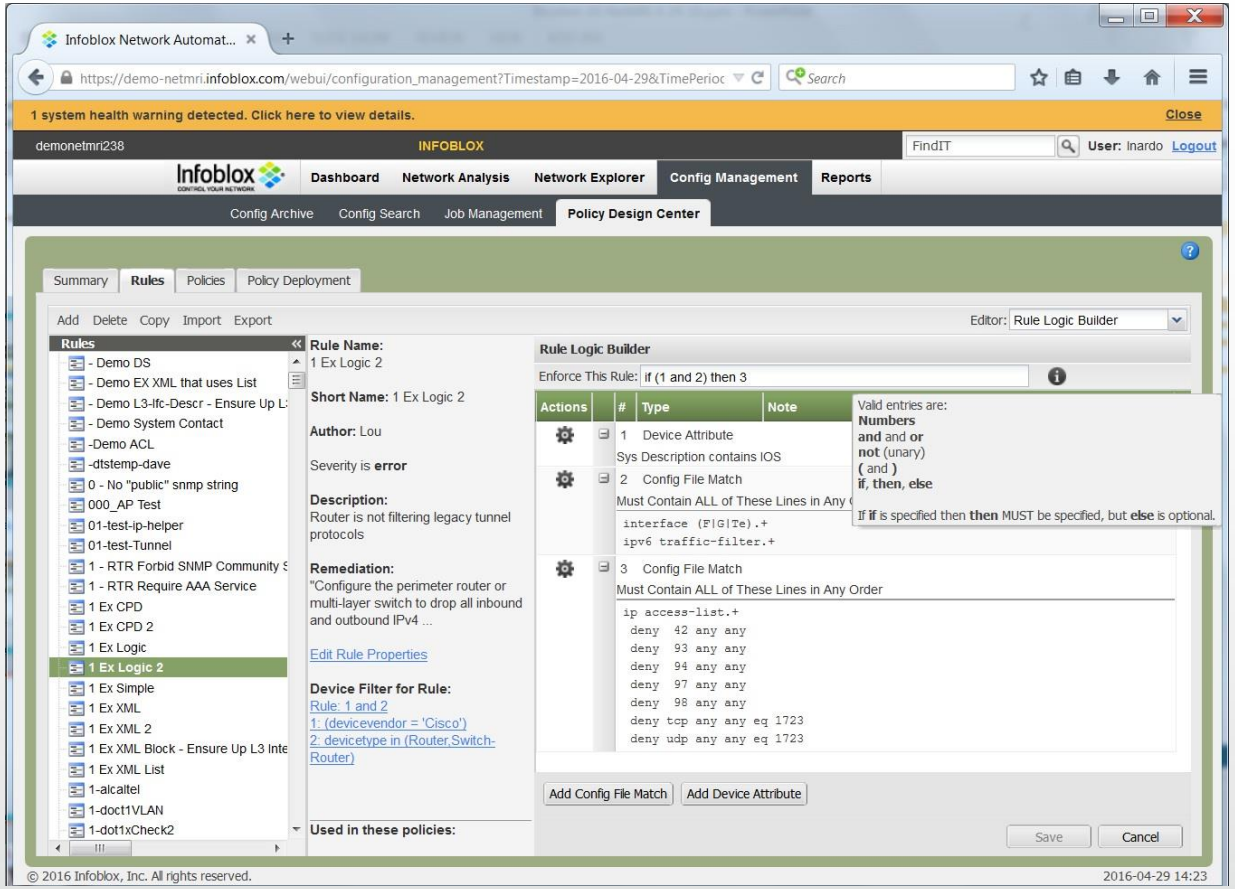
Automating Network Change, Did You Know?

- 4 methods from simple to complex
- Analysis, not just change
- Lists / dynamic variables
- Topology aware
- Multi-device sessions
- Triggered jobs
- Approval & RBAC
- Change “levels” and delegation
- API and custom data
- Community



Enforcing Policy Did You Know?

- Beyond config file checks
- 4 methods from simple to complex
- Logic builder
- XML method = “basic” language
- Lists / dynamic variables
- Non-configuration data analysis
- Topology aware
- Rule object approach
- API and custom data
- Community





Customer Examples and Q&A Forum

- Marty Adkins, Netcraftsmen & Large Federal Agency
- Gregory Whitehead, M&T Bank

Marty Adkins

CCIE #1289, CCSI #93021

Network Architect and co-founder of NetCraftsmen in 2001

Worked with Netcordia founder and NetMRI inventor Terry Slattery to define/refine initial product goals

Frequent user since 2004

adkins@netcraftsmen.com

BLOX FEST

NetMRI Rules, Policies and Scripts

Strategy:

- Have rules specify device filters for vendor, type, model, etc.
- Combine rules with a similar purpose into a policy (optionally with device filter)
- Deploy policy on device groups -- importance of well-defined device groups!

  Example:

  Policy: Site_AAA (TACACS+ plus fall back method)
  Rule: AAA IOS Line Password
  Rule: AAA IOS Local Username-Secret
  Rule: AAA IOS TACACS Routers
  Rule: AAA IOS TACACS Switches
  Rule: AAA NX-OS Local Username-Secret
  Rule: AAA NX-OS TACACS

Examples: Single Rule Into Policy

Policy: IOS Type 4 Secrets ⇒ Violation triggers script WAN_Local_Auth.ccs

Identify IOS devices with username or enable secret commands that employ type 4 (AES) secrets. Due to an implementation flaw, Cisco advised customers to revert to the older type 5 (MD5) method until the flaw was corrected.

Policy: ISR-G2 No ESE Access ⇒ Script: Prevent ESE Access

Verify that Cisco ISR-G2 routers block remote access to the Embedded Services Engine via line 2. This prevents remote connections to the Ethernet LAN IP with port numbers of 2002, 4002, 6002, 9002, which bypasses the VTY access-class ACL.

Example: Multiple Rules Into a Policy

Rules:

- IOS Syslog servers
- IOS SNMP community strings
- IOS SNMP trap servers

Policy: IOS Logging and SNMP

Policy Violation triggers immediate execution of script IOS_Set_Logging_SNMP.ccs

Script repeats the policy checks and corrects as required

Fires custom issue -- IOS Logging/SNMP Corrected

Example: Policy Testing Device Attributes

Policies to identify devices not running a site's baseline OS version

Device filter includes: $sysDescr not like /15\.3\(4\)M5/

- Useful during planned OS upgrades to enumerate ones not yet completed or requiring a reload
- Catch RMA replacement units not baselined before deployment
- Upgrade script can be executed only on those members failing the policy
- A good use for the Lists feature

(Note: can also implement this via child device groups)

Useful scripts that are not policy driven

Backup Cisco Config.ccs - supports IOS, NX-OS, CatOS, ASA

- Performs "copy running startup", optionally to TFTP server
- Can be executed on devices cited in issue “Running Config Not Saved” via “Execute Command”

AdHocCommandWithConfigSave.ccs - Infoblox AdHoc script augmented:

- Supports IOS, NX-OS, CatOS, ASA
- Detects config change made and incorporates Backup Cisco Config

When Policies Are Insufficient

Assess SSH Operation:

Script Enable_IOS_SSH_Access.ccs ⇒ custom issue: IOS SSH Now Enabled

This script tests to see if an SSH-capable crypto image is running.
If so, it then determines if a crypto key pair has been generated and if not, it does so.
It verifies that only SSH v2 access is permitted and if not, corrects that.
It then determines if VTYs permit incoming SSH connections and if not,
it changes that.
Finally, it generates the issue details for all devices which required changes.

(Enhancement needed: Verify that the crypto key modulus is 2048 bits.)

Anyone spot the catch-22? ☺

When Policies Are Insufficient

Script Check_NTP.ccs ⇒ custom issues: NTP Configuration Invalid, NTP Not Synchronized

This script checks NTP operation and configuration:

Is NTP synchronized to some NTP server?

Is it associated with all valid servers for the device group/type? If either is not true, fire issue NTP_Not_Synchronized.

Are there any extraneous servers defined?

Is there a correct NTP authentication key in MD5 format? Is the key enabled and do the server commands specify it?

If any configuration item is not correct, fire issue NTP_Config_Invalid

Holy Grail -- Bullet-Proof OS Upgrades

Script: 1921_IOS_Upgrade.ccs

- This script first performs a light clean up of flash contents by deleting unnecessary files such as: info, info.ver config.old html (directory), cpconfig*, cpexpress*, *.pkg, etc.
- Get the active image filename from "show version“
- Locate stored image files via "dir" and iterate through the list:
- If an image file = desired file, compare the MD5 hash (verify /md5) to value obtained from CCO
- If MD5 comparison fails, fire custom issue "Corrupt IOS Image File“
- Delete all image files except the desired one and the currently running one
- If desired image file is not present, copy it from an HTTP image repository server
  - Compare the MD5 as before
  - Update Boot System Commands as required

Report result, changes made, in custom issue c1921_IOS_Upgrade_Completed

Tips & Best Practices

Many customers rely almost solely on Infoblox-supplied Ad Hoc Command Batch script

- Strength: Very easy to use for small changes across many devices
- Weakness: It does exactly what you tell it to – the safety is off

Most common mistakes for Cisco devices:

1) Forgetting to enter config mode with "config term"
2) Forgetting to exit config mode with "end" ("exit" only backs out one level)
3) Forgetting to save the change to startup with "write mem“
4) Not anticipating extra (conditional) CLI prompts - use "\r“
5) Not escaping characters that are special in Ruby Perl - "$“

4-99) Failing to test on a few sample devices before spraying it on a few hundred. :-(

Tips & Best Practices

- Use device filters in rules, policies, and especially scripts!
  Do not assume that a script will only be executed on appropriate devices
- Device group definitions are VERY important –
  Spend the time to think about what makes sense for your organization, and ripple effects in evolving them
- Always test scripts on a small representative subset of devices
- If changing a blanket rule or policy and its remediation script, manually run the script once against all affected devices to create one batch instead of many triggered single jobs

Tips & Best Practices

Script testing & debugging:

Use the Regular Expression Test page to get Trigger-Variables and Trigger-Template matching on Device Output (find link on Scripts -> Edit/Add)

MUCH faster than iterating on a real device

Specify Script-Timeout, Action-Timeout and Trigger-Timeout for long-running commands and scripts

Use Trigger-Context-Prompt to temporarily specify an alternate device prompt to expect in response to a command

Perl scripting – you can connect to the integral Sandbox and have full access to Perl at the shell level

NetCraftsmen

Marty Adkins
adkins@netcraftsmen.com

BLOX FEST

Greg Whitehead

Senior Network Engineer
M&T Bank
gwhitehead@mtb.com

NetMRI Deployment:

- v7.0.5 on NT-4000
- ~ 4,000 Routers, switches, load balancers, and WAN accelerators
- Active and Back-up appliances in separate datacenters

Use Case: “Reachable” Device Custom Data Field as a Filter

- Custom Field:
  Created a device level custom field that gets populated called “reachable” that is Y or N based on SNMP availability of the device within the last 24 hours.
- Dynamic Group:
  That field is used to put devices into a group.
- Script and Rule Filters:
  That group is used in scripts and policy rules to make sure we don’t get failures for devices that are really not on the network at that point.

1) Nightly Job:

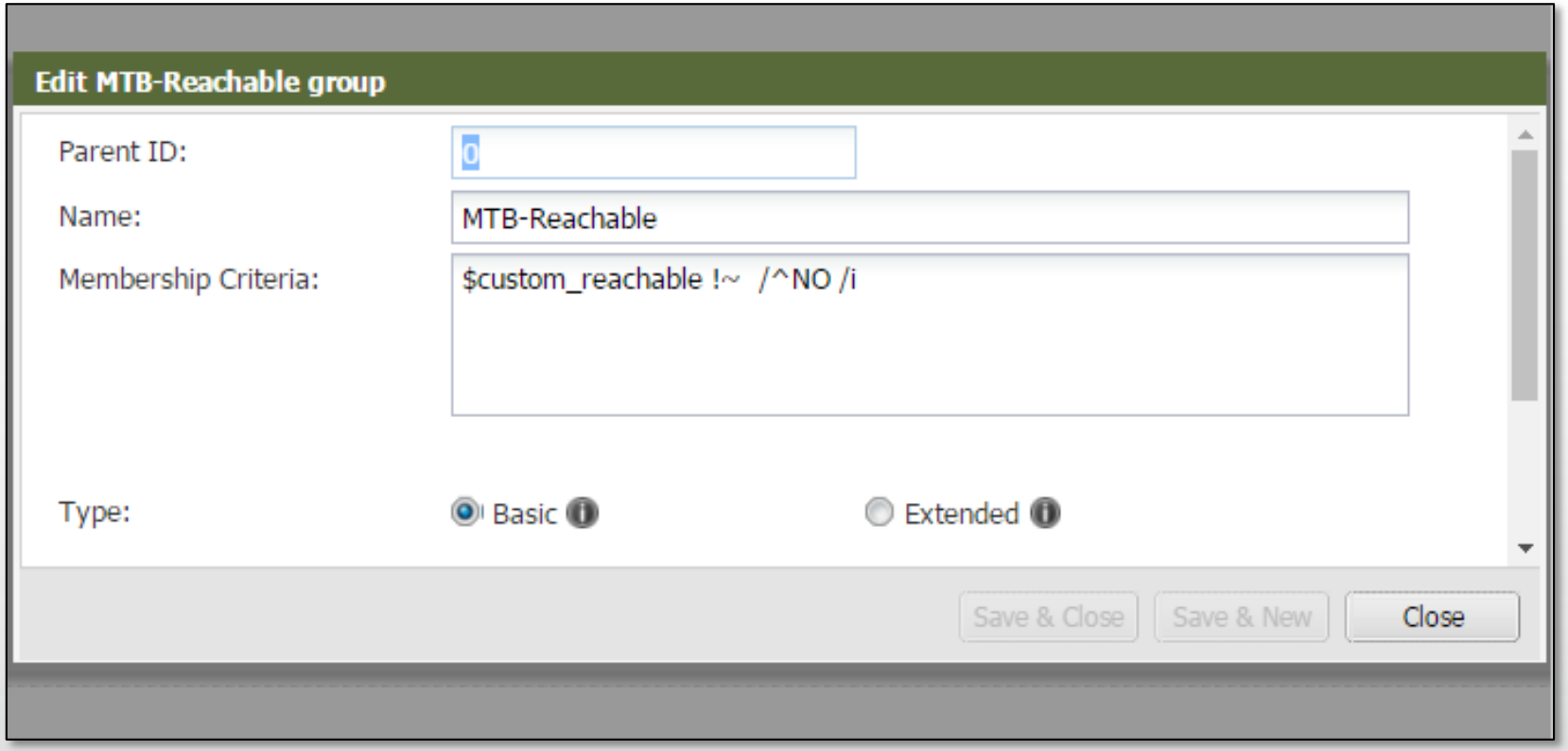
- Executes multiple commands against each device, and
- Updates custom data field: “Reachable”

```
my $last_snmp_access;
my $secs_since_last_snmp = last_snmp_access_within_last_day(
\$last_snmp_access );
if( $secs_since_last_snmp ) {
      $device->set_custom_field('reachable', "YES - last good SNMP
$last_snmp_access" );
} else {
      $device->set_custom_field('reachable', "NO - last good SNMP
$last_snmp_access" );
}

sub last_snmp_access_within_last_day {
      my( $last_snmp_access_ref  ) = @_;
```

```
      # script will return number of seconds if last successful SNMP
communication from NetMRI to device is within 1 day, or will return 0 if an
error or last SNMP access > 1 day

      my $device_settings = $easy->broker->device_setting->index({
            DeviceID    => $easy->device_id,
            include =>  'DeviceSNMPTimestamp'
      });

      my $last_snmp = $device_settings->{device_settings}[0]-
>{DeviceSNMPTimestamp};

      $easy->log_message( 'debug', "last successful SNMP access:
<$last_snmp>" );
      $$last_snmp_access_ref = $last_snmp;

      my $last_snmp_epoch;
      if( $last_snmp =~ /(\d+)-(\d+)-(\d+) (\d+):(\d+):(\d+)/ ) {
            $last_snmp_epoch = timelocal($6, $5, $4, $3, $2 - 1, $1 );
      } else {
            $easy->log_message( 'error', "cannot parse: <$last_snmp>" );
            return 0;
      }

      my $secs_since_access = ( time - $last_snmp_epoch );
      $easy->log_message( 'debug', "last successful SNMP access was this many
seconds ago: <$secs_since_access>" );

      if( $secs_since_access > 86400 ) {
            $easy->log_message( 'error', " The last time SNMP access
succeeded for this device was more than 1 day ago ($secs_since_access
seconds), do not try to run commands against this device" );
            return 0;
      }
      return $secs_since_access;
}
```

2) Grouping:

- Devices with “Reachable” device custom data field
- Automatically placed into device group “MTB-Reachable.

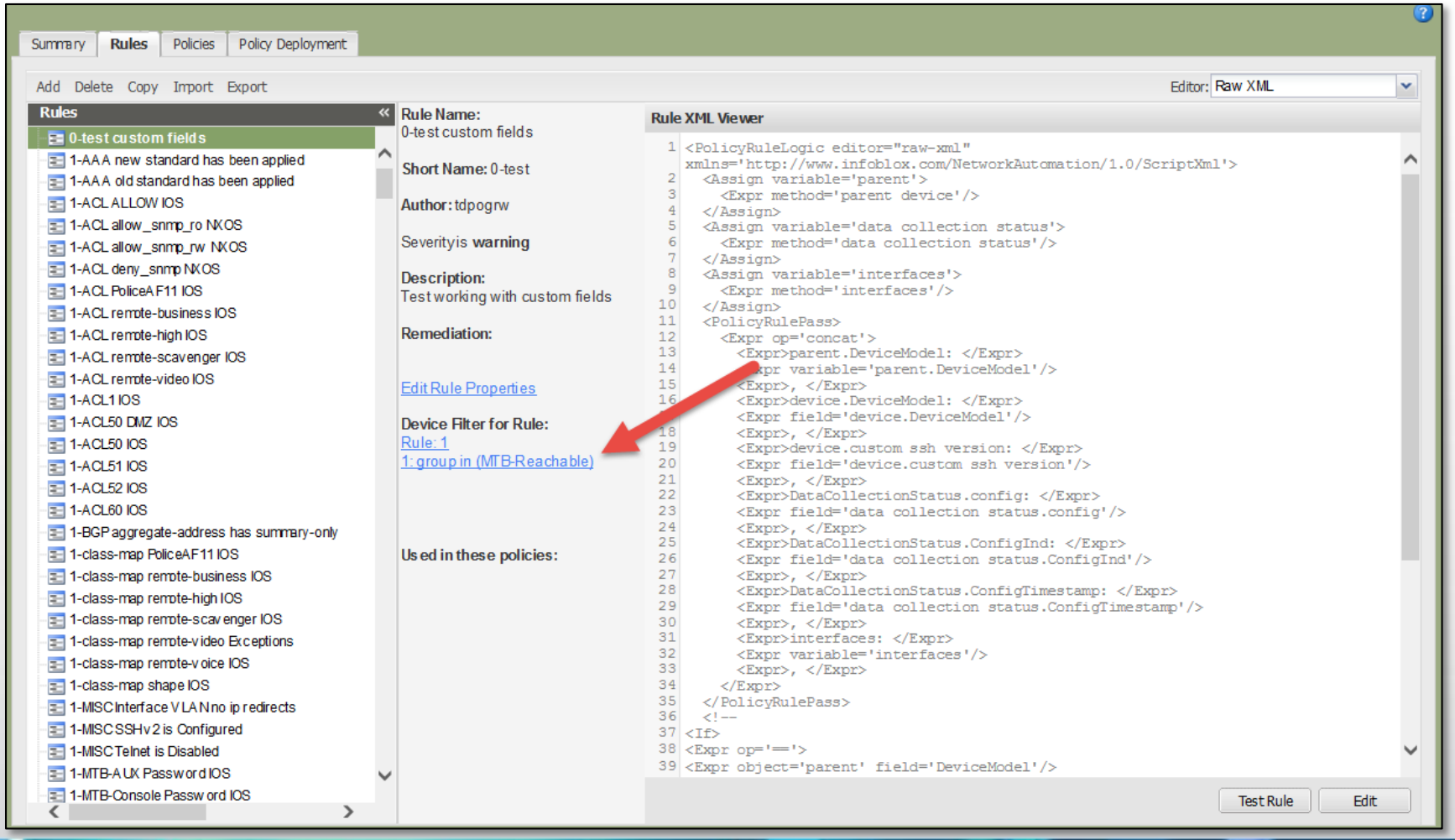


3) Group used as a filter:

- Jobs and Policies only run against devices reachable by SNMP
- Still see occasional problem where reachable but configs can't be collected.

For Perl Scripts:

```
# BEGIN-SCRIPT-BLOCK
#
# Script-Filter:
# $Vendor = "Cisco" and $sysDescr =~ /ios/i
# and memberOf[ "MTB-Reachable" ]
#
# Script-Timeout: 600
#
#
# END-SCRIPT-BLOCK
<script commands here>
```



Policy Rule Exception Lists

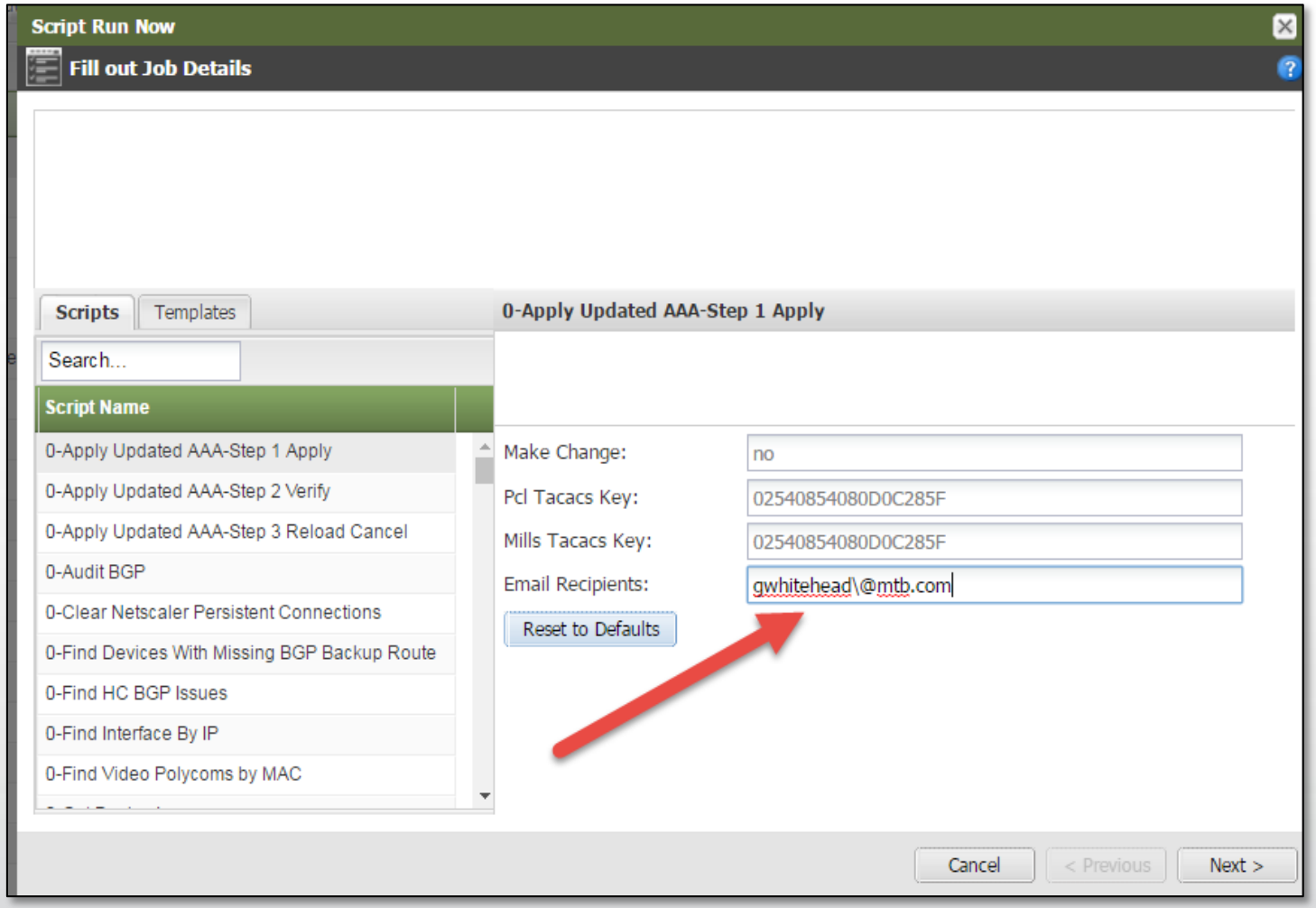
- Use lists in policy rules to indicate if there are devices that the policy rule should NOT be applied to.
- This functions as an exception list, and allows us flexibility when creating rules, because there are always devices whose configuration is an exception, and we don’t want them to fail the Policy Rule checks.

Sending Email During Job Execution

- Use the MIME::Lite module in Perl scripts (allowing the email address to be entered at time of job execution) so that detailed logging can be sent for each device a script is run against.
- This is easier than looking at log files at the OS level.

Sending Email During Job Execution

Input email recipients



The Perl Script, each writes data to file at the OS called $device.txt

```
# BEGIN-SCRIPT-BLOCK
#
# Script-Filter:
# $Vendor = "Cisco" and ( $sysDescr =~ /ios/i or $sysDescr =~ /cisco vg/i )
#
# Script-Timeout: 300
#
#
# Script-Variables:
#     $make_change            word        "no"
#     $pcl_tacacs_key         word        "02540854080D0C285F"
#     $mills_tacacs_key word          "02540854080D0C285F"
#     $email_recipients string            "gwhitehead\@mtb.com"
#
# END-SCRIPT-BLOCK

use strict;
use NetMRI_Easy;
use NetMRI::API;
use Data::Dumper;
use MIME::Lite;

our( $make_change, $pcl_tacacs_key, $mills_tacacs_key, $email_recipients,
$sysdescr, $model, $version );

<…>

my $subject = "Apply Updated AAA - Step 1 Apply - $device_name";
if( $error ) { "ERROR: Apply Updated AAA - Step 1 Apply - $device_name" }
```

```
if( $email_recipients ) {

      my $msg;
      $msg = MIME::Lite->new
            (
            From    => 'XXXXX',
            To      => "$email_recipients",
            Subject => $subject,
            Type  => 'multipart/mixed',
            Data  => 'Test data',
            ) || die "Error creating multipart container: $!";


      $filename = "$device_name.txt";
      my $full_filename = "$directory/$filename";
      $easy->log_message( 'debug', "The report file is: <$full_filename>" );

      ### Add the attachment
      $msg->attach (
            Type => 'text/plain',
            Path => $full_filename,
            Filename => $filename,
            Disposition => 'attachment'
            ) || die "Error adding the text message part: $!\n";

      $msg->send ("smtp", "10.1.1.1", debug => 1) || die "Error sending to
SMTP server - Error: $!\n";
}

if( $error ) {
      exit 1
} else {
      exit 0
}
```

Back up NXOS Licenses

- Extending beyond NetMRI’s config backup
- Script to backup NXOS licenses to a central SCP server since these are not backed up by NetMRI and can be lost.

Using Lists in Policy Rules and Scripts

- Policy rules that use lists to verify VTY, AUX, enable secret and local user account hashed passwords are correct.
- Perl scripts were written that use the exact same lists to set correct values for these hashed passwords.

Compile & Email Detailed Data From Job Execution Across Multiple Devices

- First script gathers data from each device its own .csv file on the OS
- Second script runs and concatenates all the files into a single file, emails the .csv file to selected users, and renames (or deletes) the original .csv files so upon next execution only data for devices the job was run against gets sent.
- This could be easier if scripts had ability to run a different script pre or post execution of the man script.


BLOX FEST
Infoblox