

# 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



# 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**

Section	Requirements	Supporting Evidence	Results
1.1.2	Establish firewall configuration standards that include a current network diagram with all connections to cardholder data, including any wireless networks.	NetMRI automatically provides a detailed discovery of network devices and their connection topology. Please look at table 1, below.	Pass
1.3.6	Secure and synchronize running and saved configurations.	NetMRI automatically collects and stores running and saved configurations. Please look at table 2, below.	Pass
		NetMRI automatically verifies that network devices do not logins using vendor supplied defaults and credentials. Please look at table 3, below.	Pass
		NetMRI automatically detects configuration changes, collects and stores running and saved configurations. An issue is generated when a configuration change is detected. Please look at table 4, below.	Pass
		NetMRI automatically discovers network devices and connection topology. Please look at table 1, below.	Pass



# 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

The screenshot displays the Infoblox Network Automator web interface. The browser address bar shows the URL: [https://demo-netmri.infoblox.com/webui/configuration\\_management?Timestamp=2016-04-28&TimePeriod=Daily](https://demo-netmri.infoblox.com/webui/configuration_management?Timestamp=2016-04-28&TimePeriod=Daily). The interface includes a navigation menu with options like Dashboard, Network Analysis, Network Explorer, Config Management, and Reports. The main content area shows a 'Lists' management page with a table of configuration lists. The selected list is 'Site Information', which is described as 'This list describes site specific configuration values.' The table contains the following data:

Site Code	Primary DNS	Location	Secondary DNS	SNMP Contact	Logging Server	Primary TACACS	Secondary TACACS
sti	10.50.1.100	St. Louis	10.50.1.200	10.50.2.25	10.50.1.10	10.50.1.25	10.50.2.25
dnv	10.51.1.100	Denver	10.51.1.205	10.51.2.15	10.51.1.10	10.51.1.25	10.51.2.25
dal	10.70.1.100	Dallas	10.70.1.201	10.70.2.25	10.70.1.10	10.70.1.25	10.70.2.25
sfo	10.40.1.50	San Francisco	10.40.1.10	10.40.2.25	10.40.1.10	10.40.1.25	10.40.2.25
ny	10.10.1.50	New York	10.10.2.100	Len Akers	10.10.1.10	10.10.1.25	10.10.2.25

The interface also shows a sidebar with a tree view of configuration lists, including 'MR\_retail', 'my\_newhostname', 'NetManServers', 'PAU\_List', 'RETA-IPSEC-Var', 'Servers', 'sif\_test', 'Site Details', 'Site Information', 'SMU Closets', 'stephane-test1', 'TAE Allowed DHCP Servers', 'TAE BMP Device Provisioning', 'TAE BMP Site Settings', 'TAE BMP Switch Model Interface Defs', and 'TAE Script Settings'. The bottom of the page displays the copyright notice: © 2016 Infoblox, Inc. All rights reserved. and the date: 2016-04-28 08:01.



# 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

Infoblox Network Automator

1 system health warning detected. Click here to view details.

demonetmr238 INFOBLOX FindIT User: inardo Logout

Infoblox Dashboard Network Analysis Network Explorer Config Management Reports

Config Archive Config Search Job Management Policy Design Center

Summary Rules Policies Policy Deployment

Add Delete Copy Import Export Editor: Rule Logic Builder

Rules

- Demo DS
- Demo EX XML that uses List
- Demo L3-Ifc-Descr - Ensure Up L
- Demo System Contact
- Demo ACL
- dstemp-dave
- 0 - No "public" snmp string
- 000\_AP Test
- 01-test-ip-helper
- 01-test-Tunnel
- 1 - RTR Forbid SNMP Community S
- 1 - RTR Require AAA Service
- 1 Ex CPD
- 1 Ex CPD 2
- 1 Ex Logic
- 1 Ex Logic 2
- 1 Ex Simple
- 1 Ex XML
- 1 Ex XML 2
- 1 Ex XML Block - Ensure Up L3 Inte
- 1 Ex XML List
- 1-alcate
- 1-dot11VLAN
- 1-dot1xCheck2

Rule Name: 1 Ex Logic 2

Short Name: 1 Ex Logic 2

Author: Lou

Severity is error

Description: Router is not filtering legacy tunnel protocols

Remediation: \*Configure the perimeter router or multi-layer switch to drop all inbound and outbound IPv4 ...

Edit Rule Properties

Device Filter for Rule: Rule: 1 and 2 1: (devicevendor = 'Cisco') 2: devicetype in (Router,Switch-Router)

Used in these policies:

Rule Logic Builder

Enforce This Rule: if (1 and 2) then 3

Actions	#	Type	Note
1	1	Device Attribute	Sys Description contains IOS
2	2	Config File Match	Must Contain ALL of These Lines in Any Order interface (F G Te).+ ipv6 traffic-filter.+
3	3	Config File Match	Must Contain ALL of These Lines in Any Order ip access-list.+ deny 42 any any deny 93 any any deny 94 any any deny 97 any any deny 98 any any deny top any any eq 1723 deny udp any any eq 1723

Valid entries are:  
Numbers  
and and or  
not (unary)  
( and )  
if, then, else  
If if is specified then then MUST be specified, but else is optional.

Add Config File Match Add Device Attribute

Save Cancel

© 2016 Infoblox, Inc. All rights reserved. 2016-04-29 14:23





# 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](mailto:adkins@netcraftsmen.com)**





# 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  $\Rightarrow$  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  $\Rightarrow$  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



# Greg Whitehead

Senior Network Engineer

M&T Bank

[gwhitehead@mtb.com](mailto: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.”

### Edit MTB-Reachable group

Parent ID:

Name:

Membership Criteria:

Type:  Basic ⓘ  Extended ⓘ



### 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>
```

The screenshot displays a network management tool interface with the following components:

- Summary** tab selected, with sub-tabs for **Rules**, **Policies**, and **Policy Deployment**.
- Rules** list on the left: A scrollable list of rules including "0-test custom fields" (highlighted), "1-AAA new standard has been applied", "1-AAA old standard has been applied", "1-ACL ALLOW IOS", "1-ACL allow\_snmp\_ro NKOS", "1-ACL allow\_snmp\_rw NKOS", "1-ACL deny\_snmp NKOS", "1-ACL PoliceAF11 IOS", "1-ACL remote-business IOS", "1-ACL remote-high IOS", "1-ACL remote-scavenger IOS", "1-ACL remote-video IOS", "1-ACL1 IOS", "1-ACL50 DMZ IOS", "1-ACL50 IOS", "1-ACL51 IOS", "1-ACL52 IOS", "1-ACL60 IOS", "1-BGP aggregate-address has summary-only", "1-class-map PoliceAF11 IOS", "1-class-map remote-business IOS", "1-class-map remote-high IOS", "1-class-map remote-scavenger IOS", "1-class-map remote-video Exceptions", "1-class-map remote-video IOS", "1-class-map shape IOS", "1-MISC Interface VLAN No ip redirects", "1-MISC SSHv2 is Configured", "1-MISC Telnet is Disabled", "1-MTB-AUX Password IOS", and "1-MTB-Console Password IOS".
- Rule Details** for "0-test custom fields":
  - Rule Name:** 0-test custom fields
  - Short Name:** 0-test
  - Author:** tdpogrw
  - Severity:** warning
  - Description:** Test working with custom fields
  - Remediation:** (empty)
  - Device Filter for Rule:** [Rule: 1](#)  
[1:group in \(MTB-Reachable\)](#) (indicated by a red arrow)
  - Used in these policies:** (empty)
- Rule XML Viewer** on the right, showing XML code for the rule logic. The code includes assignments for variables like "parent", "data collection status", "interfaces", and "device.DeviceModel", and a **PolicyRulePass** block with various expressions.
- Buttons:** "Test Rule" and "Edit" at the bottom right.

# 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 screenshot shows a software interface titled "Script Run Now" with a sub-header "Fill out Job Details". On the left, there is a list of scripts under the "Scripts" tab, with "0-Apply Updated AAA-Step 1 Apply" selected. On the right, there are several input fields: "Make Change:" (no), "Pcl Tacacs Key:" (02540854080D0C285F), "Mills Tacacs Key:" (02540854080D0C285F), and "Email Recipients:" (gwhitehead@mtb.com). A red arrow points to the "Email Recipients" field. At the bottom, there are buttons for "Reset to Defaults", "Cancel", "< Previous", and "Next >".

# 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"
#   $spl_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, $spl_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;
    $msg = MIME::Lite->new
        (
            From    => 'XXXXXX',
            To      => "$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 main script.



# BLOX FEST

Infoblox 