



NetBrain® Next-Gen R11.1b

Intent Based Automation Study Guide

Contents

1	Introduction	8
1.1	What is Intent-based Automation?	8
1.2	Intent-Based Automation Use Cases and Flows	10
1.2.1	Intent-Based Collaborative Troubleshooting.....	10
1.2.2	Continuous Network Assessment	12
1.2.3	Change Management and Assessment.....	14
1.3	How to Use This Cookbook?	15
1.3.1	Sign in to Netbrain Lab for the First Time	16
2	Automate Frequently Used Commands.....	21
2.1	Search and Add Devices to the Map.....	22
2.1.1	Draw Devices from a Device Group	22
2.1.2	Draw Devices from the Search Bar	23
2.1.3	Save the Device Map.....	23
2.2	Quick Intent Creation.....	24
2.3	Define Variables with a Visual Parser	26
2.3.1	Define parser for ping <destination ip>.....	26
2.3.2	Define parser for Command <i>Show ip route</i>	30
2.4	Define the Diagnosis.....	32
2.5	Define Intent Output.....	34
2.6	Duplicate the Intent to Other Devices	38
2.7	Diagnosis Tree	40
2.8	View the Result on Map	41
2.9	Create Intent and Summary Dashboard	42
2.9.1	Intent Dashboard	42
2.9.2	Summary Dashboard	44
2.9.3	Add an Intent Dashboard to an Existing Summary Dashboard	47

3	Automate Basic Device Health Checking	50
3.1	Select a Device	51
3.2	Retrieve Data and Parse Variables for Health Checks.....	53
3.2.1	Parse variables from <i>show version</i>	53
3.2.2	Parse variables from <i>Show Process CPU</i>	56
3.2.3	Parse variables from <i>Show Interface</i>	57
3.3	Define Diagnosis for Health Checks.....	59
3.3.1	Define Diagnosis for <i>Show version</i>	59
3.3.2	Define Diagnosis for <i>Show Process CPU</i>	61
3.3.3	Define Diagnosis for <i>Show interface</i>	63
3.4	Monitor F5 Virtual Server Status	67
3.4.1	Parse variables from <i>show ltm virtual detail recursive</i>	67
3.4.2	Define Diagnosis for Monitor F5 virtual server	70
3.5	Execute Intents and Create Dashboards	78
3.5.1	Execute Intents	78
3.5.2	Intent Dashboard	79
3.5.3	Summary Dashboard	81
4	Enable Others to Use Your Automation	83
4.1	Auto Intent Wizard: Replicate Intent to Multiple Devices	84
4.1.1	Intent Template	85
4.1.2	Pre Decode	89
4.1.3	Auto Intent.....	92
4.1.4	Run the Auto intent:	93
4.2	Wrapper Intent as End User Interface	96
4.2.1	Create Wrapper Intent.....	96
4.2.2	Execute Wrapper Intent.....	103
4.3	Run intent with Chatbot	106
4.3.1	Create the Netbrain Chatbot	106
4.3.2	Use Chatbot.....	111
4.3.3	Add Auto Intent to Chatbot.....	114
5	Network Assessment and Document: Essential	118

5.1	Document Your Network Inventory and Performance	119
5.1.1	Prerequisites	120
5.1.2	Create Network Intent	121
5.1.3	Use Intent Replication Wizard.....	123
5.1.4	Run the Intent	127
5.1.5	Export ADT to CSV file	129
5.1.6	Create Intent Dashboard for ADT Automation Column.....	130
5.2	ADT Drilldown	133
5.2.1	Build Base ADT.....	134
5.2.2	Build ADT from a CSV File.....	141
5.2.3	Build ADT from Intent Template	144
5.3	Assess and Document Your Network Operational Status	146
5.3.1	Create Network Intent	147
5.3.2	Add Signature Variables to ADT Base Table	154
5.3.3	Export ADT to CSV file	162
6	Network Assessment Case Study: Security Assessment.....	164
6.1	Check NIST Compliance and Vulnerability	164
6.1.1	Build ADT Base Table	165
6.1.2	Create Network Intent: Check Vty Telnet Access	168
6.1.3	Create NI: AAA Config Check.....	182
6.1.4	Create NI: Device Unused Ports Config Check	188
6.1.5	Create NI: Device Password Policy Config Check	193
6.1.6	Create NI: Device Line Session Timeout.....	195
6.1.7	Create Intent and Summary Dashboard	197
6.2	CVE Security Advisory	202
6.2.1	Prerequisites	203
6.2.2	Build Base ADT.....	205
6.2.3	Create an Intent to Check if a CVE is Affecting any Device	207
6.2.4	Create Intent to Check Device Against all CVEs	215
6.2.5	Create Intent and Summary Dashboard	216
7	Network Assessment Case Study: Configuration Drift.....	217

7.1	Check Configuration Drift Against the Golden Template	218
7.1.1	Prerequisites	218
7.1.2	Create Intent to Check ACL Against Golden Template	222
7.1.3	Modify Intent to Create CSV File	232
7.1.4	Use Intent Replication Wizard.....	238
7.1.5	Run Intent Once and Rebuild Table	241
7.1.6	View Summary Report	242
7.1.7	Create Intent and Summary Dashboard	243
7.2	Create Golden Template	244
7.2.1	Create Intent to Export ACL Configuration to a CSV File.....	244
7.2.2	Create ACL Report for the Whole Network Devices.....	248
7.2.3	Analyze the Report to Find the Golden Config	250
7.3	Check NTP Server Against Golden Template	256
7.3.1	NTP Server Golden Template.....	256
7.3.2	Create Intent to Check NTP Server Against Golden Template	256
7.3.3	Create the Summary Report and Dashboard	259
7.4	Check Configuration Settings in Public Cloud	261
7.4.1	Prerequisites	262
7.4.2	Create Intent to Check AWS EC2 Configuration	264
7.4.3	Replicate Intent to All AWS EC2	273
8	Network Assessment Case Study: Failover	274
8.1	Check Failover Status Change.....	274
8.1.1	Create HSRP Device Group	275
8.1.2	Create an intent to check the failover status change	276
8.1.3	Replicate the intents for all HSRP devices using ADT	287
8.1.4	Create the Dashboard	291
8.2	Follow-up to Check the Performance Degraded After Failover.....	293
8.2.1	Update replicated intents in ADT	297
8.3	Check Failover Consistency of HSRP Pair Devices	298
8.3.1	Create a follow-up intent.....	299
8.3.2	Create a parent intent.....	307

8.3.3	Replicate the intents for all HSRP devices using ADT	318
8.3.4	Replicate the intent to all HSRP devices	321
8.3.5	View and Run the intents in ADT	323
8.3.6	Create the Dashboard	324
9	Collaborative Troubleshooting.....	326
9.1	Document BGP Devices and Configurations	326
9.1.1	Create BGP Device Group	327
9.1.2	Create an intent to parse BGP AS numbers and configuration	328
9.1.3	Define the replicate settings and decode	333
9.1.4	Build Base ADT.....	336
9.2	Create a wrapper intent for BGP Troubleshooting.....	339
9.2.1	Create an intent to Check BGP Configuration	339
9.2.2	Create a Wrapper Intent.....	353
9.2.3	Execute the Auto Intent BGP Wrapper	357
9.3	Create frequently used BGP Troubleshooting Intents	361
9.3.1	Create NI to Check BGP Neighbor Status Check	361
9.3.2	Replicate the Intent to ADT	364
9.3.3	Create NI to Check BGP Route Reflector Client	367
9.3.4	Replicate the Intent to ADT	372
9.4	Create a chatbot for the BGP TS wrapper intent	375
9.4.1	Create the Netbrain Chatbot	375
9.4.2	Use Chatbot.....	378
10	Application Path Assessment and Troubleshooting.....	381
10.1	Document the application path when the network is healthy	382
10.2	Build and Manage Automations for Path.....	387
10.2.1	Create the ADT base table from the path browser.....	387
10.2.2	Replicate the Intents of Device Health Check.....	389
10.2.3	Create Path Intent for Path Next Hop Stability check	392
10.2.4	Create Wrapper Intent and Enable Auto Intent	397
10.3	Troubleshoot the path issue.....	405
11	Intent-based Change Verification and Assessment.....	406

11.1	Build Intents for Change Assessment.....	407
11.1.1	What has Changed	407
11.1.2	Auto Test.....	410
11.2	Intent-Driven Change Management and Assessment.....	412
11.2.1	Define Change.....	412
11.2.2	Create Intent to Analyze Route Changes.....	413
11.2.3	Organize and Run Automations Before Change	420
11.2.4	Execute Change	424
11.2.5	Verify Change	425
12	Map and Document Your Network.....	427
12.1	Create a Map	427
12.1.1	Create Map with Intent.....	428
12.1.2	Intent Data View	433
12.2	Create the Inventory Report	438
12.2.1	Built-in Inventory Reports	438
12.2.2	Use ADT to Build Inventory Report	440
12.3	Create the Report from CLI Command Results	443
12.3.1	Create CSV Report from CLI Command Results	444
12.3.2	Create Report by Pre-Replicated Intent Template	448

1 Introduction

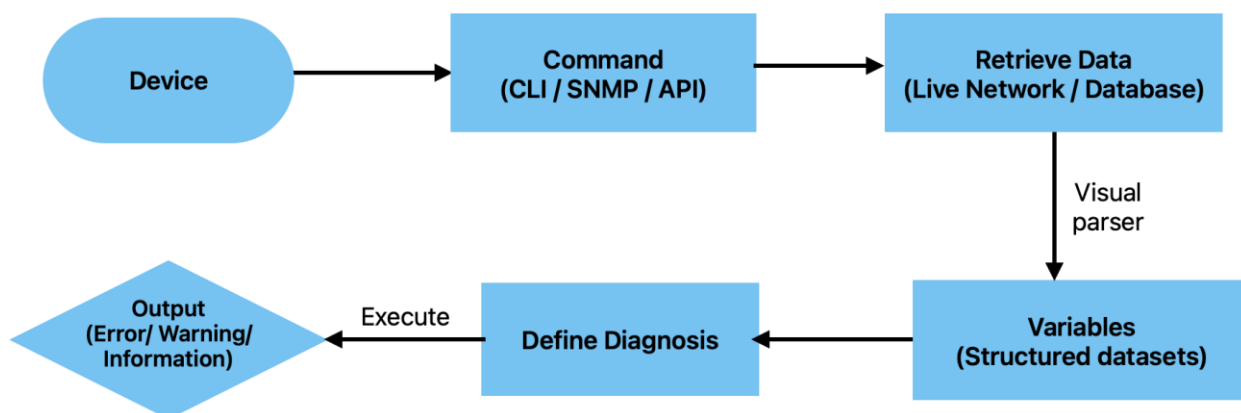
1.1 What is Intent-based Automation?

Network Intent (NI) is an automation construct that allows users to define expected design and operational state. It establishes a baseline configuration and operational state to validate the network design. Further, users can run NIs to simply detect design and operational deviations during the troubleshooting or set up the scheduled task to run a set of NIs to assess the whole network.

A standard NI has the following key components:

- **Device(s):** An intent is specific to a device or a set of devices. The intent is designed this way so that it can include the current, last, and baseline data.
- **Raw data:** The data can be retrieved from devices by the CLI, SNMP, and API.
- **Parser:** The Visual Parser transforms the raw data into structured datasets. These datasets serve as the input for the diagnosis. Visual Parser is the UI-based for users without any programming language to create one.
- **Diagnosis:** The diagnosis defines rules or conditions to evaluate different aspects of the network and the status code (the output) for different conditions. Users can then specify the appropriate actions or recommendations to be executed based on the status codes.

So, an NI takes a device and a command (CLI/SNMP/API) as the input, retrieves data from the live network or database, parses the data into the structured datasets (variables), runs the diagnosis on these variables, and outputs the network status (error/warning/info).

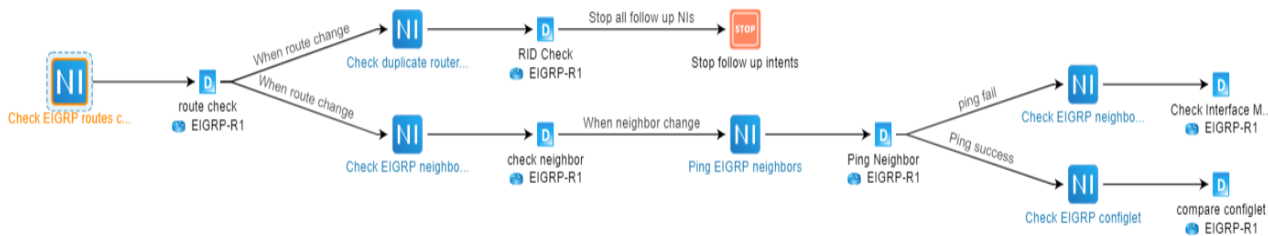


The following are some examples of NI:

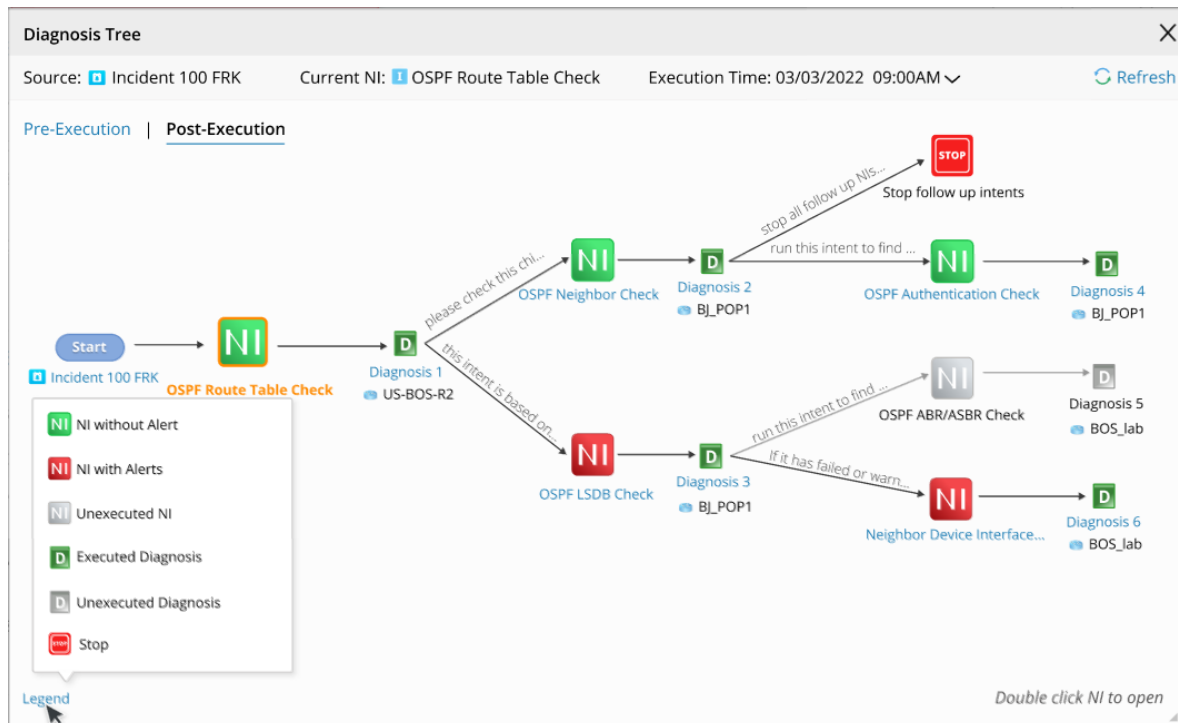
- Enforce configuration compliance and prevent configuration drift. For example, you can create an NI to ensure that an access list is configured in all Cisco firewalls.

- Continuously monitor the operational status. For example, you can create NIs to monitor the general health of a network device, such as the reachability, uptime and reboot reason, performance data, and temperature. You can also create NIs to monitor the specific vendor functions, such as the virtual server and pool member of an F5 load balancer.
- Check the failover. Create an intent to check whether the failover status changes and monitor the performance if a failover occurs.

Multiple NIs can be grouped to form a diagnosis flow. Users can define **follow-up NIs** in the Diagnosis so that these NIs can be executed under certain conditions. For example, users can add Check duplicate router ID and Check EIGRP neighbors as the follow-up NI if the EIGRP routes change. An Intent can also call itself as a follow-up intent (**follow-up self**), so the system will jump into other devices to execute the same defined logic as the original intent. For example, users can create an intent to check the BGP neighbor status and call itself for its BGP neighbors. This follow-up intent can be recursively called till no neighbor is found or it reaches a user-defined depth.



The execution results of a parent NI and its follow-up NIs are displayed in a diagnosis tree:



An intent (seed NI) can be cloned for all qualified devices through the whole network or as a set of network devices via an **Intent Replication Wizard**. An **Intent Template** (NIT) will be defined for this purpose, including the definition of the target devices, rules to replace the Macro Variables, etc. The cloned intents can be grouped into a column of an **Automation Data Table (ADT)**, which is the other key component (besides intent itself) for intent-based automation.

An ADT contains a **base table**, which can be the critical assets and be built from the device group, sites, application table, intent template, and CSV file, and many **Column groups**, which are associated with the intent and results and can be replicated primarily from the intent template.

The following is an example of an ADT table showing all BGP devices and intents associated with the BGP devices:

BGP Document and TS

Table Builder

Last Updated at: 07/08/2024 02:18 PM

Rebuild Table

Add Data Manually

Description: Type description here...

Items: 15 Rows 11 Columns

Search...

Advanced Filter: Undefined

Device	BGP ASN	Router ID	BGP Configurations	Check BGP Config Change	Intent Status Code	Device Status Code	TS BGP (Wrapper)	BGP neighbor Check	Check Reflector Cl
UK-LHR-CR01-02	65200	192.168.10.253	router bgp 65200	Check BGP configure change UK...	BGP does not change	UK-LHR-CR01-02	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check UK...	BGP Route Reflector
UK-LHR-CR01-01	65200	192.168.10.254	router bgp 65200	Check BGP configure change UK...	BGP does not change	UK-LHR-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check UK...	BGP Route Reflector
US-NY-CR01-01	65101	192.168.1.254	router bgp 65101	Check BGP configure change US...	BGP does not change	US-NY-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check US...	BGP Route Reflector
ISP-PE03	10000		router bgp 10000	Check BGP configure change IS...	BGP does not change	ISP-PE03	Wrapper intent for BGP Tr...	BGP Neighbor Stability Check IS...	BGP Route Reflector
ISP-P02	10000		router bgp 10000	Check BGP configure change IS...	BGP does not change	ISP-P02	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check IS...	BGP Route Reflector
US-BOS-CR01-01	65100	192.168.0.254	router bgp 65100	Check BGP configure change US...	BGP does not change	US-BOS-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check US...	BGP Route Reflector
JP-TYO-CR01-01	65300	192.168.20.254	router bgp 65300	Check BGP configure change JP-T...	BGP does not change	JP-TYO-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check JP...	BGP Route Reflector
US-BOS-R2	65001	10.11.11.11	router bgp 65001	Check BGP configure change US...	BGP does not change	US-BOS-R2	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check US...	BGP Route Reflector
SG-SIN-CR01-01	65301	192.168.21.254	router bgp 65301	Check BGP configure change SG...	BGP does not change	SG-SIN-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check SG...	BGP Route Reflector
ISP-PE01	10000		router bgp 10000	Check BGP configure change IS...	BGP does not change	ISP-PE01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check IS...	BGP Route Reflector
ISP-PE02	10000		router bgp 10000	Check BGP configure change IS...	BGP does not change	ISP-PE02	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check IS...	BGP Route Reflector
DE-MUC-CR01-01	65201	192.168.11.254	router bgp 65201	Check BGP configure change DE...	BGP does not change	DE-MUC-CR01-01	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check DE...	BGP Route Reflector
JP-TYO-CR01-02	65300		router bgp 65300	Check BGP configure change JP-T...	BGP does not change	JP-TYO-CR01-02	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check JP...	BGP Route Reflector
US-BOS-R1	65001	10.10.10.10	router bgp 65001	Check BGP configure change US...	BGP does not change	US-BOS-R1	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check US...	BGP Route Reflector
US-BOS-CR01-02	65100		router bgp 65100	Check BGP configure change US...	BGP does not change	US-BOS-CR01-02	Wrapper intent for BGP Troubles...	BGP Neighbor Stability Check US...	BGP Route Reflector

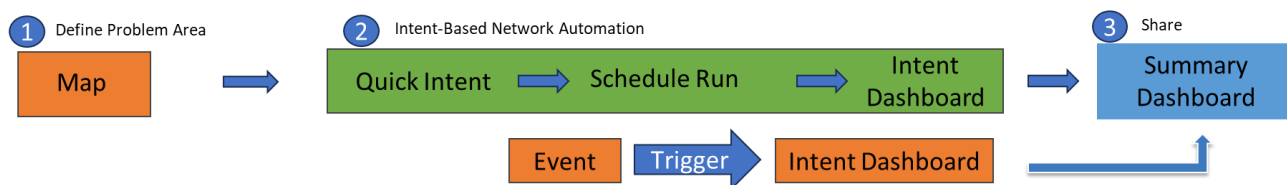
1.2 Intent-Based Automation Use Cases and Flows

Intent-based automation can be applied for many use cases. In this book, we will cover three categories of use cases which can be automated:

- Collaborative troubleshooting.
- Continuous network assessment.
- Network change and assessment.
- Map and document the network.

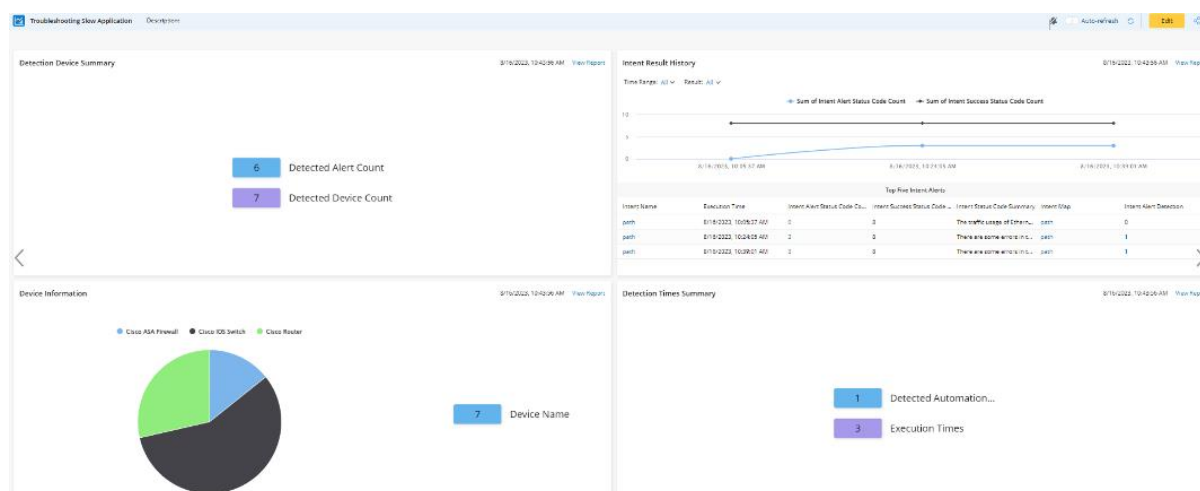
1.2.1 Intent-Based Collaborative Troubleshooting

The collaborative troubleshooting workflow starts with mapping the problem area. Different levels of users can create intents via the **Quick Intent** tab within the map and run intents to troubleshoot the same problem. The active diagnosis results and the relevant triggered event results can be shared in a **Summary Dashboard**, which summarizes the results of groups of intents.



After mapping out the problem area, a user can leverage the Quick Intent tab inside the map to quickly define an intent for a map device by following three steps: collect the data, define the logic, and run the intent. Users can repeat these three steps till they are satisfied with the results. An intent can be scheduled to run (e.g., run every minute for 20 times) to troubleshoot a transient problem.

The final quick intent can then be replicated for all devices on the map. The cloned intent can be saved as a **map intent**, and the result can be displayed on the map or in the **Intent Dashboard**.

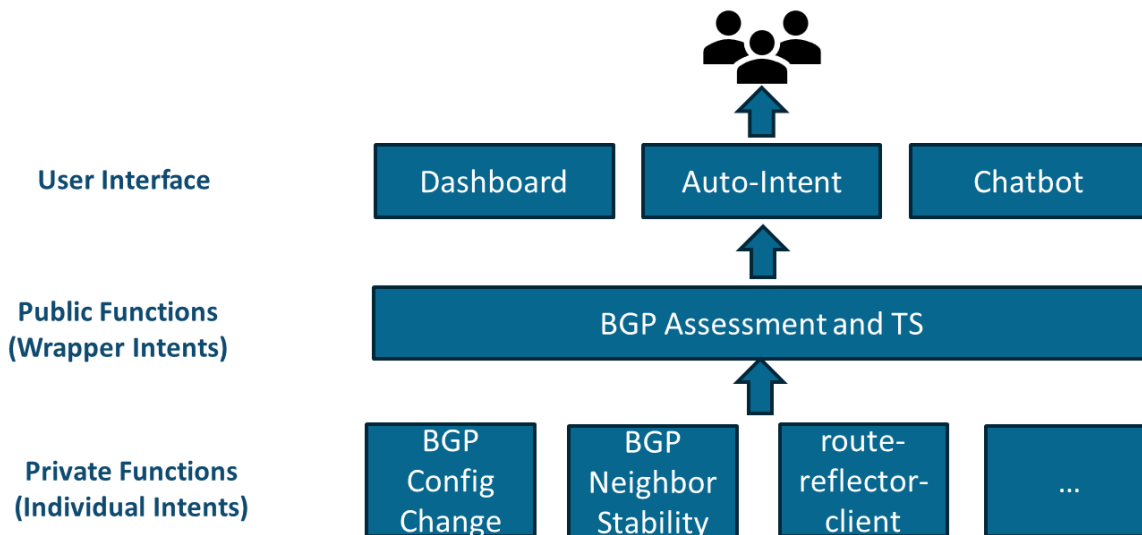


Different users at different levels can create many intents to troubleshoot the same problem. The Intent Dashboards from these intents can be added to a Summary Dashboard. The final output of the collaborative workflow is the Summary Dashboard, with each row as a diagnosis activity and each column as a device.

Dashboard and Intent Group	Device Results									
	Total Device Results	Internet	Global-Core-WAN	USA-WAN	EMEA-WAN	Distribution-WAN	NY Data Center	London Data C...	Core Network	
ServiceNow Trigger										
BGP Neighbor Stability Ch...	55 8	2 2	7 2	3 1	4 1	23 1	6 1	3	7	
BGP Routes Stability Check	67 0	4	10	4	6	25	7	4	7	
IGP Injected to BGP Redistrib...	40 0	1	5	3	3	14	5	3	6	
Level-1 NOC										
PIM Neighbor Stability Check	21 0	1	6	1	2	6	2	2	1	
Multicast Route Stability Check	15 0	1	4	1	2	3	1	2	1	
Level-2 Escalation										
Multicast RP Stability Check	21 0	1	6	1	2	6	2	2	1	
Multicast RPF Neighbor Stabili...	15 0	1	4	1	2	3	1	2	1	

The individual intents can be grouped to troubleshoot a problem. One way to do this is to create a wrapper intent, which includes all relevant intents as the follow-up intents as a public interface for

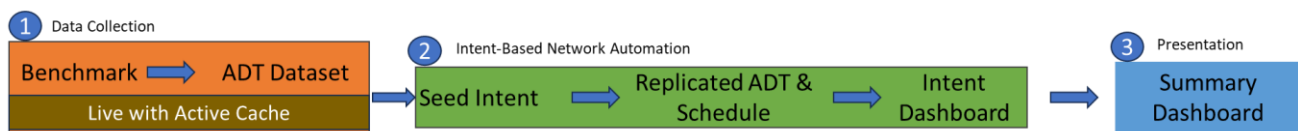
the end users so that any change of underlined intent will not change the end-user interfaces. For example, you can wrap all intents to troubleshoot the BGP-related issues, such as checking BGP config change, BGP neighbor stability, and route-reflector-client, into a wrapper intent, BGP Assessment and TS, which will be exposed to the end user interface, such as Dashboard, Auto-intent, and Chatbot.



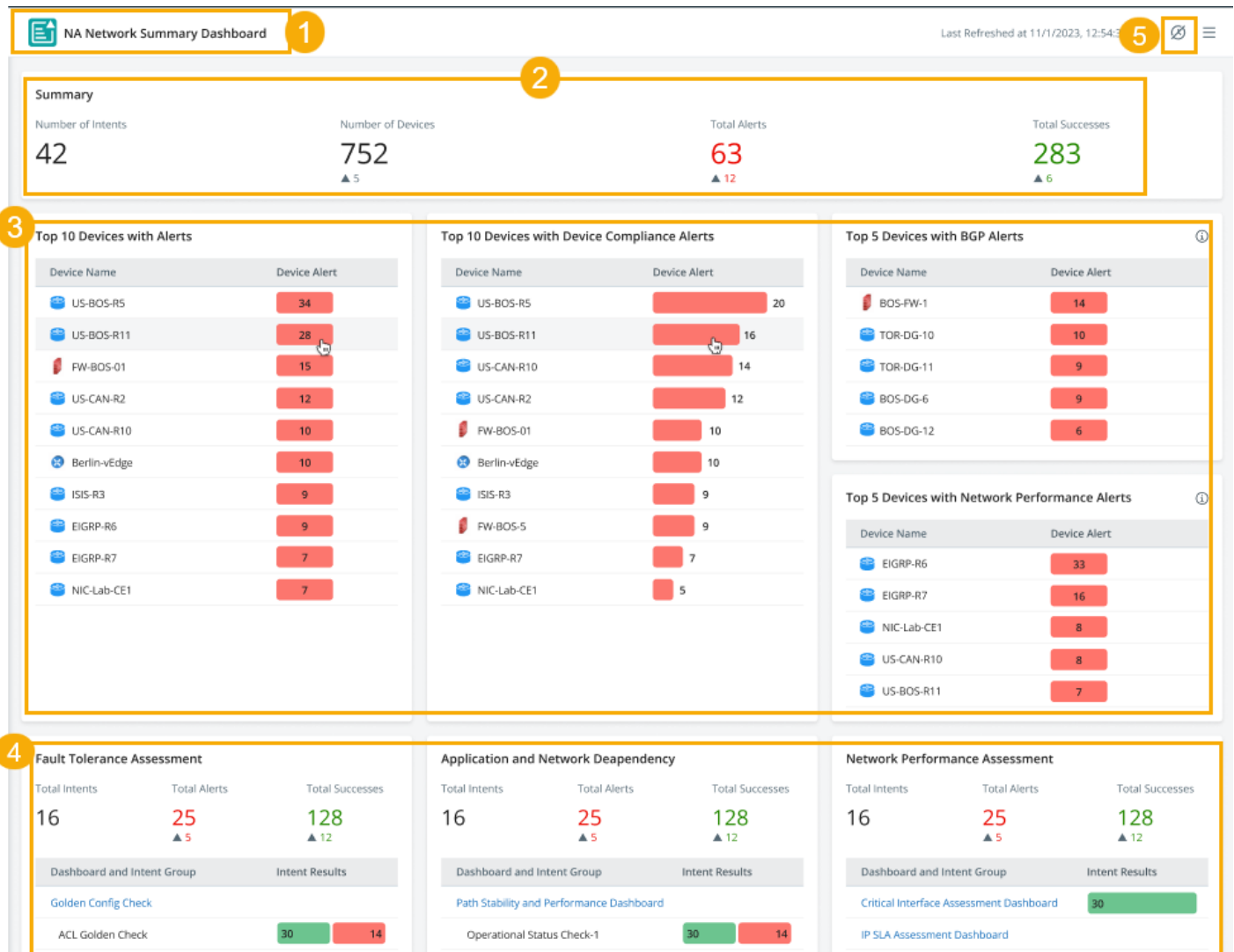
1.2.2 Continuous Network Assessment

The **Automation Data Table (ADT)** is a key component of the continuous network assessment across your whole network. You can build an **ADT Base Table** from the device group (such as all BGP devices) or from the intent template, which forms the scope for the network assessment. Then, replicate the intents to the ADT table as the **Column Group** to assess the network configurations and operational status.

In the continuous assessment workflow, the data is collected through the benchmark and saved in the ADT as the **Dataset**, which can be used to decode, replicate and run intents.



The results of the continuous assessment are presented in the Summary Dashboard. The dashboard displays the key metrics, the top devices with the most alerts, and the detail charts, with each row as an assessment point and each column as the alert counts for a device group or site.



In this book, we will study three important cases of network assessments:

- Security: check the configurations against the NIST compliance and find out the CVE vulnerability for network devices.
- Configuration Drift: check the configuration drift from the Golden template and the industry standard.
- Failover: check the configuration consistency between the failover pair, failover status, and the performance degradation after the failover.

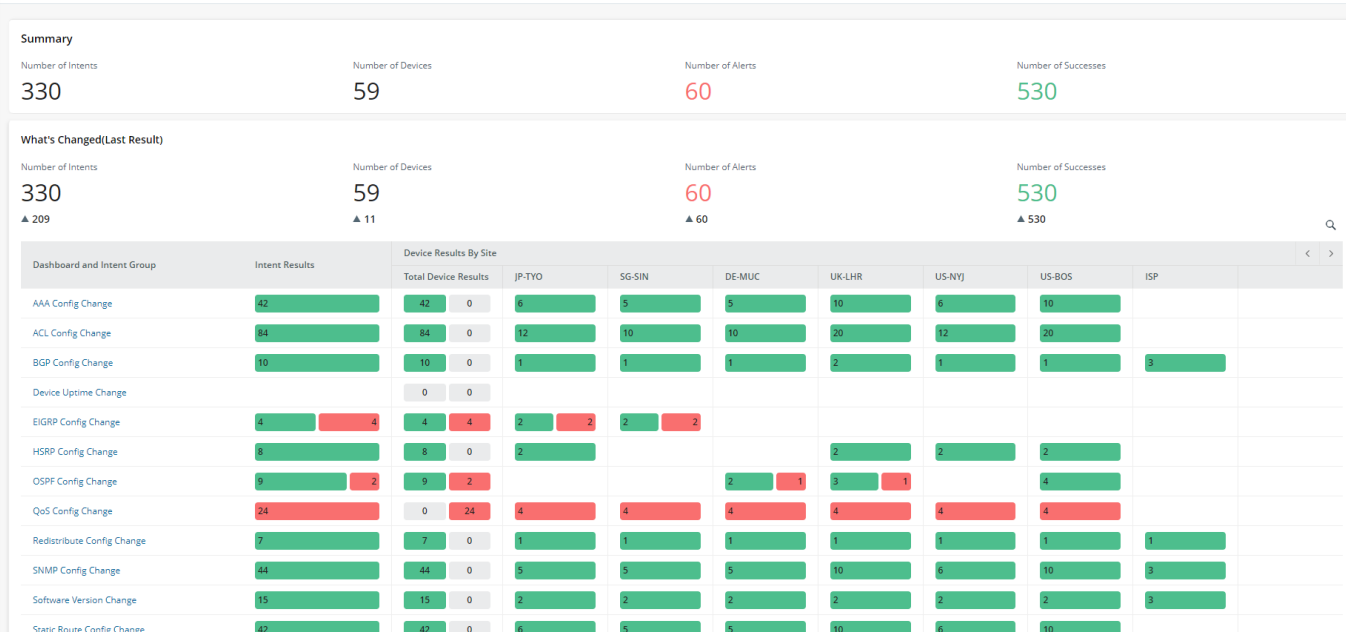
1.2.3 Change Management and Assessment

NetBrain **Change Management** module is a comprehensive solution for controlling a network change process. With intent-based automation, you can upgrade the change process with more accurate verifications of the change, for example,

The screenshot displays the NetBrain Next-Gen interface. On the left, a vertical sidebar contains navigation icons for Recents, Network, Files, Site, Path, Dashboard, Intents, Chatbot, Data, Map, and Desktop. The main workspace is divided into several panels. The top panel shows the 'Intent' tab with a workflow diagram. The workflow consists of steps: 2. Define Change, 3. Benchmark Before, 4. Auto Test Intent (Before Change), 5. Execute, 6. What is Changed Intent, 7. Auto Test Intent (After Change), 8. Benchmark After, and 9. Compare. The 'Auto Test Intent (Before Change)' step is highlighted with a red arrow pointing to a table of intents. The table has columns: Intent Name, Target Dev..., Status Code, CSV Repor..., and Actions. It lists four intents: Ping Check, Assess OSPF neighbor st, Check CPU, and Cookbook Server Farm F. Below the table, there is a filter section with 'Items:0' and a 'Filter' button. On the right side of the interface, there is a network diagram showing a central green cloud icon connected to three devices: US-BOS-R2 (Cisco Router), US-BOS-SW3 (Cisco IOS Switch), and US-BOS-SW2 (Cisco IOS Switch). The diagram includes IP addresses and interface names for each connection.

1. Run the auto-test before and after the change. You can add as many as intents to verify that the change does not violate the designs and that important operation status is normal. For example, you may want to run the batch ping for the key applications to ensure that they are still accessible.
2. Run the intent, **what is changed**, to verify that the change is successfully executed and there are no unexpected impacts. For example, for routing change, verify that no critical routes are removed.

The intent to discover **what is changed** can be useful not just for Change Management but also for troubleshooting and network assessments. You can create a dashboard to view what is changed across the whole network:



1.3 How to Use This Cookbook?

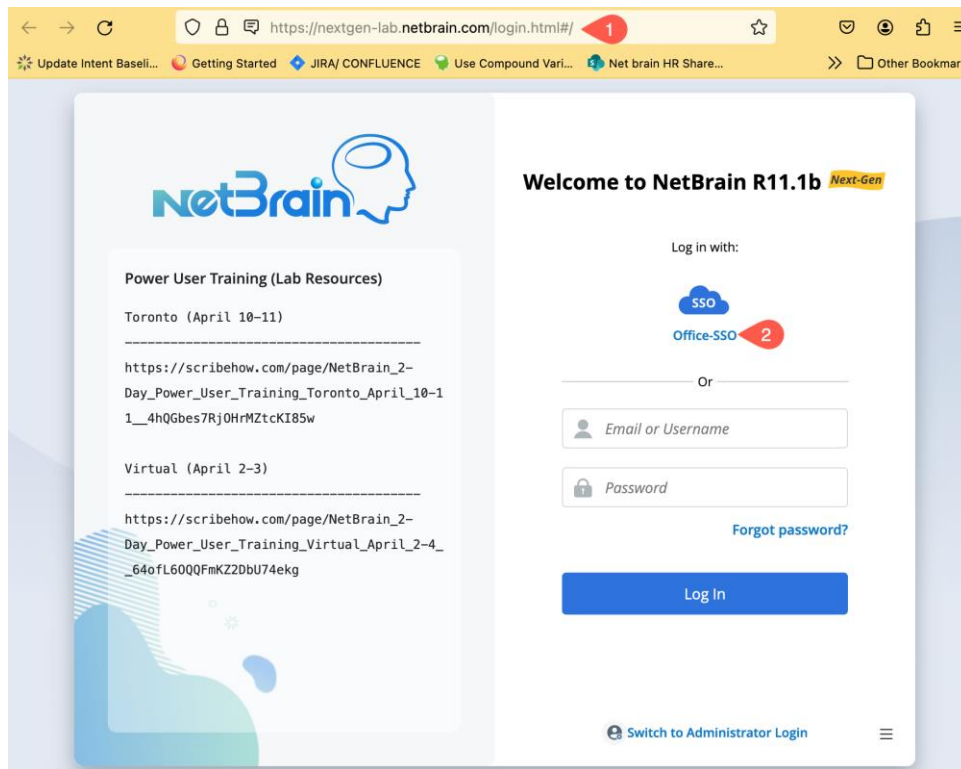
In this book, you will learn the key concepts of NetBrain **Intent-Based Automation** with real-world use cases. The book is organized into two parts:

- Chapters 2 to 5 walk you through the basics of intent-based automation, and we recommend you read all of these materials and follow the examples. In these chapters, you will learn how to automate the most frequently used commands, such as commands to probe the device and interface health. It covers the basic workflow of intent-based automation: create an intent and ADT, replicate it to auto intent and ADT, and create the dashboard and chatbot for end users.
- In chapters 6 to 12, you will apply the intent-based automation to different use cases. You can skip any of these chapters and jump to the use cases you are interested.
 - Chapters 6 to 8 study three cases of network assessments: security, configuration drift, and failover.
 - Chapter 9 covers collaboration troubleshooting, and chapter 10 teaches how to troubleshoot the path-related issues.
 - Chapter 11 studies how intent-based automation can be used to assess the impact of network change.
 - Finally, chapter 12 teaches how the intent can be used to create the map and documentation.

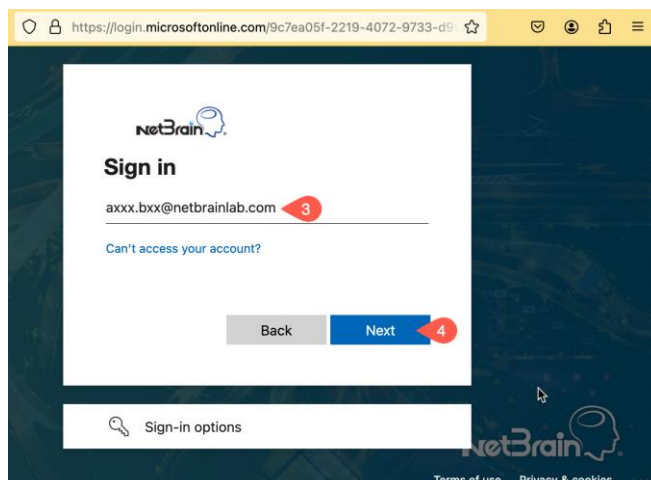
The best way for you to learn is to follow the examples step by step. You can practice using your own NetBrain system. Or, You can use the free NetBrain lab. Start by signing up for an account as detailed in the Section [Sign-in Netbrain Lab](#).

1.3.1 Sign in to Netbrain Lab for the First Time

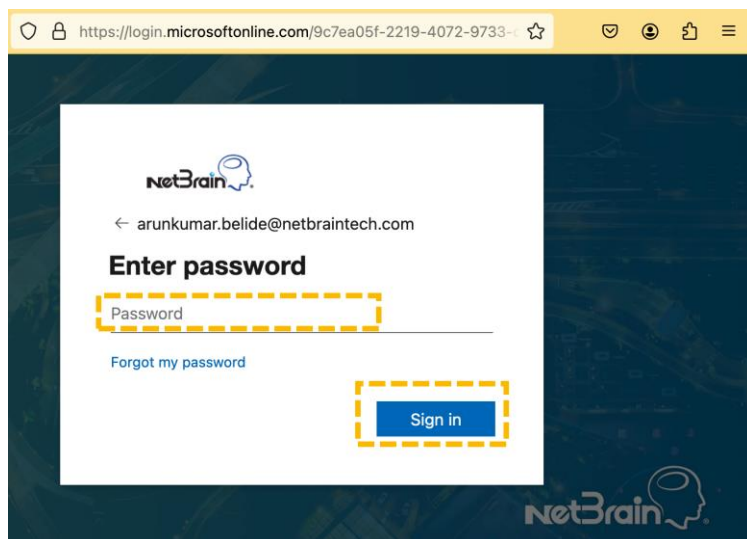
1. Navigate to nextgen-lab.netbrain.com/desktop.html in your browser.
2. Click **Office-SSO** to log in with your @netbrainlab.com account.



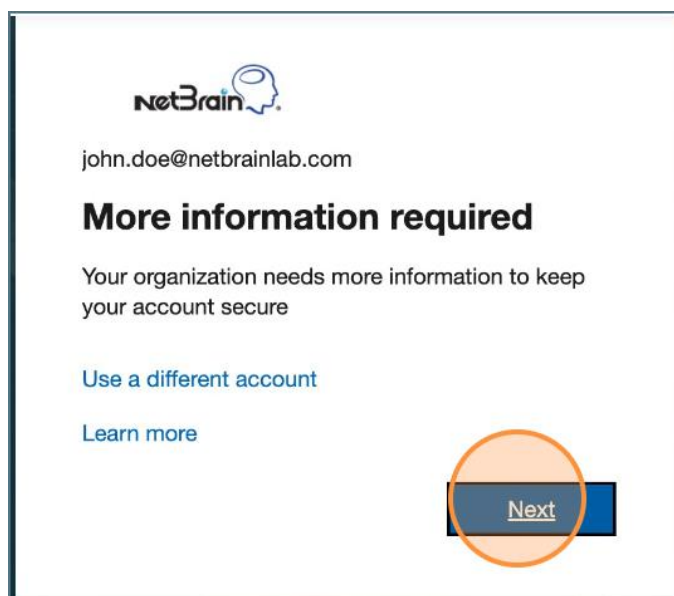
3. Enter the username. The username provided by NetBrain ends with **@netbrainlab.com**. You can reach out to NetBrain support if you do not have one.
4. Click **Next**.



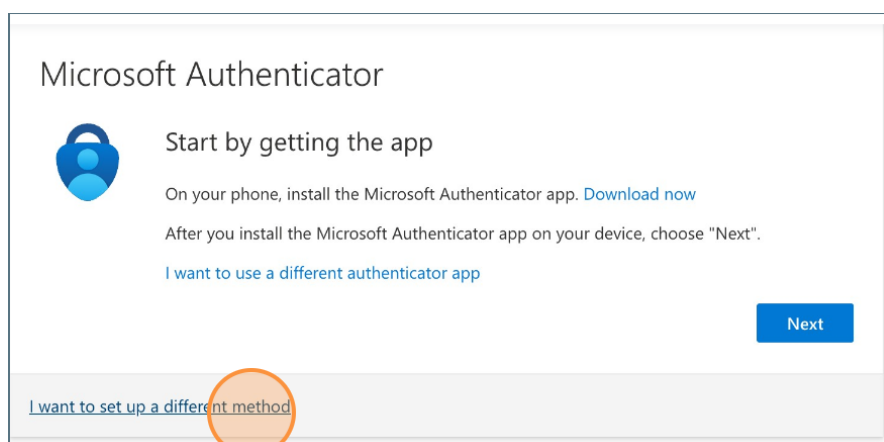
5. Enter the provided **Password** and then click the **Sign in** button.



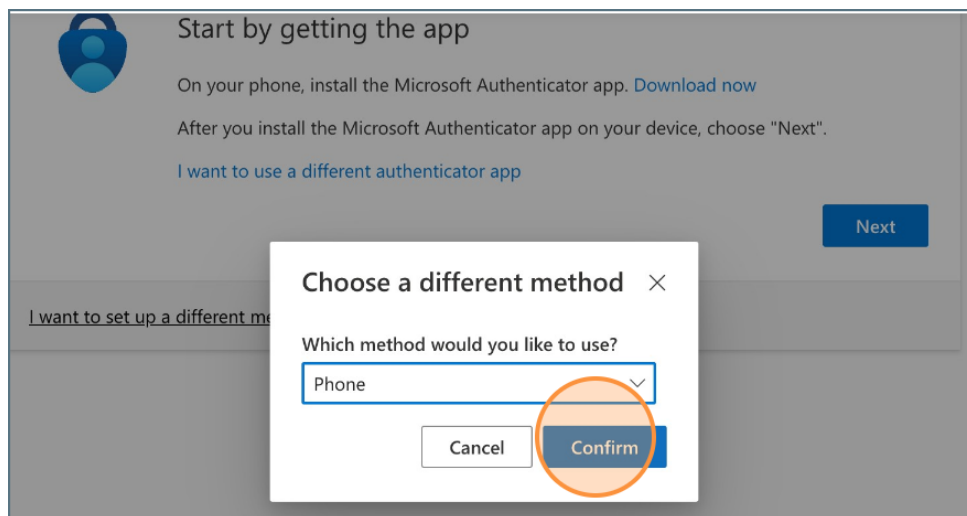
6. To set up the MFA (multi-factor authentication), click the Next button.



7. Click I want to set up a different method to set up SMS-based MFA.



8. If you want to use an Authenticator app, click **Choose a method >> Phone >> Confirm.**



Start by getting the app

On your phone, install the Microsoft Authenticator app. [Download now](#)

After you install the Microsoft Authenticator app on your device, choose "Next".

[I want to use a different authenticator app](#)

[I want to set up a different method](#)

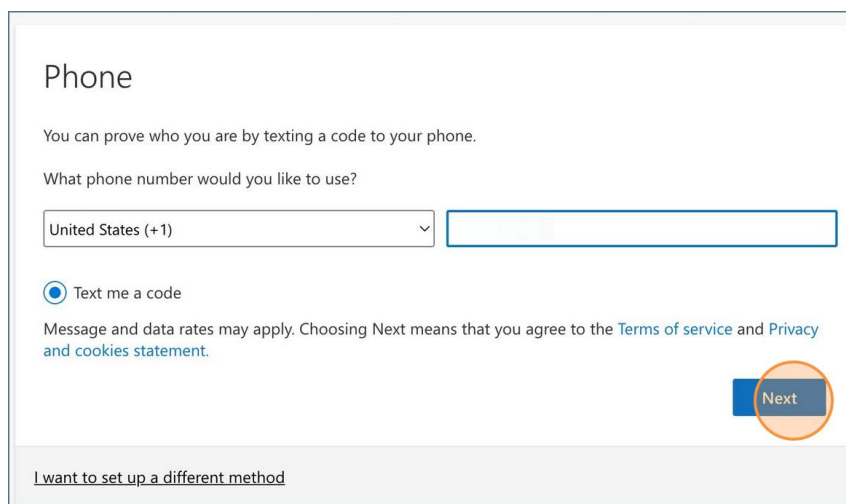
Choose a different method ×

Which method would you like to use?

Phone

Cancel Confirm

9. Enter your mobile phone number and then click **Next.**



Phone

You can prove who you are by texting a code to your phone.

What phone number would you like to use?

United States (+1)

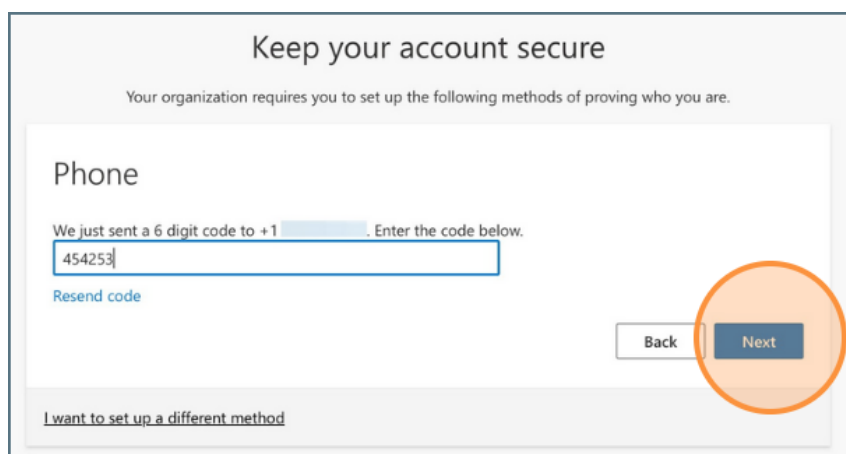
☒ Text me a code

Message and data rates may apply. Choosing Next means that you agree to the [Terms of service](#) and [Privacy and cookies statement](#).

[I want to set up a different method](#)

Next

10. Click the **Enter code** field and enter the code sent to your mobile number >> **Next.**



Keep your account secure

Your organization requires you to set up the following methods of proving who you are.

Phone

We just sent a 6 digit code to +1 . Enter the code below.

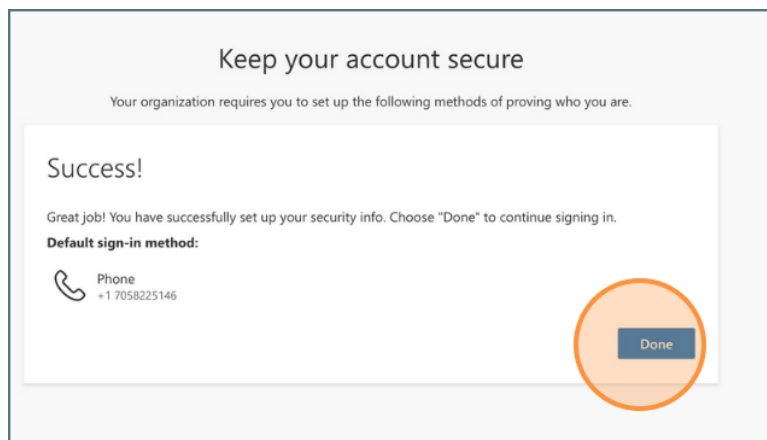
454253

[Resend code](#)

Back Next

[I want to set up a different method](#)

11. Click **Done**.



Next, you will need to change your password. Enter the provided password and a new password, and click **Sign In**.

netBrain

john.doe@netbrainlab.com

Update your password

You need to update your password because this is the first time you are signing in, or because your password has expired.

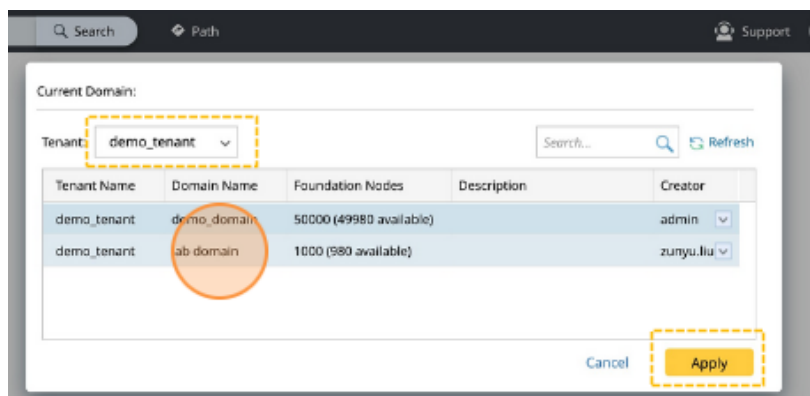
Current password

New password

Confirm password

Sign in

12. On the first login, select your Tenant **demo_tenant** >> **Lab domain** >> **Apply**.



You have now successfully setup MFA and logged into the NetBrain lab environment.

PART 1 - Intent Based Automation Essential

2 Automate Frequently Used Commands

In this chapter, you will create your first intent. You will find that it is easy to automate the frequently used CLI commands such as ***Ping <destIP>*** and ***show ip route <destIP>***. No programming knowledge is required. We will use two examples:

CLI Command	Intent Description
<i>Ping <destination ip></i>	Check whether the success rate is equal to 100% and the average round trip time is reasonable. If not, create an alert.
<i>show ip route <destination ip></i>	Compare the next hop to the last known value (baseline). If it changes, create an alert and set the baseline value.

The key steps to define an intent are:

1. Select a device, enter the CLI command, and retrieve the data.
2. [Define variables](#): from the CLI command results, find the data you are interested in and define it as the variable.
3. [Define the Diagnosis](#): compare the variable with the normal status or design and create an alert.

This chapter also covers a common intent-based automation creation flow, which has the following steps:

4. Create a new map or open an existing map.
5. Under the **Quick Intent** tag, create an intent. You can edit and run the quick intent recursively till you are satisfied with the intent results.
6. Save the intent as a map intent.
7. Create the dashboard from the intent to better view the results.

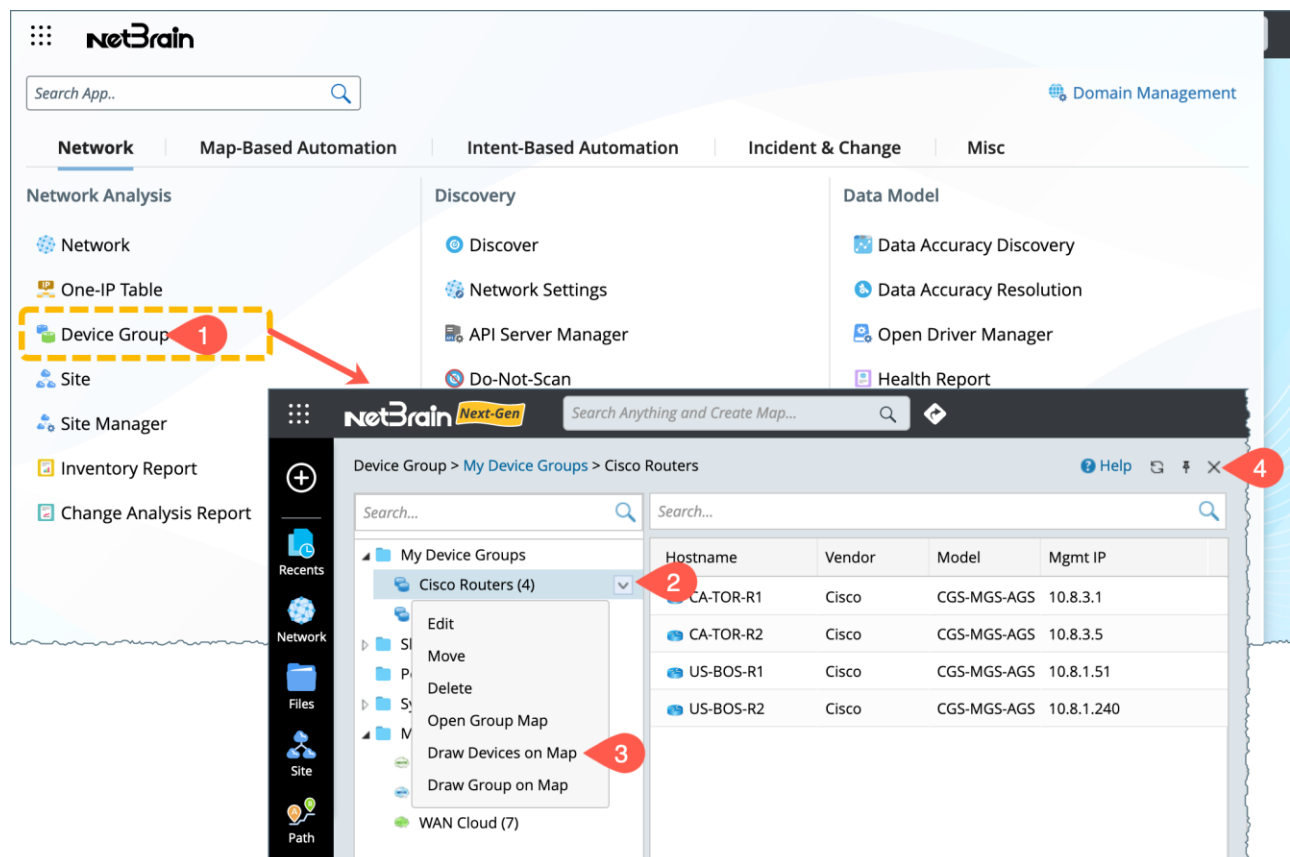
2.1 Search and Add Devices to the Map

The intent is always associated with a device or a set of devices, which are called the **seed device(s)**. You can draw these devices either by searching devices from the search bar and adding them to the map or by drawing the predefined group of devices to the map (recommended best practice).

2.1.1 Draw Devices from a Device Group

You can create a device group and draw the predefined group of devices to the map as follows:

1. Go to **Device Group** from the **start** menu ☰ from the Netbrain desktop.
2. In the device group pane, select the predefined group **Cisco Routers** and click the drop-down menu.
3. Click **Draw Devices on Map** to add the devices to a new map.
4. Close the Device group pane.

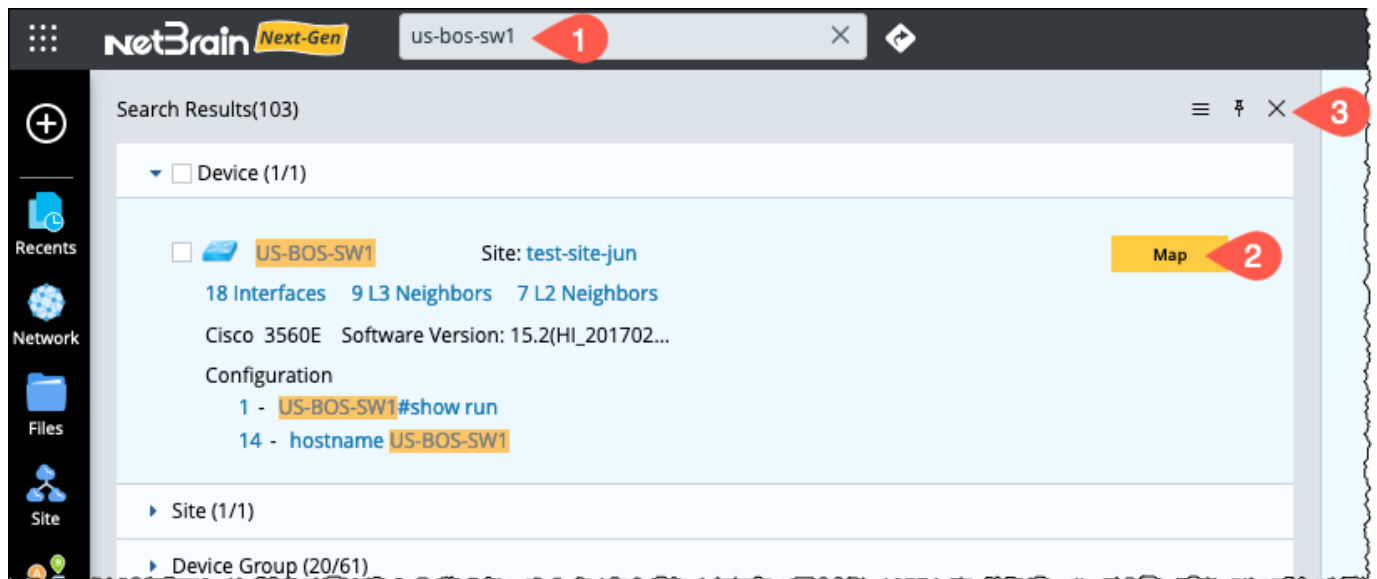


2.1.2 Draw Devices from the Search Bar

You can search and map devices from the Netbrain desktop:

Search the device **US-BOS-R1** in the search bar.

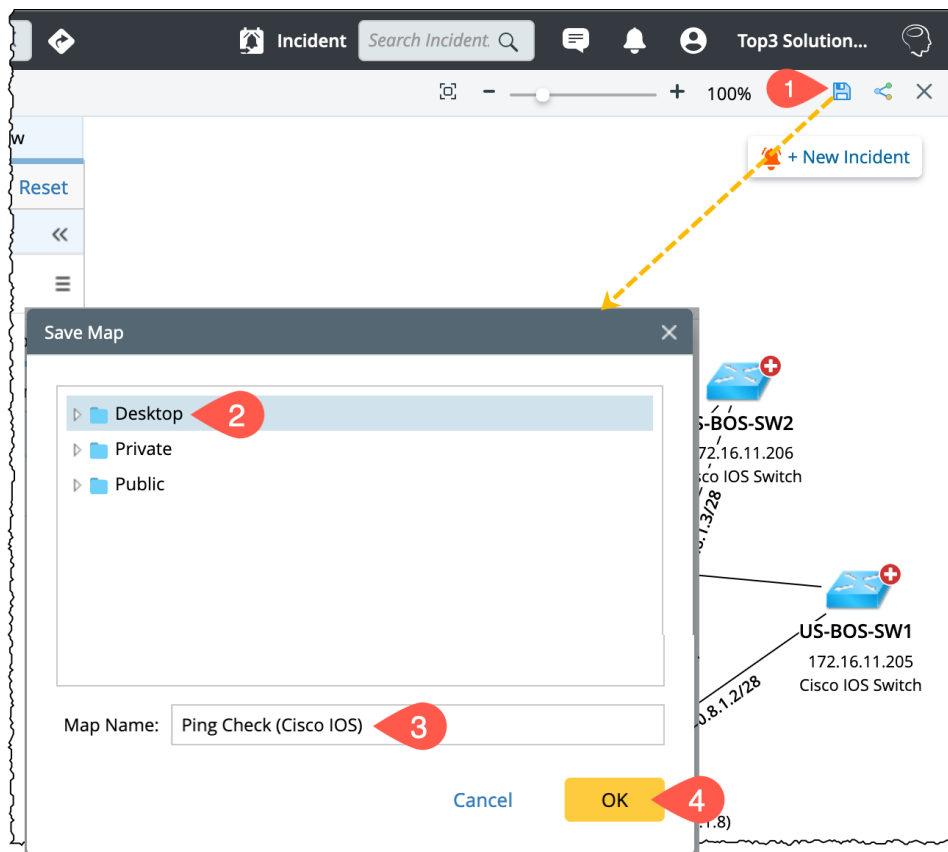
1. In the results device section, click **Map** to draw the device to the map.
2. Close the search results window and open the **Intent** pane.
3. Repeat the Step 1 thru Step 3 and add other cisco devices **US-BOS-R2** and **CA-TOR-R1**.



2.1.3 Save the Device Map

In the Upper-Right corner of the screen, click the floppy disk icon to save the current map.

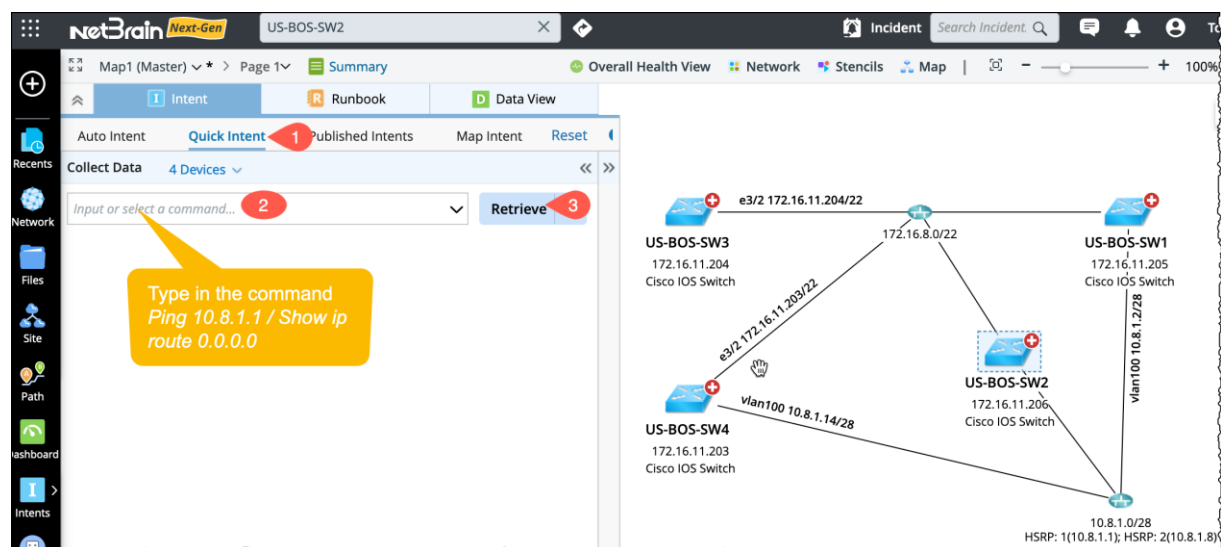
1. In the **Save Map** window, choose either a Desktop or another location to save the map.
2. Add a name [Ping Check (Cisco IOS) / Route Check (Cisco IOS)] to the map.
3. Click **OK** to save and close the window.



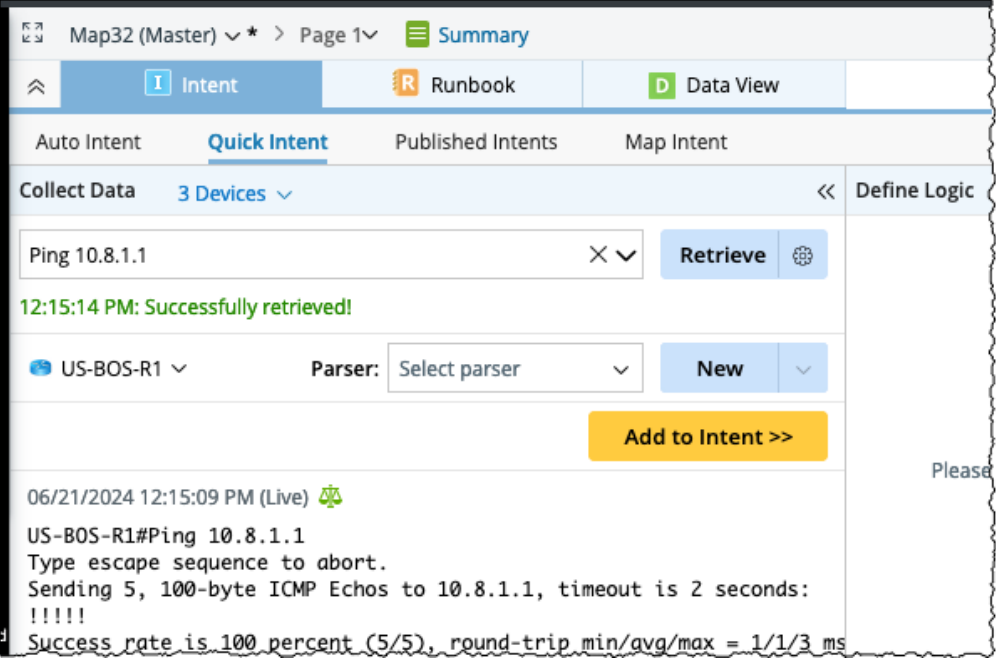
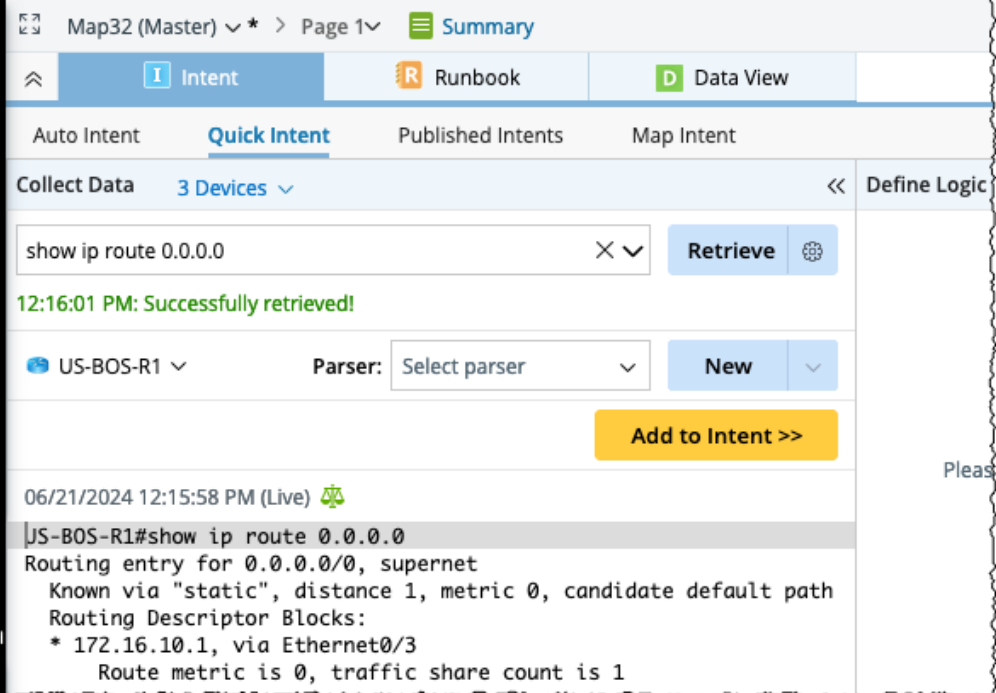
2.2 Quick Intent Creation

From the map you just created,

1. Go to the **Quick Intent** tab in the Intent section.
2. Enter the command **Ping 10.8.1.1** in **Input or Select a command...** field to collect data. For another example, you enter the command **show ip route 0.0.0.0**.
3. Click **Retrieve** to parse the data.



4. The retrieved data sample will be:

CLI	Retrieve data Sample
Ping <destination ip>	 <p>The screenshot shows the NetBrain 'Quick Intent' interface. The command 'Ping 10.8.1.1' has been entered and executed successfully on device 'US-BOS-R1'. The output shows a 100% success rate with a round-trip time of 1/1/3 ms. The interface includes tabs for 'Intent', 'Runbook', and 'Data View', and a 'Collect Data' section with a 'Retrieve' button.</p>
show ip route <destination ip>	 <p>The screenshot shows the NetBrain 'Quick Intent' interface. The command 'show ip route 0.0.0.0' has been entered and executed successfully on device 'US-BOS-R1'. The output shows the routing entry for 0.0.0.0/0, known via static, with a distance of 1 and metric of 0. The interface includes tabs for 'Intent', 'Runbook', and 'Data View', and a 'Collect Data' section with a 'Retrieve' button.</p>

2.3 Define Variables with a Visual Parser

Defining variables is simple with **Quick Intent** using the auto-parser feature. Entry-level users can quickly learn and utilize the parser. The system will automatically select the appropriate parser mode (**single** or **multiple**) based on input words.

Once the auto-parser completes the parser definition task, you can add additional variables or make adjustments to the **line pattern** to achieve the desired result.

2.3.1 Define parser for ping <destination ip>

You can define a variable with the following steps:

1. Create and add a parser using the option **New** located next to the **Parser** field.

Map32 (Master) > * > Page 1 > Summary

Intent Runbook Data View

Auto Intent Quick Intent Published Intents Map Intent

Collect Data 3 Devices << Define Logic

Ping 10.8.1.1 X v Retrieve

12:15:14 PM: Successfully retrieved!

US-BOS-R1 v Parser: Select parser 1 New v

Add to Intent >>

06/21/2024 12:15:09 PM (Live)

US-BOS-R1#Ping 10.8.1.1

Type escape sequence to abort.

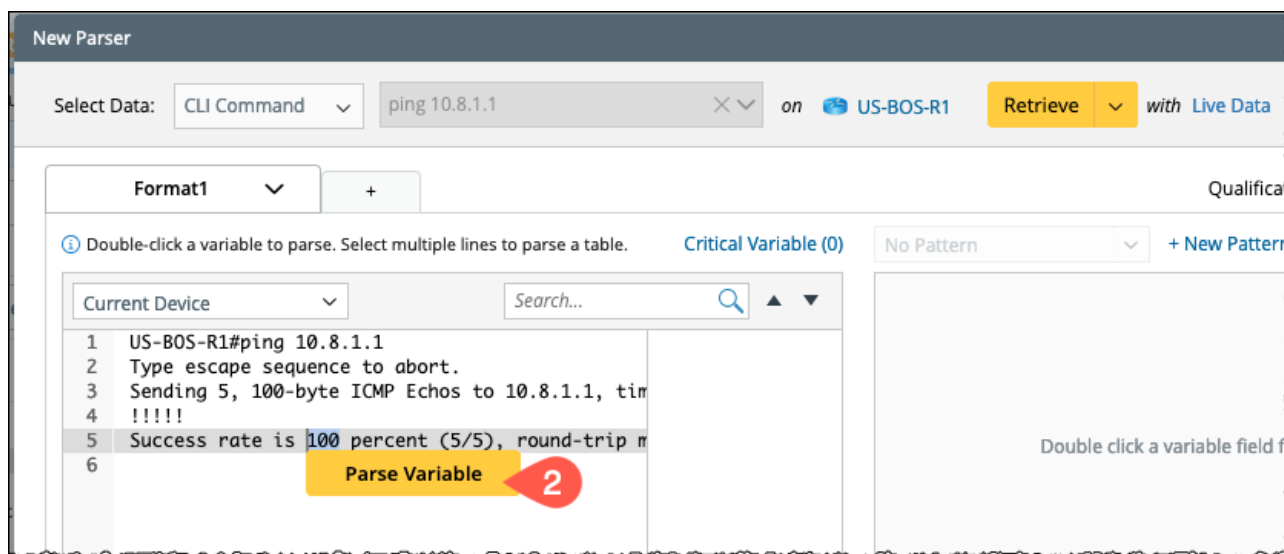
Sending 5, 100-byte ICMP Echos to 10.8.1.1, timeout is 2 seconds:

!!!!!

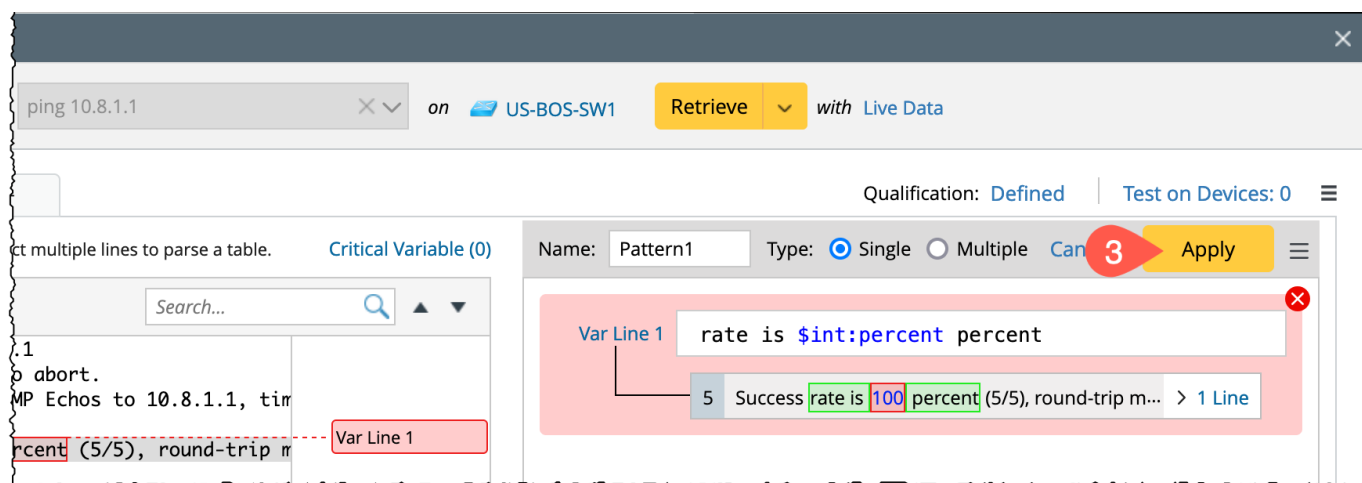
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/3 ms

Please

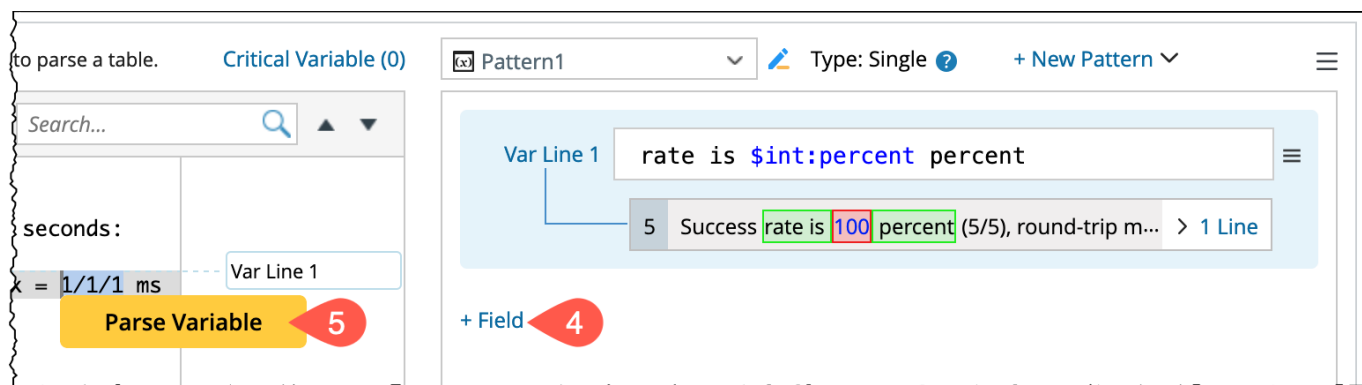
2. Select the text **100** in the sentence **rate is 100 percent** and click **Parse Variable** in the tip window. You can also double-click the text to get the same result.



3. Click **Apply**.



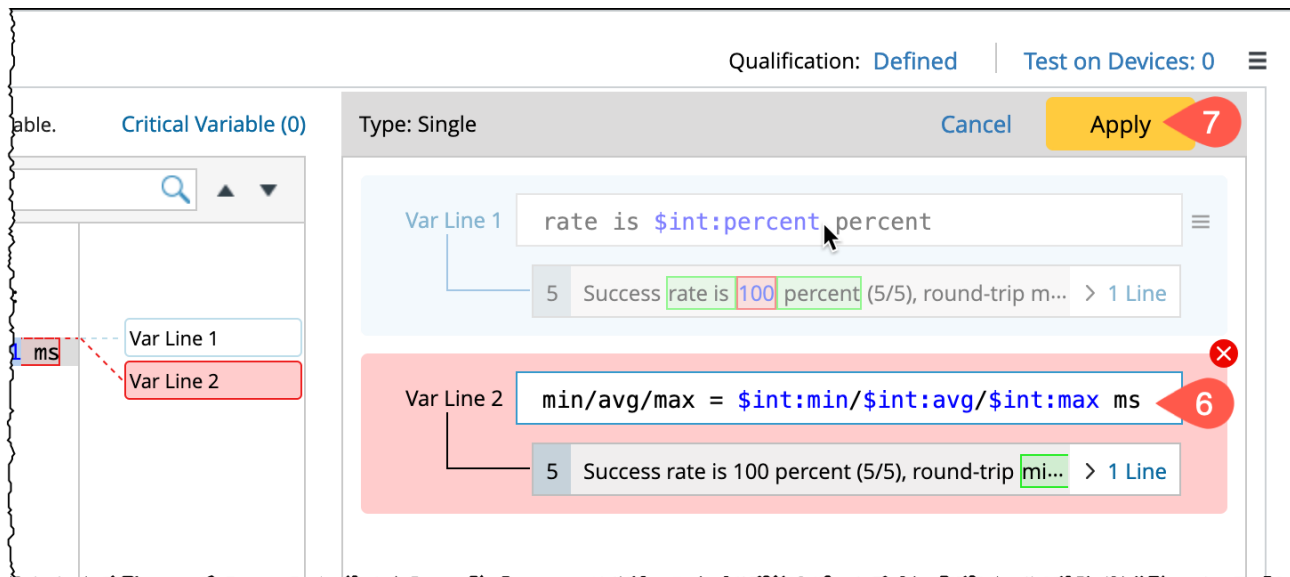
4. Click **+Field** to another variable for round trip time.
5. Similar to Step 2, select the round-trip text value **1/1/2**.



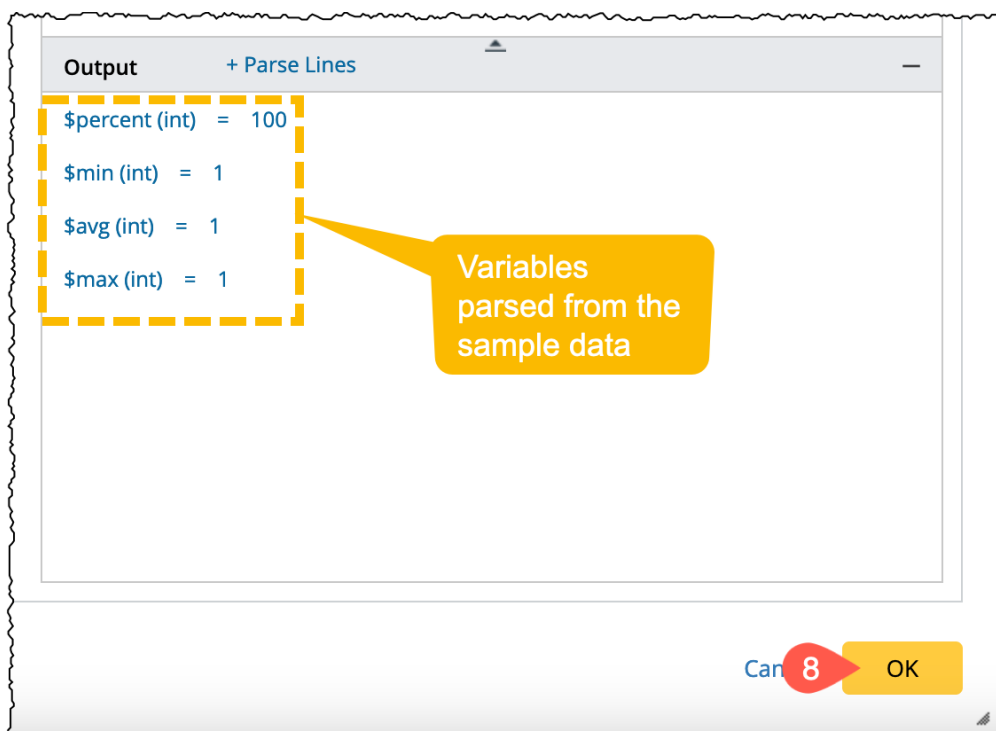
6. Modify the Variable line pattern to `min/avg/max = $int:min/$int:avg/$int:max ms`.

NOTE: The variable name is defined as `$<type>:<variable_name>`. The default variable type is string, and the type mstring is a string with spaces.

7. Click **Apply**.



8. Select **OK** to save and close the window.



9. Click **Add to Intent**, and the parsed variables are added to the diagnosis.

The screenshot displays the NetBrain interface with the 'Intent' tab selected. The 'Collect Data' section shows a command 'ping 10.8.1.1' being executed on device 'US-BOS-SW1'. A red circle with the number '9' highlights the 'Add to Intent >>' button. A red dashed arrow points from this button to the 'Define Logic' section, specifically to the 'D 1 Diagnosis' entry. The 'Define Logic' section shows the command 'ping 10.8.1.1' being added to the diagnosis. The 'Diagnosis' section shows the command 'ping 10.8.1.1' being added to the diagnosis. The 'Diagnosis' section also shows the command 'ping 10.8.1.1' being added to the diagnosis.

Map2 (Master) > * > Page 1 > Summary Overall Health View Network Stencils

Intent Runbook Data View

Auto Intent Quick Intent Published Intents Map Intent Reset Help

Collect Data 4 Devices << Define Logic <<

ping 10.8.1.1 X Retrieve

11:28:39 PM: Successfully retrieved!

US-BOS-SW1 Parser: Parser1 P1 Edit

9 Add to Intent >>

07/19/2024 11:28:10 PM (Live) Successfully parsed all 4 variables.

US-BOS-SW1#ping 10.8.1.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.8.1.1, timeout is 2 seconds:
!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

US-BOS-SW1 ping 10.8.1.1 P1 D 1 Diagnosis

Create

Diagnosis: D Diagnosis 1 + New Diagnosis

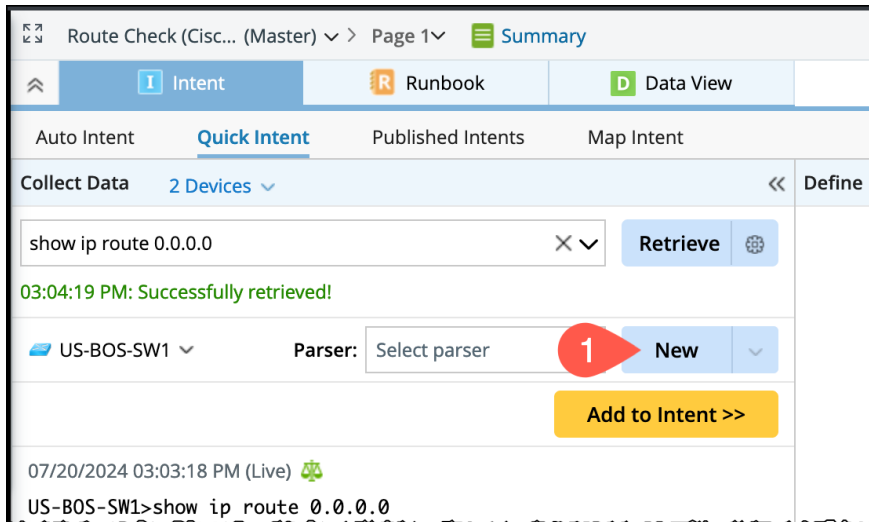
Name: Diagnosis 1 Anchor: Select Variable

Type description of the diagnosis

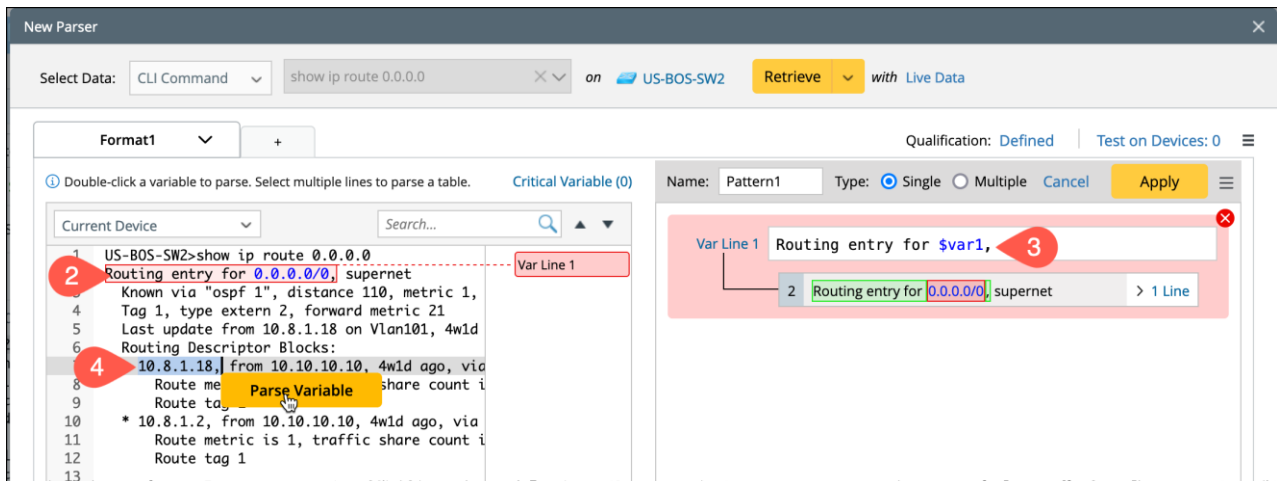
2.3.2 Define parser for Command *Show ip route*

You can define a variable with the following steps:

1. Create and add a parser using the option **New** located next to the **Parser** field.



2. Select the text **Routing Entry For 0.0.0.0/0**, and click **Parse Variable** in the tip window. You can also double-click the text to get the same result.
3. Modify the Variable line pattern to **Routing entry for \$subnet,**.
4. Similar to Step 2, copy the next hop line **10.8.1.18** from the sample data and parse the variable.

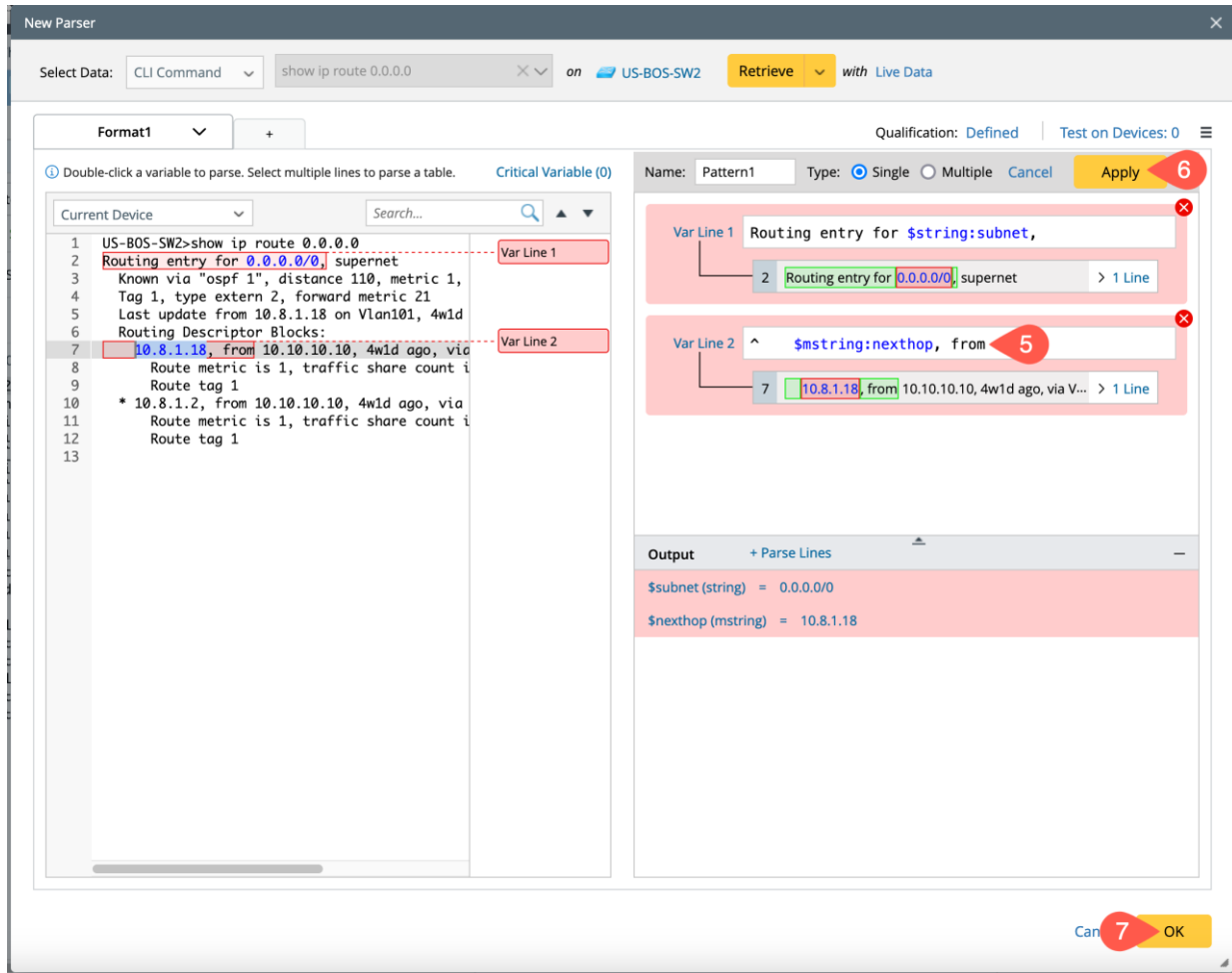


5. Modify the Variable line pattern to **^\$mstring:nexthop, from.**

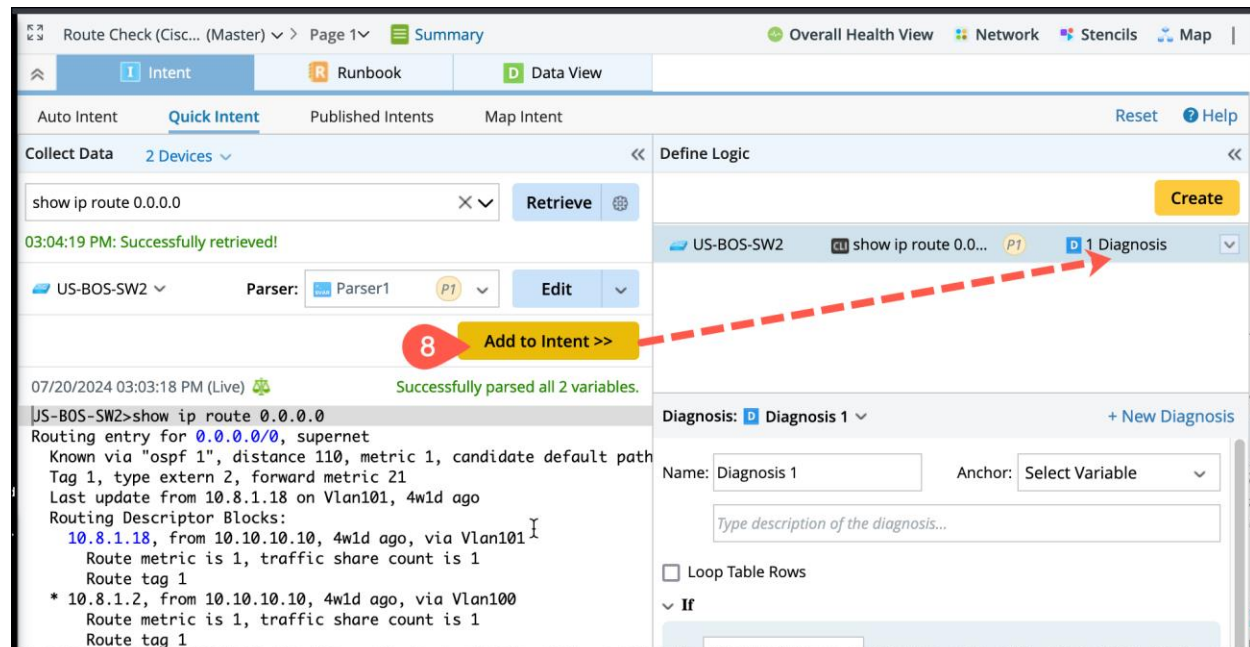
NOTE: The variable name is defined as \$<type>:<variable_name>. The default variable type is string, and the type mstring is a string with spaces. The created variables will appear under the Parsed Result section.

6. Click **Apply**.

7. Select **OK** to save and close the window.



8. Click **Add to Intent**, and the parsed variables are added to the diagnosis.



2.4 Define the Diagnosis

Go to the **Define Logic** section to define the diagnosis logic as follows:

1. Enter the diagnosis name, e.g., **Ping Cisco** or **Subnet next hop check**.
2. Define the condition as detailed in the following table and images:

Intent	Steps
Ping < destination ip>	<ol style="list-style-type: none"> a. Variable percent does not equal 100. b. Variable avg is Greater than 60. c. Boolean Expression: A or B.

The screenshot shows the NetBrain R11.1b interface. The top navigation bar includes 'Intent', 'Runbook', and 'Data View'. The 'Intent' section is active, showing 'Collect Data' and 'Define Logic' tabs. The 'Define Logic' tab is selected, and the 'Name' field is set to 'Ping Cisco' (labeled 2). The 'Anchor' field is set to '\$percent'. The 'Loop Table Rows' section (labeled 3) contains an 'If' condition. The 'If' condition has three rows: 'A' with 'percent' (labeled a) 'Does not ...' '100', 'B' with 'avg' (labeled b) 'Greater th...' '60', and 'C' with 'Select Variable'. The 'Boolean Expression' field is set to 'A or B' (labeled c).

Intent	Steps
show ip route <destination ip>	<ol style="list-style-type: none"> The variable subnet Is not empty. Variable nexthop (Current) Does not equal nexthop (Baseline). Boolean Expression: A and B.

Add Note
Add Diagnosis
Can also click a variable on the left to add automation.

Name: Subnet nexthop check 2
Anchor: ▼

Type description of the diagnosis...

☐ Loop Table Rows

▼ If 3

A CA-TOR-R1 Current ▼

a

subnet ▼

Is not empty ▼

🗑️

B CA-TOR-R1 Current ▼

b

nexthop ▼

Does not equal ▼

Baseline ▼

nexthop ▼

🗑️

C Select Variable ▼

Boolean Expression: A and B c

NetBrain R11.1b | 33

2.5 Define Intent Output

Enter a message under the **Then** and **Else** output areas to appear as the result of the diagnosis.

1. **Then:** Define a color, message and status in case **If** condition is true, as shown in the figure.
2. Click the checkbox of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

Intent	Message
Ping	<i>\$this_device</i> ping failed, and the average round trip time is <i>\$avg</i> ms

Then

Diagnosis Message:

s_device ping failed, and the average round trip time is *\$avg*.

2

☒

Set Status Code for Device:

Error

ce ping failed, and the average round trip time is *\$avg*.

2

☒

Set Status Code for Intent:

Error

ice ping failed, and the average round trip time is *\$avg*

Add Logic

Else

Diagnosis Message:

this_device ping success, and the average round trip time is *\$*

4

☒

Set Status Code for Device:

Success

\$this_device ping success, and the average round trip

4

☒

Set Status Code for Intent:

Success

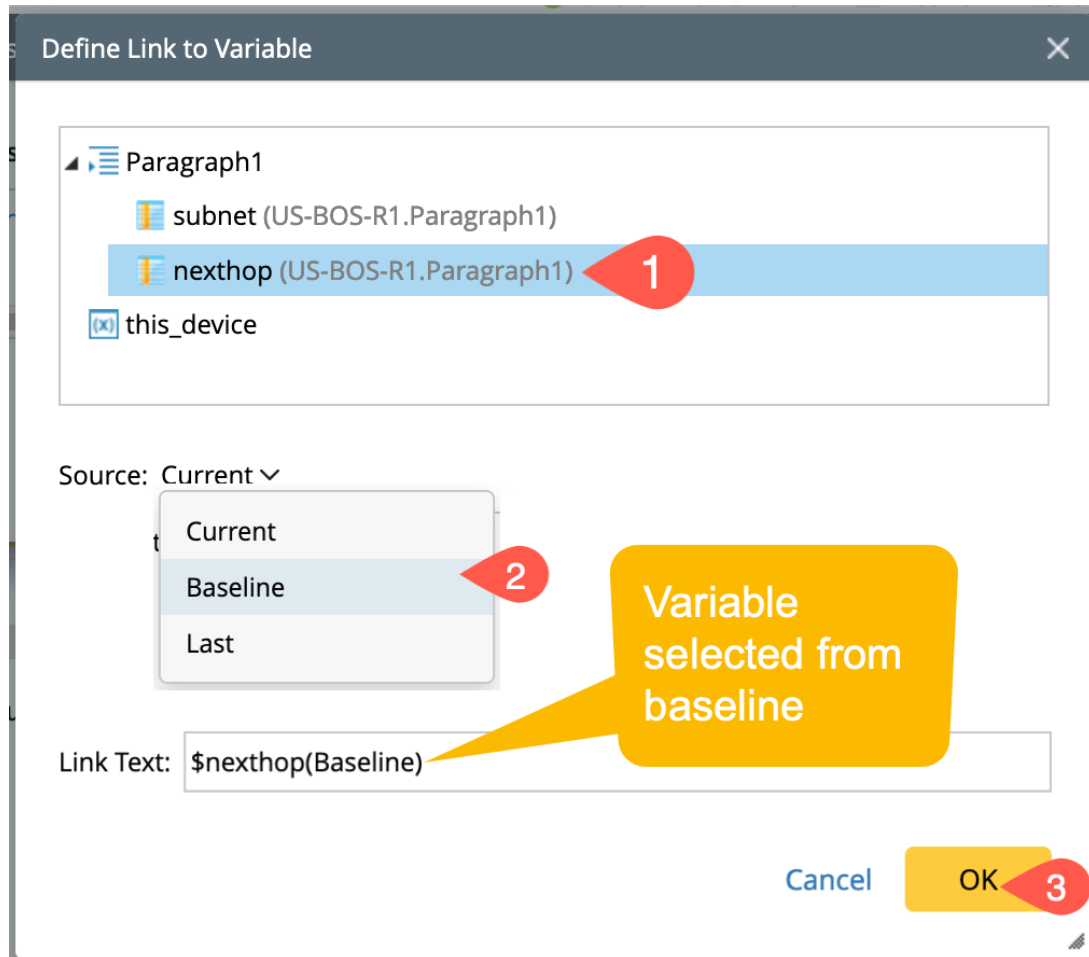
\$this_device ping success, and the average round trip

+ Elself

Intent	Message
Show ip route	On <i>\$this_device</i> subnet <i>\$subnet</i> nexthop <i>\$nextHop</i> is changed (<i>\$nextHop(Baseline)</i>).

NOTE:

- By typing \$, you can get the variable selection pop-up.
- For a variable, you can get the current value and baseline value.



3. **Else:** Define a color, message and status in case **If** condition is not true, as shown in the figure.
4. Click the checkbox of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

Intent	Message
Ping	<i>\$this_device</i> ping succeeded, and the average round trip time is <i>\$avg</i>

Show ip route

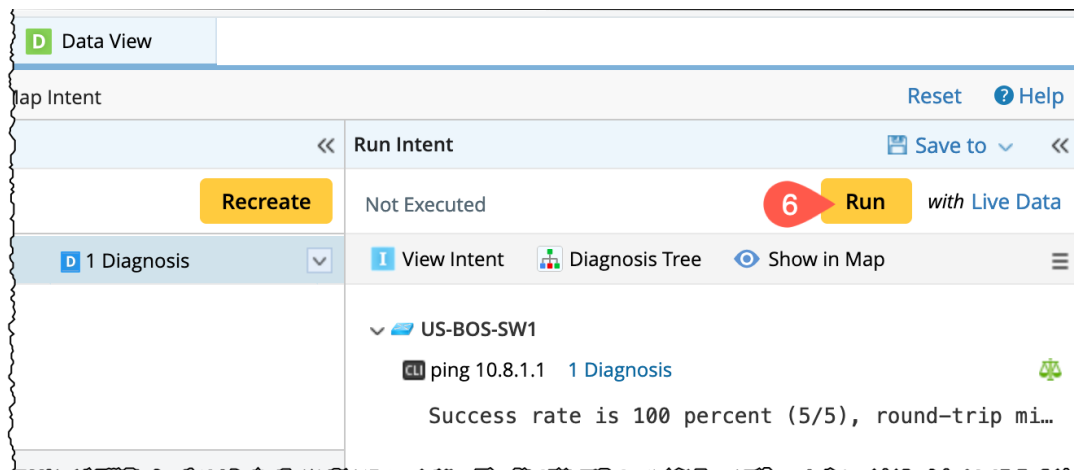
On *\$this_device* subnet *\$subnet* nexthop *\$nextHop* is not changed (*\$nextHop(Baseline)*).

The screenshot shows a configuration window with two main sections: 'Then' and 'Else'. The 'Then' section is highlighted with a red circle '1'. It contains a 'Diagnosis Message' field with a dropdown menu and a text box containing 'On *\$this_device* subnet *\$subnet* nexthop *\$nextHop* is changed (*\$nextHop(Baseline)*)'. Below this are two 'Set Status Code' fields, both with a dropdown menu and a text box containing 'On *\$this_device* subnet *\$subnet* nexthop *\$nextHop* is changed (*\$nextHop(Baseline)*)'. The 'Else' section is highlighted with a red circle '3'. It contains a 'Diagnosis Message' field with a dropdown menu and a text box containing 'On *\$this_device* subnet *\$subnet* nexthop *\$nextHop* is not changed (*\$nextHop(Baseline)*)'. Below this are two 'Set Status Code' fields, both with a dropdown menu and a text box containing 'On *\$this_device* subnet *\$subnet* nexthop *\$nextHop* is not changed (*\$nextHop(Baseline)*)'. The 'Add Logic' button is visible at the bottom. Red circles with numbers 2 and 4 point to the 'Set Status Code' fields in the 'Then' and 'Else' sections respectively.


5. Click on **Create** to save and create the intent.

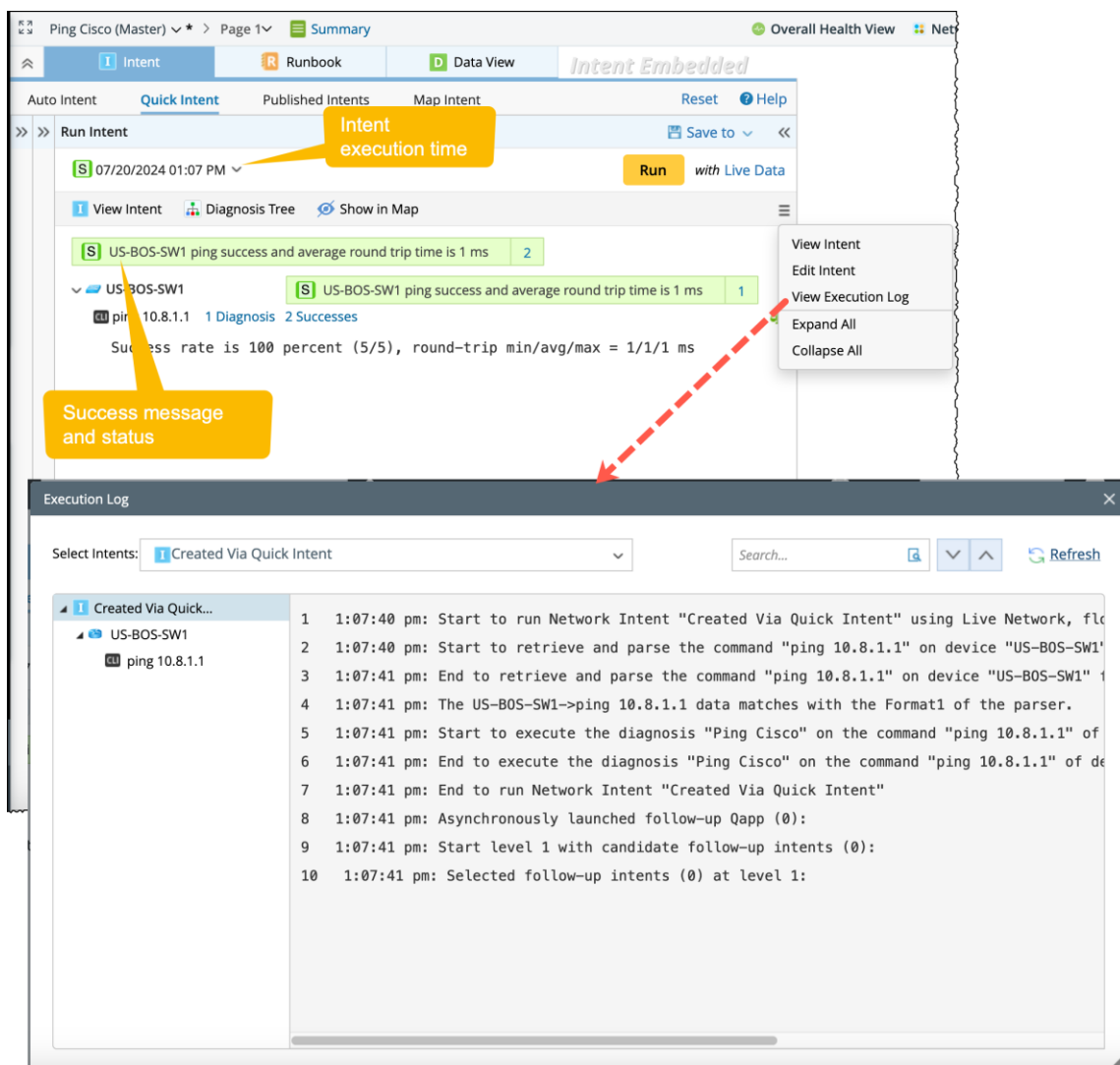
The screenshot shows the 'Quick Intent' configuration interface. The 'Collect Data' section on the left has a dropdown menu set to 'US-BOS-R1' and a text box containing 'ping 10.8.1.1'. The 'Define Logic' section on the right has a dropdown menu set to 'US-BOS-R1' and a text box containing 'show ip route 0.0.0.0'. The 'Create' button is highlighted with a red circle '5'. The 'Add to Intent >>' button is visible at the bottom. The 'Parser' dropdown is set to 'Show ip ro...'. The 'Diagnosis' dropdown is set to 'Diagnosis 1'. The 'Add to Intent >>' button is highlighted with a red circle '5'.

6. Go to the **Run Intent** pane and click **Run** to execute the diagnosis.



Upon completing the diagnosis by the system, the following result will appear:

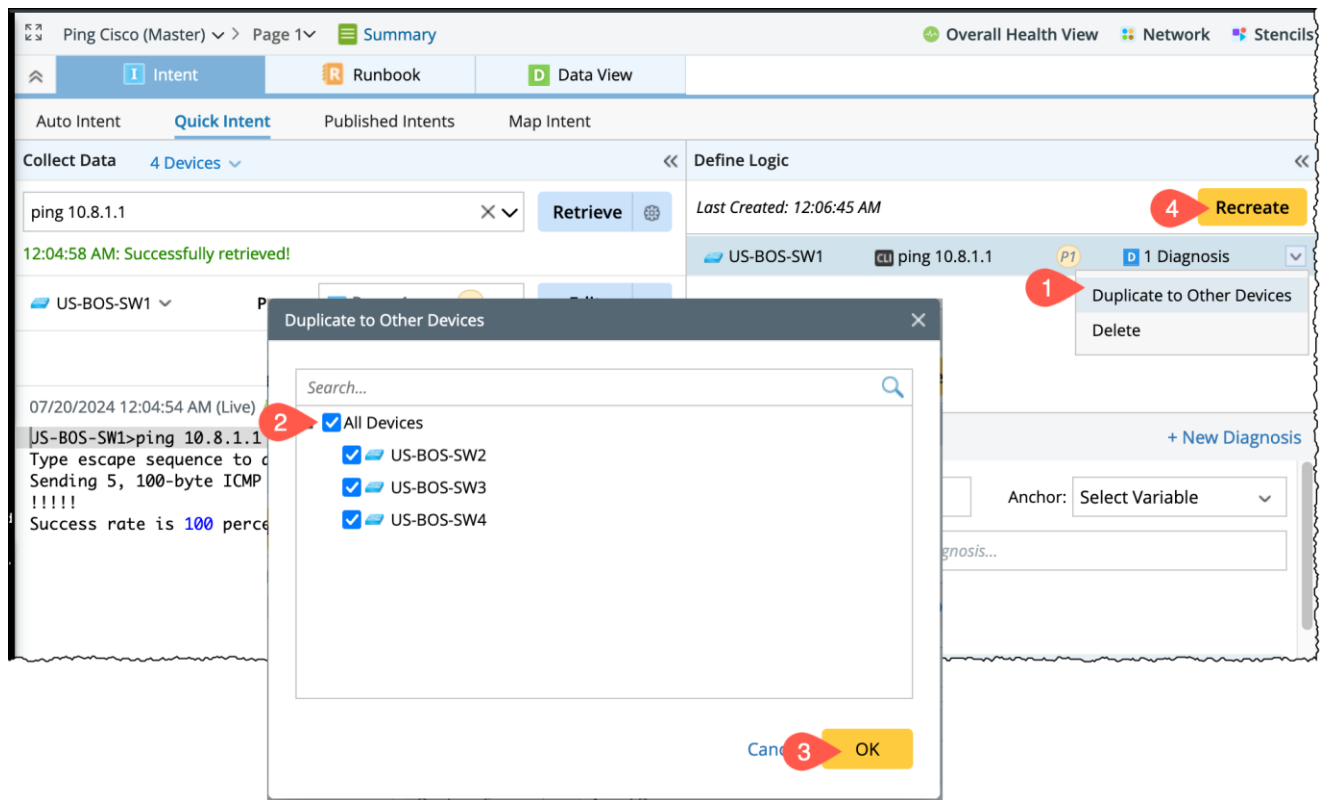
- Intent execution date and time.
- Success/Failure message and status as defined in **Then** and **Else** conditions.
- The execution log can be accessed using the **View Execution Log** located under :



2.6 Duplicate the Intent to Other Devices

An intent can be duplicated to other devices. The system will automatically copy the parser and logic to other devices.

1. You can duplicate the created intent to other devices as follows:
2. Go to the **Define Logic** pane, and from the drop-down menu, select the option **Duplicate to Other Devices**.
3. Select all the devices to which you want to copy the intent.
4. Click **OK** to save.



5. All the selected devices will be added to the list and click **Recreate**.
6. In the **Run Intent** pane, all the devices will be listed with intent configured.
7. Click **Run** to execute the intent diagnosis on all the devices. And the results are displayed.

Define Logic << >> Reset Help

Last Created: 12:14:51 AM 4 **Recreate**

Device	Command	Status	Diagnosis
US-BOS-SW1	cli ping 10.8.1.1	P1	1 Diagnosis
US-BOS-SW2	cli ping 10.8.1.1	P1	1 Diagnosis
US-BOS-SW3	cli ping 10.8.1.1	P1	1 Diagnosis
US-BOS-SW4	cli ping 10.8.1.1	P1	1 Diagnosis

Diagnosis: 1 Ping Cisco + New Diagnosis

Name: Ping Cisco Anchor: Select Variable

Type description of the diagnosis...

☐ Loop Table Rows

If

A US-B... Current

percent Does not ... 100

B US-B... Current

Run Intent Save to << >> Not Executed 6 **Run** with Live Data

I View Intent Diagnosis Tree Show in Map

- US-BOS-SW1
 - cli ping 10.8.1.1 1 Diagnosis
 - Success rate is 100 percent (5/5), roun...
- US-BOS-SW2
 - cli ping 10.8.1.1 1 Diagnosis
 - Success rate is 100 percent (5/5), roun...
- US-BOS-SW3
 - cli ping 10.8.1.1 1 Diagnosis
 - Success rate is 100 percent (5/5), roun...
- US-BOS-SW4
 - cli ping 10.8.1.1 1 Diagnosis
 - Success rate is 100 percent (5/5), roun...

8. In the **Run Intent** pane, click **Save to > Save to Map Intent** and then **OK**.

Run Intent Save to << >> Reset Help

Recreate 7 **Save to Map Intent**

S 07/20/2024 12:17 AM 1 **Save to Path Intent**

I View Intent Diagnosis Save to Common Intent

S US-BOS-SW1 ping success and average round tri... 8

US-BOS-SW1 S US-BOS-SW1 ping success an... 1

cli ping 10.8.1.1 1 Diagnosis 2 Successes

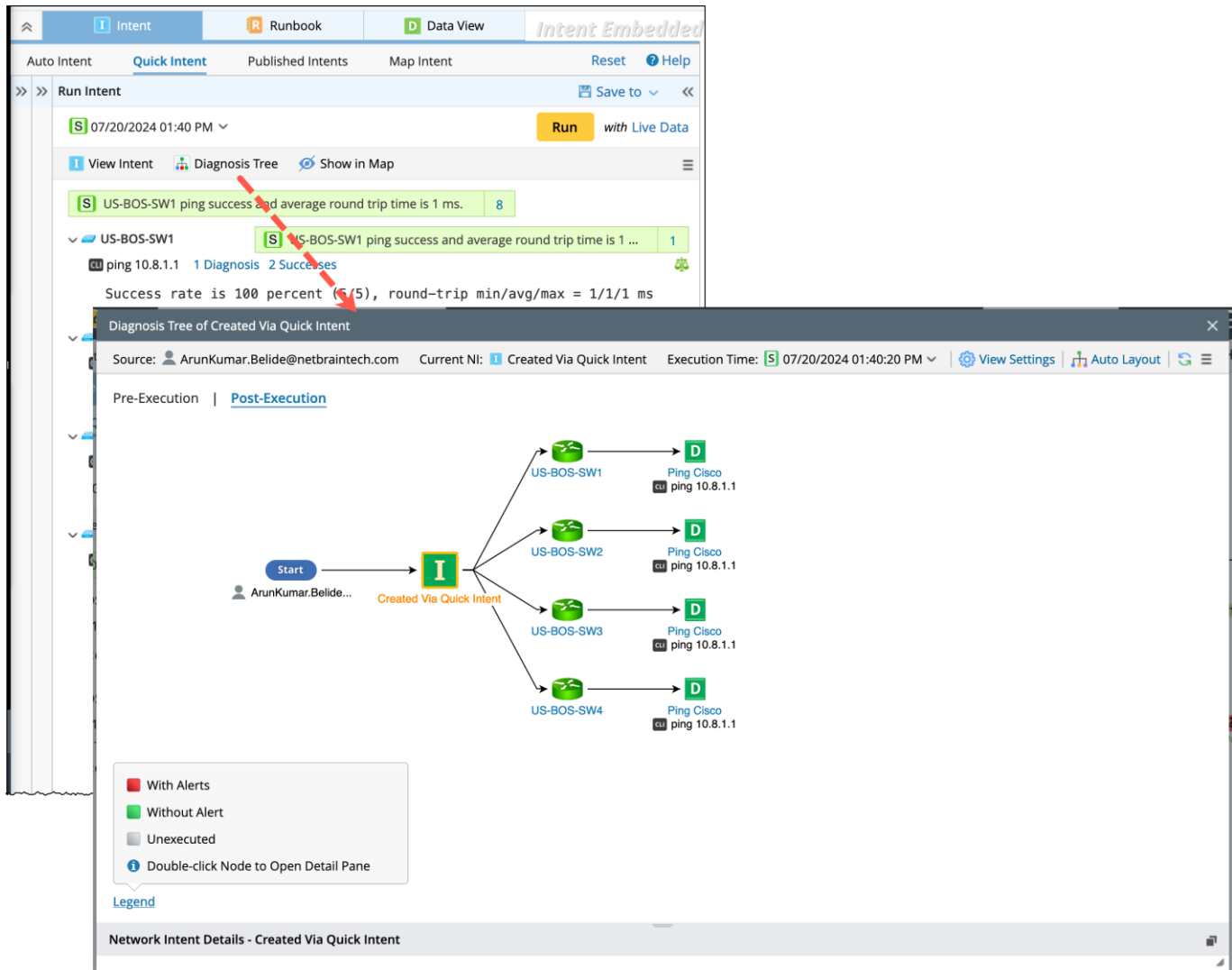
Success rate is 100 percent (5/5), roun...

+ New Diagnosis

2.7 Diagnosis Tree

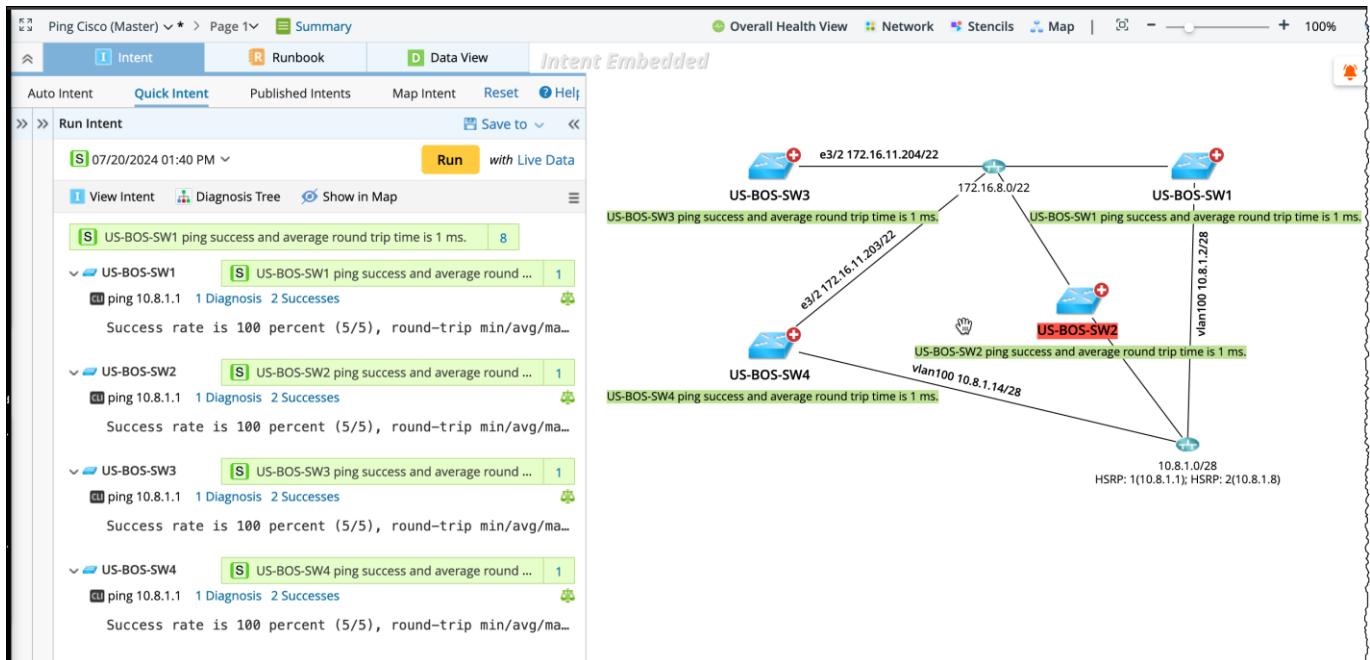
Click  to view the diagnosis tree. It includes the following details as shown in the figure:

- Current NI medium,
- Execution date and time,
- The diagnosis tree in Pre-Execution and Post-Execution mode,
- The status of execution in each device with color legends.



2.8 View the Result on Map

The result of the intent execution can be visualized in a map as shown in the following map by selecting **Show in Map** button located in the **Run Intent** pane:



2.9 Create Intent and Summary Dashboard

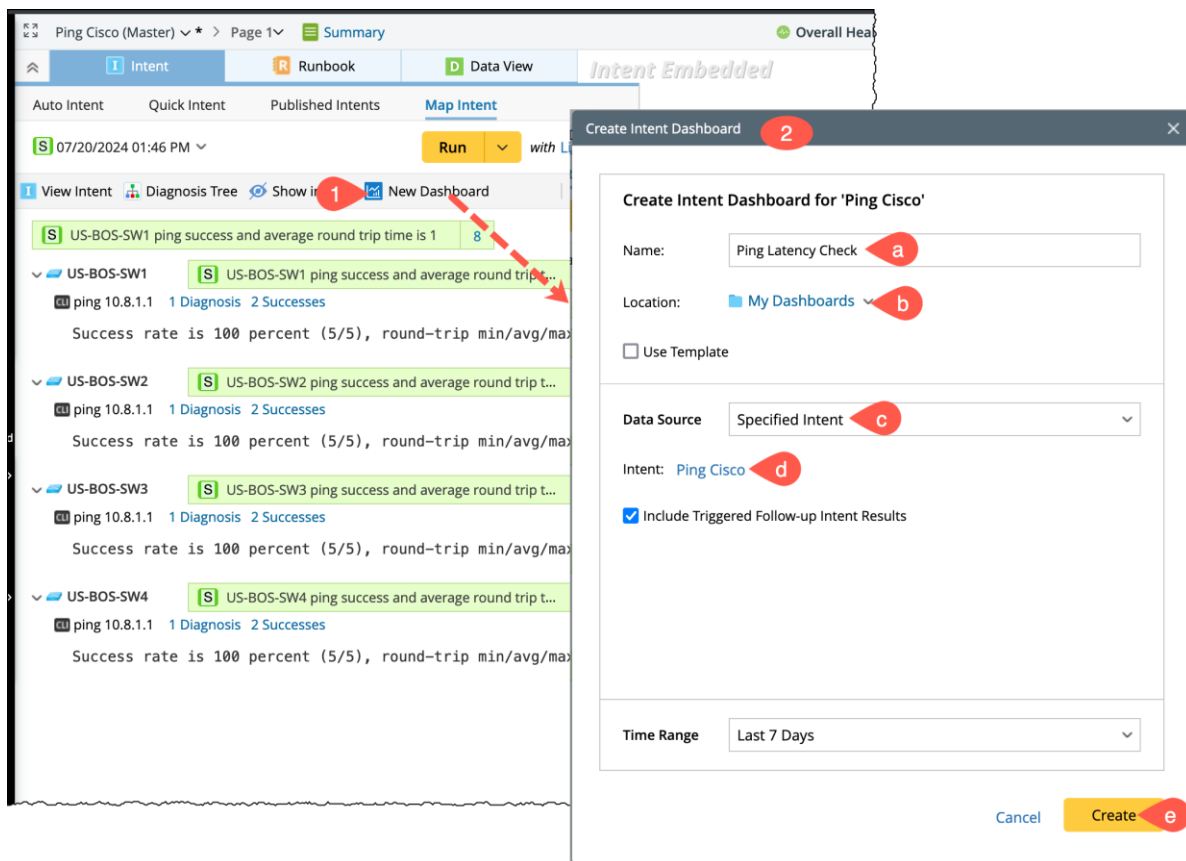
You can view the intent results through two different dashboards:

- **Intent Dashboard** to view the individual Intent results and
- **Summary Dashboard** to view the consolidated Intent results in a single view.

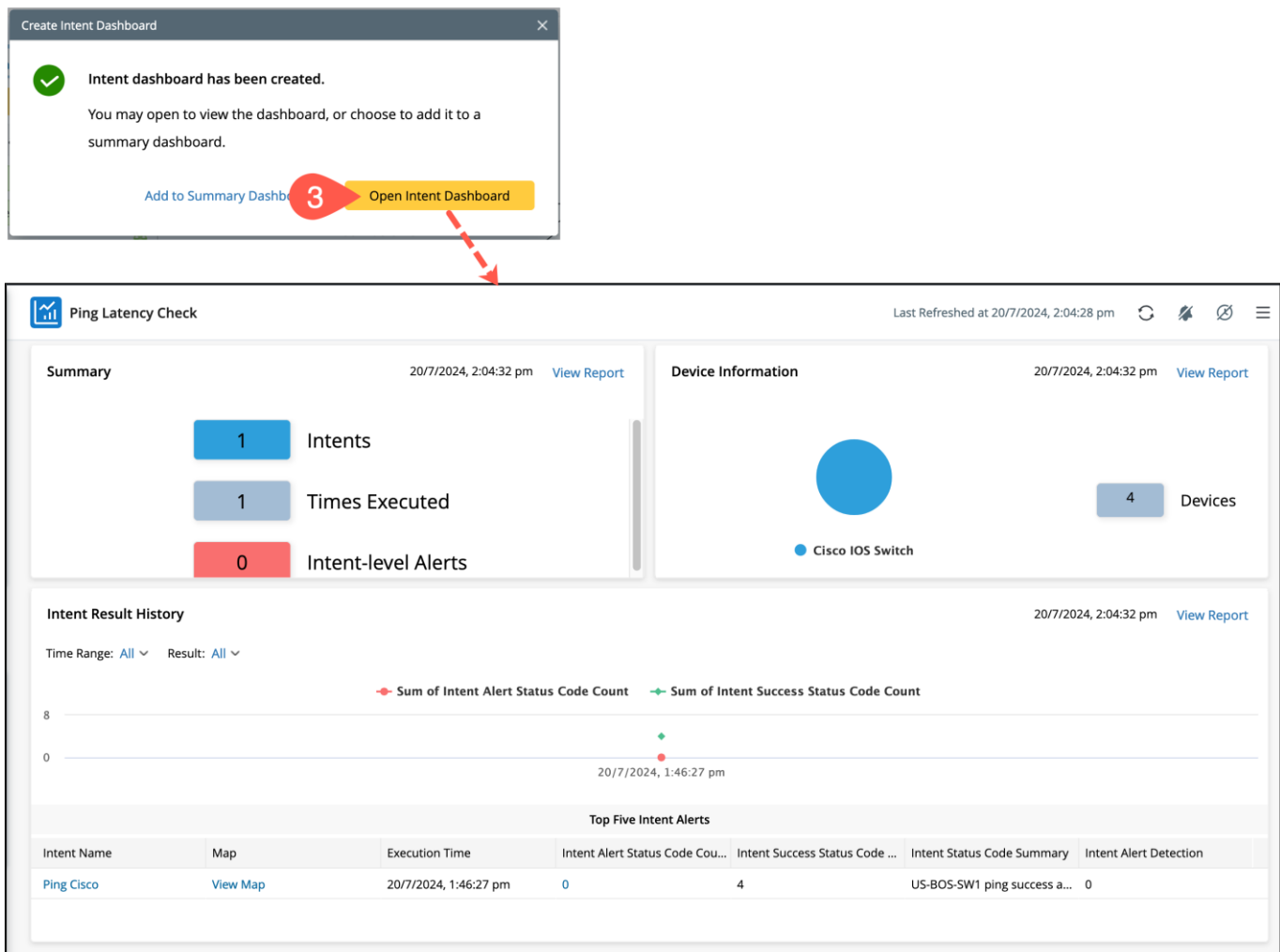
2.9.1 Intent Dashboard

The Intent Dashboard observes specific network issues with details and displays the results. You can save the frequently used dashboards as templates. Intent Dashboard can be created directly from the **Map Intent** tab as follows:

1. Go to the **Map Intent** tab and click on **New Dashboard**.
2. In the **Create Intent Dashboard** window, define the following:
 - a. Enter the Dashboard Name, **Ping Latency Check**
 - b. Select the Location to save the Intent Dashboard.
 - c. **Data Source:** By default, **Specified Intent** is selected from the dropdown.
 - d. **Intent:** Keep the default intent to create the dashboard for the same intent.
 - e. Click **Create**.




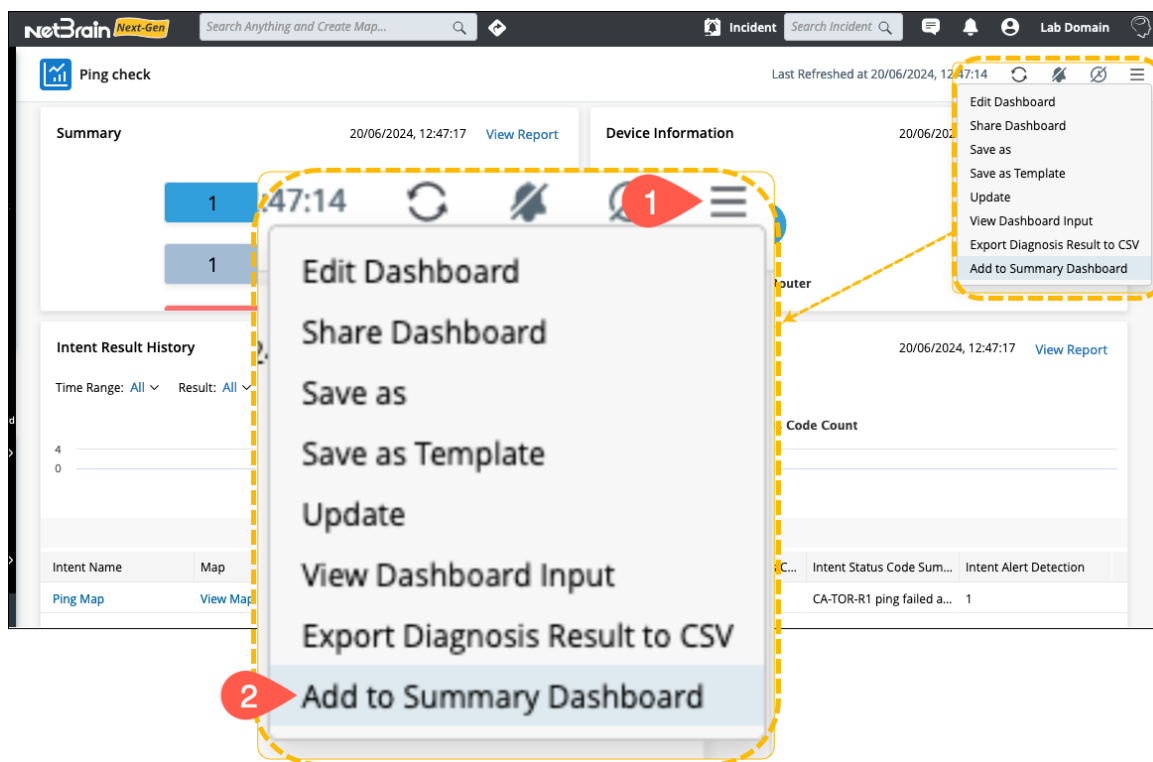
3. **Create Intent Dashboard** dialog pops up. Click **Open Intent Dashboard**, and the dashboard will be created.



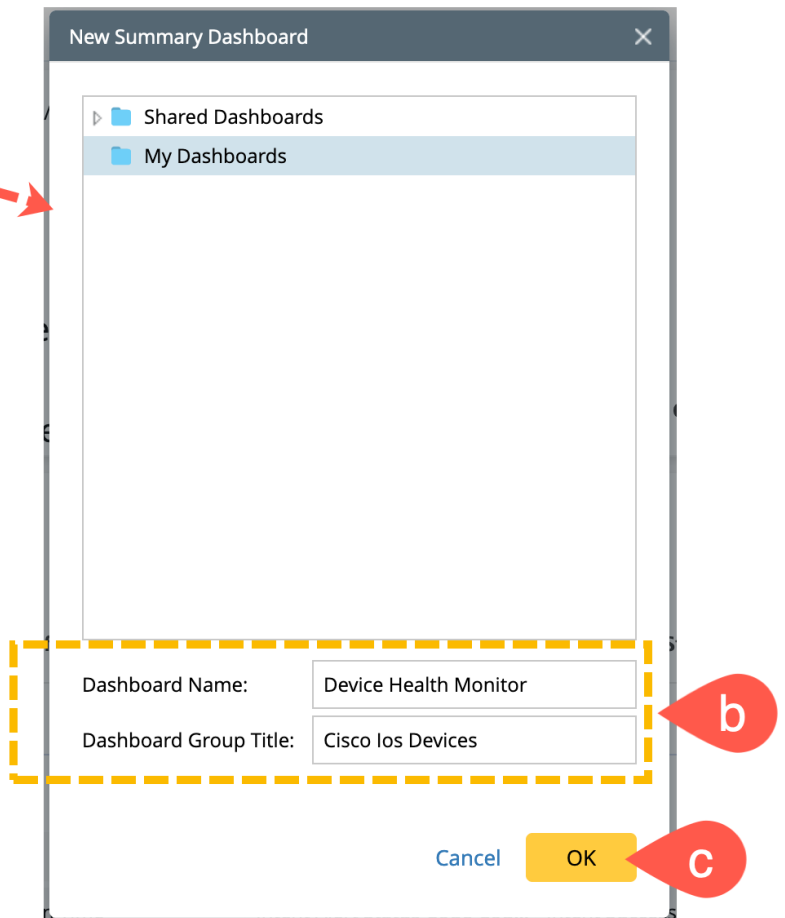
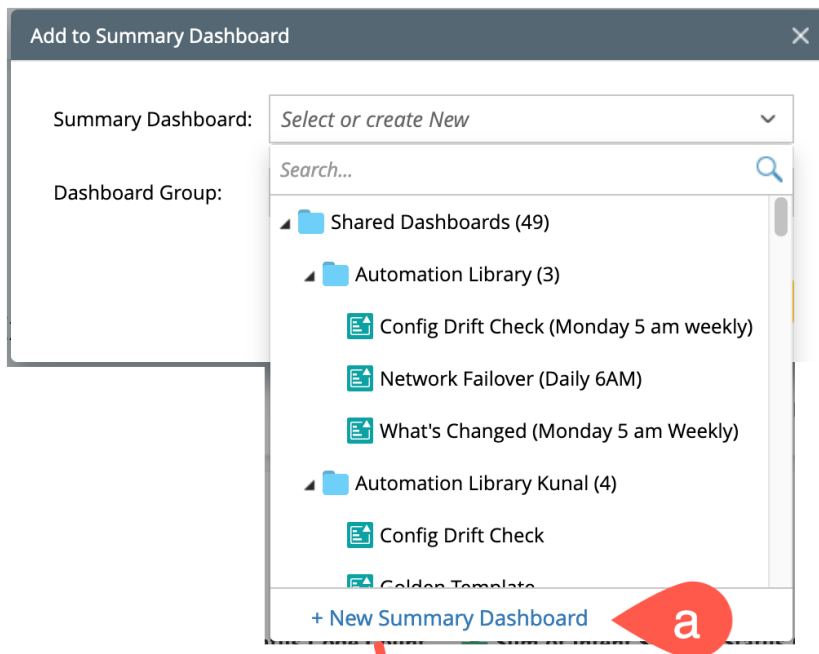
2.9.2 Summary Dashboard

The summary dashboard provides an overview displaying results from multiple Intent dashboards of the entire network or a set of network devices. With Summary Dashboard, you can group Intent Dashboards into widgets based on diagnosis purpose and display results by device, site or device groups. You can use the summary dashboard to monitor critical information across thousands of devices and discover the root cause for issues in one view.

1. Let us create a Summary Dashboard using the step-by-step instructions as follows:
2. Click  to open the menu located at the top-right corner of the dashboard window.
3. Select **Add to Summary Dashboard** to open the corresponding window for creating a summary dashboard.

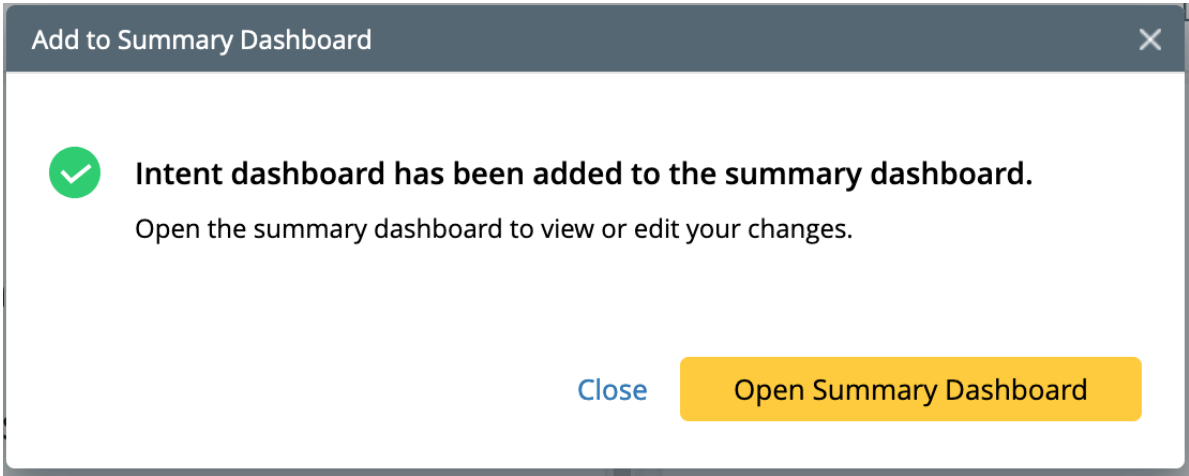


4. In the **Add to Summary Dashboard** window, let us create the new summary dashboard and group as follows:
 - a. **Summary Dashboard**: open the dropdown menu and select **+New Summary Dashboard** to pop up its dialogue.
 - b. Enter the dashboard basic details like **name**, **group title** and **location** of the summary dashboard to save.
 - c. Click **OK** to save and create the summary dashboard.

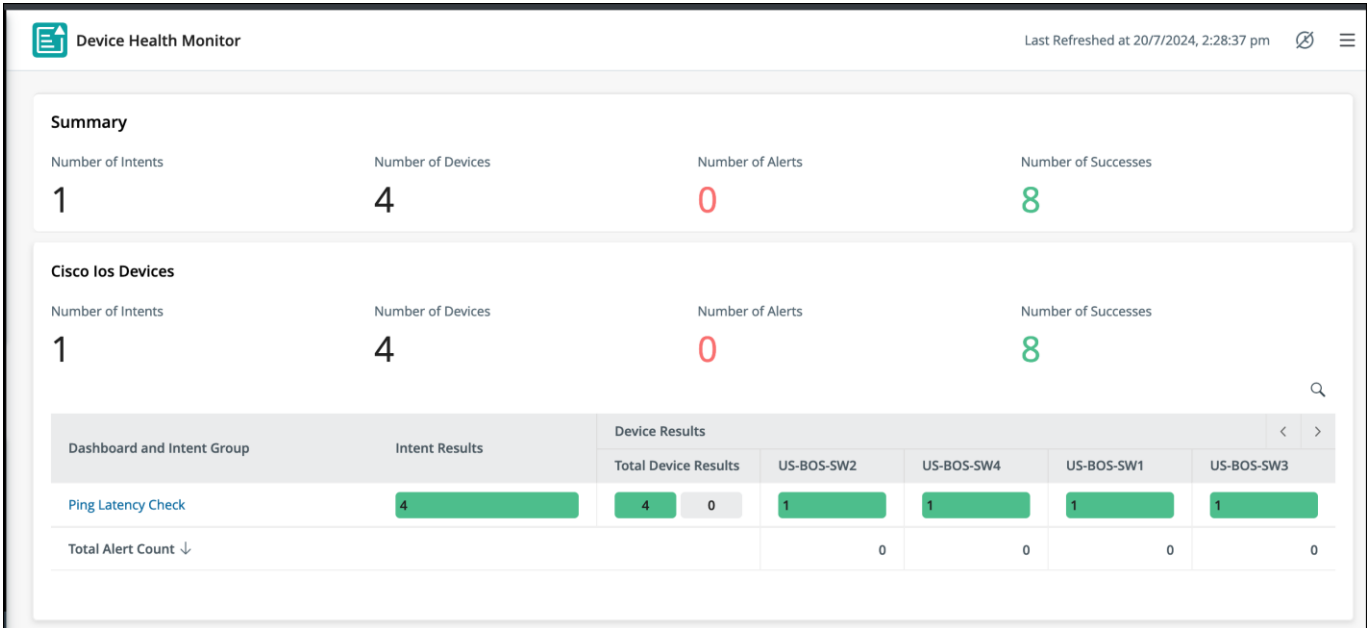


You can add more intent dashboards to the same summary dashboard by choosing this summary dashboard located under My dashboards locationAdd to Summary Dashboard dropdown menu.

5. In the **Add to Summary Dashboard** dialog, a success message prompt appears along with the option to **Open Summary Dashboard**. Click to view the dashboard.



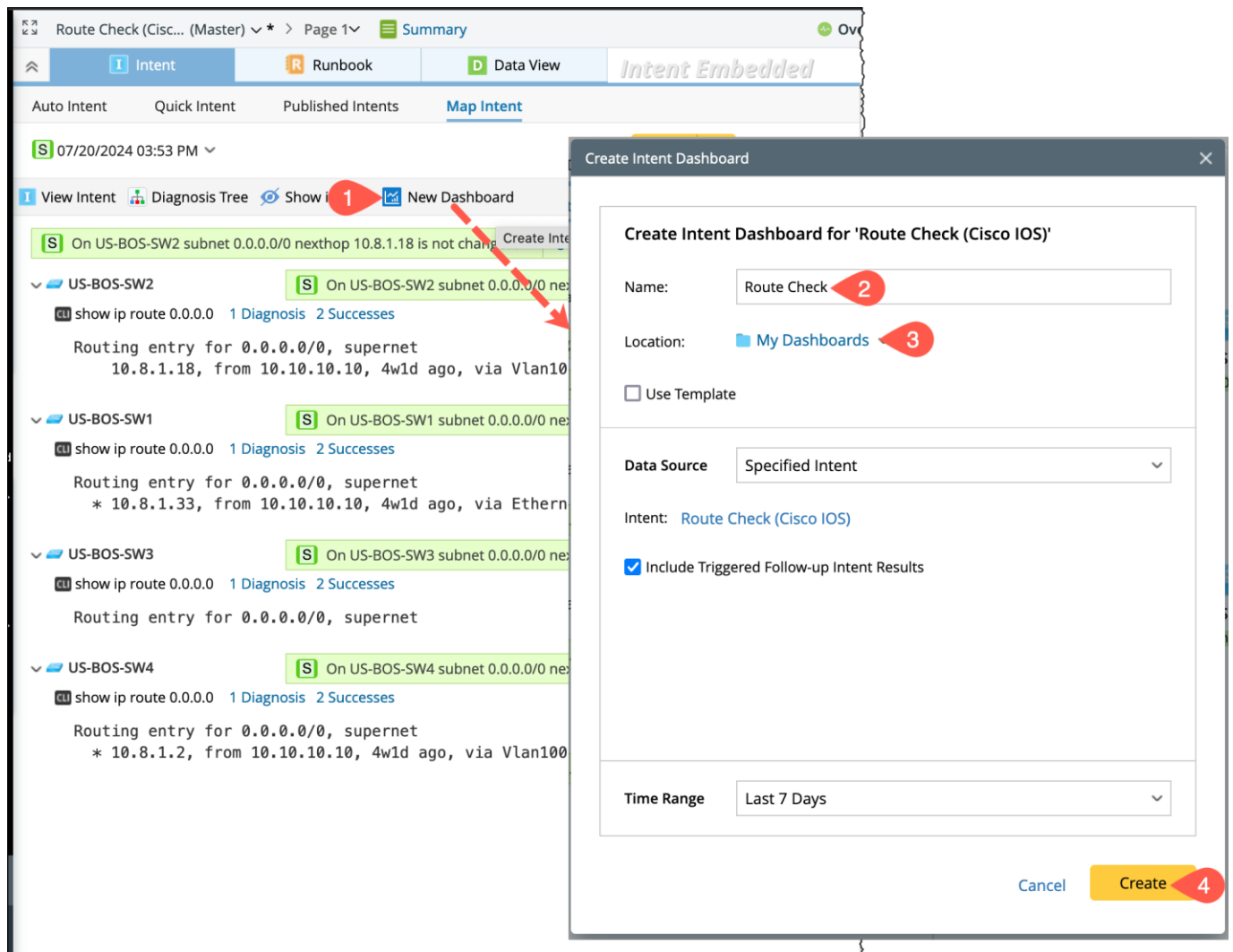
6. Review the resulting Summary Dashboard and explore the dashboard interface.



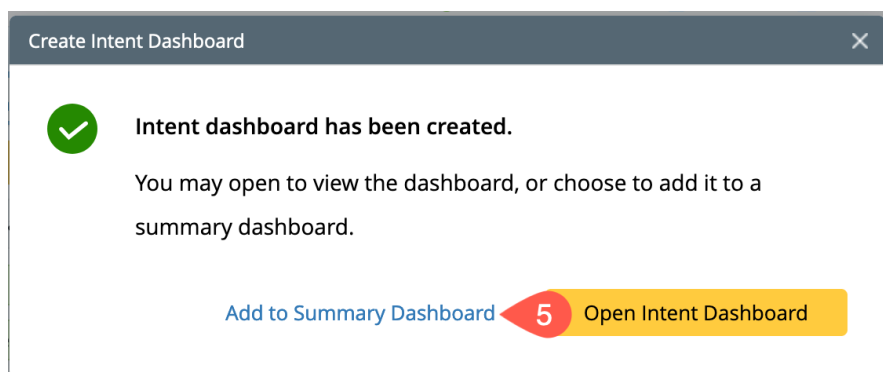
2.9.3 Add an Intent Dashboard to an Existing Summary Dashboard

Now, let us add the route check intent dashboard to the **Device Health Monitor** Summary Dashboard created in Section 2.9.2 as follows:

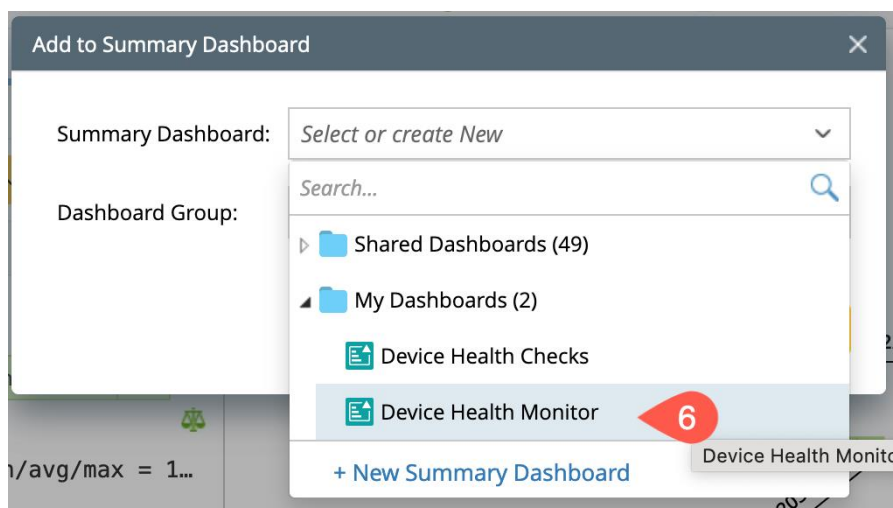
1. Go to the Map Intent tab and click New Dashboard to launch the Create Intent Dashboard window.
2. Change the default dashboard name to **Route check**.
3. Keep the default settings for Location (My Dashboards) and other fields
4. Click **Create**.



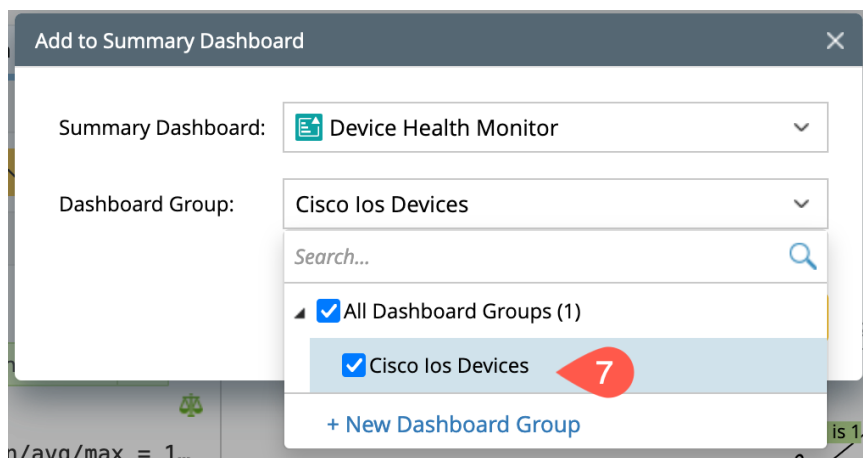
5. In the **Create Intent Dashboard** dialog, a success message prompt appears along with Click **Add to Summary Dashboard** to open its corresponding dialog.



6. In the **Add to Summary Dashboard** window, click the **Summary Dashboard** dropdown menu to open and select the **Device Health Monitor** that you have created in Section 2.9.2.



7. Choose the dashboard group that you have created in the previous section or create a new group as per the need.



8. Click **OK** to save and create the summary dashboard.
9. Click **Open Summary Dashboard** to view the dashboard.

Add to Summary Dashboard

Summary Dashboard: Device Health Monitor

Dashboard Group: Cisco Ios Devices

Cancel **8** OK

Add to Summary Dashboard

✓ Intent dashboard has been added to the summary dashboard.
Open the summary dashboard to view or edit your changes.

Cancel **9** Open Summary Dashboard

10. Review the resulting Summary Dashboard and explore the dashboard interface.

Device Health Monitor

Last Refreshed at 20/7/2024, 4:27:31 pm

Summary

Number of Intents

1

Number of Devices

4

Number of Alerts

0

Number of Successes

16

Cisco Ios Devices

Number of Intents

1

Number of Devices

4

Number of Alerts

0

Number of Successes

16

Dashboard and Intent Group

Intent Results

Device Results

Total Device Results

US-BOS-SW2

US-BOS-SW4

US-BOS-SW1

US-BOS-SW3

Ping Latency Check

4

0

1

1

1

1

Route Check

4

4

0

1

1

1

1

Total Alert Count ↓

0

0

0

0

Route Check added to the existing dashboard

3 Automate Basic Device Health Checking

While troubleshooting a network-related problem, the first step is always to check the overall health of the corresponding network devices and their interfaces/ports. Instead of manually checking, you can automate the frequently used device and interface health checks. The following table lists some basic common health checks:

Intent Name	Description
Uptime check (Cisco IOS)	Check whether the device reboots within the last 24 hours (the uptime is shorter than a day). If so, then provide the reload reason. Use the CLI command <code>show version</code> .
CPU usage check (Cisco IOS)	Check whether the device CPU utilization is normal or higher using the CLI command <code>show process CPU</code> .
Interface status check (Cisco IOS)	Check the interface status and also whether errors (Input, Output, CRC errors) are increasing using the CLI command <code>show interface</code> .

NOTE: It is recommended to name the intent as **<purpose of the intent><Device type>**.

Besides the overall health check, you may also automate some important operational status. For example, for an F5 load balancer, you may create an intent to check the status and state of the virtual servers and their pool member using the CLI command `show ltm virtual detail recursive`.

Due to the limitations in Quick Intent, we shall use the standard intent editor and create the Intents for Cisco devices. There are three main steps to define an intent:


1. Select a device, define the basic information of intent and save the intent to Intent Manager.
2. Parse the variables from retrieved command data.
3. Define the diagnosis logic and output.

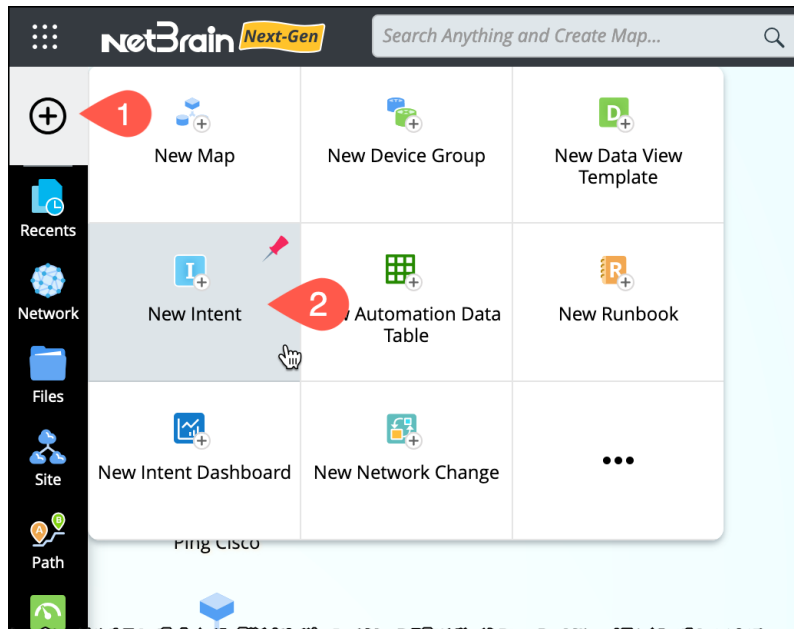
After executing the intent, you can create a dashboard to show its results:

4. [Execute the intent and create a dashboard.](#)
5. [Create an intent dashboard.](#)
6. [Add Intent dashboards to a common Summary Dashboard.](#)

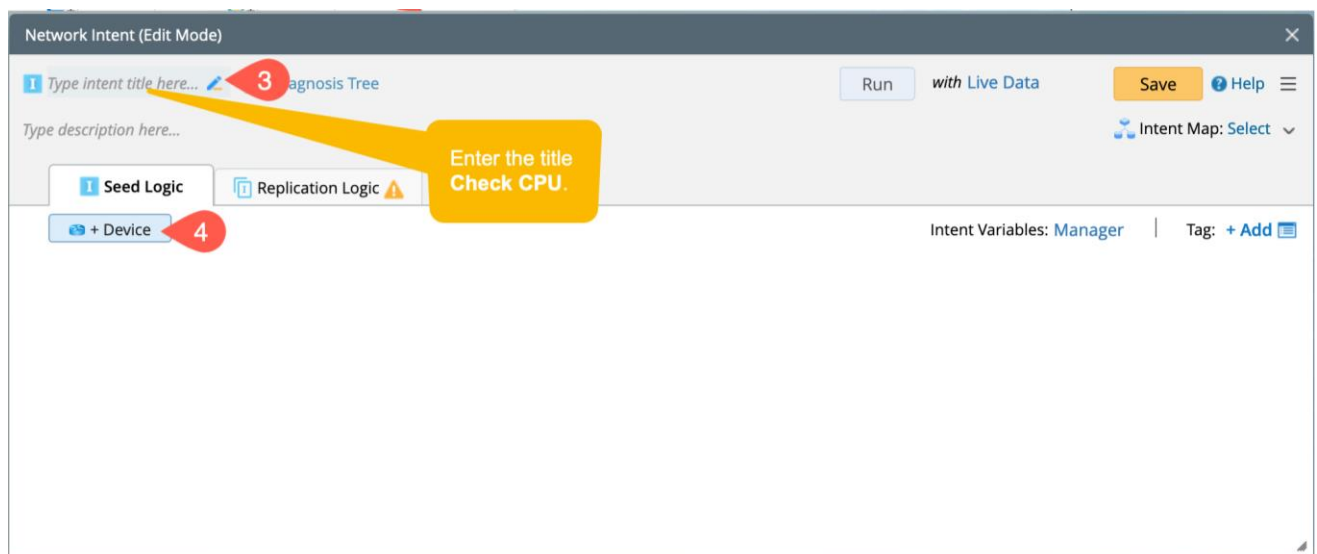
3.1 Select a Device

Define the basic information of the intent, such as name, description, and device and save it to the intent manager as follows:

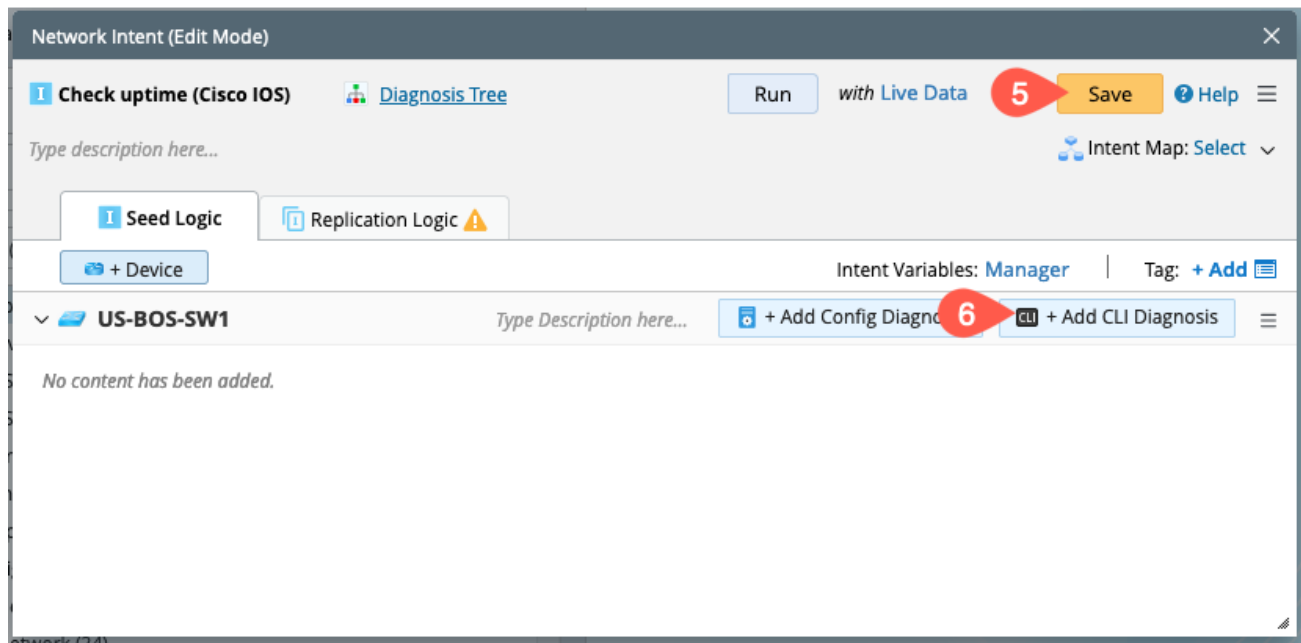
1. Click  icon from your desktop.
2. Select **New Intent**. A window **Network Intent** in edit mode will appear.



3. Enter the title and a brief description (optional) of the intent.
4. Add the seed devices by selecting the option **+Device**.



5. Click **Save** to save the intent to the Intent Manager.
6. Click **+Add CLI Diagnosis** to open the corresponding window and proceed to the next section to parse the variables and define the diagnosis.



3.2 Retrieve Data and Parse Variables for Health Checks

After choosing the device as described in [Section 3.1](#), parse the variables from the retrieved CLI command data in the following manner:

1. [Parse variables from Show Version](#)
2. [Parse variables from Show Process CPU](#)
3. [Parse variables from Show Interface](#)

3.2.1 Parse variables from *show version*

1. In the command field, enter the command **show version**.
2. Click Retrieve.
3. In the sample data, select the uptime value **3 weeks, 4 days, 11 hours, 15 minutes** and click **Parse Variable** in the tip window.

NOTE: You can also double-click the text to get the same result.

The screenshot displays the 'CLI Command Diagnosis' interface. At the top, the device 'US-BOS-SW1' is selected, and the command 'show version' is entered in the command field. A red circle '1' is around the command field, and a red circle '2' is around the 'Retrieve' button. Below the command field, a progress bar shows '1. Define Variable' and '2. Define Diagnosis'. The output of the command is displayed in a table. A red circle '3' is around the selected uptime value '30 weeks, 3 hours, 15 minutes'. A yellow 'Parse Variable' button is shown over the selected text. The interface also includes a search bar and a 'Format1' dropdown.

Line	Text
1	US-BOS-SW1>show version
2	Cisco IOS Software, Linux Software (I86BI_LINUXL2-ADV
3	Technical Support: http://www.cisco.com/techsupport
4	Copyright (c) 1986-2017 by Cisco Systems, Inc.
5	Compiled Thu 02-Feb-17 03:38 by mmen
6	
7	ROM: Bootstrap program is Linux
8	
9	US-BOS-SW1 uptime is 30 weeks, 3 hours, 15 minutes
10	System returned to ROM by reload at 0
11	System image file is "unix:/opt/unetlab/addons/iol
12	Last reload reason: Unknown reason
13	

4. In the right pane **Var Line 1** field, update the variable name to `$mstring:uptime`.

NOTE: The variable name is defined as `$<type>:<variable_name>`.

The screenshot shows the 'Define Diagnosis' step in the NetBrain interface. The 'Var Line 1' field is highlighted with a red dashed box and a red arrow pointing to the variable name `$mstring:var1`. The output shows the variable value as `31 weeks, 2 days, 11 hours, 47 minutes`.

5. Similarly, parse the reload reason and update the variable to `$mstring:reload_reason`.

The screenshot shows the 'Define Variable' step in the NetBrain interface. The 'Var Line 2' field is highlighted with a red dashed box and a red arrow pointing to the variable name `$mstring:reload_reason`. The output shows the variable value as `Unknown reason`.

6. Parsed variables will be listed under the **Output** section in the right pane. Validate the variables.
7. Click **Apply** to save and go to **Define Diagnosis** [Section 3.3.1](#).

The screenshot shows the 'Define Diagnosis' step in the NetBrain interface. The main area displays two variable definitions:

- Var Line 1:** `uptime is $mstring:uptime`. The corresponding value is `29 weeks, 5 days, 13 hours, 35 minutes`.
- Var Line 2:** `reason: $mstring:reload_reason`. The corresponding value is `Unknown reason`.

Below the variable definitions, the **Output** section (highlighted with a yellow border) shows the parsed results:

```
$uptime (mstring) = 29 weeks, 5 days, 13 hours, 35 minutes
$reason (mstring) = Unknown reason
```

At the bottom right, there is a red circle with the number **7** and a yellow **Apply** button.

3.2.2 Parse variables from *Show Process CPU*

1. In the command field, enter the command **Show Process CPU**.
2. Click **Retrieve** and the sample data is displayed under the **Define Variable** section.
3. In the sample data, from the CPU utilization line (you can search the word *CPU utilization* to find the line), select the **five minutes** value **0%** and click **Parse Variable** in the tip window.

NOTE: You can also double-click the text to get the same result.

4. In the right pane **Var Line 1** field, update the variable name to **\$int:cpu**.

NOTE: The created variables will appear under the **Output** section. The variable name is defined as **\$<type>:<variable_name>**.

5. Click **Apply** to save and go to **Define Diagnosis** [Section 3.3.2](#).

The screenshot illustrates the process of parsing variables from the 'Show Process CPU' command. The interface is divided into two main sections: '1. Define Variable' and '2. Define Diagnosis'.

1. Define Variable: This section shows a table of sample data. A red box highlights the 'five minutes: 0%' value in the 'CPU utilization' line. A red arrow points from this value to the 'Parse Variable' button in the tip window.

2. Define Diagnosis: This section shows the 'Var Line 1' field updated to 'five minutes: \$int:cpu'. A red arrow points from this field to the 'Apply' button. The 'Output' section shows the variable '\$cpu (int)' with a value of 0.

3.2.3 Parse variables from *Show Interface*

Unlike the CLI command outputs in earlier sections, the output of the **show interface** includes multiple interfaces, and each of them has an identical format. For this type of text format, you will use **Paragraph Parser** to convert multiple text sections of an identical pattern into a table variable:

1. In the command field, enter the command **show interface**.
2. Click Retrieve.
3. In the sample data, select the line *Ethernet0/0 is up, line protocol is up* and click **Parse Variable** from the tip window. This line will be defined as the ID line, which identifies the starting line of a paragraph. The ID line is a normal line pattern where you can define the variables and keywords.
4. In the right pane **Var Line 1** field, modify the line to be: `$string:interface` is `$mstring:L1status`, line protocol is `$mstring:L2status`.

NOTE: The variable names are defined as `$<type>:<variable_name>`.

NOTE: All the lines in source Lines data with interface and its status will be highlighted automatically and reflected in the **Output** pane.

5. Click **Apply**.

The screenshot shows the 'CLI Command Diagnosis' window. The 'Current Device' is 'US-BOS-SW1' and the command is 'show interface'. The 'Retrieve' button is highlighted. The '1. Define Variable' pane shows the command output with the line 'Ethernet0/0 is up, line protocol is up (connected)' selected. The '2. Define Diagnosis' pane shows the 'Pattern1' configuration. The 'ID Line' is set to '^\$var2 is \$var1, line \$mstring:dummy is \$connected'. The 'Var Line 1' field is modified to 'face is \$mstring:L1status, line protocol is \$mstring:L2status'. The 'Apply' button is highlighted. The 'Output' pane shows the resulting table with columns '\$Interface', '\$L1status', and '\$L2status'.

\$Interface	\$L1status	\$L2status
Ethernet0/0	up	up (connected)
Ethernet0/1	up	up (connected)
Ethernet0/2	up	up (connected)

Data with interface and its status are parsed to the output pane

6. Similarly, parse other variables (input errors, CRC and output errors) by referring to the following table:

Variables	Parsing Procedure
Input_errors	<ol style="list-style-type: none"> Select the input error value <code>0</code> and click Parse Variable in the tip window. Modify the Variable to <code>\$int:input_errors</code>.
CRC errors	<ol style="list-style-type: none"> Select the CRC value <code>0</code> and click Parse Variable in the tip window. Modify the Variable line to <code>\$int:CRC</code>.
Output_errors	<ol style="list-style-type: none"> Select the output error value <code>0</code> and click Parse Variable in the tip window. Modify the Variable to <code>\$int:output_errors</code>.

7. Click **Apply** to save and go to **Define Diagnosis** [Section 3.3.3](#).

The screenshot shows the NetBrain R11.1b interface during the 'Define Variable' step. The interface is split into two panes. The left pane, titled '1. Define Variable', displays a list of network interface statistics. Lines 21-25 are highlighted with a yellow box, showing values for input errors, CRC, and output errors. The right pane, titled '2. Define Diagnosis', shows two variables being defined: 'Var Line 1' and 'Var Line 2'. 'Var Line 1' is defined as '\$int:input_errors input errors, \$int:CRC CRC,' and 'Var Line 2' is defined as '\$int:output_errors output errors'. Both variables are linked to the highlighted lines in the first pane. At the bottom right, there is a red circle with the number '7' and an 'Apply' button.

3.3 Define Diagnosis for Health Checks

Let's define the logical condition and diagnosis output using the parsed variables for all the intents as follows:

3.3.1 Define Diagnosis for *Show version*

1. Go to the **Define Diagnosis** ribbon and click on **Add Diagnosis**.
2. Enter the diagnosis name as **Check Uptime**.
3. **Anchor:** Select the variable *\$uptime* from the drop-down menu.

NOTE: Selecting the anchor will draw a line from the variable to the message in the diagnosis result.

4. Define the **If** condition as follows:
 - a) A: Variable *uptime* | **Contains** | **day**.
 - b) B: Variable *uptime* | **Contains** | **week**.
 - c) Boolean Expression: **A or B**.

2. Define Diagnosis

Add Note 1 Add Diagnosis Can also click a variable on the left to add automation.

Name: Check Uptime 2 Anchor: \$uptime 3

Type description of the diagnosis...

☐ Loop Table Rows

▼ If 4

A US-BOS-S... Current ▼

a uptime Contains day

B US-BOS-S... Current ▼

b uptime Contains week

C Select Variable ▼

c Boolean Expression: A or B

▼ Then

5. **Then:** In case **If** logic is true, define the color (**green**), status (**Success**), and message as `$this_device uptime is $uptime`.

NOTE: By typing \$, you can get the variable selection pop up.

6. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

7. **Else:** In case **If** logic is not true, define the color (**red**), status (**Error**), and message as `$this_device uptime is $uptime because of $reload_reason`.

NOTE: By typing \$, you can get the variable selection pop up.

8. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

9. Click **Apply** to save the settings and then close the window.

The screenshot shows a configuration window with two main sections: **Then** and **Else**. Each section has a **Diagnosis Message** field and a **Save to Incident** checkbox. The **Then** section is configured with a green color, **Success** status, and the message `$this_device uptime is $uptime`. The **Else** section is configured with a red color, **Error** status, and the message `$this_device uptime is $uptime because of $reload_reason`. Both sections have checkboxes for **Set Status Code for Device** and **Set Status Code for Intent** checked. A yellow dashed box highlights the status code configuration area in both sections. At the bottom right, there is a **Can** button and an **Apply** button.

10. Back in the **Network Intent** window, click **Save** to save the intent to the Intent Manager.

3.3.2 Define Diagnosis for *Show Process CPU*

1. Go to the **Define Diagnosis** ribbon and click on **Add Diagnosis**.
2. Enter the diagnosis name as **Check CPU Usage** and select the variable **\$CPU** for the anchor.
NOTE: Anchor helps when viewing the resulting diagnosis. It will draw a line from the diagnosis message to the variable you select as the anchor.
3. Define the **If** condition as:
A: Variable **cpu** | **Greater than** | **50**.
NOTE: Set the condition as *cpu is Greater than 80* (or any number you think implies the high CPU in your network).
4. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**) and message ***\$this_device* CPU utilization is high *\$cpu***.
NOTE: By typing \$, you can get the variable selection pop up.
5. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

2. Define Diagnosis

Can also click a variable on the left to add automation.

1 Add Note Add Diagnosis

Name: Check CPU Usage 2 Anchor: \$cpu

Type description of the diagnosis...

☐ Loop Table Rows

▼ If

A US-BOS-S... Current ▼

3 cpu Greater than 50

B Select Variable ▼

▼ Then

Diagnosis Message: ☐ Save to Incident

4 \$this_device CPU utilization is high \$cpu

☒ Set Status Code for Device:

5 Error \$this_device CPU utilization is high \$cpu

☒ Set Status Code for Intent:

Error \$this_device CPU utilization is high \$cpu

6. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as `$this_device CPU utilization is normal`.

NOTE: By typing \$, you can get the variable selection pop up.

7. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
8. Click **Apply** to save the settings and then close the window.

▼ Else Delete

Diagnosis Message: ☐ Save to Incident ⋮

6 ▼ `$this_device CPU utilization is normal`

7 ☒ ⋮ Set Status Code for Device: ✓ Success `$this_device CPU utilization is normal`

☒ ⋮ Set Status Code for Intent: ✓ Success `$this_device CPU utilization is normal`

+ Add Logic

+ Add Elself

Can 8 Apply

9. Back in the **Network Intent** window, click **Save** to save the intent to the Intent Manager.

Network Intent (Edit Mode)

1 Cisco Device Level CPU Check Diagnosis Tree Run with Live Data 8 Save Help ⋮

Type description here...

Seed Logic Replication Logic + Device Intent Variables: Manager Tag: + Add

▼ US-BOS-SW1 Type Description here... + Add Config Diagnosis + Add CLI Diagnosis Edit Diagnosis

show process cpu Type Description here...

1	US-BOS-SW1>show process cpu									
2	CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%									
3	PID	Runtime(ms)	Invoked	uSecs	5Sec	1Min	5Min	TTY	Process	
4	1	0	9	0	0.00%	0.00%	0.00%	0	Chunk Manager	
5	2	130752	3604525	36	0.00%	0.00%	0.00%	0	Load Meter	
6	3	0	30	0	0.00%	0.00%	0.00%	0	SpanTree Flush	

Check CPU Usage Message Status Code

3.3.3 Define Diagnosis for *Show interface*

In this section, we need to create two diagnoses separately to [Check Status](#) and [errors](#) as follows:

3.3.3.1 Create a Diagnosis to check Interface Status

1. Go to the **Define Diagnosis** ribbon and click on **Add Diagnosis**.
2. Enter the diagnosis name as **Interface Status** and select the variable **\$interface** for the anchor.
3. Check in the **Loop Table Rows** and select the table **Pattern1**.
4. Select the variable **\$interface** for the Table Key.
5. Define the **If** condition as follows:
 - a) A: Variable *L1status* | Contains | down.
 - b) B: Variable *L2status* | Contains | down.
 - c) Boolean Expression: **A or B**.

2. Define Diagnosis

Add Note **1** Add Diagnosis Can also click a variable on the left to add automation.

Name: Interface Status **2** Anchor: Pattern1.\$interface

Type description of the diagnosis...

3 ☒ Loop Table Rows **Pattern1** Table Key: interface **4**

5 **If**

A US-BOS-S... Current
L1status Contains down

B US-BOS-S... Current
L2status Contains down

C Select Variable

Boolean Expression: A or B

Then

6. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**), and message **On *\$this_device*, Interface *\$interface* is down, Current L1 status is *\$L1status* and L2 status is *\$L2status*.**
7. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
8. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as **On *\$this_device*, Interface *\$interface* is UP.**
9. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
10. Click **Apply** to save and create the intent.

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Then

Diagnosis Message: ☐ Save to Incident

6 on *\$this_device*, Interface *\$interface* is down, Current L1 status is *\$L1status* and L2

7 ☒ Set Status Code for Device: ☐ Save to Incident

Error on *\$this_device*, Interface *\$interface* is down, Current L1 status is *\$L1status*

☒ Set Status Code for Intent:

Error on *\$this_device*, Interface *\$interface* is down, Current L1 status is *\$L1status*

Add Logic

Else Delete

8 Diagnosis Message: ☐ Save to Incident

On *\$this_device*, Interface *\$interface* is UP

☒ Set Status Code for Device:

9 Success On *\$this_device*, Interface *\$interface* is UP

☒ Set Status Code for Intent:

Success On *\$this_device*, Interface *\$interface* is UP

+ Add Elself

Can 10 Apply

3.3.3.2 Create a Diagnosis to check interface errors

1. Go to the **Define Diagnosis** ribbon and click on **Add Diagnosis**.
2. Enter the diagnosis name as **Check Errors**.
3. Select the variable **\$input_error** for the anchor.
4. Check in the **Loop Table Rows** and select the table **Pattern1**.
5. Select the variable **\$interface** for the Table Key.
6. Define the **If** condition as follows:
 - a) A: Variable *input_errors* (Current) | Does not equal | *input_errors* (Last)
 - b) B: Variable *CRC* (Current) | Does not equal | *CRC* (Last)
 - c) C: Variable *output_errors* (Current) | Does not equal | *output_errors* (Last)
 - d) Boolean Expression: **A or B or C**

2. Define Diagnosis

Add Note Add Diagnosis 1 Can also click a variable on the left to add automation.

Name: Check Errors 2 Anchor: Pattern1.\$input_err... 3

Type description of the diagnosis...

4 ☒ Loop Table Rows ☐ Pattern1 Table Key: interface 5

If

A US-BOS-S... Current Last
input_errors Does not equal input_errors

B US-BOS-S... Current Last
CRC Does not equal CRC

C US-BOS-S... Current Last
output_errors Does not equal output_errors

D Select Variable

Boolean Expression: A or B or C

7. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**) and message as:
`$this_device Interface $interface has:`
`Current input errors $input_errors and last input errors $input_errors(Last)`
`Current CRC $CRC and last CRC $CRC(Last)`
`Current output errors $output_errors and last $output_errors(Last)`
 NOTE: By typing \$, you can get the variable selection pop up.
8. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
9. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as
`Errors on $this_device has not increased.`
10. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
11. Click **Apply** to save and create the intent.

The screenshot shows the 'Network Intent' configuration window. It has two main sections: 'Then' and 'Else'.
 In the 'Then' section:
 - A 'Diagnosis Message' field contains the text: '\$this_device, Interface \$interface has: Current input errors \$input_errors and last input errors \$input_errors(Last)'.
 - A 'Set Status Code for Device' checkbox is checked, with a dropdown menu showing 'Error' (indicated by callout 8).
 - A 'Set Status Code for Intent' checkbox is checked, with a dropdown menu showing 'Error' (indicated by callout 8).
 - A 'Save to Incident' checkbox is unchecked.
 In the 'Else' section:
 - A 'Diagnosis Message' field contains the text: 'Errors on this device \$this_device have not increased'.
 - A 'Set Status Code for Device' checkbox is checked, with a dropdown menu showing 'Success' (indicated by callout 10).
 - A 'Set Status Code for Intent' checkbox is checked, with a dropdown menu showing 'Success' (indicated by callout 10).
 - A 'Save to Incident' checkbox is unchecked.
 At the bottom right, there are 'Cancel' and 'Apply' buttons (indicated by callout 11).

12. Back in the **Network Intent** window, click **Save** to save the intent to the Intent Manager.

3.4 Monitor F5 Virtual Server Status

Besides the overall health check, you may also automate some important operational status. For example, for an F5 load balancer, you may create an intent to check the status and state of the virtual servers and their pool member using the CLI command `show ltm virtual detail recursive`.

3.4.1 Parse variables from `show ltm virtual detail recursive`

The output of the command `show ltm virtual detail recursive` may include the multiple virtual servers, and so you use the Paragraph Parser to parse the data like what you have done for the command `show interface`. A paragraph of a virtual server may have multiple pool members, so you need to create a Paragraph Parser for the Paragraph. In other words, you create a Paragraph Parser within a Paragraph Parser (the Parent Parser).

The result will be two tables: the virtual server and the pool member. And you can create two separate diagnoses from the same sample data.

1. In the command field, enter the command `show ltm virtual detail recursive`.
2. Click Retrieve.
3. Use Paragraph Parser to parse the virtual server since there can be multiple virtual servers. Define the ID line as **Virtual Server: \$Virtual_Server**. Parse the variables and modify them as listed in the following table:

Variable	ID/Var Line
Virtual Server	Virtual Server: <code>\$Virtual_Server</code>
Status	^ Availability: <code>\$mstring:Status</code>
State	^ State: <code>\$string:state</code>
CPU Usage	Last 5 Minutes <code>\$int:CPU_Usage_Minutes</code>

- Change the name of the paragraph parser to **Virtual Server**.

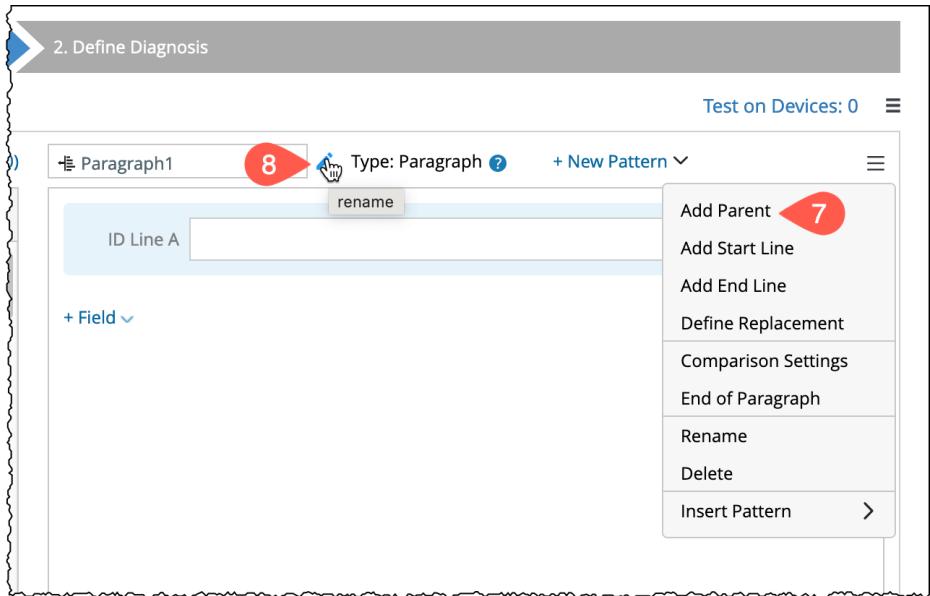
The screenshot shows the CLI Command Diagnosis interface. On the left, a list of commands is displayed, with 'Ltm::Virtual Server: Common/F5-lab' selected. A yellow box highlights this selection, and a red circle with the number 3 points to it. On the right, the 'Virtual_Server' pattern is configured. A red circle with the number 4 points to the pattern name dropdown. The pattern is of type 'Paragraph'. It contains three variable lines: 'ID Line A' with the value 'Virtual Server: \$Virtual_Server', 'Var Line 1' with the value 'Availability : \$mstring:Status', and 'Var Line 2' with the value 'State : \$string:state'. A red circle with the number 3 points to the 'Availability' value in Var Line 1. A red circle with the number 4 points to the 'Virtual_Server' pattern name dropdown.

- Create a second pattern to extract the pool member variables and add it as a child to the pattern **Virtual_server** (the parent parser).
- Go to the **+New Pattern** drop down menu and choose **Paragraph**.

The screenshot shows the '2. Define Diagnosis' step. The 'Virtual_Server' pattern is selected. A dropdown menu is open, showing options: 'Auto Pattern', 'Single Variable', 'Table', 'Paragraph', and 'Advanced'. A red circle with the number 6 points to the 'Paragraph' option. The background shows the same pattern configuration as the previous screenshot.

- In the new paragraph, Go to the menu  and select **Add Parent**.

8. Rename the pattern name to **Pool_members**.

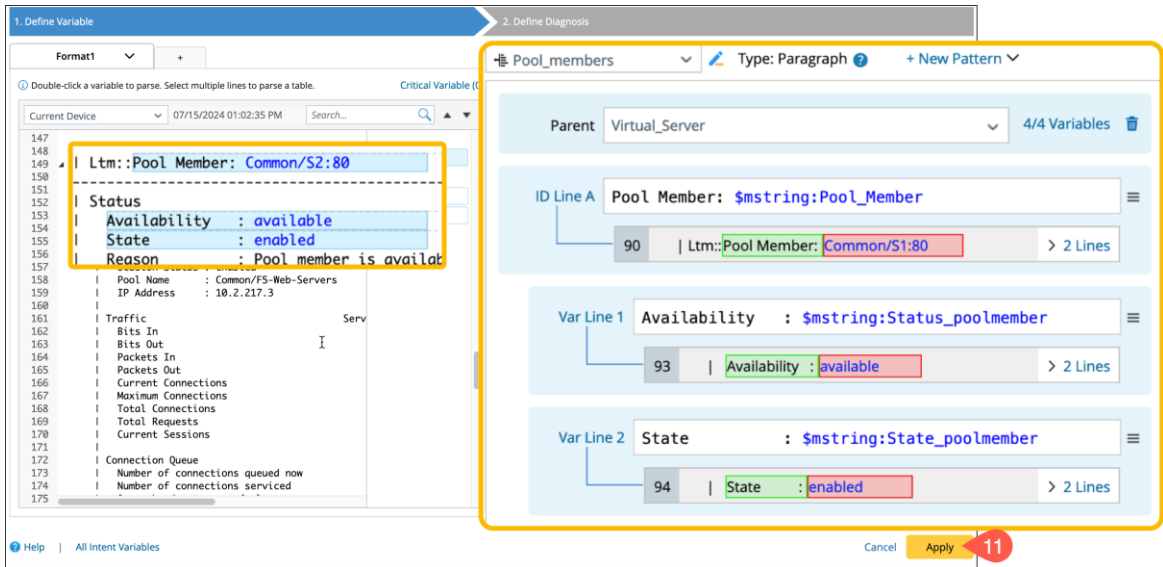


9. In the Parent field, select **Virtual_Server** from the drop down menu.

10. Parse the pool member, state and availability as listed in the following table:

Variable	ID/Var Line
Pool Member	Pool Member: <i>\$mstring:poolmember</i>
Status	Availability: <i>\$mstring:Status_poolmember</i>
State	State: <i>\$string:state_poolmember</i>

11. Click **Apply** to save and go to **Define Diagnosis**.



3.4.2 Define Diagnosis for Monitor F5 virtual server

In this use case, we shall create three different diagnoses each for:

1. [Virtual server status and state](#)
2. [Checking CPU usage](#)
3. [Pool member status and state](#)

3.4.2.1 Create a Diagnosis to check the virtual server Status and State

1. Go to the **Define Diagnosis** section and click on **Add Diagnosis**.
2. Enter the diagnosis name as Virtual server status and state.
3. **Anchor:** Select the variable *\$Virtual_server* from the drop-down menu.
4. Check in the **Loop Table Rows** and select the table **Virtual_server**.
5. Select the variable **\$Virtual_server** for the Table Key.
6. Define the **If** condition as follows:
 - a) A: Variable *status (Current)* | Does not equal | *Status* (Baseline).
 - b) B: Variable *State (Current)* | Does not equal | *State* (Baseline).
 - c) Boolean Expression: **A or B**.

2. Define Diagnosis

Add Note **Add Diagnosis** 1 Can also click a variable on the left to add automation.

Name: 2 Anchor: 3

4 ☒ Loop Table Rows Virtual_Server 5

If

A I2 Current Baseline

Status Does not equal Status

B I2 Current Baseline

state Does not equal state

C Select Variable

Boolean Expression:

6

Then

7. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**) and message as:
 On `$this_device Virtual Server $Virtual_Server` status or state has changed:
 Current status is `$Status` and Baseline status was `$Status(Baseline)`
 Current state is `$state` and Baseline state was `$state(Baseline)`
8. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
9. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as
`$this_device Virtual Server $Virtual_Server status $Status or state$ state is not changed.`
10. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
11. Click **Apply** to save the settings and then close the window.

2. Define Diagnosis

Add Note
Add Diagnosis
Can also click a variable on the left to add automation.

Diagnosis Message:
Save to Incident

7

On `$this_device` Virtual Server `$Virtual_Server` status or state has changed: Current

8

Set Status Code for Device:
Error
On `$this_device` Virtual Server `$Virtual_Server` status or state has changed:

Set Status Code for Intent:
Error
On `$this_device` Virtual Server `$Virtual_Server` status or state has changed:

Add Logic

Else
Delete

Diagnosis Message:
Save to Incident

9

vice Virtual Server `$Virtual_Server` status: `$Status` and state: `$state` is not changed

10

Set Status Code for Device:
Success
`$this_device` Virtual Server `$Virtual_Server` status: `$Status` and state: `$state` is

Set Status Code for Intent:
Success
`$this_device` Virtual Server `$Virtual_Server` status: `$Status` and state: `$state` is

Add Logic

+ Add Elself

Can 11 Apply

72 | NetBrain R11.1b

3.4.2.2 Create a Diagnosis to check virtual server CPU usage

1. Go to the **Define Diagnosis** section and click on **Add Diagnosis**.
2. Enter the diagnosis name as **Check CPU Usage**.
3. **Anchor:** Select the variable `$Virtual_server` from the drop-down menu.
4. Check in the **Loop Table Rows** and select the table **Virtual_server**.
5. Select the variable **\$Virtual_server** for the Table Key.
6. Define the **If** condition as:

A: Variable `CPU_Usage_Minutes` | Greater than | 50

The screenshot shows the '2. Define Diagnosis' interface. At the top, there is a blue header with a right-pointing arrow and the text '2. Define Diagnosis'. Below the header, there are two buttons: 'Add Note' (with a notepad icon) and 'Add Diagnosis' (with a blue square icon). A red circle with the number '1' points to the 'Add Diagnosis' button. To the right of these buttons is the text 'Can also click a variable on the left to add automation.' Below this, there is a form with several fields. The 'Name' field contains 'Check CPU usage', with a red circle '2' pointing to it. The 'Anchor' field contains 'Virtual_Server.\$Virt...', with a red circle '3' pointing to it. Below the 'Name' field is a text area with the placeholder 'Type description of the diagnosis...'. Below the text area, there is a section for 'Loop Table Rows'. A red circle '4' points to the checked checkbox for 'Loop Table Rows'. To the right of the checkbox is a dropdown menu showing 'Virtual_Server'. Below this, there is a 'Table Key' field containing 'Virtual_Server', with a red circle '5' pointing to it. Below the 'Table Key' field is a section for 'If' conditions. A red circle '6' points to the first condition, which is 'A'. The condition is defined as 'CPU_Usage_Minutes' (selected from a dropdown), 'Greater than' (selected from a dropdown), and '50' (entered in a text field). Below the 'If' section is a 'Then' section, which is currently empty.

7. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**) and message as "`$this_device Virtual server $Virtual_Server, CPU usage $CPU_Usage_Minutes is high`".

NOTE: By typing \$, you can get the variable selection pop up.

8. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

9. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as `$this_device Virtual Server $Virtual_Server, CPU usage $CPU_Usage_Minutes is normal`.
10. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
11. Click **Apply** to save the settings and then close the window.

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Then

Diagnosis Message: ☐ Save to Incident

7 \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes is high

8 ☒ Set Status Code for Device: Error \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes

☒ Set Status Code for Intent: Error \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes

Add Logic

Else Delete

9 \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes is normal

10 ☒ Set Status Code for Device: Success \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes

☒ Set Status Code for Intent: Success \$this_device Virtual server \$Virtual_Server, CPU usage \$CPU_Usage_Minutes

+ Add Else

Cancel 11 Apply

12. Back in the **Network Intent** window, click **Save** to save the intent to the Intent Manager.

The screenshot shows the 'Network Intent (Edit Mode)' window. At the top, there's a title bar and a toolbar with buttons for 'Run', 'with Live Data', 'Save' (highlighted with a red circle and the number 12), and 'Help'. Below the toolbar, there's a section for 'Monitor F5 Virtual Server' with a 'Diagnosis Tree' icon. A text input field says 'Type description here...'. Below this, there are tabs for 'Seed Logic' and 'Replication Logic' (with a warning icon), and a '+ Device' button. To the right, there's a section for 'Intent Variables: Manager' and a 'Tag: + Add' button. The main area is divided into two panes. The left pane shows a list of intent items, with 'I2' selected. The right pane shows the details for 'I2', which is 'show ltm virtual detail recursive'. The details include a 'Type Description here' field and a list of intent items. The first item is 'root@(I2)(cfg-sync Sync Failed)(Active)(/)(tmsh)#show ltm virtual detail recursive'. Below this, there's a table showing the output of the command. The table has columns for 'ClientSide', 'Ephemeral', and 'General'. The output shows that the virtual server is available and has a destination of 10.1.217.50:80. On the right side of the details pane, there are three diagnostic actions: 'Check CPU usage', 'Pool Member State and Statu...', and 'Virtual Server Status and State'. Each action has a 'Message' and a 'Status Code' button.

Network Intent (Edit Mode)

1 Monitor F5 Virtual Server Diagnosis Tree

Run with Live Data 12 Save Help

Type description here...

Seed Logic Replication Logic

+ Device

Intent Variables: Manager Tag: + Add

I2 Type Description here

+ Add Config Diagnosis + Add CLI Diagnosis

show ltm virtual detail recursive Type Description here

Edit Diagnosis

```
1 root@(I2)(cfg-sync Sync Failed)(Active)(/)(tmsh)#show ltm virtual detail recursive
2
3 -----
4 Ltm::Virtual Server: Common/F5-Lab
5 -----
6 Status
7 Availability : available
8 State : enabled
9 Reason : The virtual server is available
10 CMP : enabled
11 CMP Mode : all-cpus
12 Destination : 10.1.217.50:80
13
14 Traffic ClientSide Ephemeral General
15 Bits In 176.0K 0 -
16 Bits Out 177.7K 0 -
```

Check CPU usage Message Status Code

Pool Member State and Statu... Message Status Code

Virtual Server Status and State Message Status Code

3.4.2.3 Create a Diagnosis to check pool member Status and State

1. Go to the **Define Diagnosis** section and click on **Add Diagnosis**.
2. Enter the diagnosis name as Pool Member State and Status check.
3. **Anchor:** Select the variable `$Pool_member` from the drop-down menu.
4. Check in the **Loop Table Rows** and select the table **Pool_members**.
5. Select the variable **`$Pool_member`** for the Table Key.
6. Define the **If** condition as:
 - a) A: Variable `Status_poolmember(Current)` | Does not equal | `Status_poolmember(Baseline)`
 - b) B: Variable `State_poolmember(Current)` | Does not equal | `State_poolmember(Baseline)`
 - c) Boolean Expression: **A or B**.

The screenshot shows the '2. Define Diagnosis' configuration window. It includes a header bar with 'Add Note' and 'Add Diagnosis' buttons. Below this, there are fields for 'Name' (set to 'Pool Member State and Status check') and 'Anchor' (set to 'Pool_members.\$Po...'). A description field is also present. The 'Loop Table Rows' section is checked, and 'Pool_members' is selected as the table. The 'Table Key' is set to 'Pool_Member'. The 'If' condition section is expanded, showing two conditions: 'A' (Status_poolmember Current vs Baseline, Does not equal) and 'B' (State_poolmember Current vs Baseline, Does not equal). A third condition 'C' is set to 'Select Variable'. The 'Boolean Expression' field at the bottom is set to 'A or B'.

2. Define Diagnosis

Add Note Add Diagnosis 1 Can also click a variable on the left to add automation.

Name: Pool Member State and Status check 2 Anchor: Pool_members.\$Po... 3

Type description of the diagnosis...

4 ☒ Loop Table Rows Pool_members Table Key: Pool_Member 5

If 6

A I2 Current Baseline

a Status_poolmember Does not equal Status_poolmember

B I2 Current Baseline

b State_poolmember Does not equal State_poolmember

C Select Variable

Boolean Expression: A or B c

7. **Then:** In case **If** logic is true, define the color (**red**), status (**Error**) and message as “On `$this_device`, Pool Member `$Pool_Member` status or state has changed: Current status is `$Status_poolmember` and Baseline status was `$Status_poolmember(Baseline)`
Current state is `$State_poolmember` and Baseline state was `$State_poolmember(Baseline)`”

NOTE: By typing \$, you can get the variable selection pop up.

8. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

9. **Else:** In case **If** logic is not true, define the color (**green**), status (**Success**) and message as `$this_device` Pool Member `$Pool_Member` status or state not changed.

NOTE: By typing \$, you can get the variable selection pop up.

10. Check-in both the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

11. Click **Apply** to save the settings and then close the window.

The screenshot shows a configuration window with two main sections: **Then** and **Else**.

Then Section:

- 7. A red callout points to the color selection dropdown, which is set to red.
- 8. A yellow dashed box highlights the **Set Status Code for Device** and **Set Status Code for Intent** checkboxes, both of which are checked. Each checkbox has a red callout pointing to its status code dropdown, which is set to **Error**.

Else Section:

- 9. A red callout points to the color selection dropdown, which is set to green.
- 10. A yellow dashed box highlights the **Set Status Code for Device** and **Set Status Code for Intent** checkboxes, both of which are checked. Each checkbox has a green callout pointing to its status code dropdown, which is set to **Success**.

At the bottom right, a red callout 11 points to the **Apply** button.

3.5 Execute Intents and Create Dashboards

Now, let us create intent dashboards for each intent created in Chapter 3 and previous Chapter 2. And then a summary dashboard which provides an overview displaying results from multiple Intent dashboards.

1. **Intent Dashboard** to view the individual Intent results and
2. **Summary Dashboard** to view the consolidated Intent results in a single view.

3.5.1 Execute Intents

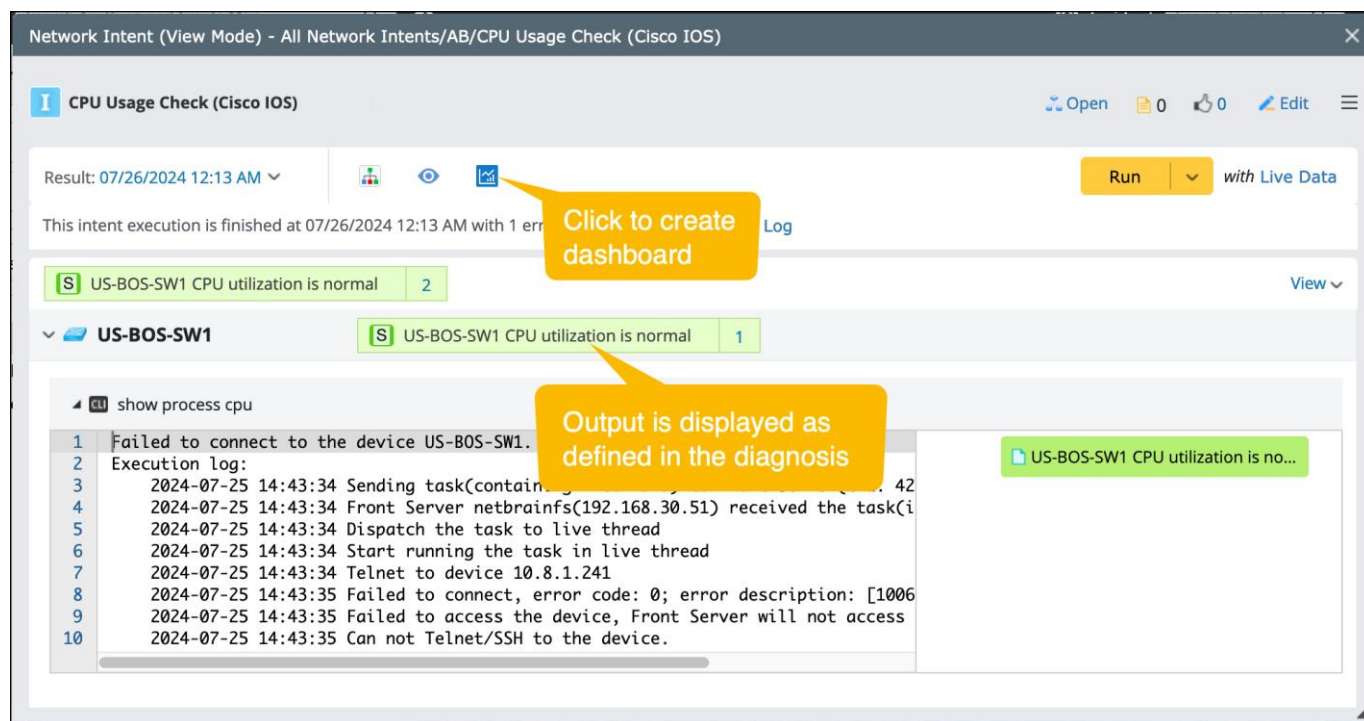
Let us execute all the intents created in this chapter and add them to a dashboard as follows:

1. Open the intent CPU Usage Check (Cisco IOS) from the intent manager.
2. Click **Run**.

The screenshot displays the NetBrain Next-Gen interface. The top navigation bar includes the NetBrain logo, a search bar, and an 'Incident' button. The left sidebar contains icons for Recents, Network, Files, Site, Path, Dashboard, Intents, Chatbot, and Data. The main content area is divided into two sections. The top section, 'Network Intent Manager > Common Intent', shows a list of intents under the 'AB (3)' category. The 'CPU Usage Check (Cisco IOS)' intent is highlighted, and a red circle with the number '1' points to the 'Open' button next to it. The bottom section, 'Network Intent (View Mode) - All Network Intents/AB/CPU Usage Check (Cisco IOS)', shows the details of the selected intent. It includes a 'Run' button with a red circle and the number '2' next to it. Below the 'Run' button, the text 'No result available because this intent has not been executed.' is displayed. At the bottom, a table shows the output of the 'show process cpu' command for the device 'US-BOS-SW1'.


1	US-BOS-SW1>show process cpu									
2	CPU utilization for five seconds: 0%/0%; one minute: 0%; five minutes: 0%									
3	PID	Runtime(ms)	Invoked	uSecs	SSec	1Min	5Min	TTY	Process	
4	1	0	9	0	0.00%	0.00%	0.00%	0	Chunk Manager	
5	2	130752	3604525	36	0.00%	0.00%	0.00%	0	Load Meter	
6	3	0	30	0	0.00%	0.00%	0.00%	0	SpanTree Flush	
7	4	1295180	2435550	531	0.00%	0.01%	0.00%	0	Check heaps	
8	5	8060	300377	26	0.00%	0.00%	0.00%	0	Pool Manager	
9	6	0	1	0	0.00%	0.00%	0.00%	0	DiscardQ Backgr	
10	7	0	2	0	0.00%	0.00%	0.00%	0	Timers	
11	8	6404	1339739	4	0.00%	0.00%	0.00%	0	WATCH_AFS	
12	9	4	1	4000	0.00%	0.00%	0.00%	0	OIR Handler	
13	10	0	1	0	0.00%	0.00%	0.00%	0	IFS Agent Manag	

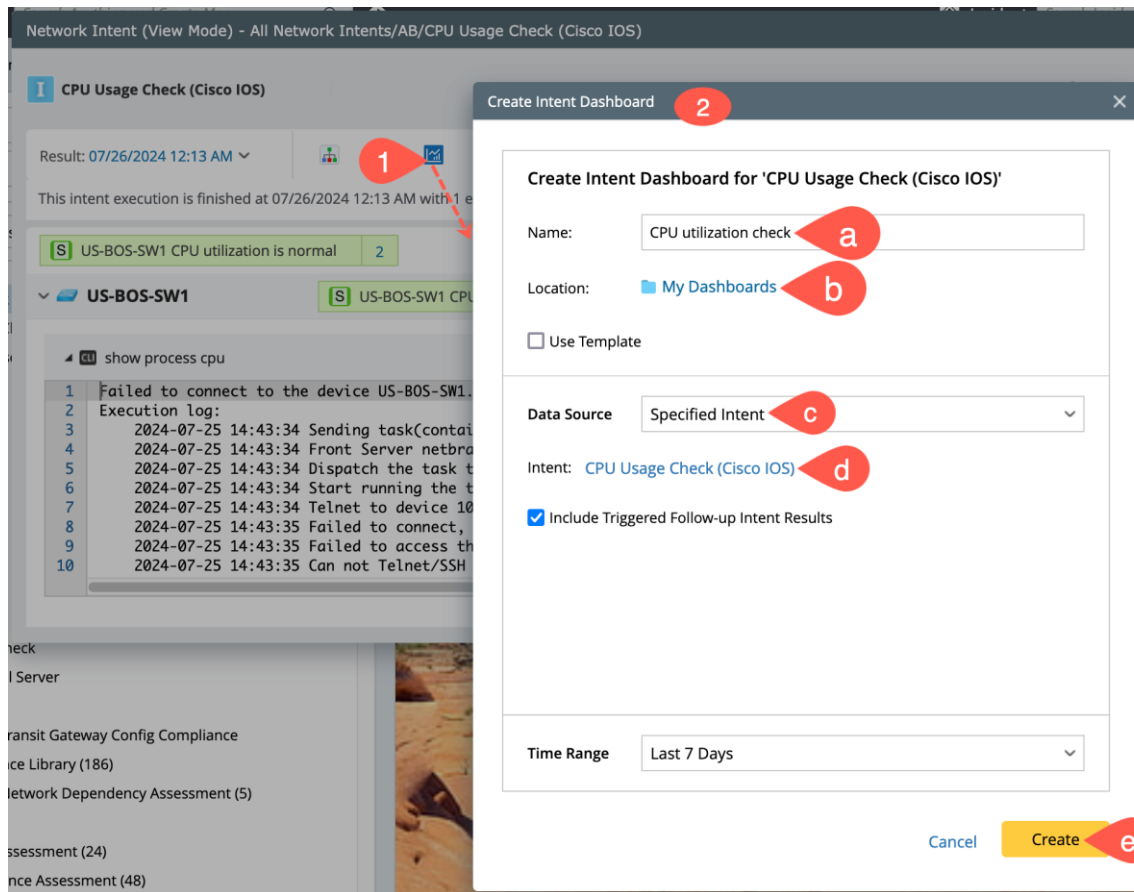
- After successful execution, the results will appear as defined in the diagnosis section. To create the intent dashboard from this window, go to next Section 3.4.2.



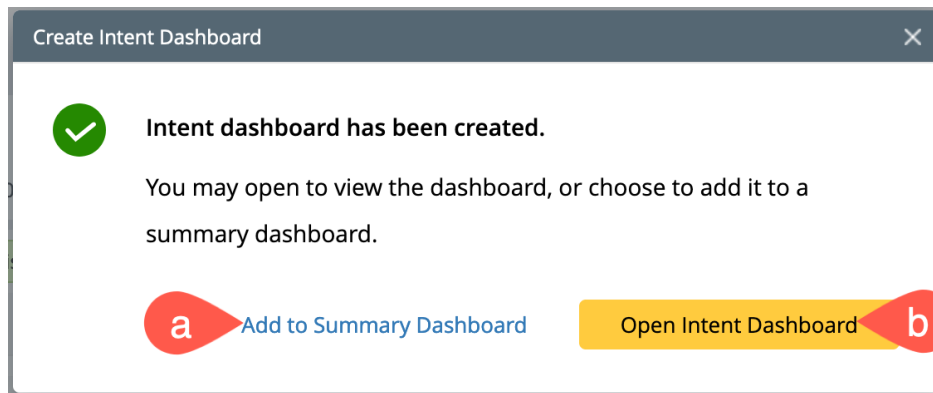
3.5.2 Intent Dashboard

Create an Intent Dashboard directly from the Network Intent results window as follows:

- Click the **Create Intent Dashboard** icon  in the intent results window to open the corresponding window.
- In the Create Intent Dashboard window:
 - Enter the Dashboard Name as **CPU utilization check**.
 - Select the **Location** to save the Intent Dashboard.
 - Data Source:** By default, **Specified Intent** is selected from the dropdown.
 - Intent:** Created intent is selected by default. Keep it the same.
 - Click **Create**.



3. A prompt window appears with the success message Intent Dashboard has been created with two options: Open Intent Dashboard or Add to Summary Dashboard.



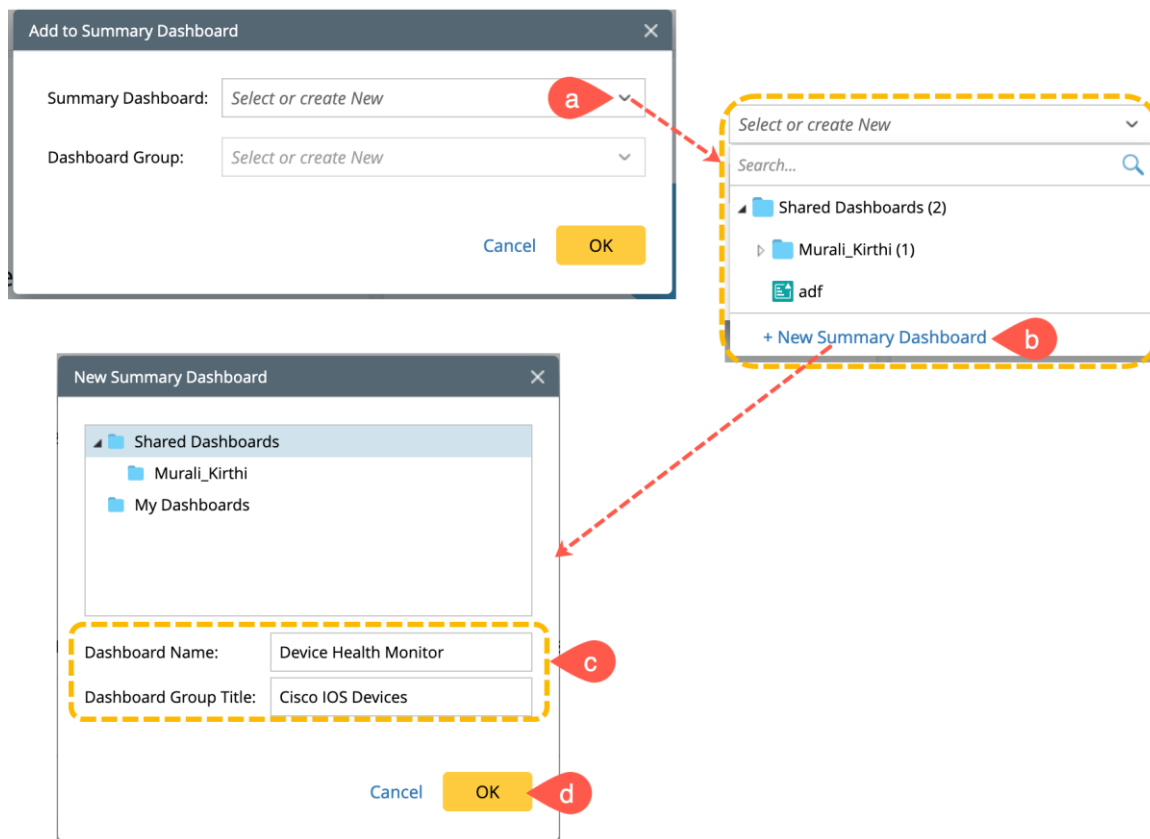
- a) **Add to Summary Dashboard:** The Summary dashboard provides an overview displaying results from multiple Intent dashboards.
 - b) **Open Intent Dashboard:** Displays the intent executed results of the created intent. Refer to Section 2.9.2 for more information.
4. Click **Add to Summary Dashboard**. Refer to next [Section 3.4.3](#) to view all the Cisco device network intent results together in a single dashboard.

3.5.3 Summary Dashboard

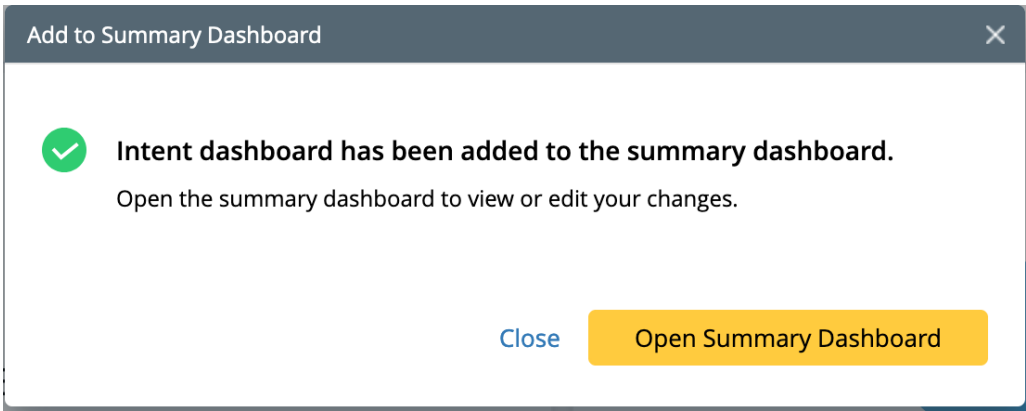
The summary dashboard provides an overview displaying results from multiple Intent dashboards of the entire network or a set of network devices. With Summary Dashboard, you can group Intent Dashboards into widgets based on diagnosis purpose and display results by device, site or device groups. You can use the summary dashboard to monitor critical information across thousands of devices and discover the root cause for issues in one view.

Let us create a Summary Dashboard using the step-by-step instructions as follows:

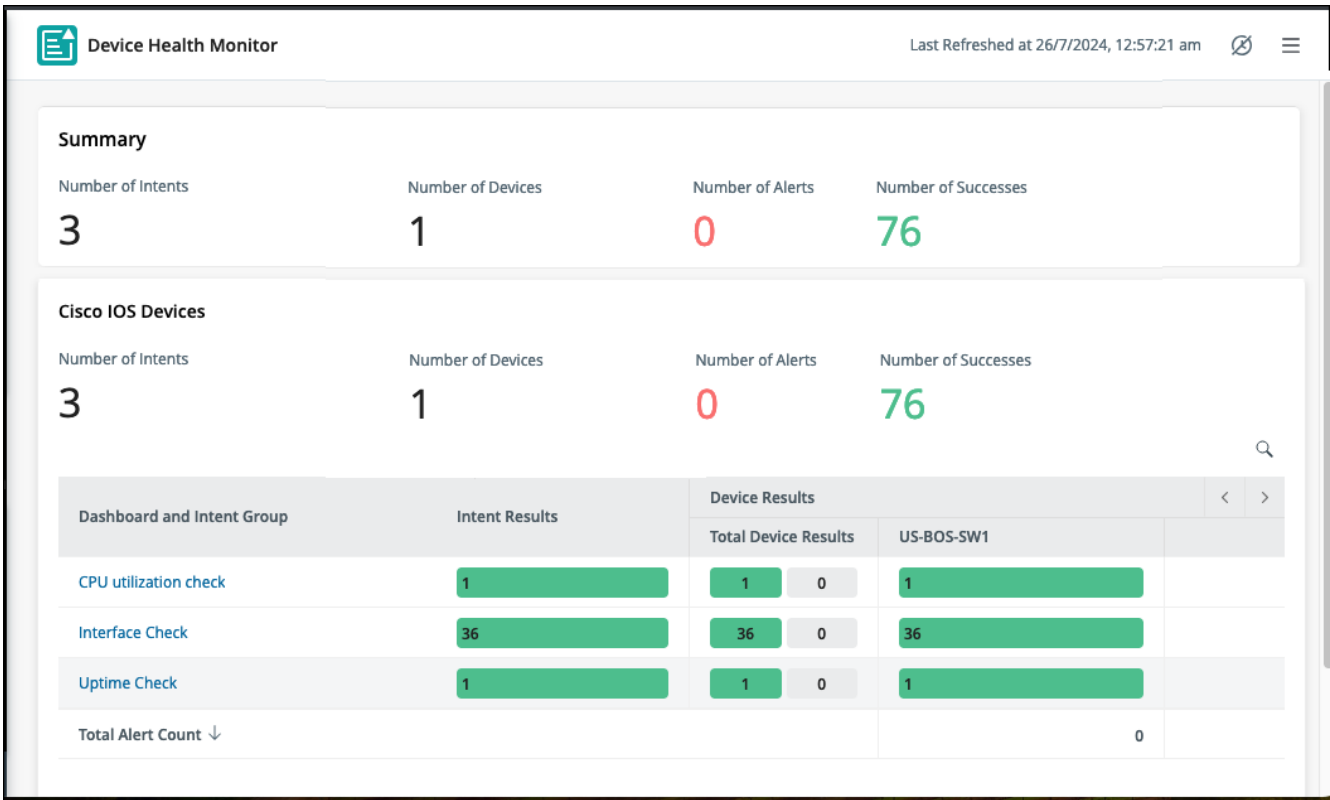
1. After selecting **Add to Summary Dashboard**, its corresponding window will appear.
2. In the **Add to Summary Dashboard** window, let us create a new summary dashboard and new dashboard group as follows:
 - a) **Summary Dashboard**: open the dropdown menu.
 - b) Select **+New Summary Dashboard** to pop up its dialogue.
 - c) Enter the dashboard basic details like **name**, **group title** and **location** of the summary dashboard to save.
 - d) Click **OK** to save and create the summary dashboard.



3. A prompt window appears with the success message "Intent dashboard has been added to the summary dashboard".



4. Close the window and add two other intents, **Uptime check** and **Interface status check**, to dashboards created for Cisco IOS switch devices in Chapter 3. The final summary dashboard will be:



4 Enable Others to Use Your Automation

In the last two chapters, you learned how to create intents to automate a task usually done via the CLI commands. The intent results can be viewed in the map, the diagnosis tree, and in the Dashboard. An intent is always associated with one or multiple devices. The intent diagnosis can be duplicated to other devices, which are included as the seed devices. However, this duplication method is not scalable. In this chapter, you will learn a scalable way to replicate an intent to other devices via an easy-to-follow wizard, **Auto intent wizard**. While a user opens a map, The intents (**auto intents**) that are qualified for the devices in the map will be listed in the **Auto Intent** pane.

In this chapter, we will introduce a key feature, **follow up intent**, and teach how you can use the follow up intent to encapsulate a set of intents into a parent intent. This parent intent, sometimes called **wrapper intent**, hides the detailed intent implementation from the end user and can function as a public interface to the end user, which will be accessed as an auto intent or as a chatbot.

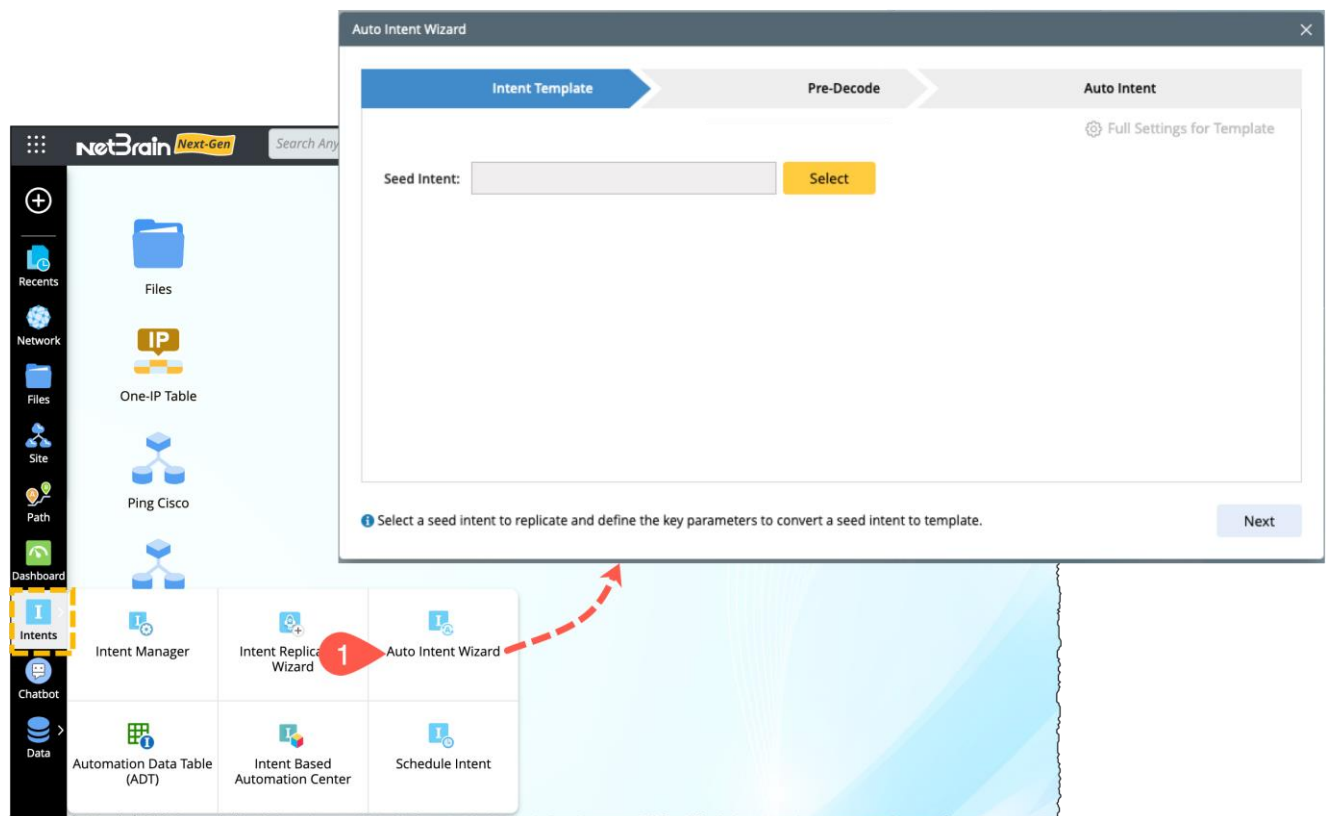
This chapter will reuse the intents created in previous [Section 2](#) and [Section 3](#) and cover the following flows:

1. [Use Auto Intent Wizard for others to use the automation.](#)
2. [Use Follow up intent to create a wrapper intent.](#)
3. [Chatbot to run intents.](#)

4.1 Auto Intent Wizard: Replicate Intent to Multiple Devices

Replicating the intents to multiple devices with **Auto Intent Wizard** is straightforward. The steps involved are detailed as follows:

1. Click the **Intents** menu located on the desktop sidebar and select **Auto Intent Wizard** to open its corresponding wizard.
2. Input the data for replication in the following three ribbons of the Auto Intent Wizard:
 - [Intent Template](#)
 - [Pre Decode](#)
 - [Auto Intent](#)



4.1.1 Intent Template

In this ribbon, you will provide the basic details like seed intent, target devices and template settings such as **macro variables**.

1. **Seed Intent:** Click **Select** and choose the intent **Ping Cisco** from the **Select Intent** dialog.
2. **Intent Template for:** Choose the **Device-based Replication**.
3. Intent Qualification: Click the **Select** drop-down menu beside **Define via Device Group/Sites**.
4. Click **Select Device Groups** to open its corresponding window.
 - a) Check the selection box of the device group you want to replicate.
 - b) Click **OK**.

Auto Intent Wizard - Ping Cisco

Intent Template Pre-Decode Auto Intent

Full Settings for Template

Seed Intent: Ping Cisco **Select** 1 modified at: 12:25:21 AM 07/20/2024

Intent Template for: ☒ Device-based Replication 2 ☐ Path-based Replication

Intent Qualification: ☒ via Device Groups/Sites: **Select** 3 ☐ via Dynamic Search: Undefined

Define Macro Variables and Rules for Their Substitution

Items: 4

Seed Device	Seed Command
US-BOS-SW1	cu ping 10.8.1.1
US-BOS-SW2	cu ping 10.8.1.1
US-BOS-SW3	cu ping 10.8.1.1
US-BOS-SW4	cu ping 10.8.1.1

Please click to select an entry from the list

Select Device Groups 4

- Select Device Groups
- Select Sites
- New Device Group

Add Device Group

- All Device Groups
- My Device Groups
 - ☒ Cisco IOS devices (4) a
 - ☐ Cisco IOS switches (4)
- Shared Device Groups
 - ☐ Ahmed
 - ☐ Ajay
 - ☐ Anurag
 - ☐ Automation Library

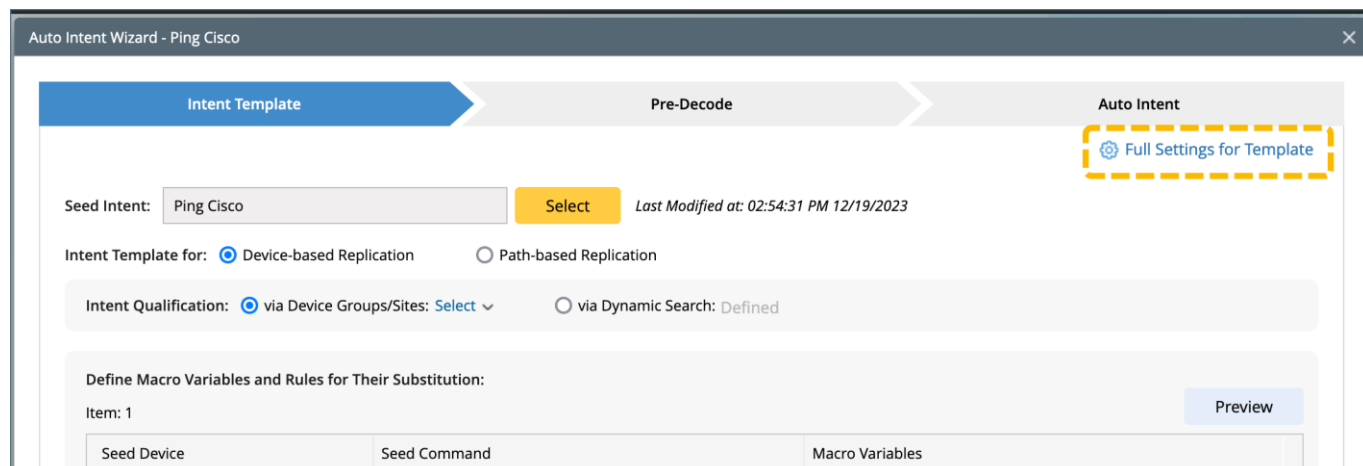
Cancel **OK** b

1 Select a seed intent to replicate and define the key parameters to convert a seed intent to template.

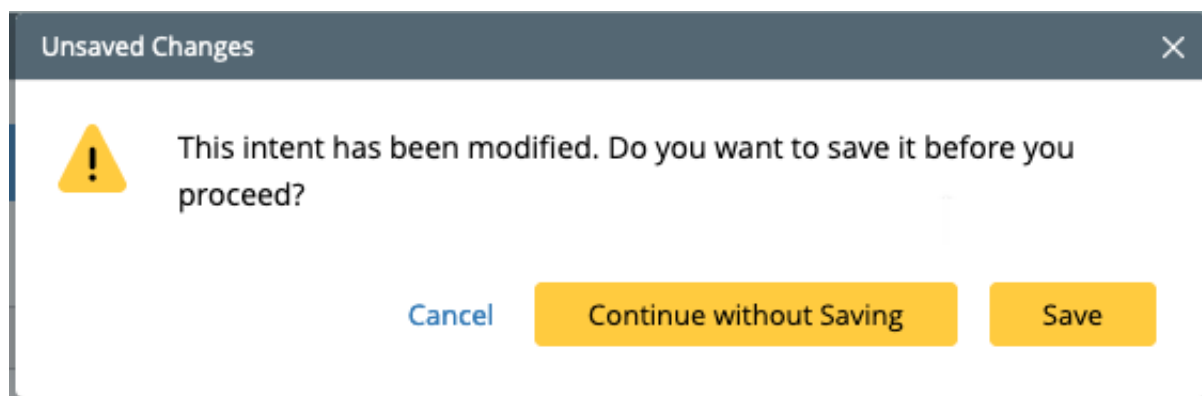
Next

Now, set the option to pass an interface using **Full Settings for Template** located at the top right corner.

5. Click **Full Settings for Template**.



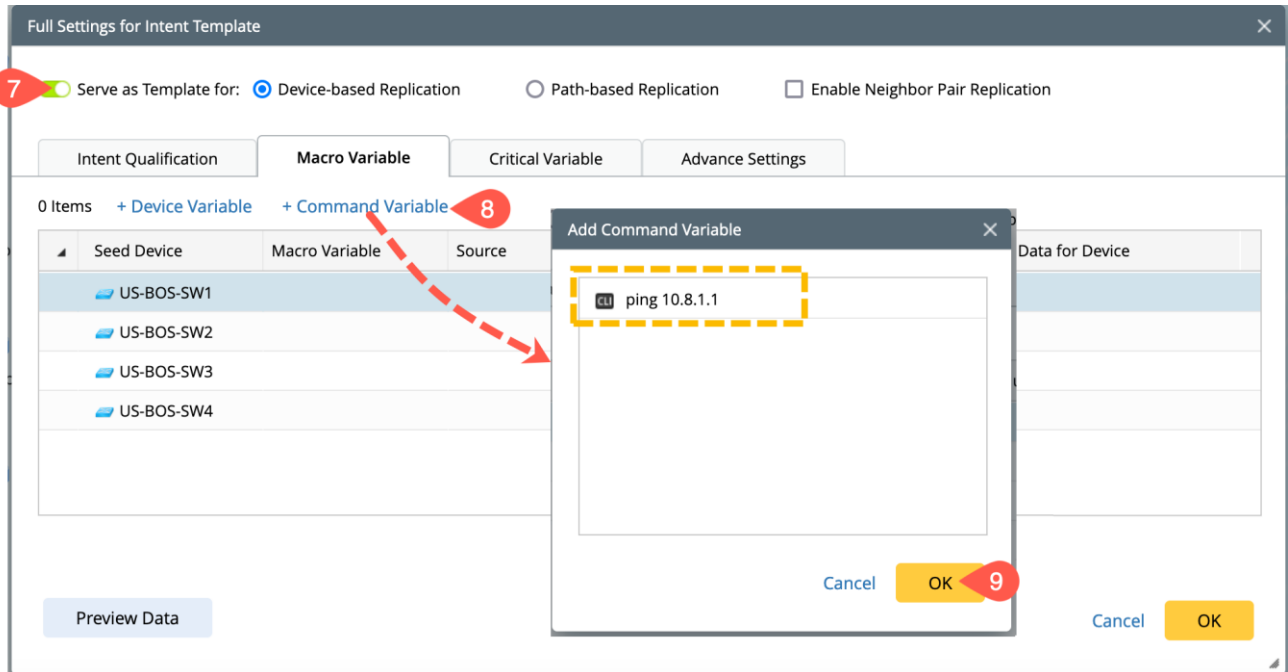
6. A prompt will appear to save the intent modifications before proceeding further, Click **Save**.



7. Toggle the Intent Template on by clicking on the **Serve as Template for**.
8. Go to the **Macro Variable** tab and click **+ Command Variable** to select the command to customize.

A **Macro Variable** is a special intent variable that can be used as an input of an intent. For example, for the intent using the command **ping 10.8.1.1**, you can replace the value **10.8.1.1** as a macro variable **\$server_ip** (the command becomes **ping \$server_ip**) so that a user can enter any server IP address while executing the intent. Besides functioning as the user input for the CLI command, Macro Variables can also be used to pass the values from a parent intent to follow up intents, which will be covered later.

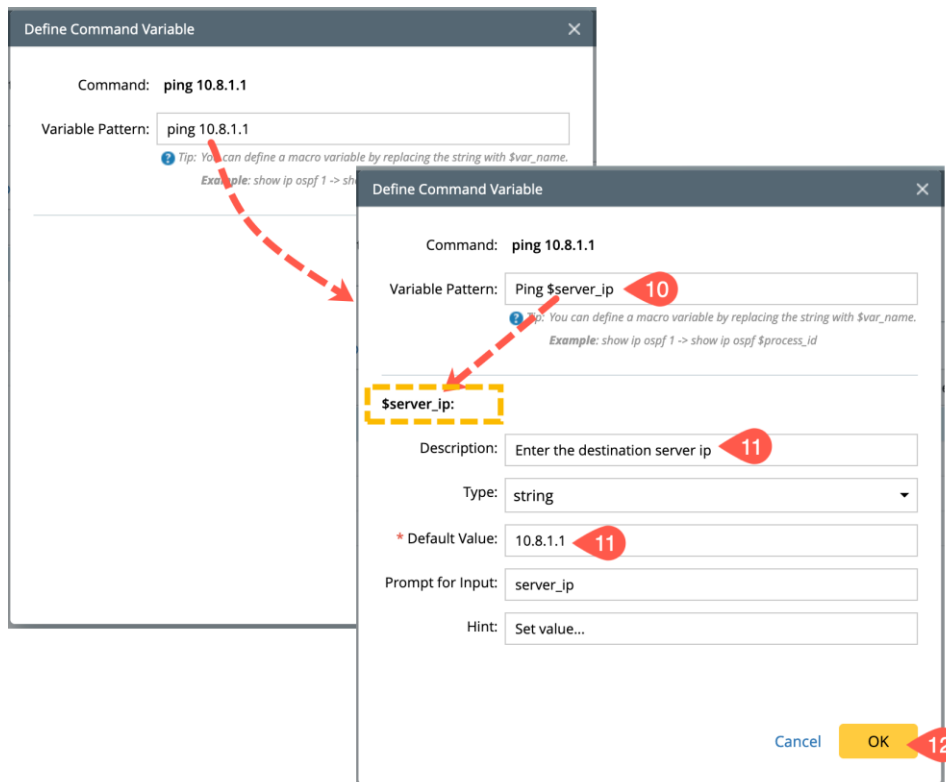
9. Select the **ping 10.8.1.1** command and click **OK** to open the variable pattern modifier window.



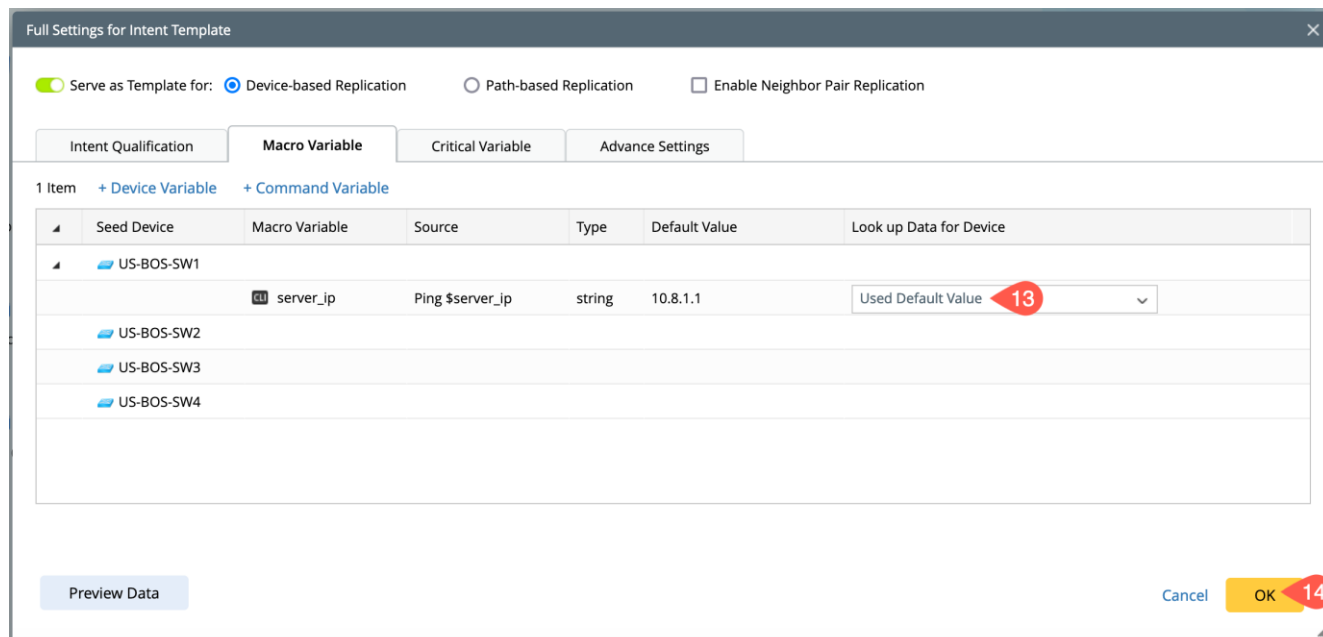
10. In the Variable Pattern dialog, modify **Ping 10.8.1.1** with **Ping \$server_ip**.

11. Add the **Description** and set the **Default Value** to **10.8.1.1**.

12. Click **OK** to save and close your settings.



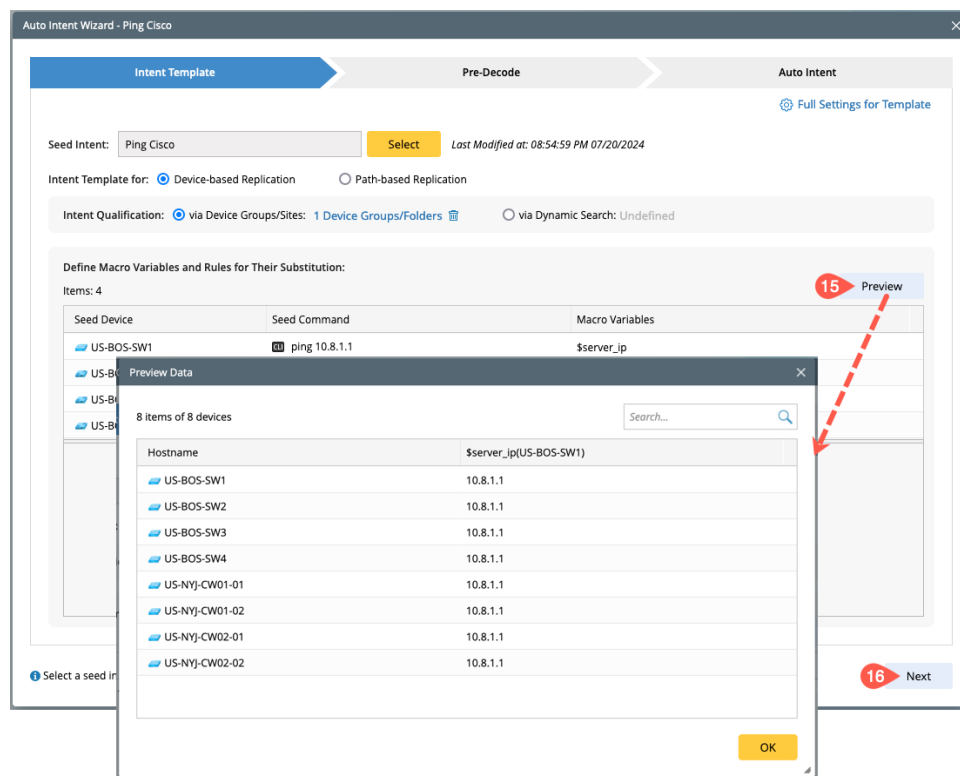
13. You will be directed back to template settings. Set the **Used Default Value** option in the **Look up Data for Device** column from the dropdown menu.
14. Click **OK** to save and close the window.



The 'Full Settings for Intent Template' window shows the 'Macro Variable' tab. It contains a table with columns: Seed Device, Macro Variable, Source, Type, Default Value, and Look up Data for Device. The table has one item, 'US-BOS-SW1', with a macro variable 'server_ip' and a default value of '10.8.1.1'. The 'Look up Data for Device' column has a dropdown menu with 'Used Default Value' selected. A red circle with the number 13 points to this dropdown. At the bottom right, there is a yellow 'OK' button with a red circle and the number 14 next to it, and a 'Cancel' button.

Seed Device	Macro Variable	Source	Type	Default Value	Look up Data for Device
US-BOS-SW1	server_ip	Ping \$server_ip	string	10.8.1.1	Used Default Value
US-BOS-SW2					
US-BOS-SW3					
US-BOS-SW4					

15. Click the **Preview** button to validate the list of target devices configured with Macro Variables.
16. Click **Next** in the Intent Replication Wizard to move to the **Pre-Decode** step.



The 'Auto Intent Wizard - Ping Cisco' window shows the 'Pre-Decode' step. It has a 'Seed Intent' field with 'Ping Cisco' and a 'Select' button. Below it, there are radio buttons for 'Intent Template for: Device-based Replication' (selected) and 'Path-based Replication'. Under 'Intent Qualification', there are radio buttons for 'via Device Groups/Sites: 1 Device Groups/Folders' (selected) and 'via Dynamic Search: Undefined'. A 'Preview' button is highlighted with a red circle and the number 15. A 'Next' button is highlighted with a red circle and the number 16. A 'Preview Data' window is open, showing a table with columns: Hostname, \$server_ip(US-BOS-SW1), and a search bar. The table lists 8 items of 8 devices, including US-BOS-SW1, US-BOS-SW2, US-BOS-SW3, US-BOS-SW4, US-NYj-CW01-01, US-NYj-CW01-02, US-NYj-CW02-01, and US-NYj-CW02-02. An 'OK' button is at the bottom right of the 'Preview Data' window.

Seed Device	Seed Command	Macro Variables
US-BOS-SW1	ping 10.8.1.1	\$server_ip
US-BOS-SW2		
US-BOS-SW3		
US-BOS-SW4		

Hostname	\$server_ip(US-BOS-SW1)
US-BOS-SW1	10.8.1.1
US-BOS-SW2	10.8.1.1
US-BOS-SW3	10.8.1.1
US-BOS-SW4	10.8.1.1
US-NYj-CW01-01	10.8.1.1
US-NYj-CW01-02	10.8.1.1
US-NYj-CW02-01	10.8.1.1
US-NYj-CW02-02	10.8.1.1

4.1.2 Pre Decode

With the settings configured in this ribbon **Pre Decode**, the seed intent will be decoded to the target devices. If the Netbrain system fails to retrieve and parse the critical variable of the target devices, then the device will not be qualified for intent replication.

1. **Install Intent Template to:** Select a group from the dropdown menu to add the intent in the IBA center.
2. Click the **Install & Decode** button to install the intent to the target devices.

NOTE: The status of the intent installation is displayed next to the **Install & Decode** button.

Auto Intent Wizard - Ping Cisco

Intent Template **Pre-Decode** Auto Intent

Intent Based Automation Center

Install Intent Template to: Automation Cookbook 1 ▼ [Decoding Settings](#)

2 **Install & Decode** Status: Uninstalled

2 **Decode Now** Status: Not Started ↺

Output: 0 devices decoded for this intent

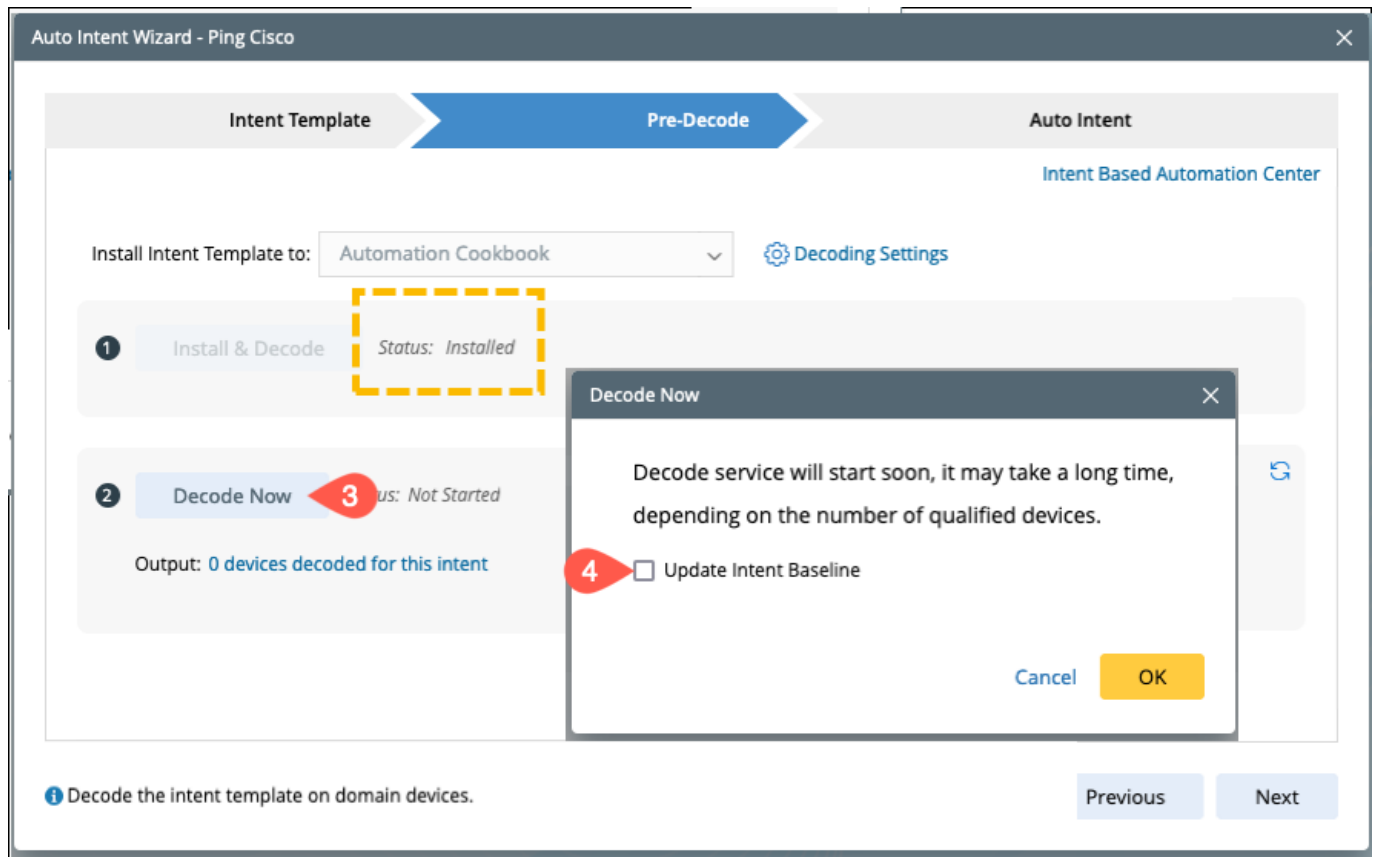
i Decode the intent template on domain devices. Previous Next

3. Click **Decode Now** to start decoding the intent to the target devices.

NOTE: Click the **Refresh** icon after a minute to see the updated decode status. The updated decoded devices can be seen at the bottom of the **Decode Now** button.

4. Check in the **Update Intent Baseline** to update the seed intent baseline information before replicating it to target devices and click **OK**.

NOTE: At this step, wait for the decode to complete by refreshing the status until you see a number of devices have been decoded.



View decoded results from Intent Based Automation Center:

5. Click **Intent Based Automation Center** at the top right corner to open it in a new window.

a) In the Intent Based Automation Center, go to the intent template **Ping Cisco**.

NOTE: Its corresponding results can be seen on the left side of the screen.

b) Click **4 Devices decoded** to view the detailed report and the log of the decoded devices.

Intent Based Automation Center

Installed Intent Templates | Published Intents | Auto Intent | Auto Intent Profile | NetBrain Download

Items: 163 + Add Intent Template Filter: All Search... Refresh

Intent Template Name: Ping Cisco

View Decode Result

Last Decode Task Run at 07/20/2024 09:09:52 PM View Decode Task Creation Log

Total Decoded Devices: 8 Filter: NoAll Trigger Sources,Auto In... Search... Refresh

Matched Device	Matched Seed Device	Matched Command	Create Intent	Last Decoded At	Baseline Data Updated At
US-BOS-SW2	US-BOS-SW3	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:51 PM	07/20/2024 09:08:54 PM
US-BOS-SW1	US-BOS-SW2	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW4	US-BOS-SW2	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW1	US-BOS-SW4	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW4	US-BOS-SW3	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW3	US-BOS-SW3	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW2	US-BOS-SW2	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW3	US-BOS-SW4	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW4	US-BOS-SW4	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW1	US-BOS-SW3	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM
US-BOS-SW3	US-BOS-SW2	ping 10.8.1.1	All Trigger Sourc...	07/20/2024 09:09:52 PM	07/20/2024 09:08:54 PM

8 Devices decoded Apply

6. Go back to Auto Intent Wizard and click **Next** to move on to the Auto Intent section.

Auto Intent Wizard - Ping Cisco

Intent Template | **Pre-Decode** | Auto Intent

Install Intent Template to: Automation Cookbook Decoding Settings

1 Install & Decode Status: Installed

2 Decode Now Status: Last Decoded at 09:09 PM 07/20/2024

Output: 8 devices decoded for this intent

Decode the intent template on domain devices.

Previous **6** Next

4.1.3 Auto Intent

The **Auto Intent** section is the final step:

1. **Enable for Auto Intent:** Enable the option by clicking the radio button.
2. **Select Folder:** Choose a folder location for the Auto Intent by clicking on **Select**.
3. Auto Intent Description: Add a description, e.g., Latency Check.
4. Choose a profile from **Select Profiles**, or create a new profile using **+Add Profile** and click **OK**.
5. Click **Finish** to save and close the Auto Intent Wizard for NIT **Ping Cisco**.

Auto Intent Wizard - Ping Cisco

Intent Template Pre-Decode Auto Intent

Auto Intent Manager

1 ☒ Enable for Auto Intent

Select Folder: AB 2

Auto Intent Description:

Ping Latency Check 3

Member this Intent to Auto Intent Profiles:

1 Profile

My Profiles/Ping

4 Select Profiles

Select Profile

Search...

Shared Profiles

My Profiles

Ping

+Add Profile


Cancel OK

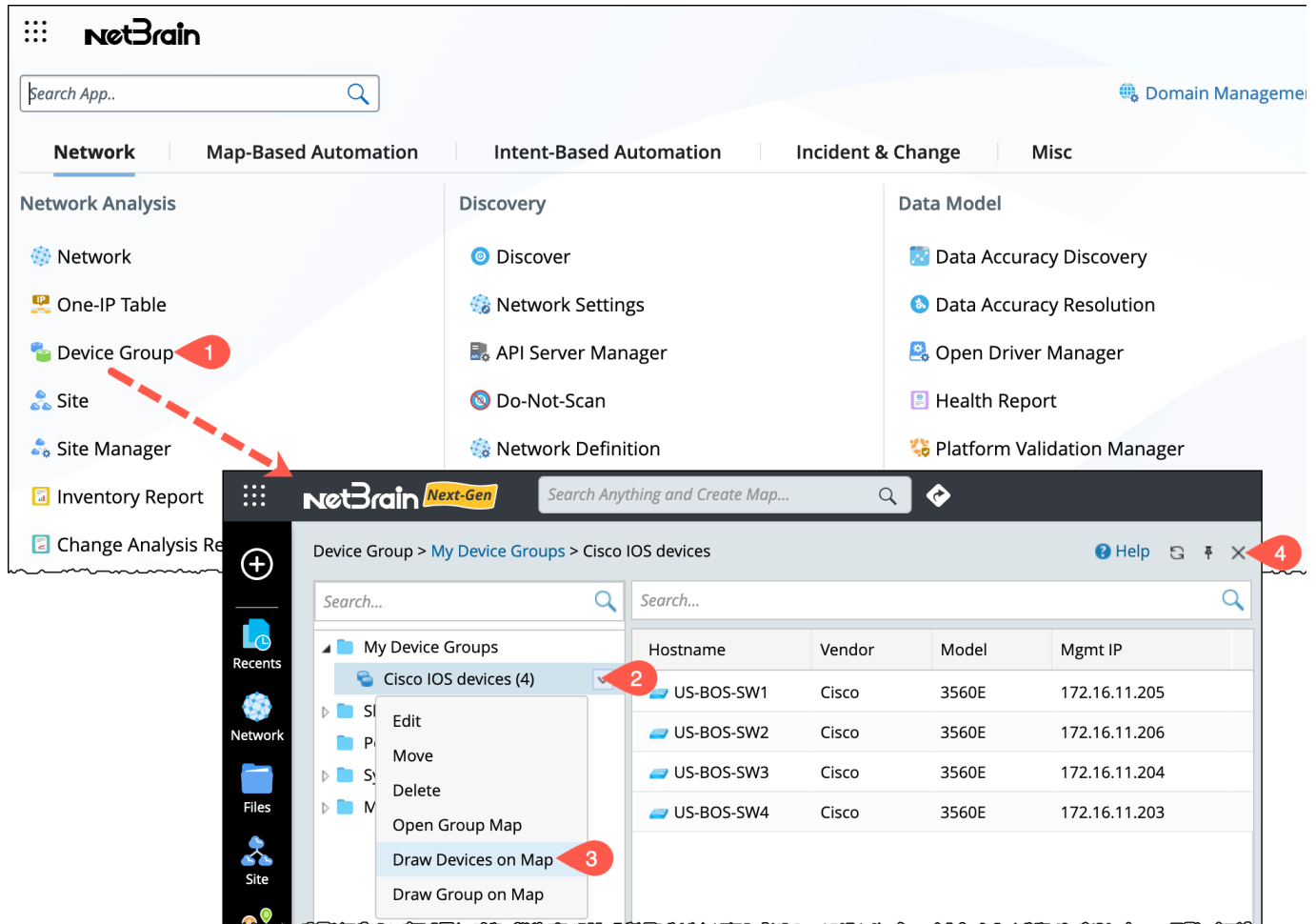
1 Enable Auto Intent for this intent template.

Previous 5 Finish

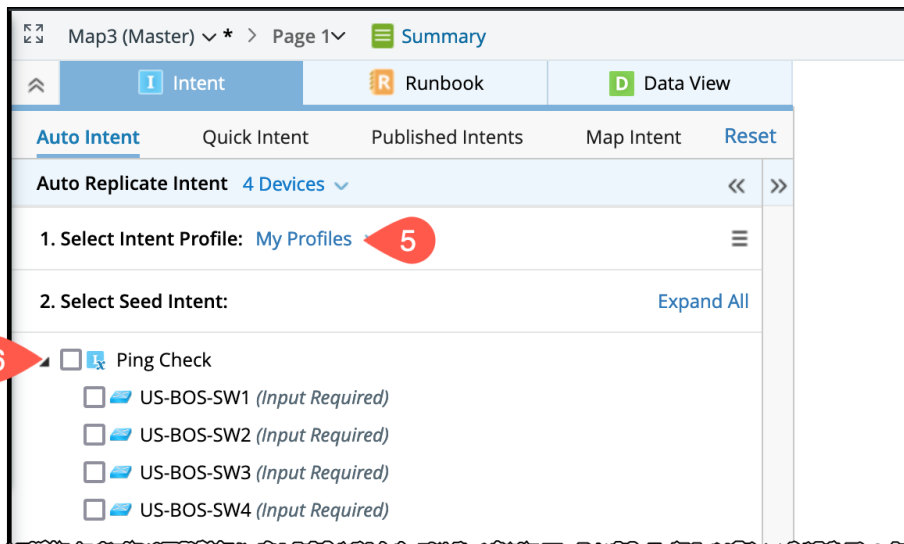
4.1.4 Run the Auto intent:

Execute the intent by drawing the targeted device group to map as follows:

1. On the desktop, go to the  menu located on the top right corner and select **Device Group**.
2. In the device group pane, select the predefined group, **Cisco IOS devices**, and click the drop-down menu to open.
3. Select **Draw Devices on Map** to add the devices to a new map.
4. Close the Device group pane.



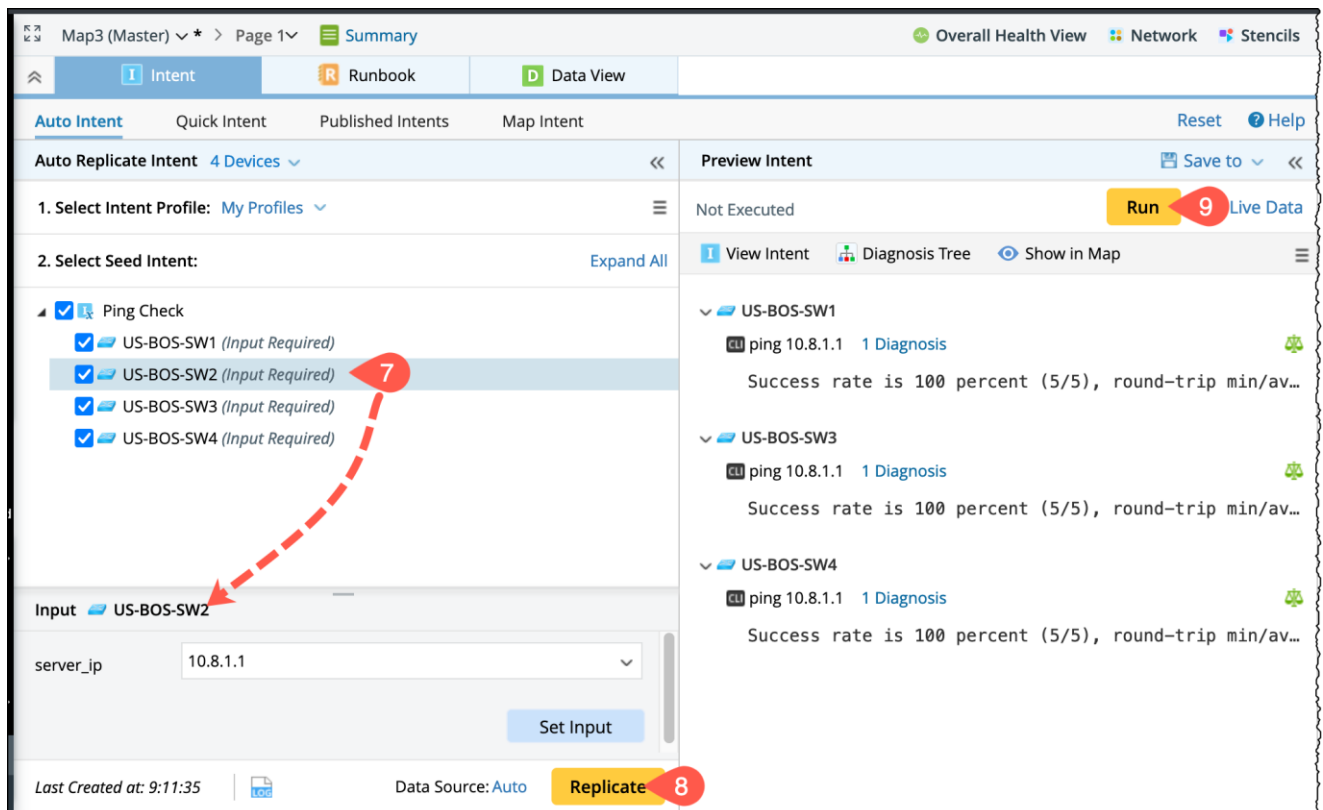
5. Open the **Auto Intent** pane under the **Intent** tab and **Select Profile (My Profiles)** that you have created in the previous section.
6. Check the selection boxes of Intent **Ping Check** and the devices listed underneath.



7. Select each device and the **Input** field appears at the bottom of the intent pane.

NOTE: The default value for the destination **server_ip** will appear for every device. It can be modified as required by clicking **Set Input**.

8. Click **Replicate**, and all the devices will appear under the new **Preview Intent** section.
9. Click **Run** to execute the diagnosis.



10. The result will appear next to each corresponding device in **Preview Intent** and on **Map**.
11. **Save** the Map with all the devices and results for future reference.
12. **Save to Map Intent** to view and run the intent (apart from schedule) as and when needed.

The screenshot displays the NetBrain interface with the 'Map3 (Master)' window. The 'Intent' tab is active, showing the 'Preview Intent' section. A dropdown menu is open, highlighting the 'Save to Map Intent' option, which is associated with step 11. The 'Preview Intent' section shows a list of devices and their ping results:

- US-BOS-SW1: ping success and average round trip time is 1 ms. (1 Success)
- US-BOS-SW3: ping success and average round trip time is 3 ms. (1 Success)
- US-BOS-SW4: ping success and average round trip time is 1 ms. (1 Success)

The 'Map' view on the right shows a network diagram with four switches: US-BOS-SW1, US-BOS-SW3, US-BOS-SW4, and US-BOS-SW2. The results from the 'Preview Intent' section are displayed next to each device in the map. A yellow callout points to the results in the map view, stating: "Results appearing next to each device".

4.2 Wrapper Intent as End User Interface

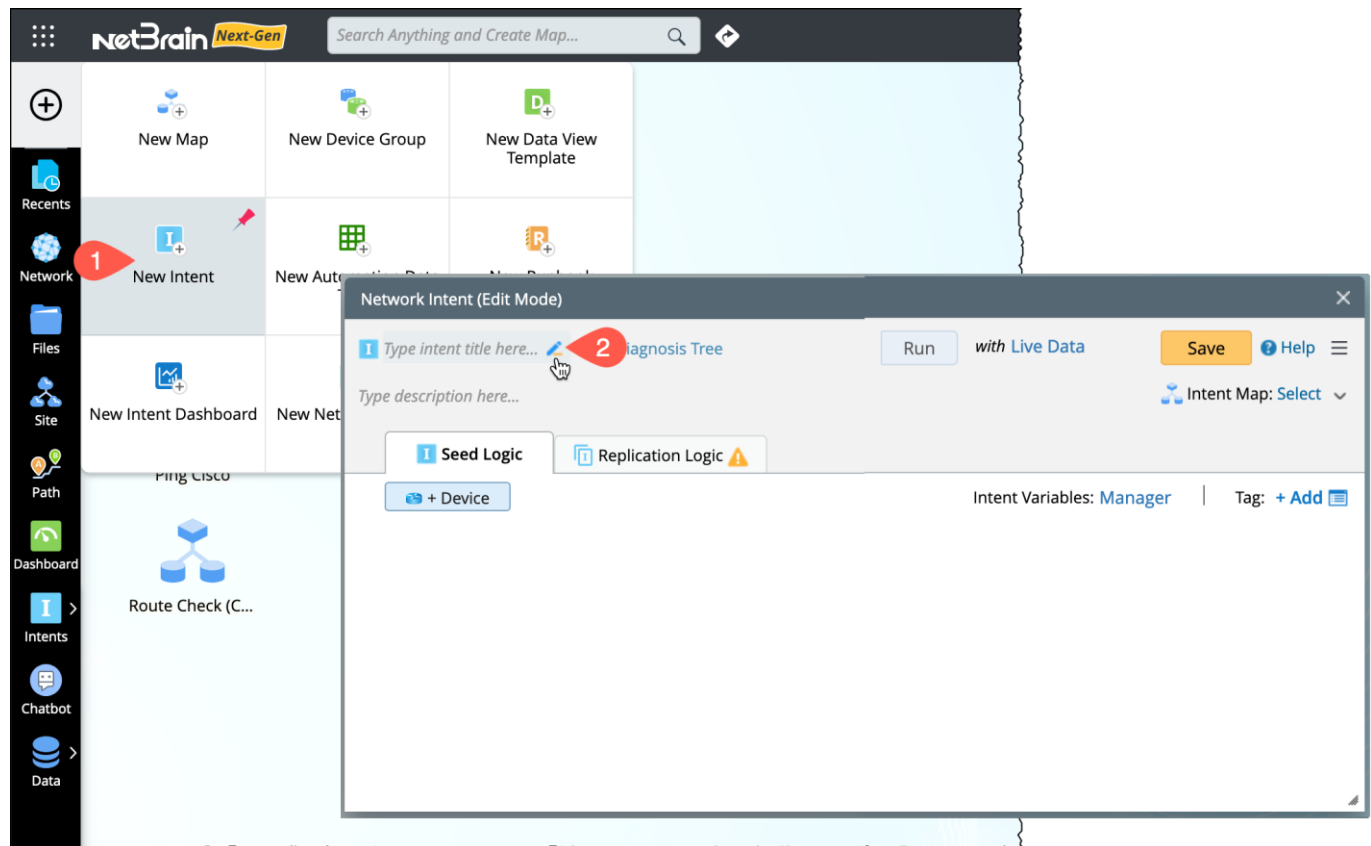
Often, you want to hide an intent implementation from an end user so that any improvement of your intent will not affect how others run the intent. You can use the **follow up intent** to encapsulate a set of intents into a parent intent. This parent intent, sometimes called **wrapper intent**, hides the detailed intent implementation from the end user and can function as a public interface to the end user, which will be accessed as an auto intent or as a chatbot.


This section covers two topics:

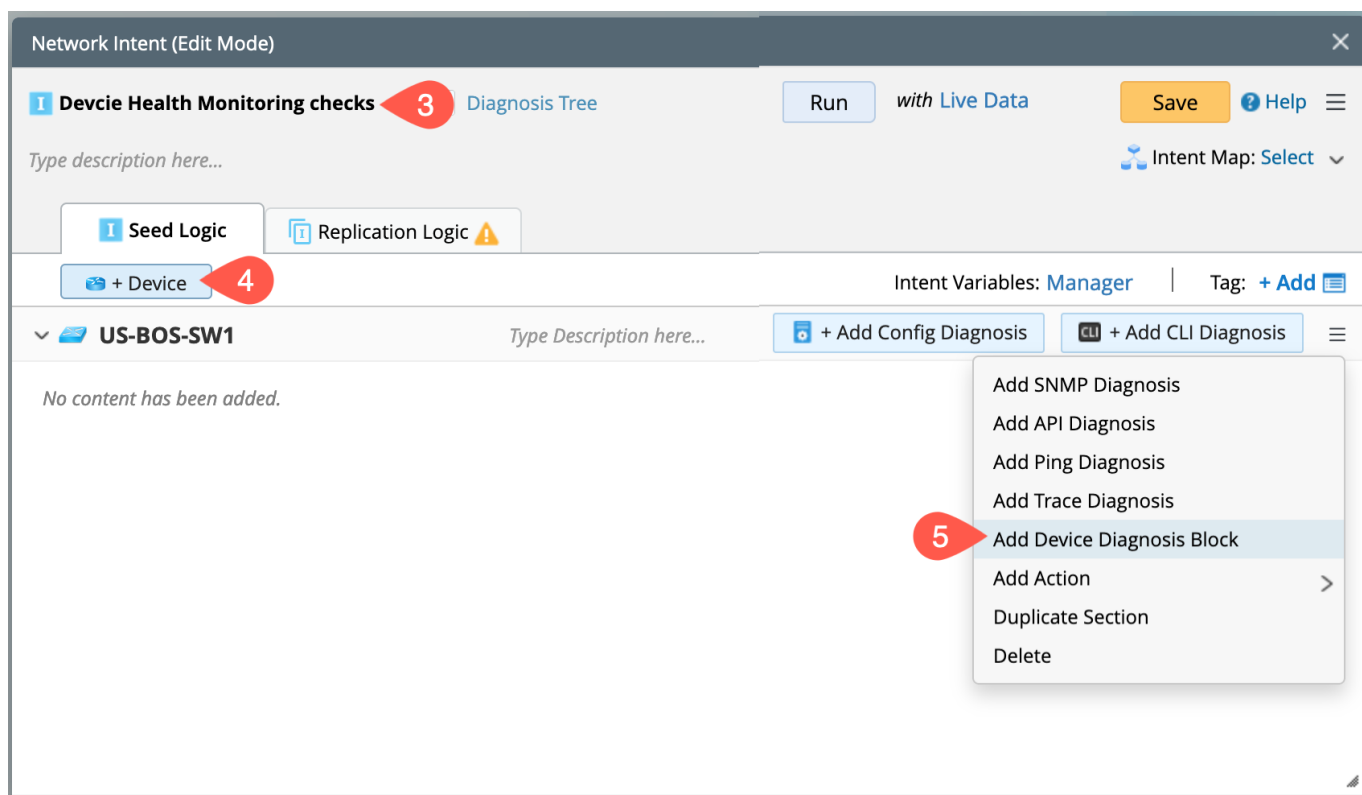
1. [Create a Wrapper Intent](#)
2. [Execute the wrapper intent](#)

4.2.1 Create Wrapper Intent

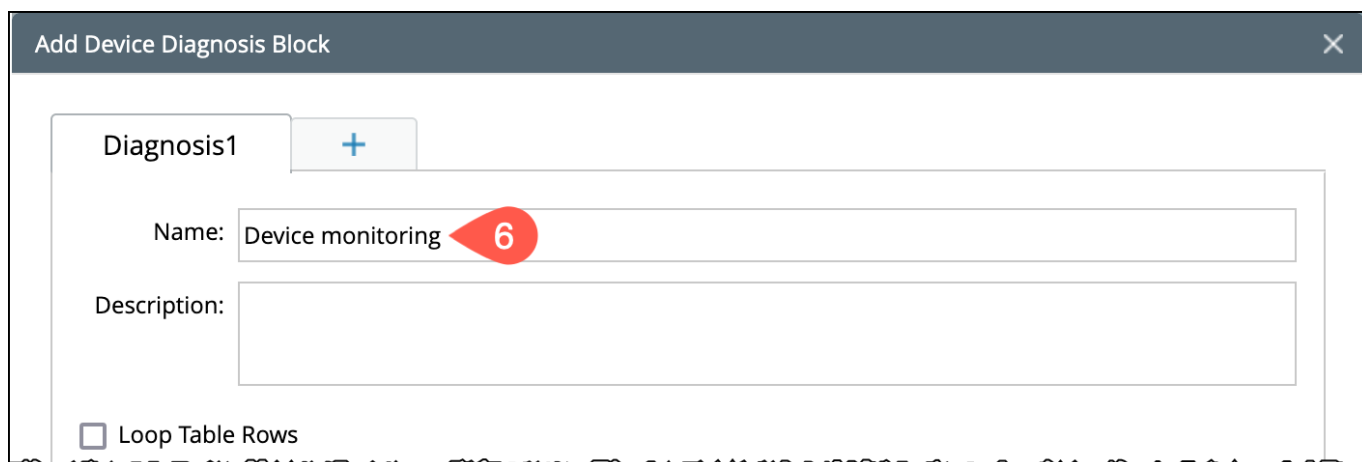
1. From the NetBrain Desktop, open the + menu and click **New Intent**.
2. In the Network Intent (Edit Mode) window, rename the Intent by clicking on the blue pen icon beside the "Type intent title here..." field.



3. Update the intent name to **Device Health Monitoring checks**.
4. Click **+Device** to add a device to the Intent,e.g., **US-BOS-SW1**.
5. Click  located beside the add diagnosis buttons, then click **Add Device Diagnosis Block**.



6. Rename Diagnosis 1 to **Device monitoring**.



7. In the **If** condition, select **True** from the Select Variable dropdown.

NOTE: Using **True** as the variable selection tells the NetBrain to always call the follow-up action without performing any diagnosis. As we always want to call the follow-up action, the True option is correct.

8. Remove the **Diagnosis Message** by clicking on the menu in the top right of it's section, then click **Delete**.

Add Device Diagnosis Block

Diagnosis1 +

Name: Device monitoring 6

Description:

☐ Loop Table Rows

▼ If

A US-BOS-SW1

True 7

B Select Variable

▼ Then

Diagnosis Message: ☐ Save to Incident

\$intf is down...

Pop up

Delete 8

Set Status Code for Device:

Set Status Code for Intent:

Add Logic ▼

+ Add Elself + Add Else

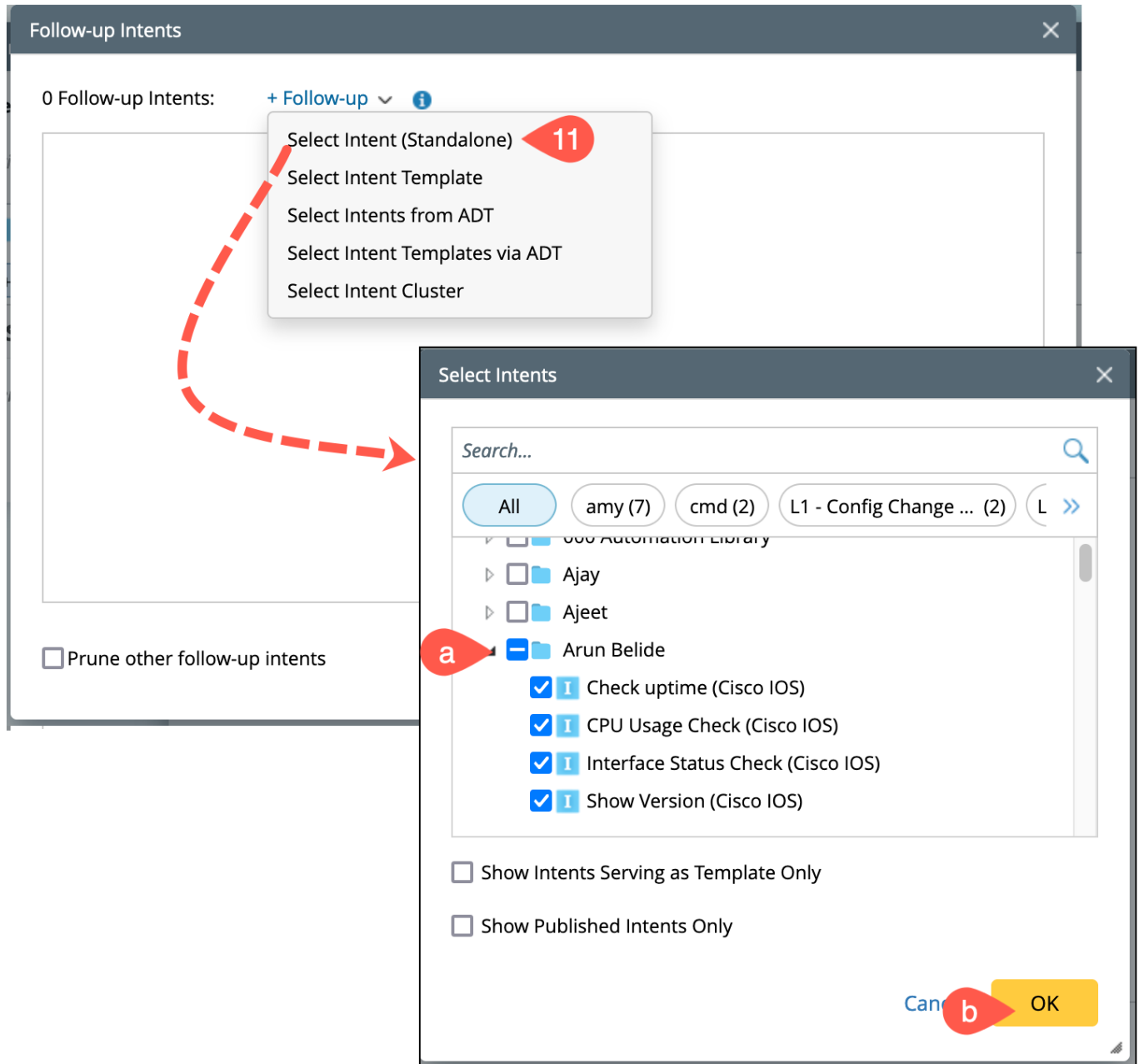
Cancel OK

9. Add a follow-up action by selecting **Follow-up Intent** from the **Add logic** dropdown.
10. After the Follow-up Intent option is added, click the **Network Intent** link to select the Network Intent in a new window, **Follow-up Intents**.

The screenshot shows the 'Add Device Diagnosis Block' dialog box. The 'Diagnosis1' tab is active, showing a 'Name' field with 'Device monitoring' and an empty 'Description' field. Below these fields is a checkbox for 'Loop Table Rows'. A dropdown menu is open, showing options: 'Create Diagnosis Message', 'Intent Data View', 'Draw Map', 'Send Email', 'Follow-up Intent' (highlighted with a red dashed arrow), and 'Advanced'. The 'Add Logic' button is also visible, with a red circle and the number '9' next to it. A red arrow points from the 'Follow-up Intent' option to a secondary dialog box titled 'Follow-up Intents'. This secondary dialog box has a 'Loop Table Rows' checkbox and an 'If' section with two conditions: 'A US-BOS-SW1' (set to 'True') and 'B Select Variable'. Below the 'If' section is a 'Then' section with a 'Follow-up Intent' dropdown set to 'Network Intent' (highlighted with a red circle and the number '10'), and options for 'Current Intent (Self)' and 'Stop'. The 'Add Logic' button is also present at the bottom of this dialog. The main dialog has '+ Add Elself' and '+ Add Else' buttons at the bottom left, and 'Cancel' and 'OK' buttons at the bottom right.

11. Click **Select Intent (Standalone)**.

- a) In the Select Intents dialog, navigate to the intents you have created in Chapter 3 and checkin all.
- b) Click **OK**.



- c) Ensure the correct intents are added here in Follow-up Intents.
- d) Click **Save**.

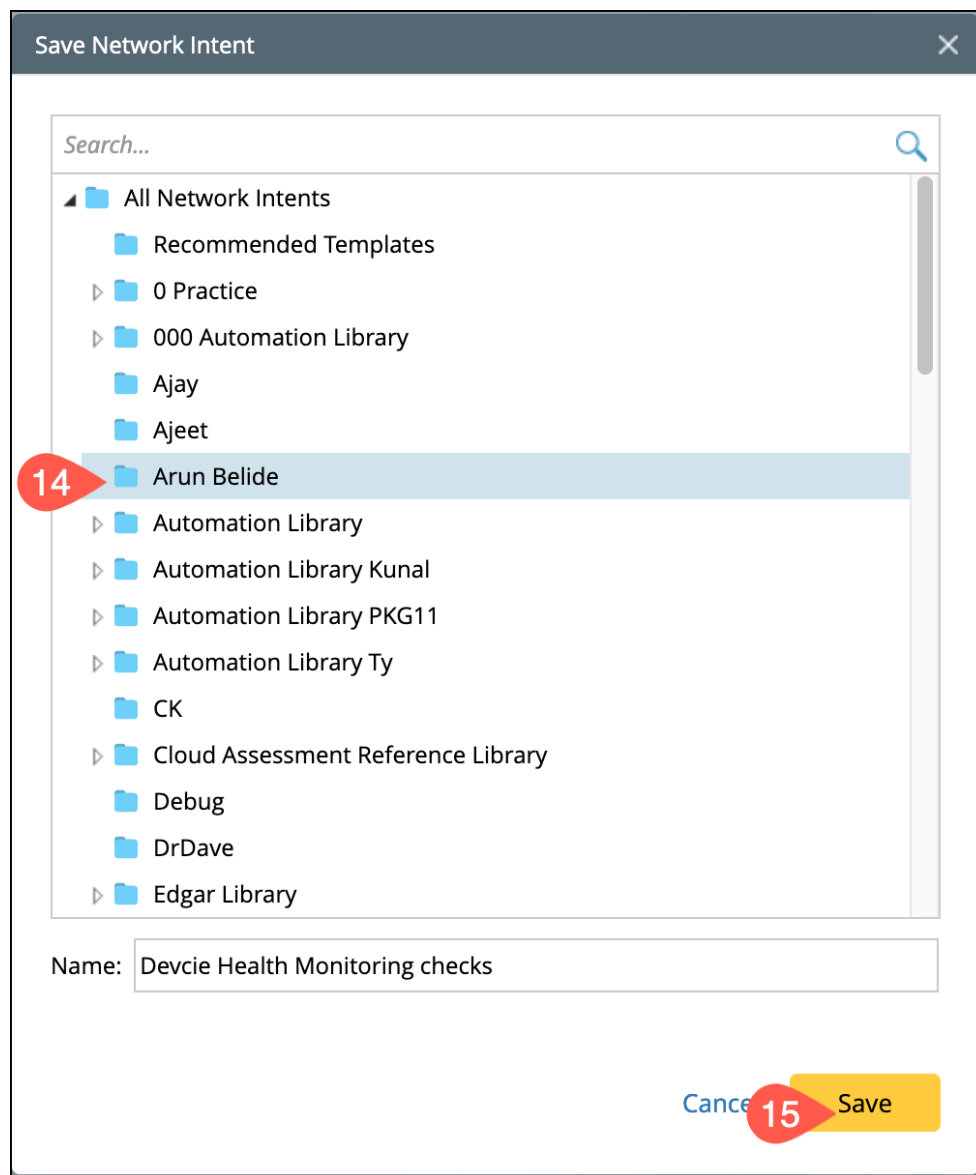
12. Click **OK** in the Add Device Diagnosis Block to save all the data and close the window.

12

13. Back in the Network Intent (Edit Mode) dialog, click **Save**.

13

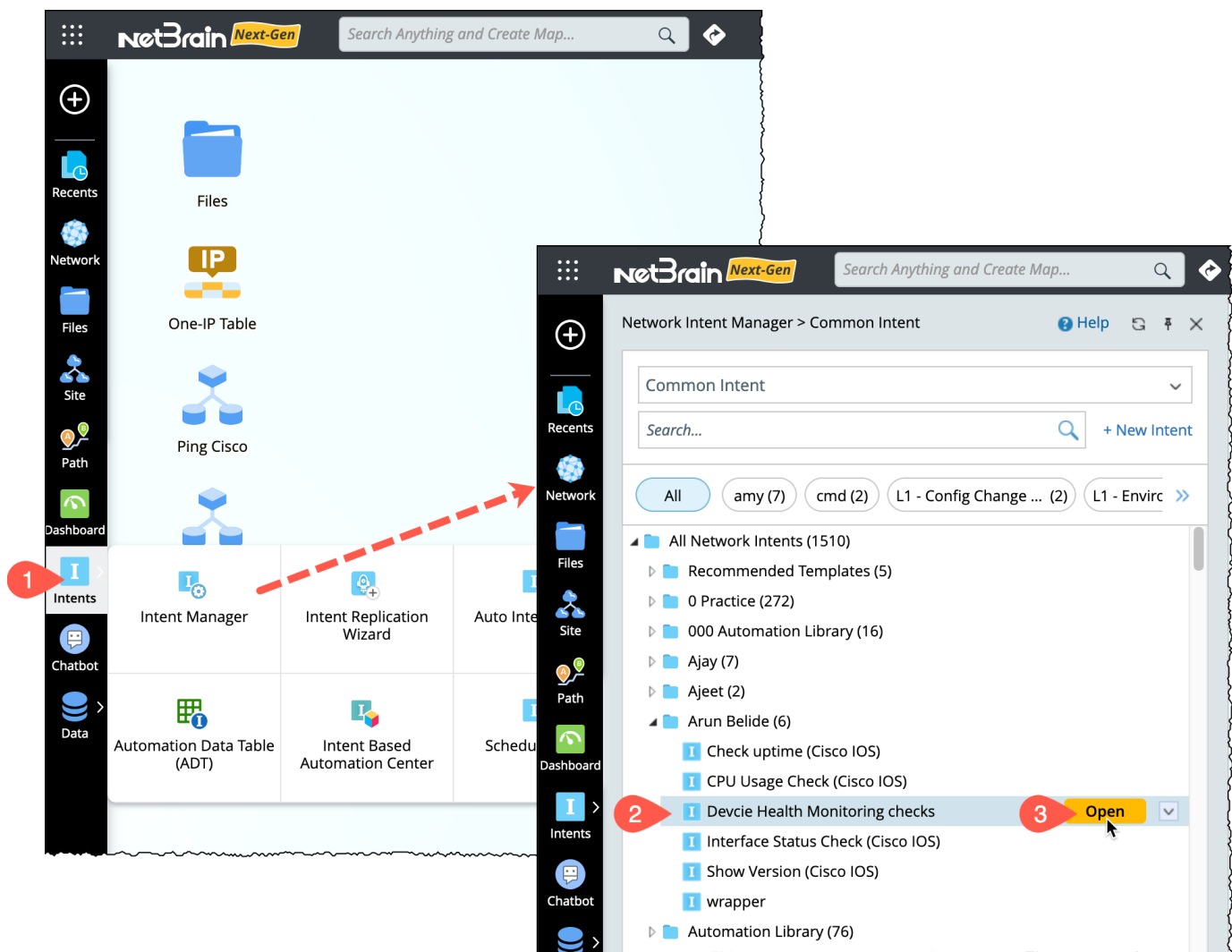
14. Browse to the **All Network Intents > [Your folder location]**.
15. Click **Save** to save and close the window.



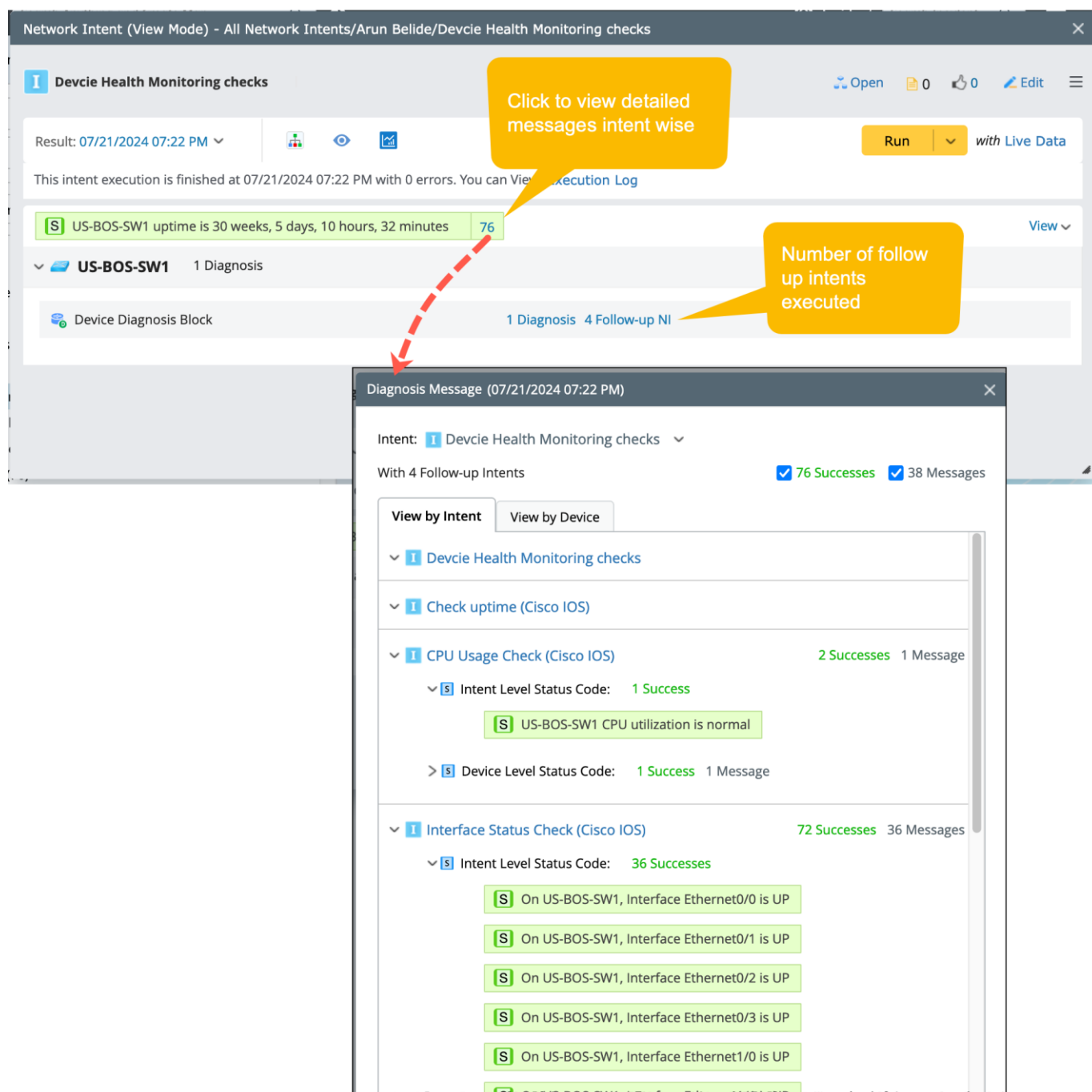
4.2.2 Execute Wrapper Intent

Let us validate and execute the wrapper intent created in the previous section.

1. In the sidebar of the End User Desktop, click **Intents** > **Intent Manager**.
2. In the Intent Manager, navigate to the wrapper intent **Device Health Monitoring checks** created in the previous section.
3. Click **Open** to view the intent in view mode.



4. Click **Run** to execute all the configured follow up intents.
5. Results will be displayed with the status, message and number of executed follow up intents.
6. Click the number in the diagnosis message to view the messages by intent.



Network Intent (View Mode) - All Network Intents/Arun Belide/Devcie Health Monitoring checks

I Devcie Health Monitoring checks

Result: 07/21/2024 07:22 PM

This intent execution is finished at 07/21/2024 07:22 PM with 0 errors. You can View Execution Log

S US-BOS-SW1 uptime is 30 weeks, 5 days, 10 hours, 32 minutes **76**

US-BOS-SW1 1 Diagnosis

Device Diagnosis Block

1 Diagnosis 4 Follow-up NI

Click to view detailed messages intent wise

Number of follow up intents executed

Diagnosis Message (07/21/2024 07:22 PM)

Intent: **I Devcie Health Monitoring checks**

With 4 Follow-up Intents **76 Successes** **38 Messages**

View by Intent **View by Device**

I Devcie Health Monitoring checks

I Check uptime (Cisco IOS)

I CPU Usage Check (Cisco IOS) **2 Successes** **1 Message**

S Intent Level Status Code: **1 Success**

S US-BOS-SW1 CPU utilization is normal

S Device Level Status Code: **1 Success** **1 Message**

I Interface Status Check (Cisco IOS) **72 Successes** **36 Messages**

S Intent Level Status Code: **36 Successes**


S On US-BOS-SW1, Interface Ethernet0/0 is UP

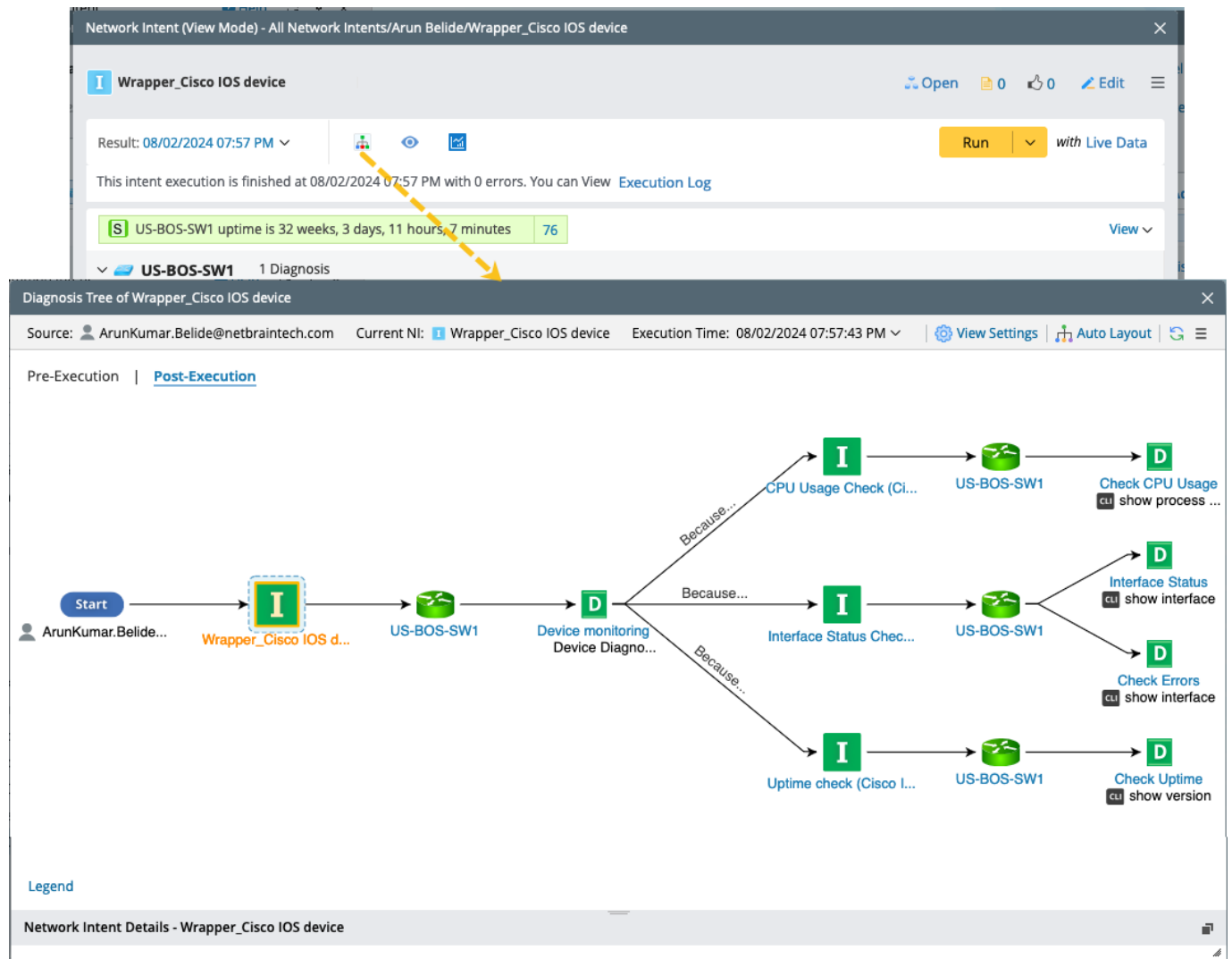
S On US-BOS-SW1, Interface Ethernet0/1 is UP

S On US-BOS-SW1, Interface Ethernet0/2 is UP

S On US-BOS-SW1, Interface Ethernet0/3 is UP

S On US-BOS-SW1, Interface Ethernet1/0 is UP

7. View the results in the diagnosis tree to check whether all the follow up intents are executed. Click the diagnosis tree button  in the results window to open the corresponding view:

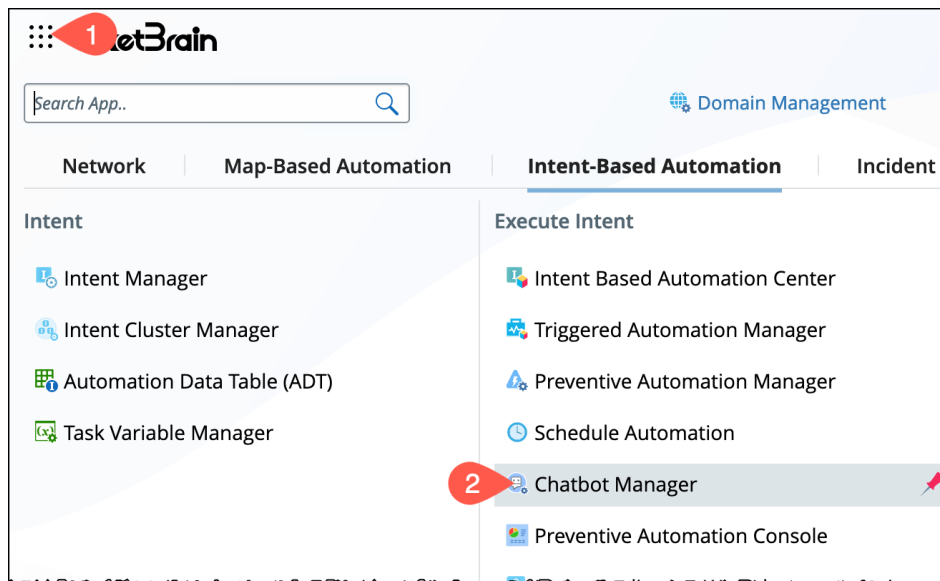


4.3 Run intent with Chatbot

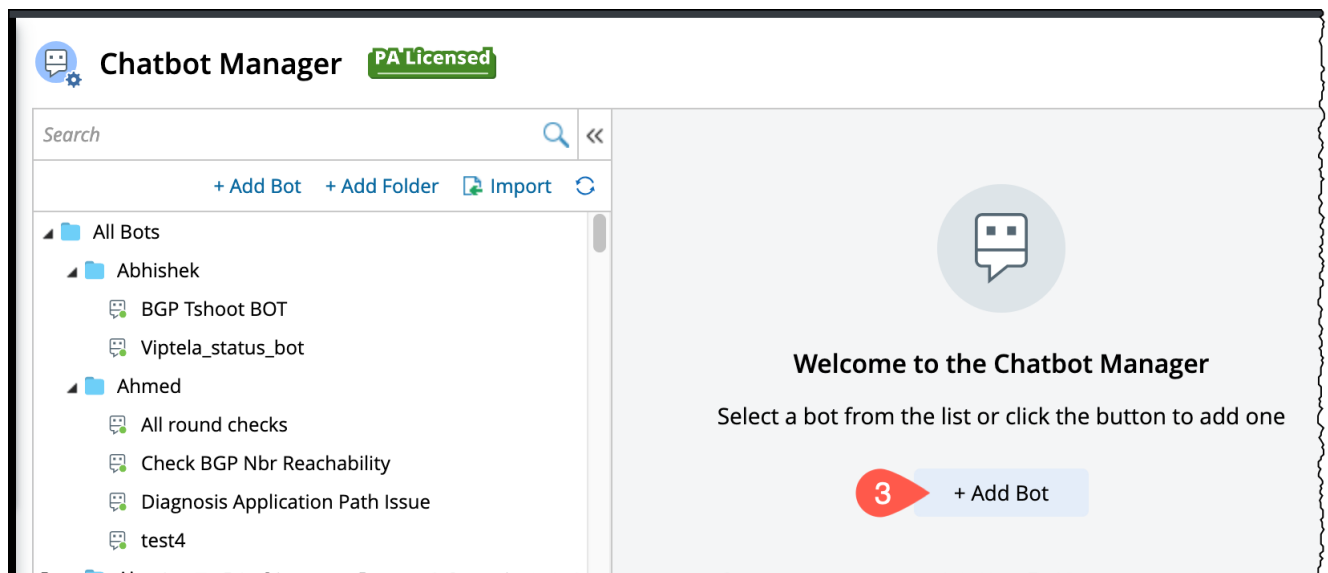
Chatbot allows power users to build interactive chatbots, so end users can execute intent-based automation to solve real-world challenges without accessing NetBrain system UI. Building a chatbot flow is straightforward.

4.3.1 Create the Netbrain Chatbot

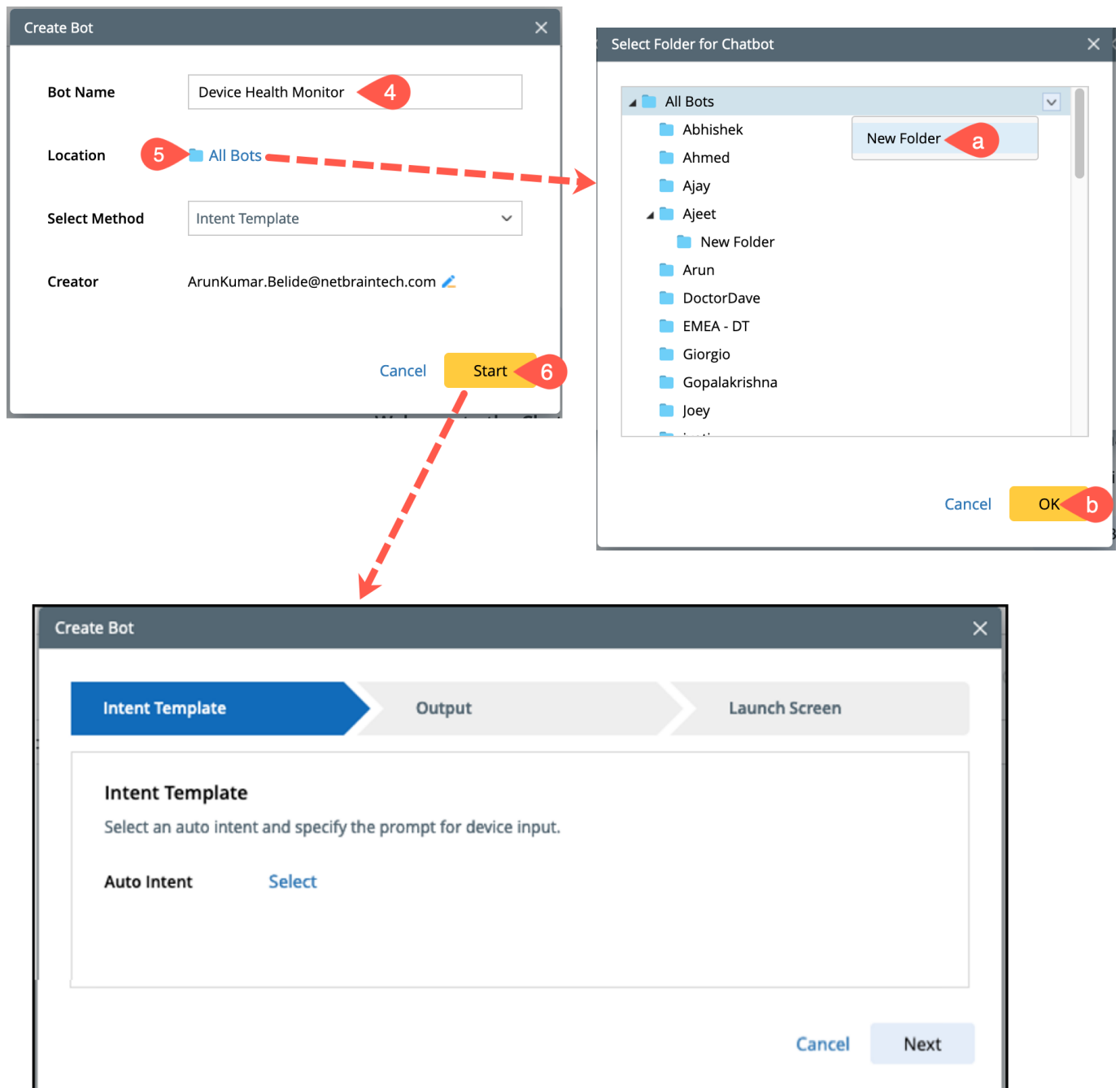
1. In the Upper-Left corner of the desktop, click ☰
2. Select **Chatbot Manager** under the **Intent-Based Automation** tab.



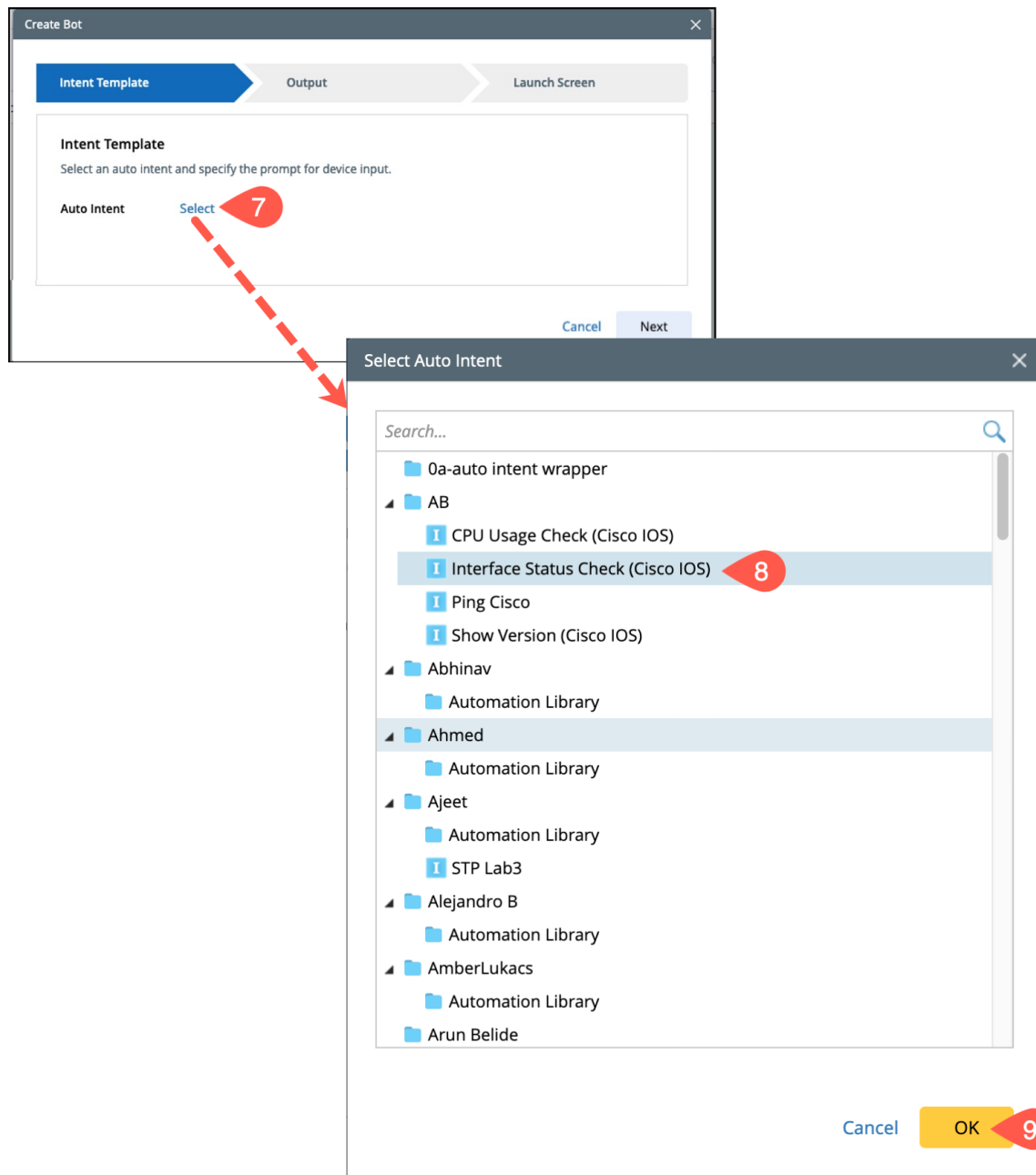
3. In the **Chatbot Manager**, click **+ Add Bot**.



4. Name the bot **Device Health Monitor**.
5. Location: To choose a location for the bot, click **All Bot**.
 - a) In the dialog, **Select Folder for Chatbot**, select All Bots or create a folder under it using the option **New Folder** from the dropdown menu.
 - b) Click **OK**.
6. Click **Start** to begin the **Chatbot creation wizard**.



7. In the first ribbon of the **Create Bot Wizard (Intent Template)**, click **Select** located next to **Auto Intent**.
8. From the **Select Auto intent** dialog, choose the intent **Interface Status Check**.
9. Click **OK** to select and close the window.



10. Modify the **Input Prompt** as required or retain the default prompt.
11. Check the checkbox, **Allow multiple selection**, to choose multiple device selection.

NOTE: This ensures to run intent multiple times (once per unique device).

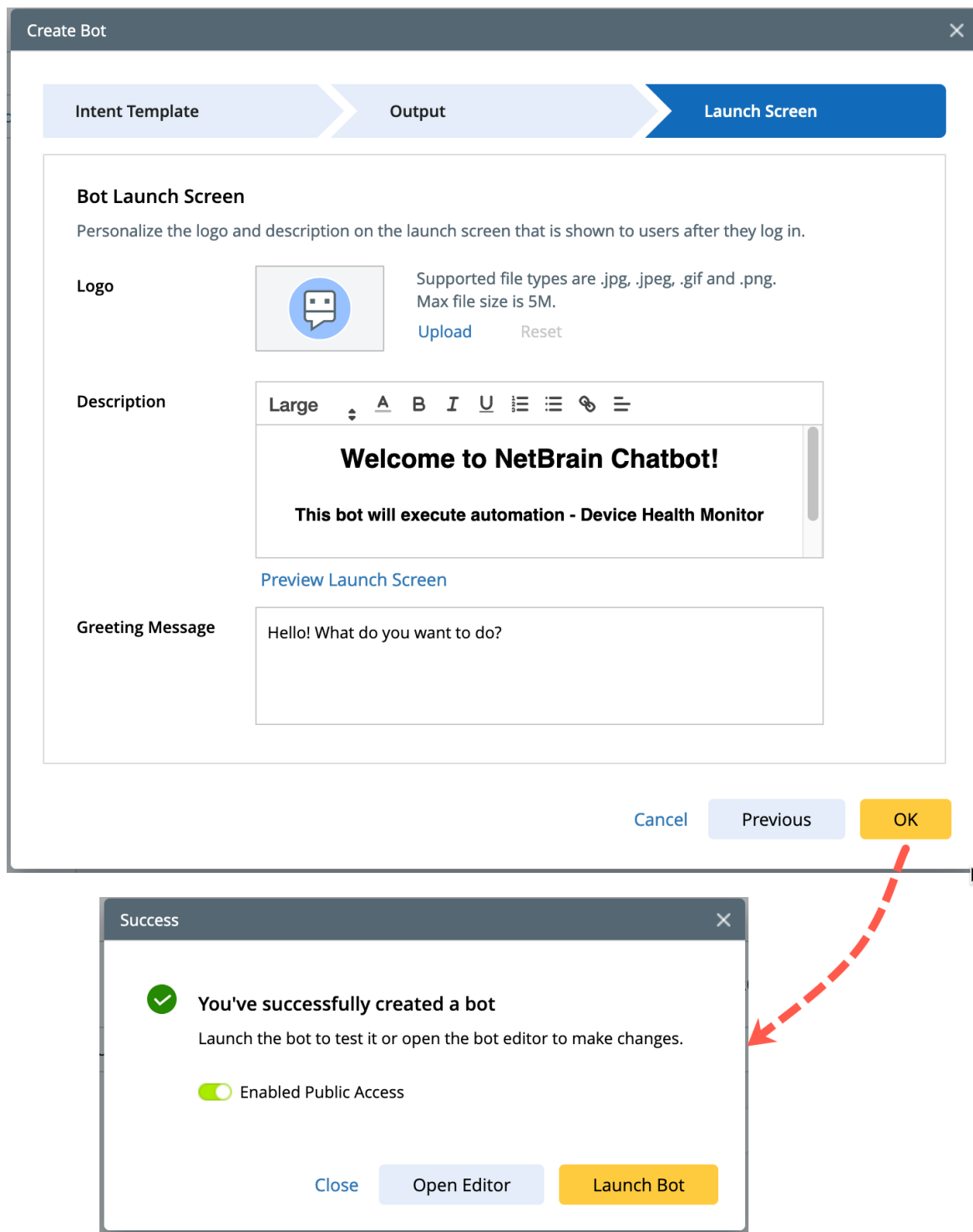
12. Click **Next** to go to the next ribbon Output.

The screenshot shows the 'Create Bot' dialog with the 'Intent Template' ribbon selected. The 'Auto Intent' is set to '/AB/Interface Status Check (Cisco IOS)'. The 'Input Prompt' is 'Please select a device:' with a red callout '10' pointing to the text and '11' pointing to the selection area. There is a checkbox for 'Allow multiple selection' which is checked. At the bottom right, there are 'Cancel' and 'Next' buttons, with a red callout '12' pointing to the 'Next' button.

13. In the **Output** ribbon, review the potential output (or click the Pencil icon to edit it), then click **Next**.

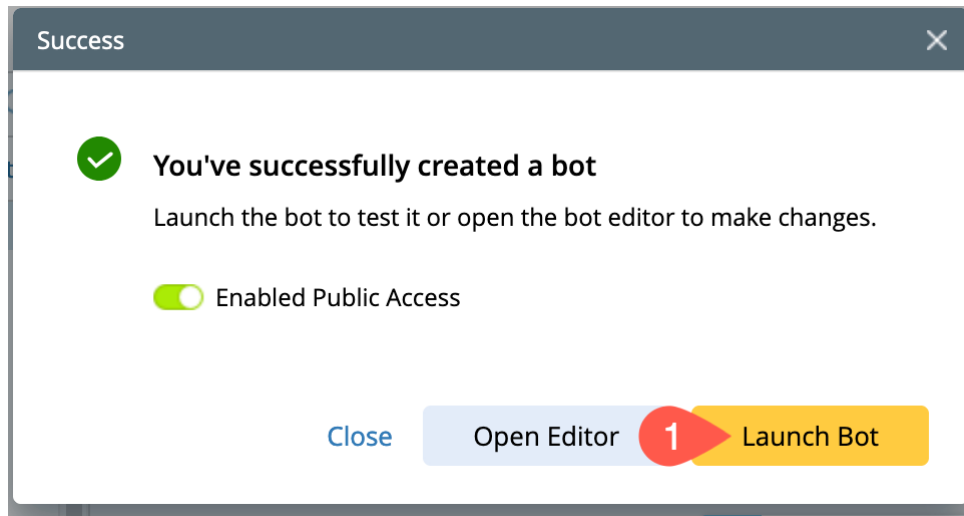
The screenshot shows the 'Create Bot' dialog with the 'Output' ribbon selected. The 'Automation Output' section displays a sample output message: '"\$intent_name" executed on device \$all_devices. \$count_of_ni_error_messages errors found: \$ni_all_status_code_message'. Below this, there are links for 'View Map: \$intent_output_map' and 'View Diagnosis Tree: \$diagnosis_tree', and a '+ Add' link. A yellow callout bubble points to a pencil icon with the text 'click the Pencil icon to modify output'. At the bottom right, there are 'Cancel', 'Previous', and 'Next' buttons, with a red callout '13' pointing to the 'Next' button.

14. In the **Launch Screen**, review the default details (make any desired edits).
15. Click **OK** to complete the configuration of the chatbot.
16. A success prompt appears on the screen upon bot creation.



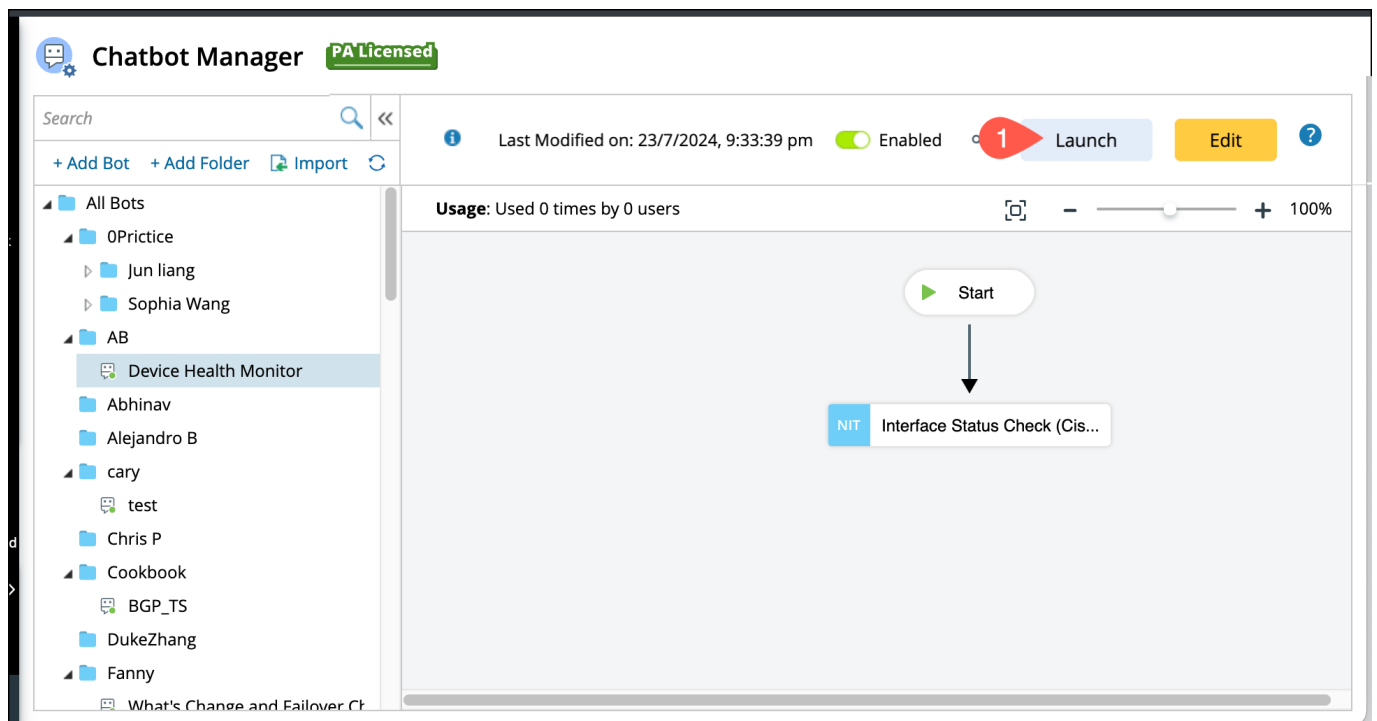
4.3.2 Use Chatbot

1. Click **Launch Bot** from the **Success** prompt window of the chatbot creation.

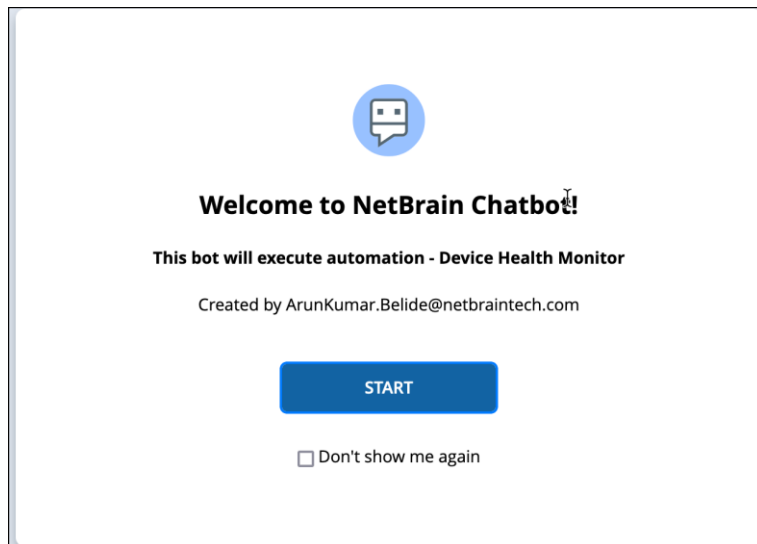


(or)

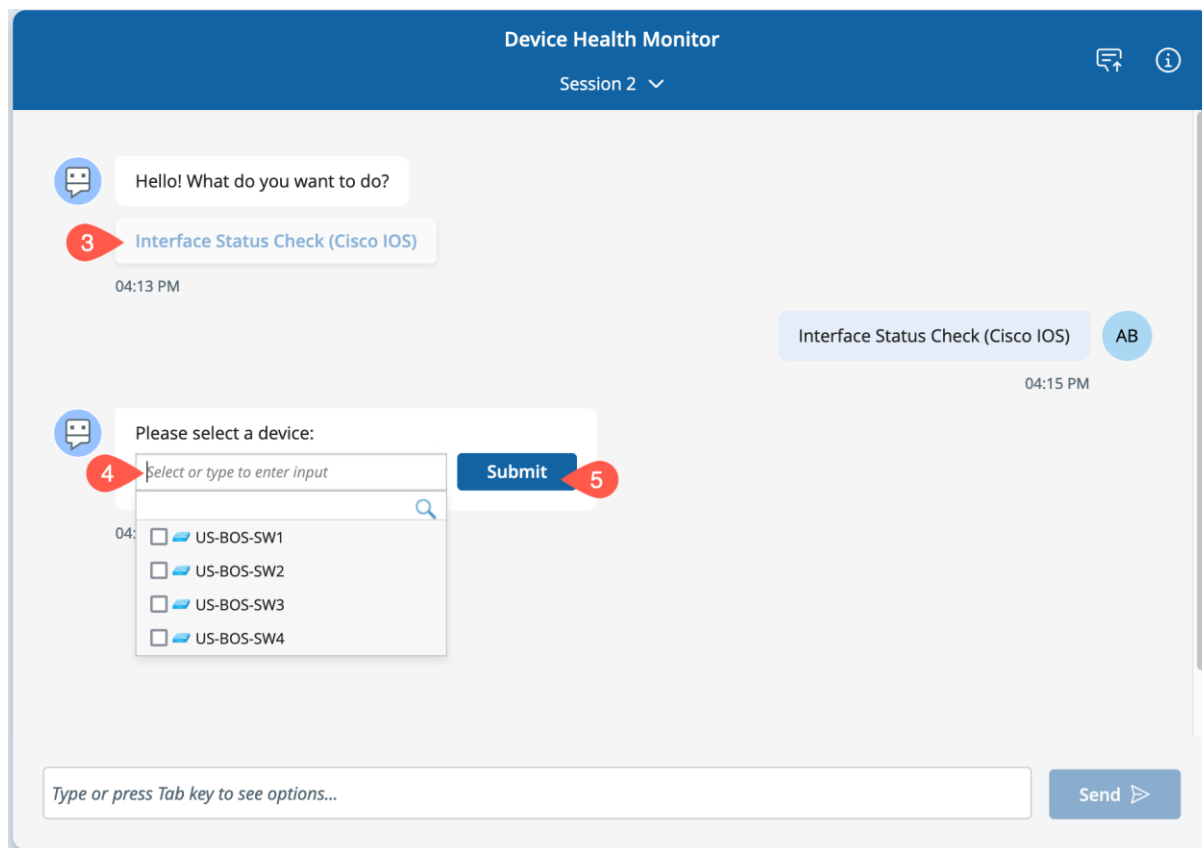
In the chatbot Manager, click the **Launch** button located on the top right corner of the window.



2. In the newly opened browser tab, click **START** to launch the Chatbot.



3. Click the Intent **Interface Status Check (Cisco IOS)**.
4. Under **Please Select a Device**, click **Select or type to enter input** field and select the number of devices from the drop-down as required. E.g., select two devices, **US-BOS-SW1** and **US-BOS-SW2**.
5. Click **Submit** to execute the intent.



- The chatbot will execute the intent and provide the diagnosis output.

NOTE: Intent Diagnosis Message indicates the ethernet is UP.

NOTE: Map and Diagnosis tree links to view the error details if any.

The screenshot displays the NetBrain Device Health Monitor interface. At the top, the title bar reads "Device Health Monitor" with "Session 3" and a dropdown arrow. Below this, a chat window shows a sequence of messages:

- A system message: "Interface Status Check (Cisco IOS)" with a timestamp of "04:19 PM".
- A user message (AB): "Interface Status Check (Cisco IOS)" with a timestamp of "04:19 PM".
- A system message: "Please select a device:" with a dropdown menu showing "--Select--" and a "Submit" button. The timestamp is "04:19 PM".
- A user message (AB): "US-BOS-SW1" with a timestamp of "04:19 PM".
- A system message: **"Interface Status Check (Cisco IOS) US-BOS-SW1 1"** executed on device US-BOS-SW1. 0 errors found:
 - A list of 20 interface status checks, all marked as "UP":
 - On US-BOS-SW1, Interface Ethernet0/0 is UP
 - On US-BOS-SW1, Interface Ethernet0/1 is UP
 - On US-BOS-SW1, Interface Ethernet0/2 is UP
 - On US-BOS-SW1, Interface Ethernet0/3 is UP
 - On US-BOS-SW1, Interface Ethernet1/0 is UP
 - On US-BOS-SW1, Interface Ethernet1/1 is UP
 - On US-BOS-SW1, Interface Ethernet1/2 is UP
 - On US-BOS-SW1, Interface Ethernet1/3 is UP
 - On US-BOS-SW1, Interface Ethernet2/0 is UP
 - On US-BOS-SW1, Interface Ethernet2/1 is UP
 - On US-BOS-SW1, Interface Ethernet2/2 is UP
 - On US-BOS-SW1, Interface Ethernet2/3 is UP
 - On US-BOS-SW1, Interface Loopback0 is UP
 - On US-BOS-SW1, Interface Loopback1 is UP
 - On US-BOS-SW1, Interface Loopback2 is UP
 - On US-BOS-SW1, Interface Port-channel35 is UP
 - On US-BOS-SW1, Interface Vlan100 is UP
 - On US-BOS-SW1, Interface Vlan101 is UP
 - A summary line: "Errors on this device US-BOS-SW1 have not increased"
- A system message: "View Map: [Intent Output Map](#)
View Diagnosis Tree: [Diagnosis Tree](#)" with a timestamp of "04:19 PM".
- A final system message: "This is the last step. You can type 'restart' to start over." with a timestamp of "04:19 PM".

At the bottom, there is a text input field with the placeholder "Type '/' for shortcut commands" and a "Send" button with a right-pointing arrow.

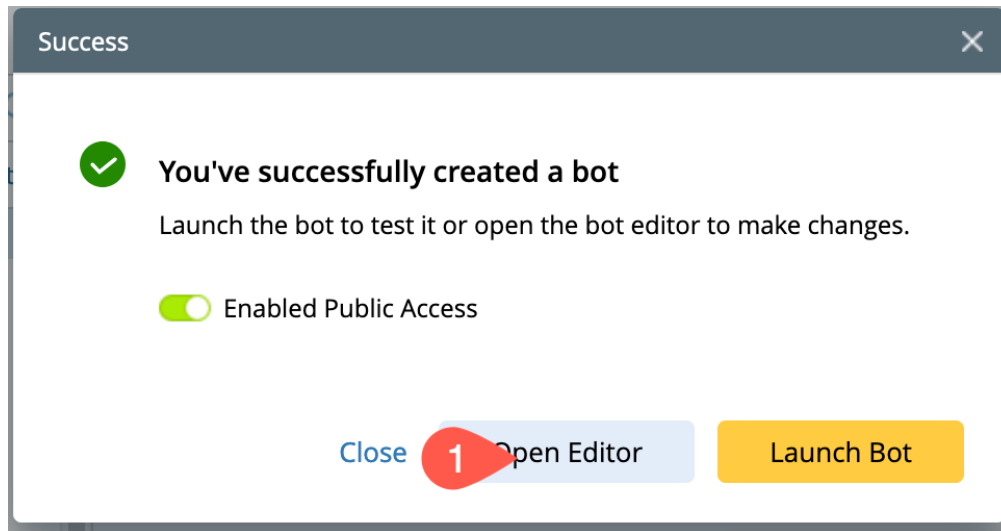
Two yellow callout boxes highlight specific features:

- A callout box points to the list of interface status checks, stating: "Ethernet status is UP".
- A callout box points to the "View Map" and "View Diagnosis Tree" links, stating: "Map and Diagnosis tree links".

- Close all browser tabs except for the **NetBrain desktop**.

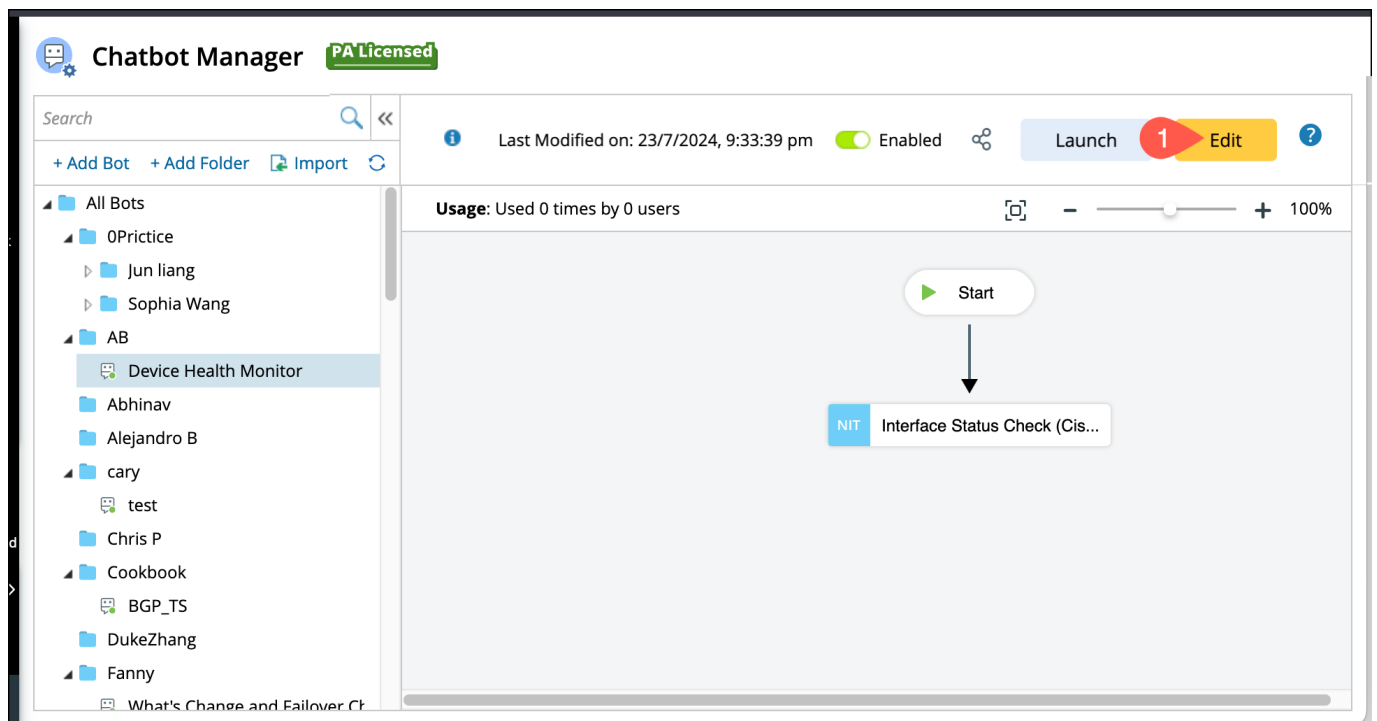
4.3.3 Add Auto Intent to Chatbot

1. Click **Launch Bot** from the **Success** prompt window of the chatbot creation.

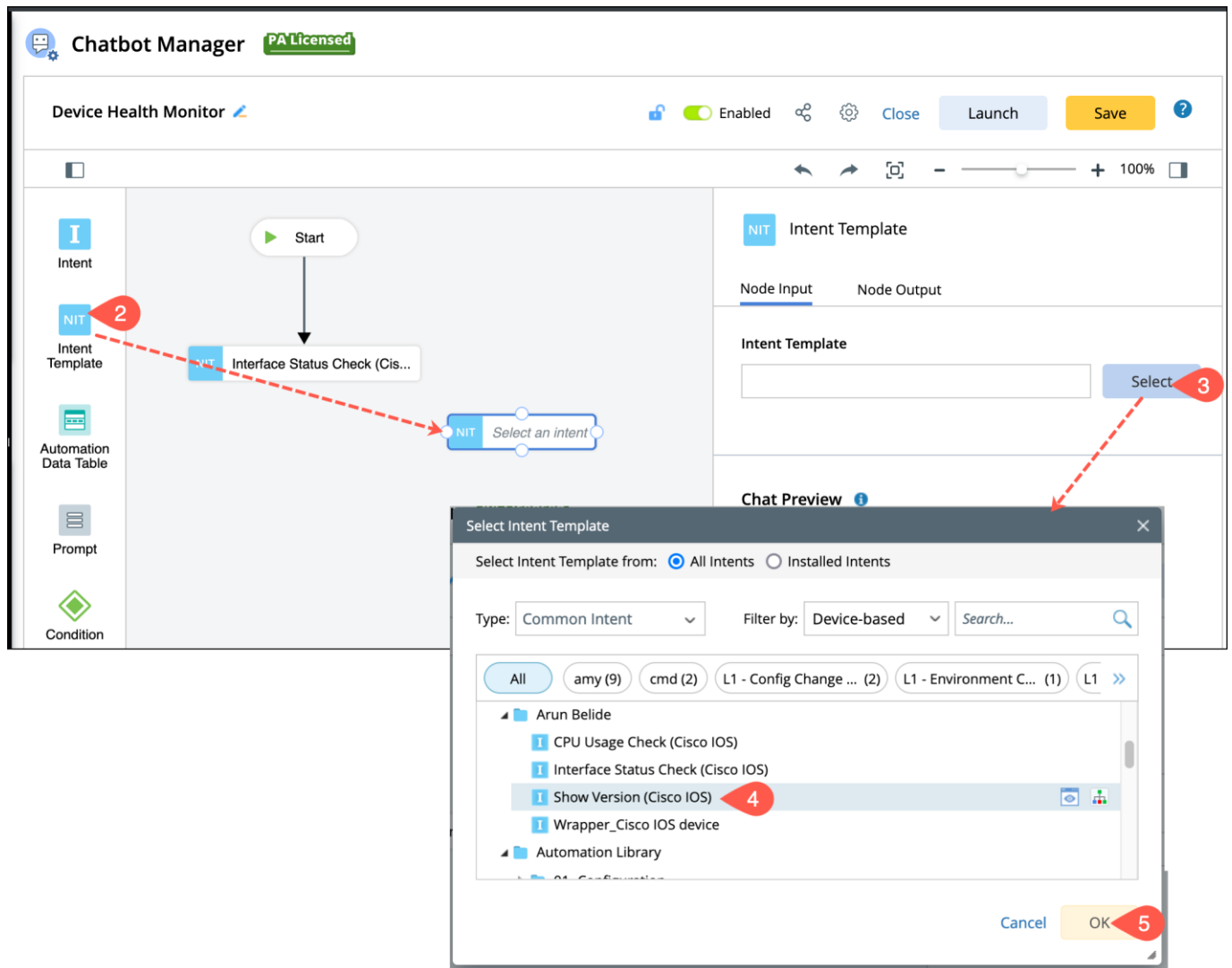


(or)

Click **Edit** located on the top right corner of the chatbot manager window.



2. In the **Chatbot Manager (Visual Editor)**, click and drag the element **Intent Template** into the editor.
3. In the right pane, click **Select** in the Intent Template editor to open the **Select Intent Template** dialog.
4. In the **Select Intent Template**, navigate to your required intent, e.g., **Show Version**.
5. Click **OK**.

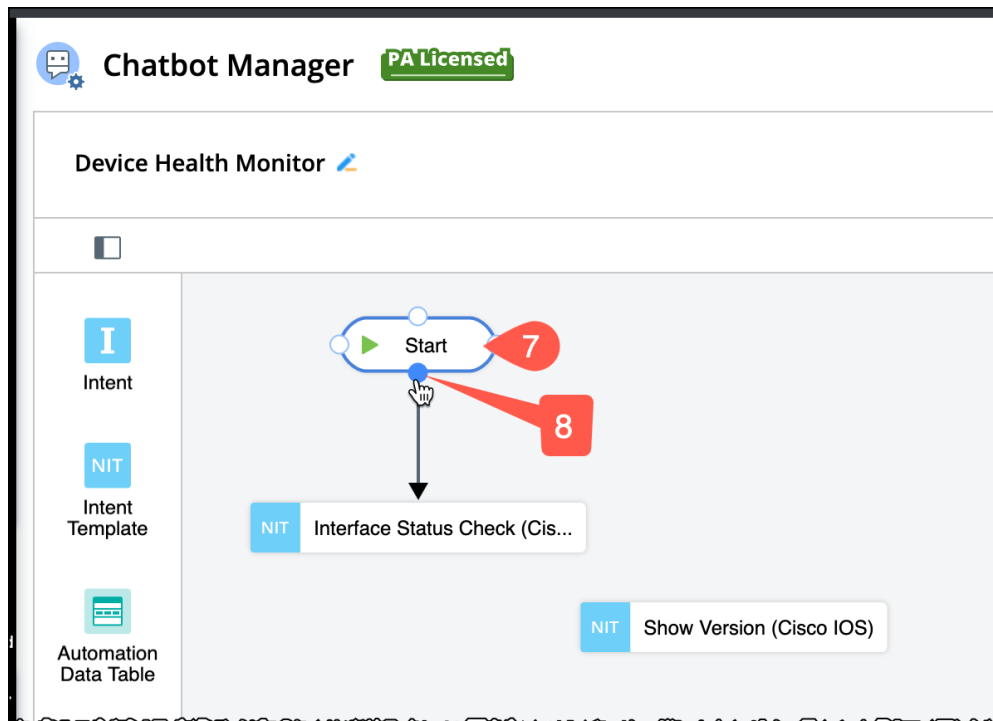


6. Under **Replicated Device Candidates**, click "--- Select ---" and:

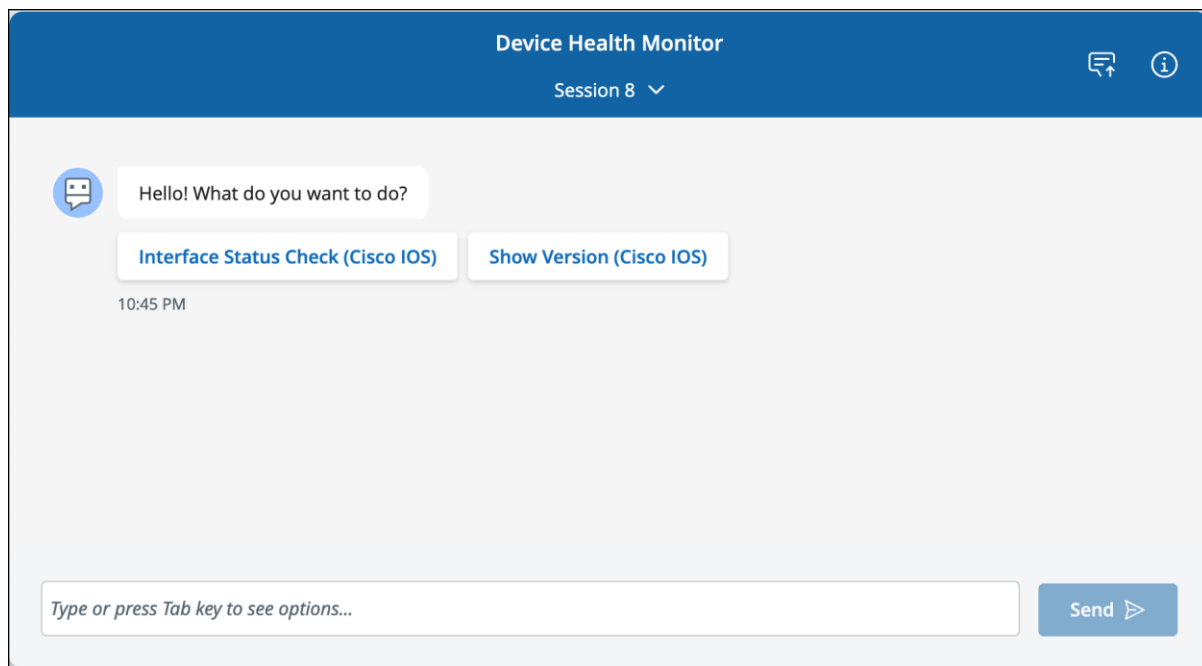
- a) User Input Mode: Choose **Multiple Selection** from the drop down menu.
- b) **Input Prompt**: Keep the default (**Please provide input**). Or modify it to give a hint for the user to provide input.
- c) **Candidate Value Source**: These values will appear as a drop-down menu for input selection, choose the option **Manual Input** and add the list of devices.

- d) Click **OK**.
7. Click the **Start** element.

8. Hover the mouse on the element, click the blue circle and click-drag the resulting arrow connecting it to **Show Version (Cisco IOS)**.



9. Click **Save**, then **Launch**.
10. In the newly opened browser tab, click **START** to launch the Chatbot. The final chatbot with two intents will be as follows:



5 Network Assessment and Document: Essential

In the last three chapters, you learn how to create an intent and replicate an intent to auto intents, which can be used by the end users for troubleshooting. In this chapter, you will learn how to use the intents for the network assessment across the whole network.

Network Assessments have always been a ‘necessary evil’ to satisfy the need to understand the network and its vulnerabilities and bottlenecks. However, this meant costly projects that spanned months, only to get a rudimentary understanding of your infrastructure as it existed at a certain point in time. The problem is that they quickly become out of date with today’s constantly changing networks.

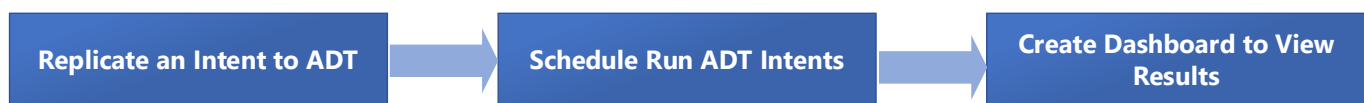
NetBrain intent-based no-code automation technology provides an ideal platform for Network Assessment, transferring it from a static document to a dynamic continuous health check. In this chapter, you will learn how to apply an intent you have created in the last three chapters to the network assessment across the whole network. You will learn an important concept, **Automation Data Table (ADT)**, a global table to manage your network assets, such as core routers and critical applications, and automations associated with these objects. ADT and NI are essential for the network assessment. Some important use cases of the network assessments will be covered in the next three chapters:

- Security assessment
- Configuration drift
- Failover assessment

In this chapter, we will use the following examples to illustrate the essential concepts and workflows of the network assessment:

- Create an ADT for your network devices and the performance data, such as CPU.
- Create a report for all down interfaces within your network. This report can help your company assess the capacity of your network.

The workflow of the network assessment will be as follows:

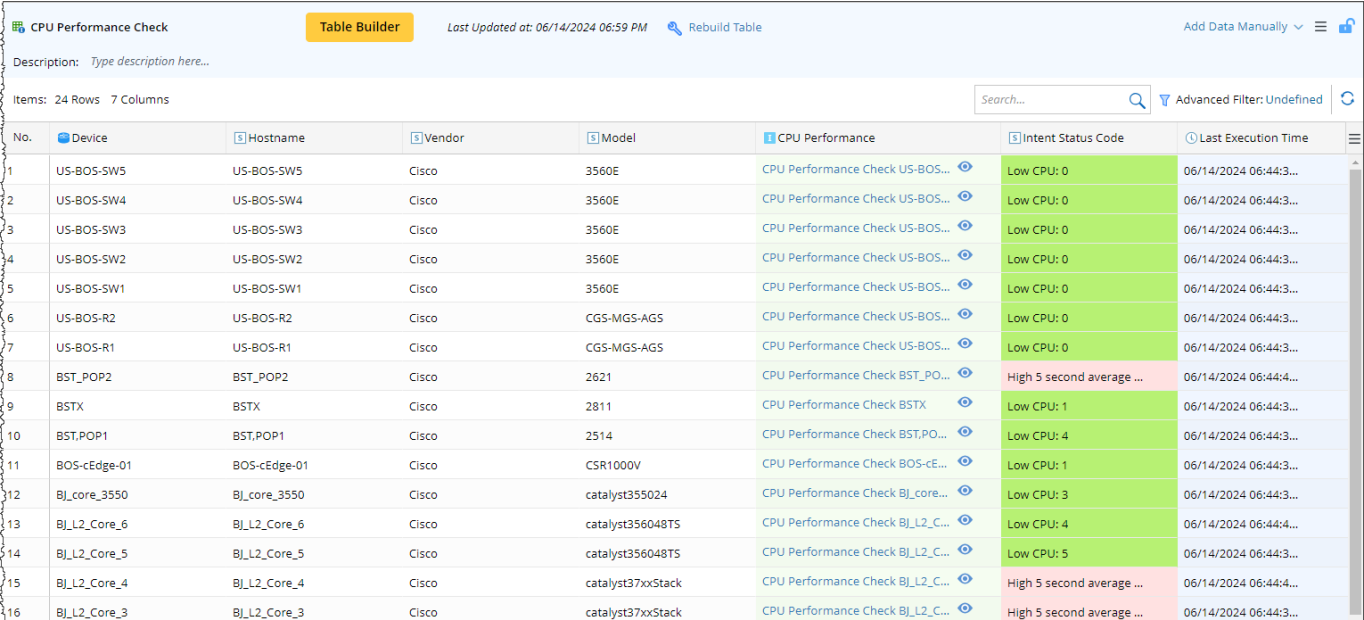


5.1 Document Your Network Inventory and Performance

You will start your first network assessment with a simple example: monitoring the CPU usage of the network devices, which can offer real-time insights into the performance of your network devices. By proactively managing CPU utilization, you can detect potential issues early and implement preventive measures. High CPU usage is often a symptom of performance bottlenecks.

In this section, you will learn how to diagnose high CPU usage within your network ecosystem, enabling you to address issues before they affect overall network performance.

You will create an ADT as follows, including the device's built-in properties, the replicated intents, and the intent status code:



The screenshot displays the NetBrain Table Builder interface. At the top, it shows the title 'CPU Performance Check', a 'Table Builder' button, and a timestamp 'Last Updated at: 06/14/2024 06:59 PM'. Below this is a search bar and an 'Advanced Filter: Undefined' button. The table itself has 8 columns: No., Device, Hostname, Vendor, Model, CPU Performance, Intent Status Code, and Last Execution Time. It contains 16 rows of data, each representing a different network device and its associated CPU performance check. The 'Intent Status Code' column uses color-coding: green for 'Low CPU: 0', yellow for 'Low CPU: 1', and red for 'High 5 second average ...'.

No.	Device	Hostname	Vendor	Model	CPU Performance	Intent Status Code	Last Execution Time
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
6	US-BOS-R2	US-BOS-R2	Cisco	CG5-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
7	US-BOS-R1	US-BOS-R1	Cisco	CG5-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
8	BST_POP2	BST_POP2	Cisco	2621	CPU Performance Check BST_PO...	High 5 second average ...	06/14/2024 06:44:4...
9	BSTX	BSTX	Cisco	2811	CPU Performance Check BSTX	Low CPU: 1	06/14/2024 06:44:3...
10	BST.POP1	BST.POP1	Cisco	2514	CPU Performance Check BST,PO...	Low CPU: 4	06/14/2024 06:44:3...
11	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	CPU Performance Check BOS-cE...	Low CPU: 1	06/14/2024 06:44:3...
12	BJ_core_3550	BJ_core_3550	Cisco	catalyst355024	CPU Performance Check BJ_core...	Low CPU: 3	06/14/2024 06:44:3...
13	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_C...	Low CPU: 4	06/14/2024 06:44:4...
14	BJ_L2_Core_5	BJ_L2_Core_5	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_C...	Low CPU: 5	06/14/2024 06:44:3...
15	BJ_L2_Core_4	BJ_L2_Core_4	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...	High 5 second average ...	06/14/2024 06:44:4...
16	BJ_L2_Core_3	BJ_L2_Core_3	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...	High 5 second average ...	06/14/2024 06:44:3...

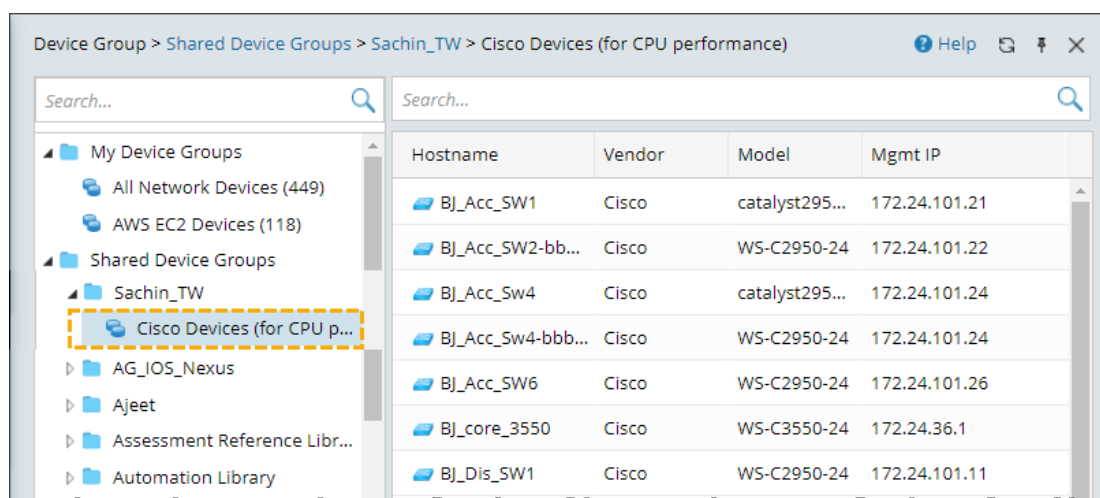
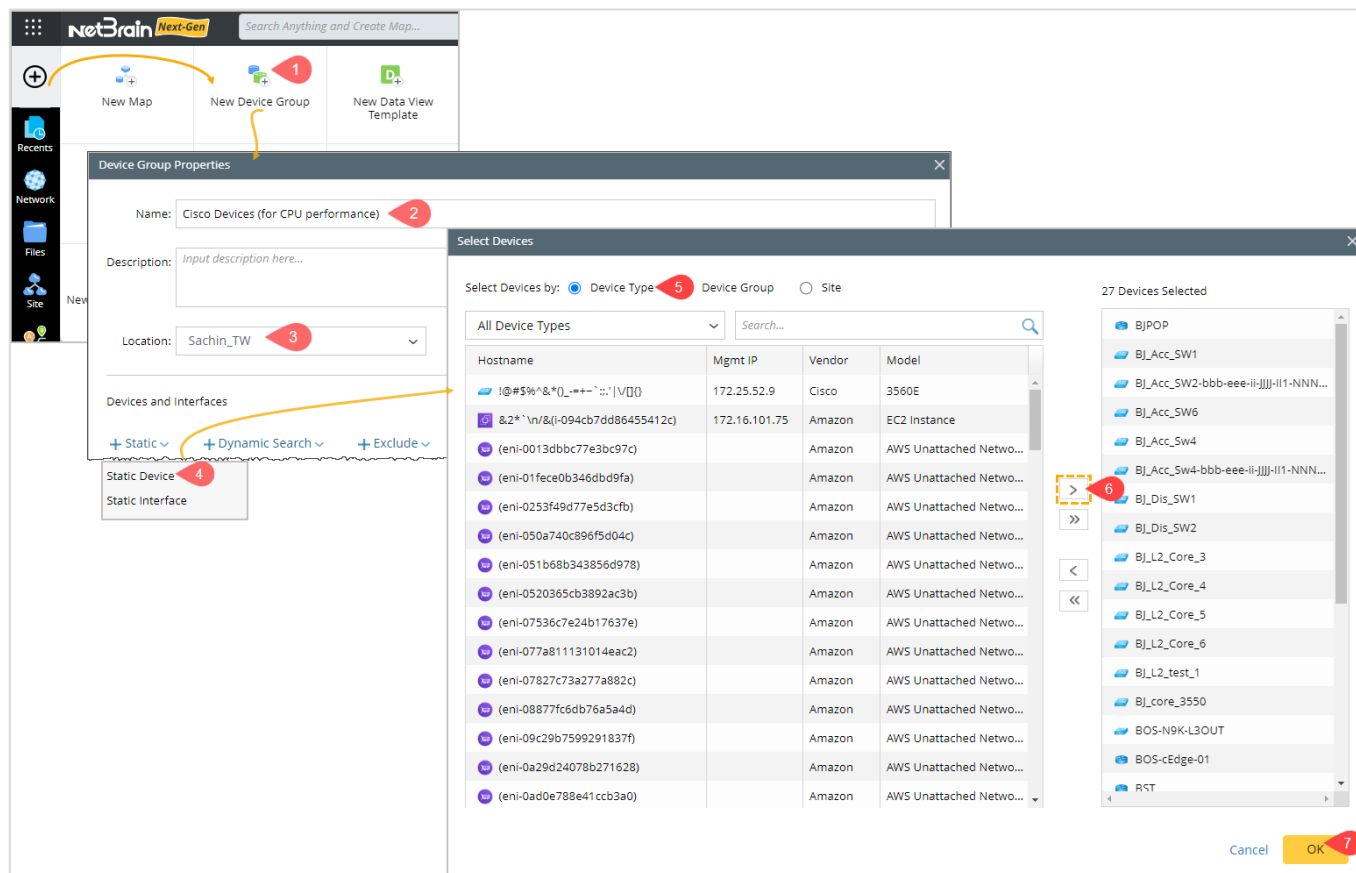
This section includes the following main steps:

- [Prerequisites](#)
- [Create Network Intent](#)
- [Use Intent Replication Wizard](#)
- [Run the Intent](#)
- [Export ADT to CSV file](#)
- [Create Intent Dashboard for ADT Automation Column](#)

5.1.1 Prerequisites

The first step is to create a device group to be assessed. This device group will be used during the **Intent Replication Wizard** and for creating the **ADT** for the base table using the **Devices of Device Group** method. If the device group already exists, you can skip these steps.

Here, you will create a device group for all the Cisco devices. You can create this device group by **Dynamic Search**: set the **Device Type** to be Cisco IOS switch or router.



5.1.2 Create Network Intent

You will reuse the network intent you created in Chapter 3. Let us recap the key steps to create an intent:

- Select a seed device and add a **CLI Diagnosis**:

Command	<i>show process cpu</i>
Line Pattern	Auto Pattern - Single
Var Line 1	Five seconds: <i>\$int:cpu_5seconds</i>

Network Intent (Edit Mode)

CPU Performance Check

Diagnosis Tree

Run

with Live Data

Save

Help

Intent Map: Select

Type description here...

Seed Logic

Replication Logic

+ Device

Intent Variables: Manager

Tag: + Add

US-BOS-R1

Type Description here...

+ Add Config Diagnosis

+ Add CLI Diagnosis

CLI Command Diagnosis

US-BOS-R1

show process cpu

Retrieve

with Live Data

1. Define Variable

2. Define Diagnosis

Format1

Test on Devices: 0

Double-click a variable to parse. Select multiple lines to parse a table.

Critical Variable (0)

Pattern1

Type: Single

+ New Pattern

Current Device

06/14/2024 03:09:14 PM

Search...

Var Line 1

1 US-BOS-R1#show process cpu

2 CPU utilization for five seconds: 0% minute: 0%

3 PID Runtime(ms) Invoked uSecs sSecs 1Min 5

4 1 4 106 37 0.00% 0.00% 0.

5 2 174348 1959534 88 0.00% 0.00% 0.

6 3 61176 9534704 6 0.00% 0.00% 0.

7 4 0 1 0 0.00% 0.00% 0.

8 5 0 1 0 0.00% 0.00% 0.

9 6 1339208 1324618 1011 0.00% 0.02% 0.

10 7 4804 163367 29 0.00% 0.00% 0.

11 8 0 1 0 0.00% 0.00% 0.

12 9 0 2 0 0.00% 0.00% 0.

13 10 5612 1310323 4 0.00% 0.00% 0.

14 11 0 1 0 0.00% 0.00% 0.

15 12 9340 1958204 4 0.00% 0.00% 0.

16 13 816 163286 4 0.00% 0.00% 0.

17 14 0 1 0 0.00% 0.00% 0.

18 15 0 1 0 0.00% 0.00% 0.

19 16 53564 9532356 5 0.00% 0.00% 0.

20 17 48908 9532355 5 0.00% 0.00% 0.

21 18 0 1 0 0.00% 0.00% 0.

22 19 0 1 0 0.00% 0.00% 0.

23 20 2884 559758 5 0.00% 0.00% 0.

24 21 0 1 0 0.00% 0.00% 0.

25 22 0 1 0 0.00% 0.00% 0.

26 23 4912 979770 5 0.00% 0.00% 0.

27 24 14716 1959147 7 0.00% 0.00% 0.

28 25 589856 23415322 25 0.00% 0.00% 0.

29 26 69316 10195196 6 0.00% 0.00% 0.

30

Var Line 1

five seconds: \$int:cpu_5seconds

2 CPU utilization for five seconds: 0%/0%; one minute: 0%; fi...

+ Field

Output

+ Parse Lines

\$cpu_5seconds (int) = 0

Sample Data

Help

All Intent Variables

Apply

- Define the diagnosis

2. Define Diagnosis 1

Add Note **Add Diagnosis** 2 Can also click a variable on the left to add automation.

Name: CheckCPU usage 3 Anchor: \$cpu_5seconds 4

Type description of the diagnosis...

☐ Loop Table Rows

If

A US-BOS-R1 5 Current

cpu_5seconds Greater than 5

B Select Variable

Then

Diagnosis Message: 6 High 5 second average CPU: \$cpu_5seconds% ☐ Save to Incident

☒ Set Status Code for Device:

Error 1 High 5 second average CPU: \$cpu_5seconds%

☒ Set Status Code for Intent:

Error 1 High 5 second average CPU: \$cpu_5seconds%

Else Delete

Diagnosis Message: 7 Low CPU: \$cpu_5seconds ☐ Save to Incident

☒ Set Status Code for Device:

Success 2 Low CPU: \$cpu_5seconds

☒ Set Status Code for Intent:

Success 2 Low CPU: \$cpu_5seconds

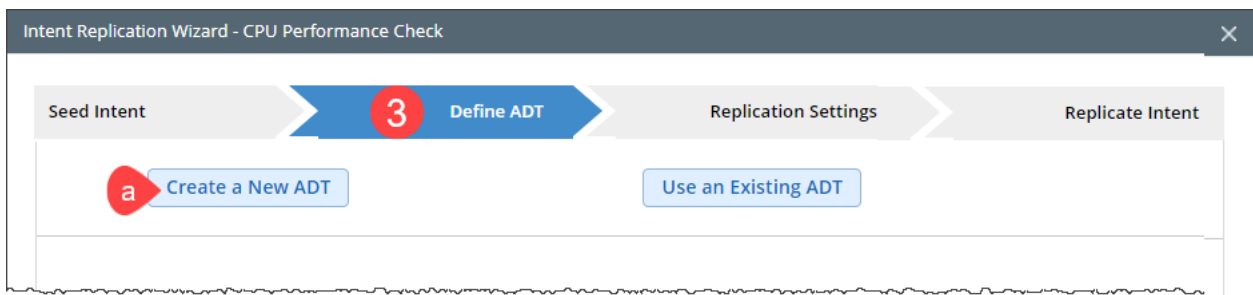
+ Add Elself

Cancel Apply 8

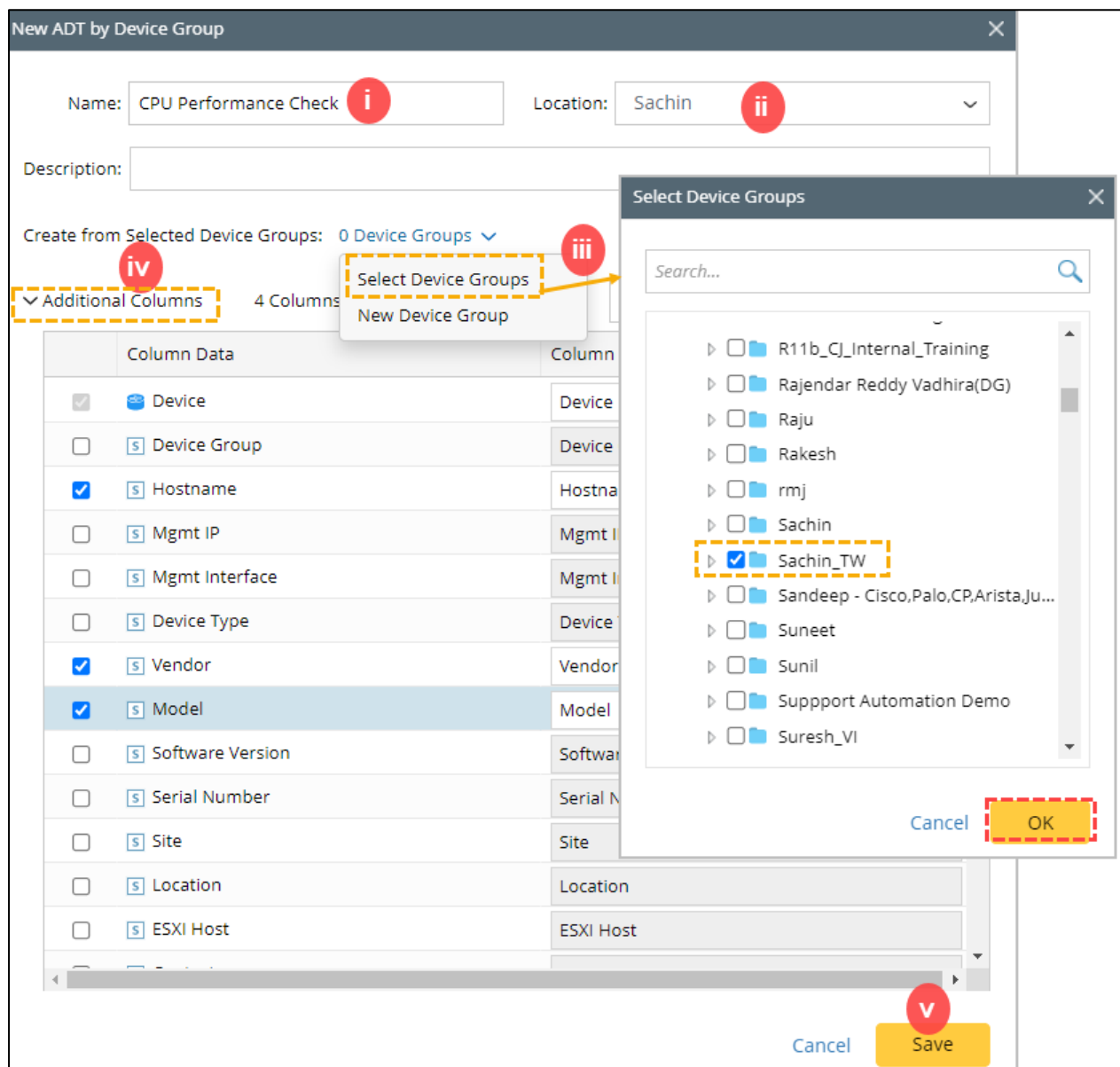
5.1.3 Use Intent Replication Wizard

In this section, you will use the **Intent Replication Wizard** to replicate the seed intent to a set of devices (of a device group). The replicated intents will be added to the Automation Column in the ADT table. The New ADT will be created.

1. Edit the intent. From the ☰ menu, open the **Intent Replication Wizard**.
2. In the **Seed Intent** tab, check for your NI (**CPU Performance Check**) and then click **Next** to go to the **Define ADT** tab.
3. Define ADT.
 - a) In the Define ADT tab, click **Create a New ADT** option to create a new ADT using Device Group.



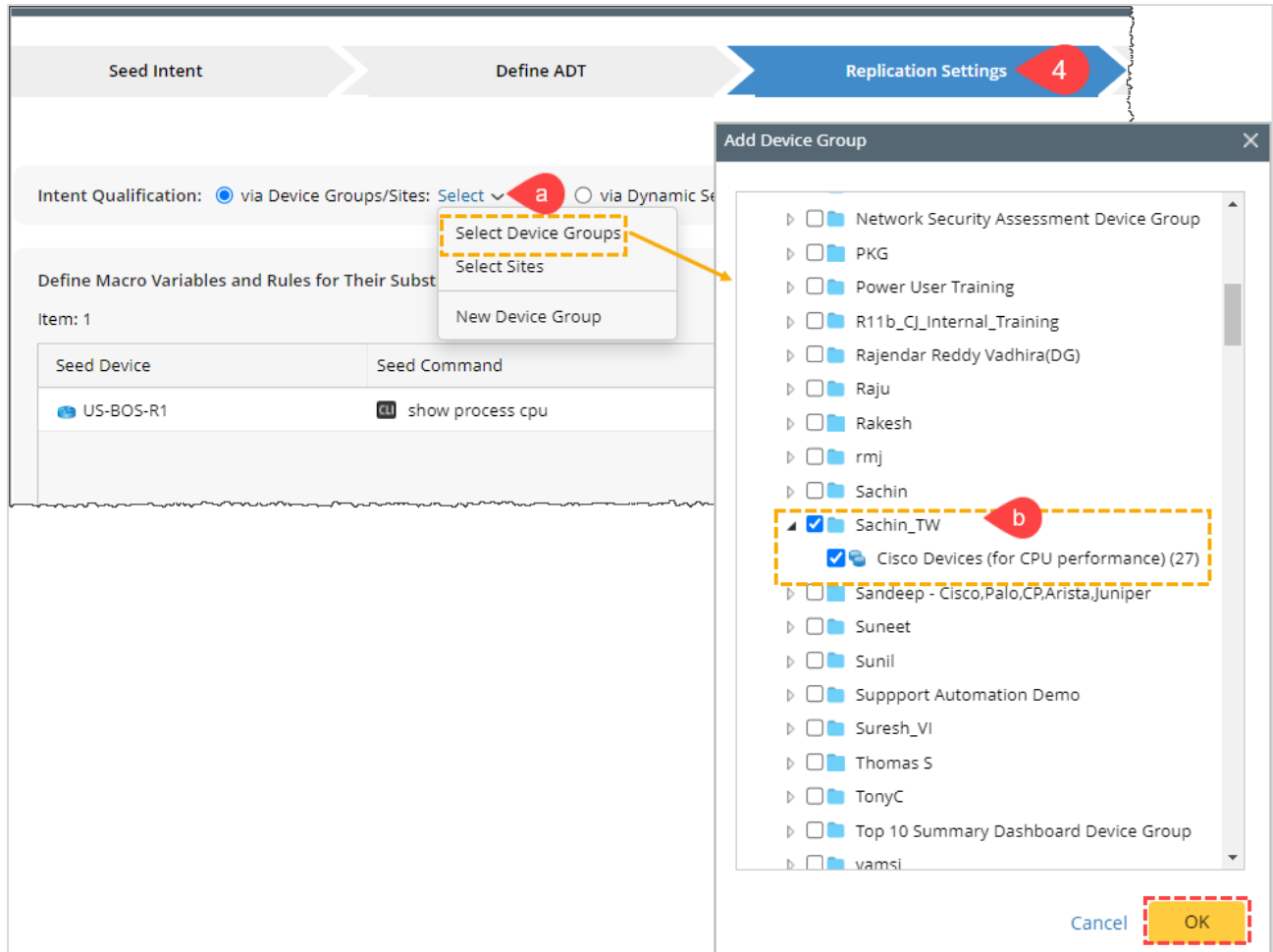
- i. Specify the ADT table name as **CPU Performance Check**. The description is optional.
- ii. Select the **Location** where you want to save the table.
- iii. From the **Device Groups** dropdown, click **Select Device Group** and then select a device group from your specific folder.
- iv. Click the **Additional Column** dropdown and select the column for ADT (e.g., Hostname, Vendor, and Model).
- v. Click **Save**.



- b) In the **Replicate Intent** section, enter the Intent group as an Intent column group name.
- c) Click **Next** to go to the **Replicate Intent** tab.

4. Replication Settings.

- In the **Intent Qualification** section, click the **Select** link to add Device Group.
- Add the device group.
- Click **Next** to go to the **Replicate Intent** tab.



5. Replicate Intent:

- In the **ADT Columns** section, you can see the **Column Name** as **Replicated Intent**. You can change this name to **CPU Performance**.
- Add more columns by ticking the checkboxes in the **Additional Columns** dropdown. Add columns, **Intent Status Code**, and **Last Execution Time**.
- Click **Save and Replicate** to save all the settings.
- You can see the **Open Output ADT** option after the successful submission of the replication request. Click **Open Output ADT** to check the replicated Intent column in the ADT Manager.

Intent Replication Wizard - CPU Performance Check

Seed Intent > Define ADT > Replication Settings > **5 Replicate Intent**

ADT Columns:

Column Data	Column Name	Tag
1 Replicated Intent	CPU Performance	0 tags
5 Intent Status Code	Intent Status Code	
1 Last Execution Time	Last Execution Time	

Additional Columns ▾

- ☒ Replicated Intent
- ☐ Intent Message
- ☒ Intent Status Code
- ☐ Device Status Code
- ☐ Intent Devices
- ☐ Intent Map
- ☐ Intent CLI Comma...
- ☒ Last Execution Time

Save and Replicate

Replication Request submitted at: 08/10/2024 01:49 PM

Open Output ADT

Selection Mode: Device-based Replication, ADT: CPU Performance Check, 0 Macro Variables.

Previous Finish

The table will now be populated with devices and the replicated Intents (**CPU Performance**).

6. Review the new Intent columns.

NOTE: All the columns related to the Intent results are empty because the Intents are not executed yet.

Automation Data Table Manager

CPU Performance Check Table Builder Last Updated at: 08/10/2024 01:49 PM Rebuild Table Add Data Manually


Description: Type description here...

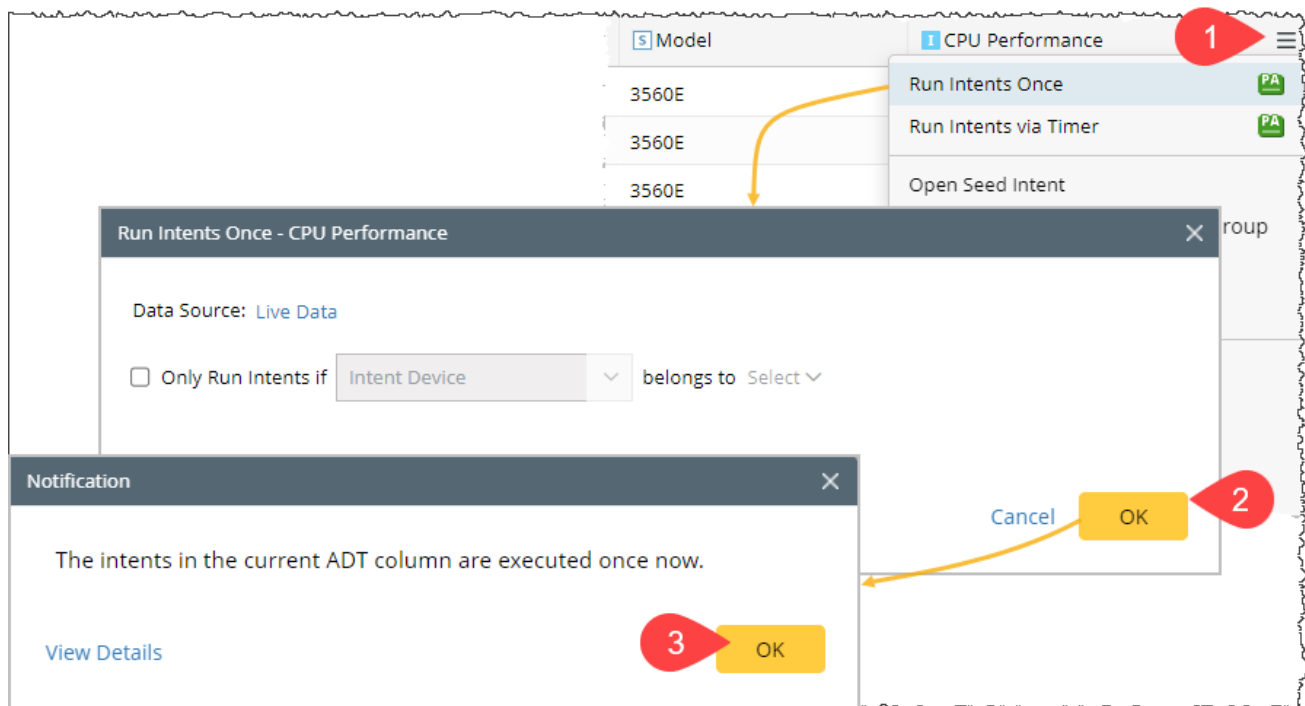
Items: 24 Rows 9 Columns Search... Advanced Filter: Undefined

No.	Device	Hostname	Vendor	Model	CPU Performance	Intent Status Code	Last Execution Time
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	CPU Performance Check US-BOS...		
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	CPU Performance Check US-BOS...		
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	CPU Performance Check US-BOS...		
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	CPU Performance Check US-BOS...		
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	CPU Performance Check US-BOS...		
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...		
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...		
8	BST_POP2	BST_POP2	Cisco	2621	CPU Performance Check BST_PO...		
9	BSTX	BSTX	Cisco	2811	CPU Performance Check BSTX		
10	BST,POP1	BST,POP1	Cisco	2514	CPU Performance Check BST,PO...		
11	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	CPU Performance Check BOS-cE...		
12	BJ_core_3550	BJ_core_3550	Cisco	catalyst355024	CPU Performance Check BJ_core...		
13	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048T5	CPU Performance Check BJ_L2_C...		
14	BJ_L2_Core_5	BJ_L2_Core_5	Cisco	catalyst356048T5	CPU Performance Check BJ_L2_C...		
15	BJ_L2_Core_4	BJ_L2_Core_4	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...		
16	BJ_L2_Core_3	BJ_L2_Core_3	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...		

5.1.4 Run the Intent

In this section, you will run intents once to populate all the results in the Intent columns (the Run Intent Once is useful for testing purposes. For continuous network assessments, you can schedule running intents by **Run Intent via Timer**). Follow the step-by-step instructions to **Run Intent Once** and **Rebuild Table**:

1. Hover over the Intent column, click  menu and then click **Run Intents Once** to execute all the Intents.
2. In the **Run Intents Once** dialog, various options are available for specifying the data source and filtering the Intents to be executed.
 - a) **Select Data Source**: Any one of the three Data Sources can be selected: **Live Data**, **Current Baseline**, and **ADT Dataset**. In this example, we will use **Live Data** as a Data Source.
 - b) Click **OK** to run the Intent once.
3. In the **Notification** window, click **OK**.

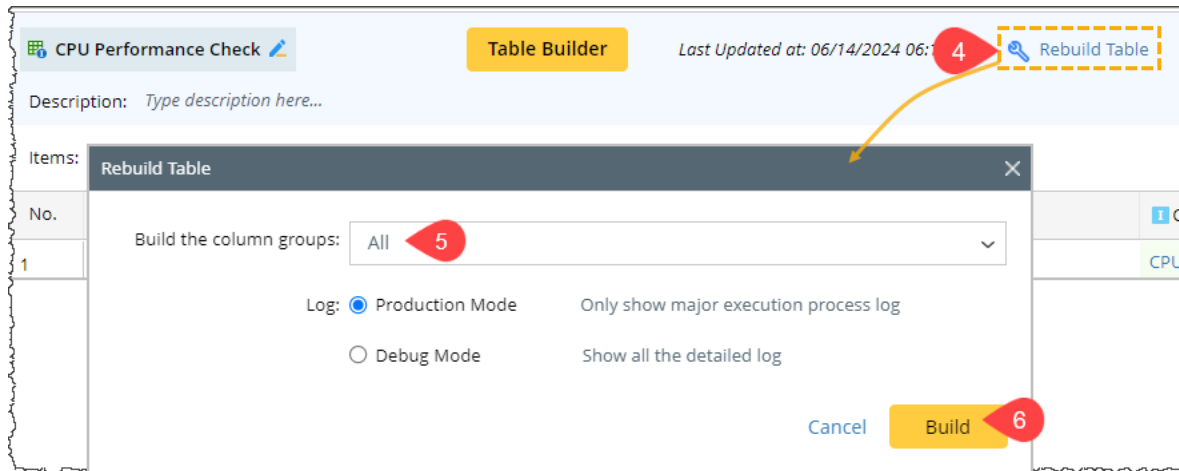


4. Wait for the system to finish executing the intents. Then, click **Rebuild Table** to open the **Rebuild Table** dialog.

Note: To ensure the intents are executed successfully, randomly click any intent row and check the diagnosis message for confirmation.

5. Select the **Column Group** from the dropdown and tick the **Production Mode** as per your preference.
6. Click **Build** to update the Intent column.

Please wait a few minutes for the Intents to execute, then check the Intent status code and execution time.




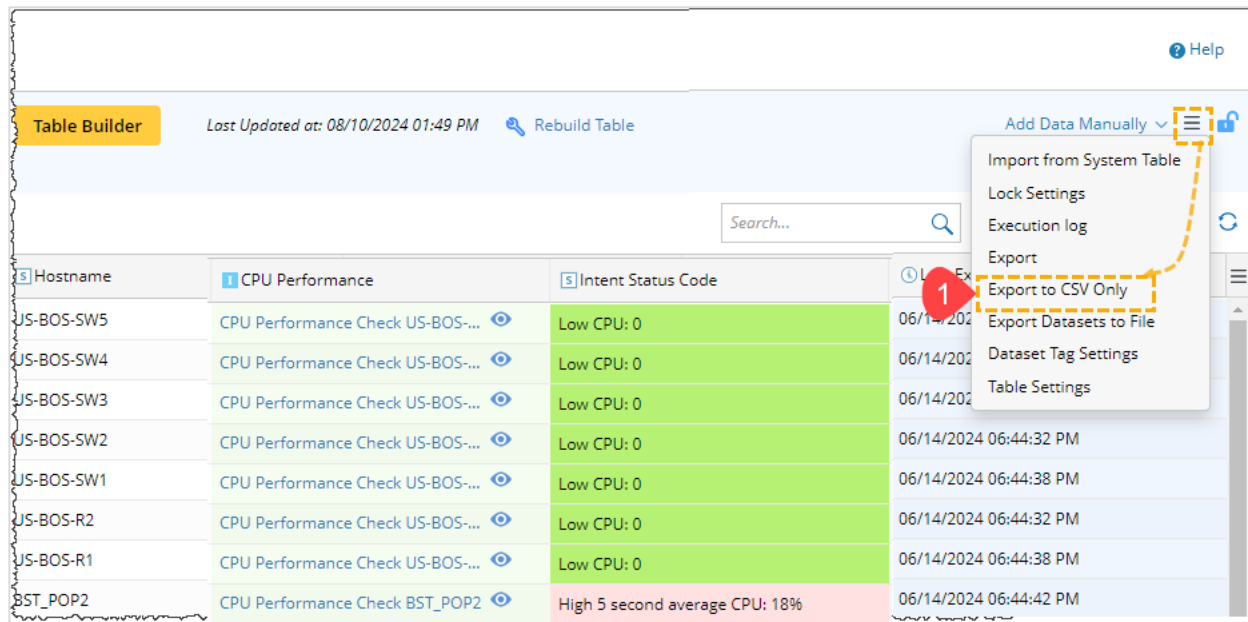
You can see the updated ADT with the Intent Status Code and the Last Execution columns. The ADT table supports the standard operations of a table. For example, you can search the keyword **High** to filter the devices with the alert status message, **high CPU**. The ADT table can be exported to a CSV file.

No.	Device	Hostname	Vendor	Model	CPU Performance	Intent Status Code	Last Execution Time
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:32 PM
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:32 PM
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:32 PM
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:32 PM
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:38 PM
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:32 PM
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:38 PM
8	BST_POP2	BST_POP2	Cisco	2621	CPU Performance Check BST_POP2	High 5 second average CPU: 18%	06/14/2024 06:44:42 PM
9	BSTX	BSTX	Cisco	2811	CPU Performance Check BSTX	Low CPU: 1	06/14/2024 06:44:34 PM
10	BST_POP1	BST_POP1	Cisco	2514	CPU Performance Check BST_POP1	Low CPU: 4	06/14/2024 06:44:35 PM
11	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	CPU Performance Check BOS-cEd...	Low CPU: 1	06/14/2024 06:44:32 PM
12	BJ_core_3550	BJ_core_3550	Cisco	catalyst355024	CPU Performance Check BJ_core...	Low CPU: 3	06/14/2024 06:44:38 PM
13	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_C...	Low CPU: 4	06/14/2024 06:44:42 PM
14	BJ_L2_Core_5	BJ_L2_Core_5	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_C...	Low CPU: 5	06/14/2024 06:44:35 PM
15	BJ_L2_Core_4	BJ_L2_Core_4	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...	High 5 second average CPU: 11%	06/14/2024 06:44:40 PM
16	BJ_L2_Core_3	BJ_L2_Core_3	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_C...	High 5 second average CPU: 6%	06/14/2024 06:44:39 PM

5.1.5 Export ADT to CSV file

ADT can be exported as a CSV file for documentation. To export the file, follow these steps:

1. In the ADT, click  menu from the upper-right corner and select the option **Export to CSV Only**.




The exported CSV file will be saved automatically to your computer's default download location, typically the "Downloads" folder (e.g., **C:\Users<your username>\Downloads**).

	A	B	C	D	E	F	G
1	Device	Hostname	Vendor	Model	CPU Performance	Intent Status Code	Last Execution Time
2	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	CPU Performance Check US-BOS-SW5	Low CPU: 0	2024-06-14T13:14:32.236Z
3	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	CPU Performance Check US-BOS-SW4	Low CPU: 0	2024-06-14T13:14:32.236Z
4	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	CPU Performance Check US-BOS-SW3	Low CPU: 0	2024-06-14T13:14:32.236Z
5	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	CPU Performance Check US-BOS-SW2	Low CPU: 0	2024-06-14T13:14:32.236Z
6	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	CPU Performance Check US-BOS-SW1	Low CPU: 0	2024-06-14T13:14:38.533Z
7	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS-R2	Low CPU: 0	2024-06-14T13:14:32.236Z
8	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS-R1	Low CPU: 0	2024-06-14T13:14:38.548Z
9	BST_POP2	BST_POP2	Cisco	2621	CPU Performance Check BST_POP2	High 5 second average CPU: 18%	2024-06-14T13:14:42.064Z
10	BSTX	BSTX	Cisco	2811	CPU Performance Check BSTX	Low CPU: 1	2024-06-14T13:14:34.392Z
11	BST_POP1	BST_POP1	Cisco	2514	CPU Performance Check BST_POP1	Low CPU: 4	2024-06-14T13:14:35.752Z
12	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	CPU Performance Check BOS-cEdge-01	Low CPU: 1	2024-06-14T13:14:32.236Z
13	BJ_core_3550	BJ_core_3550	Cisco	catalyst355024	CPU Performance Check BJ_core_3550	Low CPU: 3	2024-06-14T13:14:38.267Z
14	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_Core_6	Low CPU: 4	2024-06-14T13:14:42.486Z
15	BJ_L2_Core_5	BJ_L2_Core_5	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_Core_5	Low CPU: 5	2024-06-14T13:14:35.892Z
16	BJ_L2_Core_4	BJ_L2_Core_4	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_Core_4	High 5 second average CPU: 11%	2024-06-14T13:14:40.923Z
17	BJ_L2_Core_3	BJ_L2_Core_3	Cisco	catalyst37xxStack	CPU Performance Check BJ_L2_Core_3	High 5 second average CPU: 6%	2024-06-14T13:14:39.642Z
18	BJ_Dis_SW2	BJ_Dis_SW2	Cisco	catalyst295024			
19	BJ_Dis_SW1	BJ_Dis_SW1	Cisco	catalyst295024	CPU Performance Check BJ_Dis_SW1	High 5 second average CPU: 10%	2024-06-14T13:14:37.22Z
20	BJ_Acc_Sw4-bbb	BJ_Acc_Sw4-bbb-ee	Cisco	catalyst295024	CPU Performance Check BJ_Acc_Sw4-bbb-ee	High 5 second average CPU: 8%	2024-06-14T13:14:44.267Z
21	BJ_Acc_Sw4	BJ_Acc_Sw4	Cisco	catalyst295024	CPU Performance Check BJ_Acc_Sw4	Low CPU: 3	2024-06-14T13:14:37.439Z
22	BJ_Acc_SW6	BJ_Acc_SW6	Cisco	catalyst295024	CPU Performance Check BJ_Acc_SW6	High 5 second average CPU: 6%	2024-06-14T13:14:38.689Z
23	BJ_Acc_SW2-bbb	BJ_Acc_SW2-bbb-ee	Cisco	catalyst295024	CPU Performance Check BJ_Acc_SW2-bbb-ee	Low CPU: 1	2024-06-14T13:14:34.251Z
24	BJ_Acc_SW1	BJ_Acc_SW1	Cisco	catalyst295024			
25	BJPOP	BJPOP	Cisco	2811	CPU Performance Check BJPOP	Low CPU: 0	2024-06-14T13:14:38.252Z

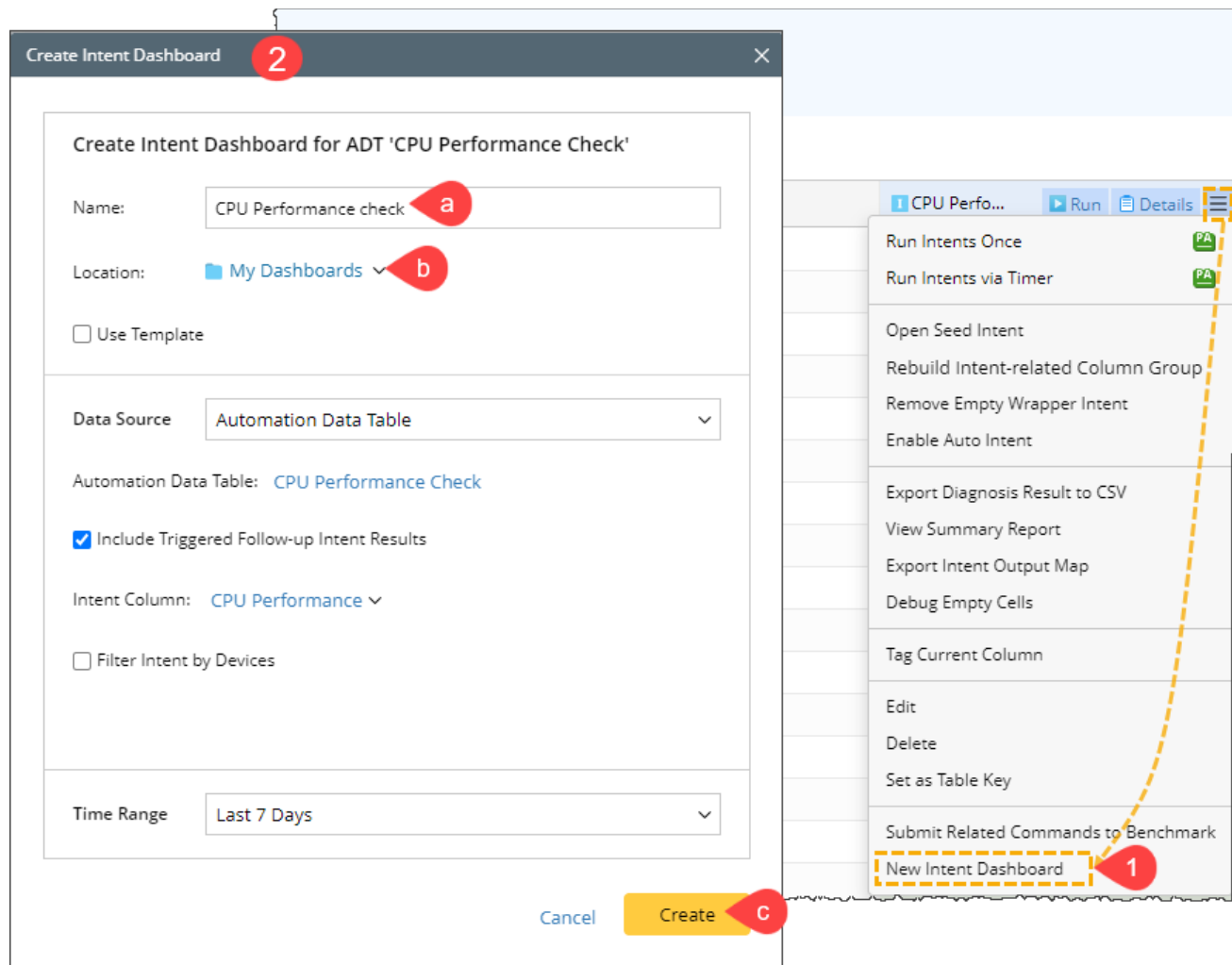
5.1.6 Create Intent Dashboard for ADT Automation Column

The dashboard provides an easy-to-understand and visual way to look into the ADT data, including the intent results. Especially the user can have an overview of the data with a glance. Creating the Dashboard from the ADT is easy as follows:

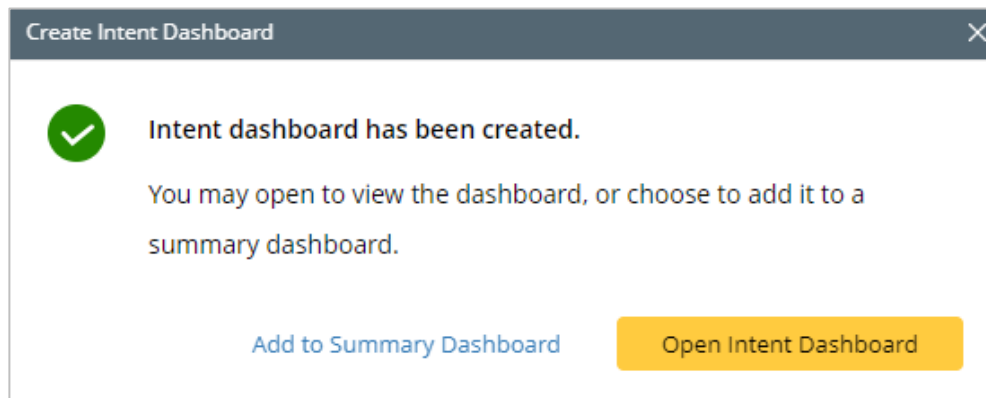
1. Hover over the Intent column, click  menu and select the **New Intent Dashboard**.
2. In the **Create Intent Dashboard** window, define the following:
 - a) Enter the **Dashboard Name**, such as **CPU performance check**.
 - b) Select the **Location** where you wish to save the Intent Dashboard.

The **Data Source** section is already set to be predefined ADT and Intent Columns since you are creating the dashboard using the automation column.

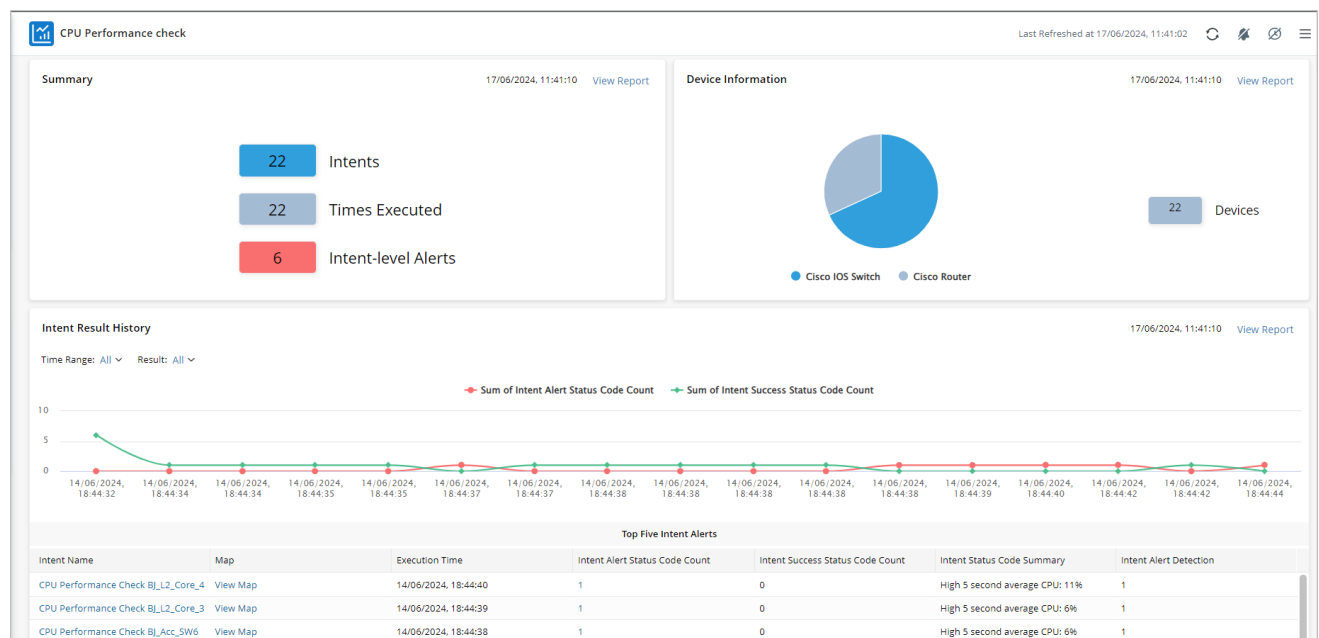
- c) Click **Create**.




3. In the Create Intent Dashboard dialog, click Open Intent Dashboard.



NOTE: The Intent Dashboard observes specific network issues with details, while the summary dashboard provides an overall view by displaying Intent results from multiple Intent dashboards. With Summary Dashboard, you can group Intent Dashboards into widgets based on diagnosis purposes and display Intent results by device, site or device groups. You can use the summary dashboard to monitor critical information across thousands of devices and discover the root cause for issues in one view.



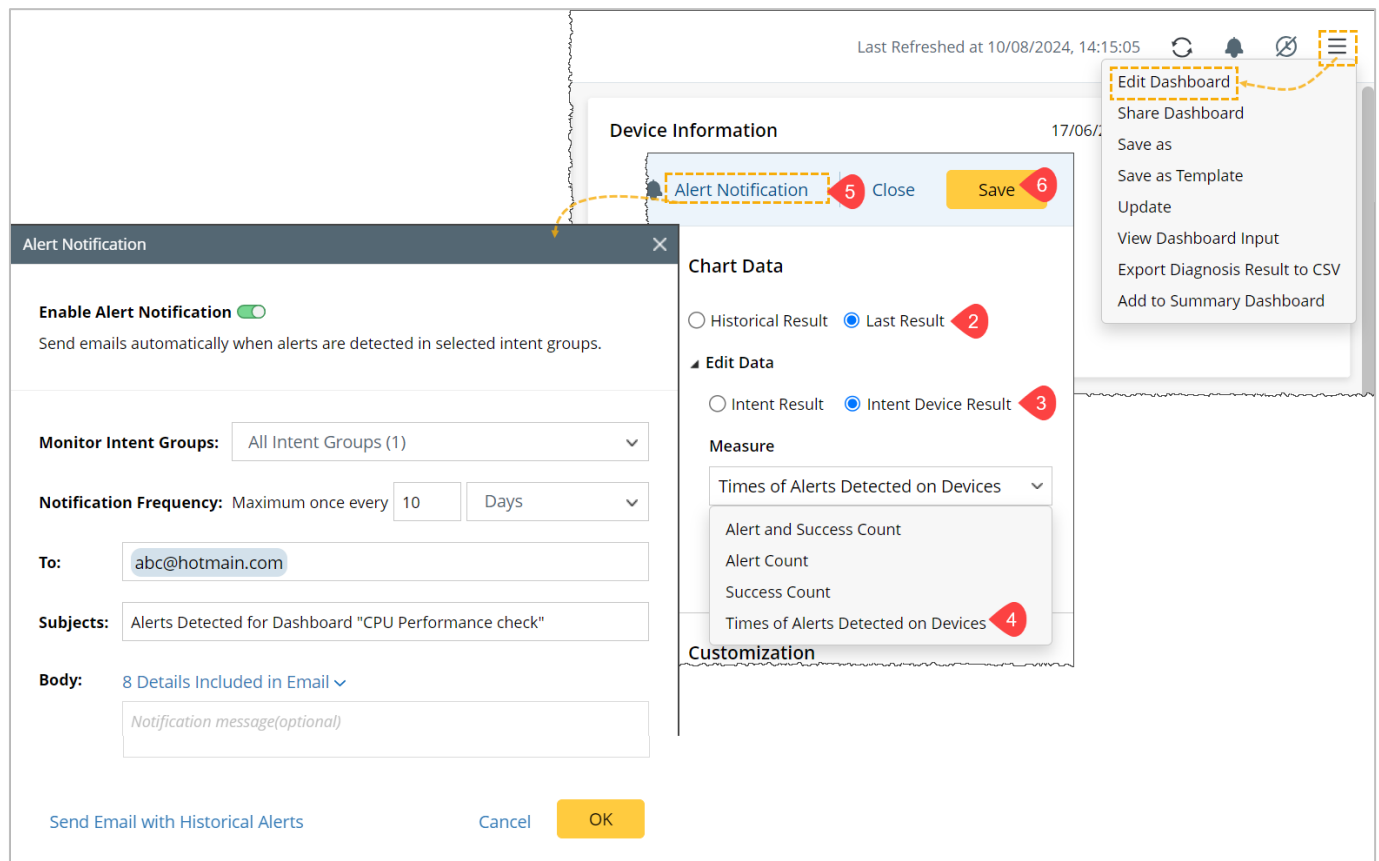
5.1.6.1 Edit Intent Dashboard

You can modify the Intent Dashboard in edit mode by selecting **Edit Dashboard** from  menu located at the top-right corner of the dashboard window:

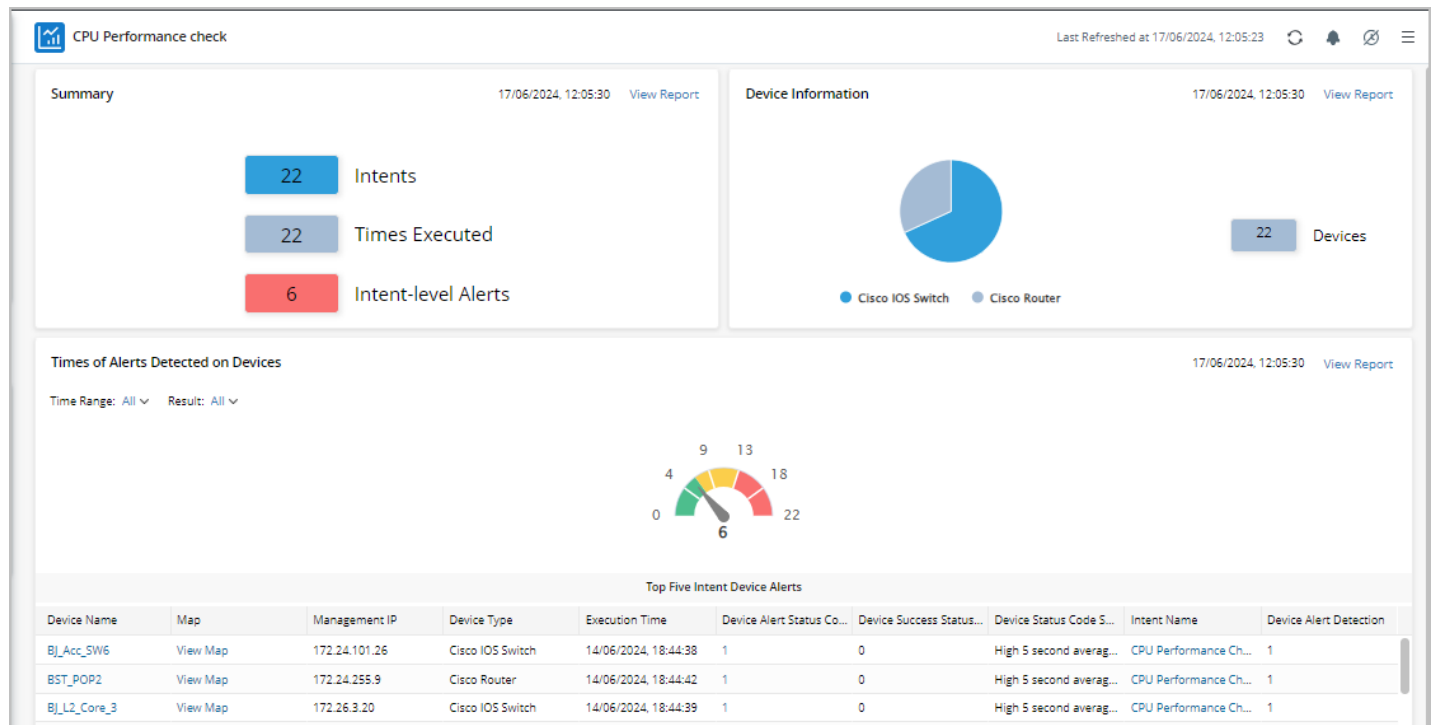
1. Click the **Intent Result History** section to view the dashboard data to be edited.
2. Change the Historical Result to the Last Result.
3. Click Edit Data and select Intent Device Result
4. Change Measure to Times of Alerts Detected on Devices.

From the **Dashboard**, you can observe that 6 devices out of 22 devices have **High 5 second average CPU**.

5. Use the **Alert Notifications** feature to enable Alert Notification to send alerts to the recipient's email address. You can adjust the Notification Frequency.
6. Click **Save** to save the Intent Dashboard and close the Dashboard.



The final Intent Dashboard will be like:



5.2 ADT Drilldown

In the last section, you learn how to create a new ADT via the **Intent Replication Wizard**. The key steps are:

- Create a new ADT from devices of a device group and add the built-in device properties to the ADT column. The devices and their properties form the **Base Table** of ADT, corresponding to the critical asset of your network.
- The seed intent is replicated, and cloned intents are added to the ADT group, which is called the **Column Group**, corresponding to the intents associated with the critical assets.

In this section, we will drill down deep into the ADT functions and operations:


- Create an ADT from scratch.

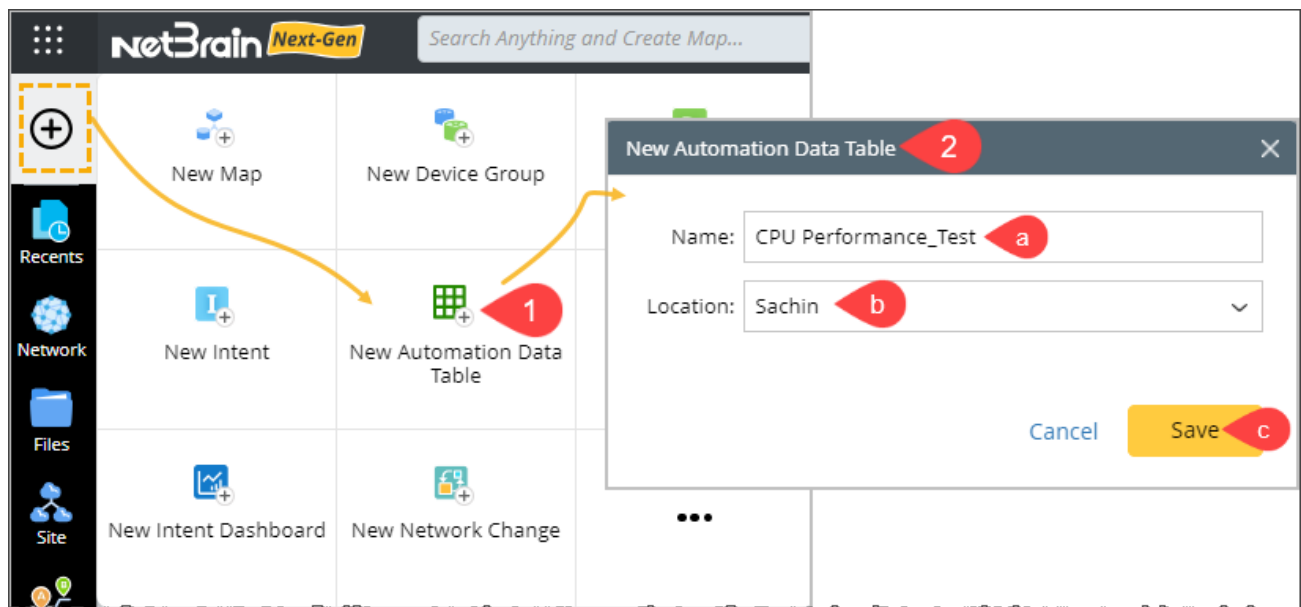
- Build ADT base from two common methods: from **Devices of a Device Group**, and from a **CSV file**. Another method, from the **Pre-replicated Intent Template**, is useful for the network assessment and document and will be covered in Section 5.3.
- Build ADT group from the intent template.
- Manually adjust the ADT.

You will use the same **Device Group** and a **Seed Intent** from Section 5.1 (CPU Performance Check) so that you can focus on the concepts and functions of the ADT. The goal is to obtain the same ADT as in Section 5.1, providing you with another way to understand ADT and its features.

5.2.1 Build Base ADT

To create an ADT and build the base table with the data in devices, follow the steps below:

1. Click the plus icon  and click **New Automation Data Table**.
2. In the New Automation Data Table popup:
 - a) Enter ADT Name.
 - b) Select the **Location** you wish to store the ADT.
 - c) Click **Save** and wait a moment for ADT to open.



3. Click **Table Builder** to open **Automation Data Table Builder** to define the Base Group.
4. Under the **Base** tab of Automation Data Table Builder, define the following settings:
 - a) Input **Description** for the base table to describe its use and function.
 - b) Select **Method**, the **Devices of Device Group** from the dropdown to build the base table.

- c) Click the **Select** link to select the created device group for building the base table with the devices in the device group, and then click **OK**.
- d) Within the Built-in Fields section, choose each column and move it into the **Column Group (Base)** pane.

You are going to build an ADT table with the following columns: **Device**, **Hostname**, **Vendor**, and **Model**.

- e) Click **Save and Build**. The **Build Table** dialog appears, define the settings as per your preferences and then click **Build** to save all the settings.

The screenshot shows the 'Automation Data Table Builder' window. At the top, the 'Column Header' section displays a table with four columns: c1 (Device), c2 (Hostname), c3 (Vendor), and c4 (Model). Below this, the 'Base' section contains a 'Description' field with the text 'ADT to check CPU performance' (annotated with 'a'), a 'Select Method to Build Base Table' dropdown set to 'Devices of Device Group' (annotated with 'b'), and a 'Select Devices by Device Group' dropdown set to 'Shared Device ...' (annotated with 'c'). The 'Built-in Fields' section on the left lists various fields, with 'Device', 'Hostname', 'Vendor', and 'Model' highlighted by a dashed orange box (annotated with 'd'). The 'Column Group (Base)' section on the right shows a grid of these four fields. A 'Select Device Groups' dialog is open in the foreground, showing a search bar and a list of device groups, with 'Sachin_TW' and 'Cisco Devices (for CPU perfor...)' selected (annotated with 'e'). The bottom of the window features an 'Auto-Build' status bar and a row of buttons: 'Cancel', 'Save', and 'Save and Build'.

The Base ADT will be like:

Automation Data Table Manager

Help

CPU Performance_Test

Table Builder

Last Updated at: 06/18/2024 01:46 PM

Rebuild Table

Add Data Manually

Description: Type description here...

Items: 27 Rows 4 Columns

Search...

Advanced Filter: Undefined

No.	Device	Hostname	Vendor	Model	
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	
8	BST_POP2	BST_POP2	Cisco	2621	
9	BSTX	BSTX	Cisco	2811	
10	BST,POP1	BST,POP1	Cisco	2514	
11	BST	BST	Cisco	2503	
12	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	
13	BOS-N9K-L3OUT	BOS-N9K-L3OUT	Cisco	NexusC9372TXE	
14	BJ_core_3550	BJ_core_3550	Cisco	WS-C3550-24	
15	BJ_L2_test_1	BJ_L2_test_1	Cisco	WS-C3750-24TS	
16	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048TS	

5.2.1.1 Edit ADT Table to Add Additional Fields

You will add additional fields **Mgmt IP** using table builder:

1. Click **Table Builder**.
2. From the **Built-in Fields** section, drag and drop **Mgmt IP** to the **Column Group (Base)** section.
3. Click **Save and Build**. The **Build Table** dialog appears, define the settings as per your preferences and then click **Build** to save all the settings.

The screenshot shows the 'Table Builder' interface. On the left, under 'Built-in Fields', the 'Mgmt IP' field is highlighted with a dashed orange box and a red circle with the number '2'. An orange arrow points from this field to the 'Column Group (Base)' section on the right. In this section, a table structure is visible with columns labeled 'c1' through 'c5'. Column 'c5' is also highlighted with a dashed orange box. At the bottom right, the 'Save and Build' button is highlighted with a red circle and the number '3'. The interface includes a 'Base' tab, a 'Description' field, a 'Select Method to Build Base Table' dropdown, and a 'Select Devices by Device Group' dropdown.


Here is the ADT after adding the additional column:

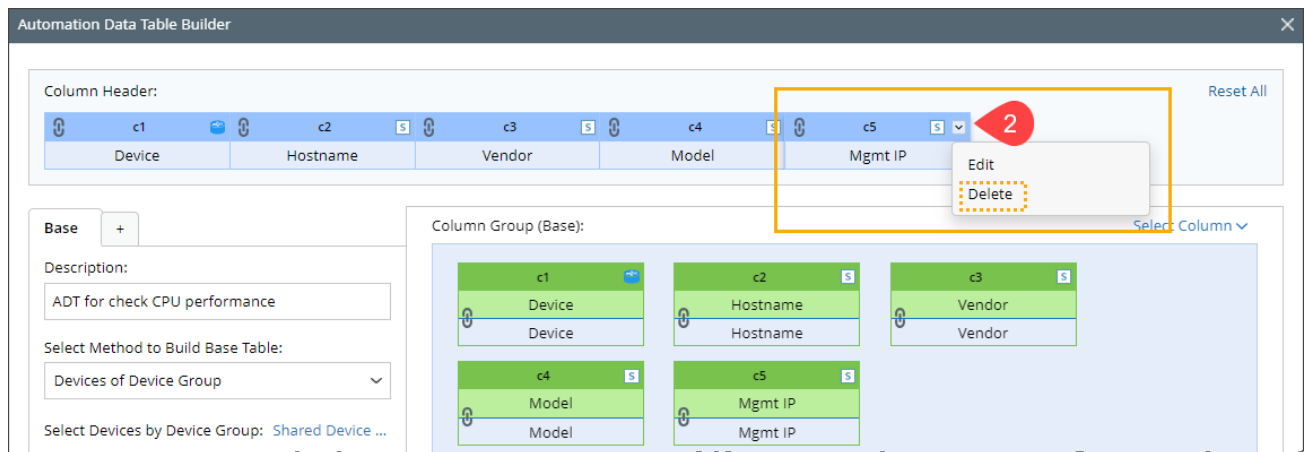
The screenshot shows the 'CPU Performance_Test' table in the 'Table Builder' interface. The table has 27 rows and 5 columns. The columns are 'No.', 'Device', 'Hostname', 'Vendor', and 'Mgmt IP'. The 'Mgmt IP' column is highlighted with a dashed orange box. The table data is as follows:

No.	Device	Hostname	Vendor	Model	Mgmt IP
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	10.8.1.52
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	10.8.1.244
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	10.8.1.243
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	10.8.1.242
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	10.8.1.241
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	10.8.1.240
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	10.8.1.51

5.2.1.2 Remove a Column



Follow the steps to delete a column:

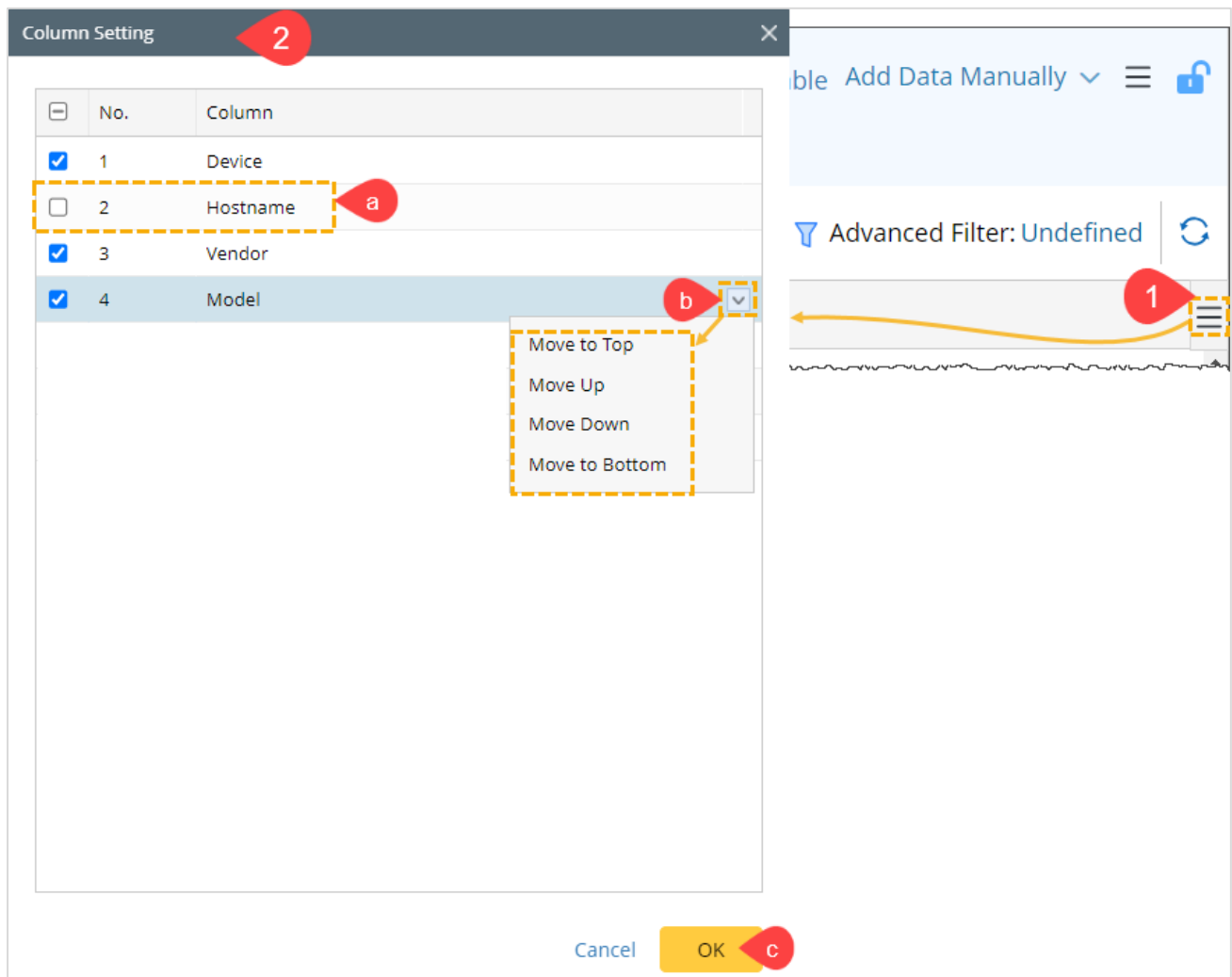
1. Click **Table Builder**.
2. From the **Column Header** section, hover over the **Mgmt IP** field, click  and then click Delete from dropdown.
3. Click **Save**.



5.2.1.3 Adjust ADT Column Order

After an ADT is created, you are allowed to change the order of its columns. Further, the columns can be set to be invisible to display only essential column data. Follow the below steps to adjust the ADT columns:

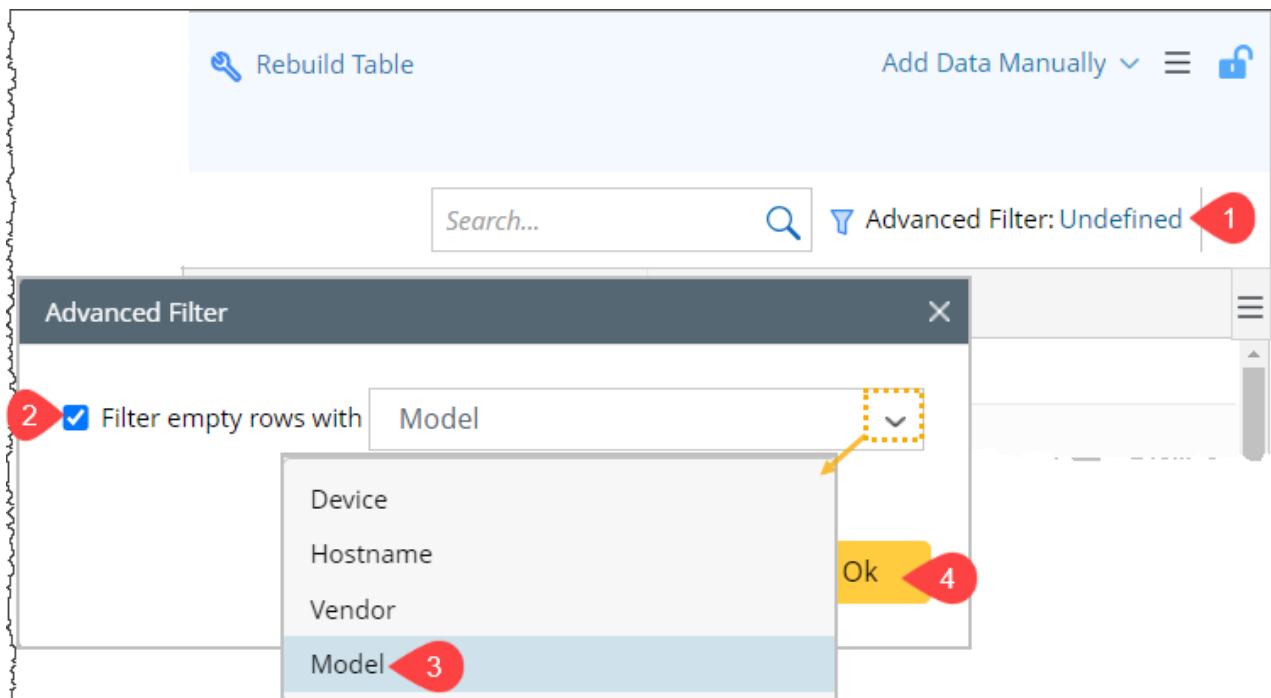
1. Click  menu in the top right corner of the column headers.
2. In the **Column Setting** window, you can perform the following operations:
 - a) Untick the checkbox of the column you wish to make invisible from the ADT.
 - b) Hover over the column you wish to adjust and click items listed in the  drop-down menu of a column to change their order.
 - c) Click **OK** to save the settings.



5.2.1.4 Use of Advanced Filter in ADT

In the ADT table, **Advanced Filter** is used to accumulate all the empty rows based on the selected column. Follow the below steps to define Advanced Filter:

1. In the ADT table header, click **Undefined** of the **Advanced Filter** field.
2. In the Advanced Filter popup, tick the **Filter empty row with** the checkbox.
3. From the dropdown, select the column you wish to filter. In this example, the Model column is selected.
4. Click **OK** to apply the filter.



5. When the filter is applied, you will see the two rows are filtered based on the column selection.

CPU Performance_Test

Table Builder

Last Updated at: 06/17/2024 05:0...

Description: Type description here...

Items: 27 Rows 4 Columns

Search...

Advanced Filter: Defined

No.	Device	Hostname	Vendor	Model
1	US-BOS-R1	US-BOS-R1	Cisco	
2	US-BOS-R2	US-BOS-R2	Cisco	

5.2.2 Build ADT from a CSV File


In the last section, you learned how to build base ADT using the **Devices of Device Groups** method. This section teaches how to create the ADT base table by **Importing a CSV File**.

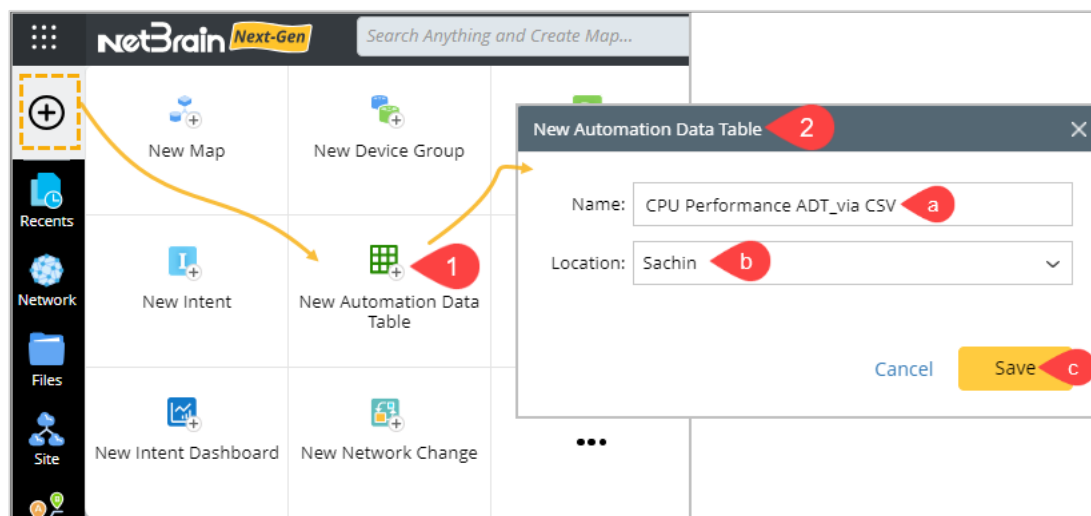
NOTE: Make sure you have a CSV file available on your local computer. Once imported, the contents of the CSV file will be loaded into the ADT.

Sample of CSV file content:

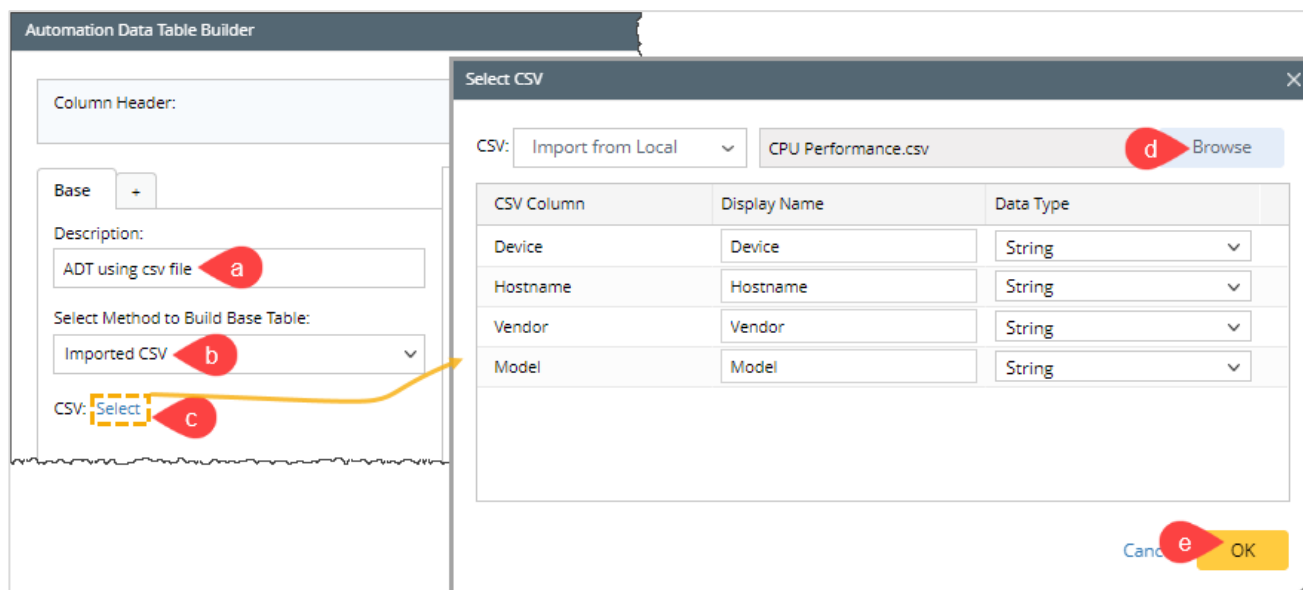
Device	Hostname	Vendor	Model
US-BOS-SW5	US-BOS-SW5	Cisco	3560E
US-BOS-SW4	US-BOS-SW4	Cisco	3560E
US-BOS-SW3	US-BOS-SW3	Cisco	3560E
US-BOS-SW2	US-BOS-SW2	Cisco	3560E
US-BOS-SW1	US-BOS-SW1	Cisco	3560E
US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS
US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS
BST_POP2	BST_POP2	Cisco	2621
BSTX	BSTX	Cisco	2811
BST	BST	Cisco	2514

To build the base table with the data in devices, follow the steps below:

1. Click the plus icon  and click **New Intent-Based Automation**.
2. In the **New Automation Data Table** popup:
 - a) Provide an ADT Name.
 - b) Select the **Location** you wish to store the ADT.
 - c) Click **Save** and wait a moment for ADT to open.

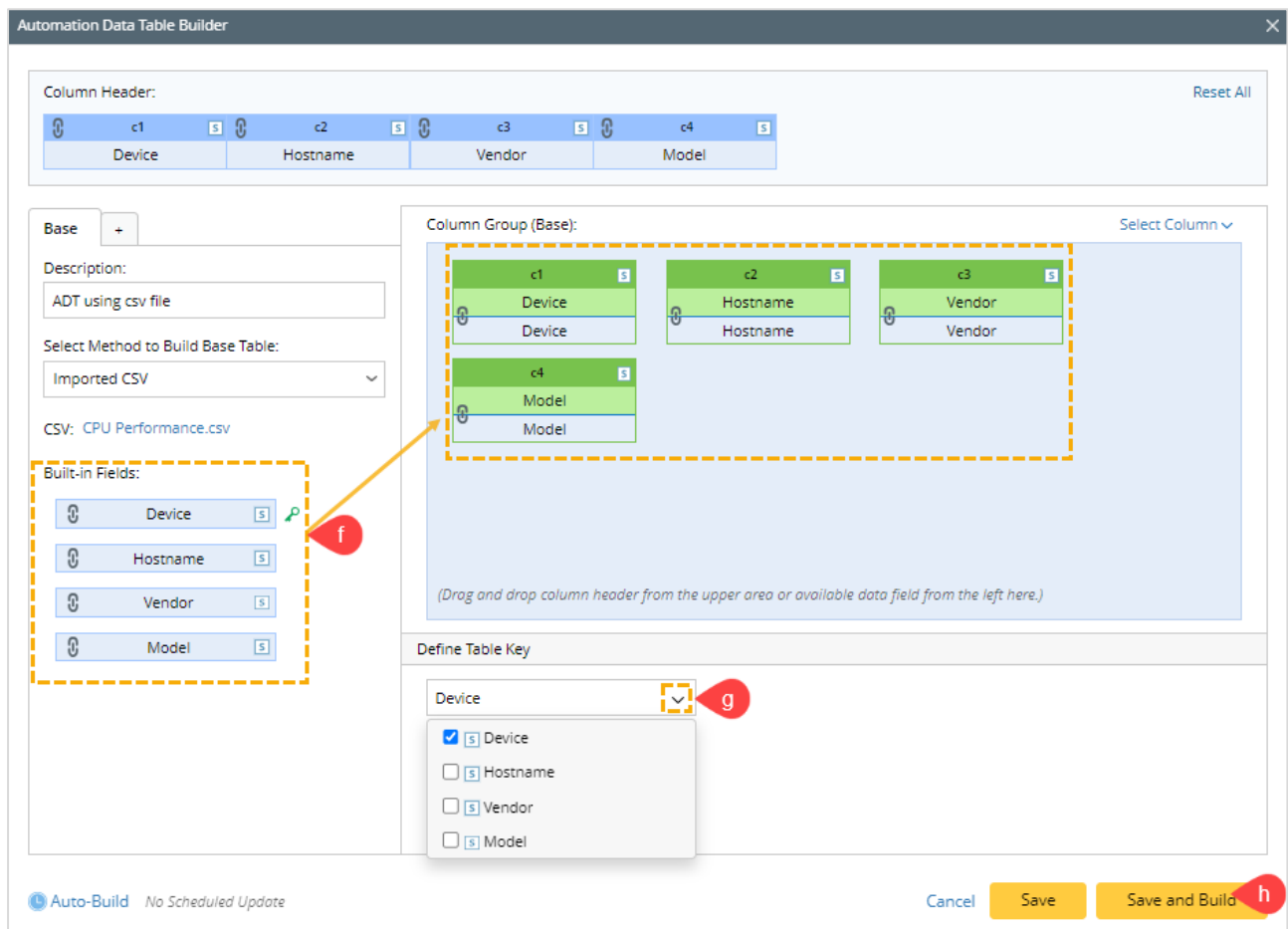


3. Click **Table Builder** to open **Automation Data Table Builder** to define the Base Group.
4. Under the **Base** tab of **Automation Data Table Builder**, define the following settings:
 - a) Input descriptions for the base table to describe its use and function.
 - b) Select Method **Imported CSV** from the dropdown to Build Base Table.
 - c) Click the **Select** link to import the CSV file.
 - d) Click **Browser** and select the CSV file from your local computer.
 - e) Click **OK**.



Under Built-in Fields, you can see Device, Hostname, Vendor, and Model fields.

- f) Within the **Built-in Fields** section, choose all the fields and move them into the **Column Group (Base)** pane.
- g) Under the **Define Table Key** dropdown, tick the **Device** checkbox.
- h) Click **Save and Build**. The Build Table dialog appears, define the settings as per your preferences and then click **Build** to save all the settings.



The ADT Base table is successfully built from the imported CSV file, which contains the columns.

Automation Data Table Manager

CPU Performance ADT_via CSV

Description: Type description here...

Items: 10 Rows 4 Columns

Search...

Advanced Filter: Undefined

No.	Device	Hostname	Vendor	Model
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS
8	BST_POP2	BST_POP2	Cisco	2621
9	BSTX	BSTX	Cisco	2811
10	BST,POP1	BST,POP1	Cisco	2514

5.2.3 Build ADT from Intent Template

After building a base table, you can add the automation to the **Column Group**. The Column Group can be built with the same types of methods to build the base table. The most frequently used one is the **Intent Template**.

Let us add an Intent group in the ADT (choose any of the Base ADTs created in the previous sections) using the Seed Intent that you have created in Section 5.1.3. Follow the steps to add Intent columns in the ADT using the **Intent Template** method:

1. In the ADT, click **Table Builder** to open **Automation Data Table Builder**.


The screenshot displays the **Automation Data Table Builder** window. The interface is divided into several sections:

- Column Header:** Shows a table with columns c1 (Device), t1 (Last Execution Time), c4 (Model), c5 (Replicated Intent), and c6 (Intent Status Code). A **Create Group** dialog is open, showing the **Intent Group** name.
- Base:** A dropdown menu shows **Intent Gro...** with a red circle 2 next to it.
- Description:** A text input field.
- Select Method to Build Group Table:** A dropdown menu shows **Intent Template** with a red circle 3 next to it.
- Intent Template:** A dropdown menu shows **CPU Performance Check** with a red circle 4 next to it.
- Built-in Fields:** A list of fields including **Replicated Intent**, **Intent Message**, **Intent Status Code**, and **Last Execution Time**. A red circle 5 is next to the **Replicated Intent** field.
- Column Group (Intent Group):** A table showing the columns c5 (Replicated Intent), c6 (Intent Status Code), and t1 (Last Execution Time). A red circle 6 is next to the **Replicated Intent** column.
- Define Logic to Populate New Columns for Each Row:** A section with a dropdown menu showing **Device** and a checkbox for **Device** (checked). A red circle 6 is next to the dropdown.
- Buttons:** **Auto-Build**, **Cancel**, **Save**, and **Save and Build** (with a red circle 7 next to it).

The **Select Intent Template** dialog is also shown, with the following details:

- Select Intent Template from:** **All Intents** (selected).
- Type:** **Common Intent**.
- Filter by:** **Device-based** and **sachin**.
- Intents:** A list of intents including **Check AWS EC2 Configuration Against Baseline**, **CP1**, **CPU Performance Check** (highlighted), **Path test**, and **PCEC2**.
- Buttons:** **Cancel** and **OK**.



- Click the  icon to open the **Create Group** popup, type the group name as **Intent Group**, and then click **OK**.
- Type **Description** (optional), and check for the method selected as **Intent Template**.
- Select previously created Intent, i.e., **CPU Performance Check**.
- Drag and drop required fields from the **Built-in Fields** and **Intent Output** sections into the **Column Group (Intent Group)** section. Add **Replicated Intent**, **Intent Status Code** (which usually includes the intent execution results), and **Last Execution Time** (knowing when the intent was executed last time can be useful for the end users).
- Select **Device** from the dropdown to replicate Intents to the device column.
- Click **Save and Build**. The **Build Table** dialog appears, define the settings as per your preferences and then click **Build** to save all the settings.

Wait a moment for ADT to populate the Intent columns.

Automation Data Table Manager							
CPU Performance_Test							
Description: Type description here...							
Items: 27 Rows 7 Columns							
No.	Device	Hostname	Vendor	Model	Replicated Intent	Intent Status Code	Last Execution Time
1	US-BOS-SW5	US-BOS-SW5	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
2	US-BOS-SW4	US-BOS-SW4	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
3	US-BOS-SW3	US-BOS-SW3	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
4	US-BOS-SW2	US-BOS-SW2	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
5	US-BOS-SW1	US-BOS-SW1	Cisco	3560E	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
6	US-BOS-R2	US-BOS-R2	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
7	US-BOS-R1	US-BOS-R1	Cisco	CGS-MGS-AGS	CPU Performance Check US-BOS...	Low CPU: 0	06/14/2024 06:44:3...
8	BST_POP2	BST_POP2	Cisco	2621	CPU Performance Check BST_PO...	High 5 second average CPU: ...	06/14/2024 06:44:4...
9	BSTX	BSTX	Cisco	2811	CPU Performance Check BSTX	Low CPU: 1	06/14/2024 06:44:3...
10	BST_POP1	BST_POP1	Cisco	2514	CPU Performance Check BST_PO...	Low CPU: 4	06/14/2024 06:44:3...
11	BST	BST	Cisco	2503			
12	BOS-cEdge-01	BOS-cEdge-01	Cisco	CSR1000V	CPU Performance Check BOS-cE...	Low CPU: 1	06/14/2024 06:44:3...
13	BOS-N9K-L3OUT	BOS-N9K-L3OUT	Cisco	NexusC9372TXE			
14	BJ_core_3550	BJ_core_3550	Cisco	WS-C3550-24	CPU Performance Check BJ_core...	Low CPU: 3	06/14/2024 06:44:3...
15	BJ_L2_test_1	BJ_L2_test_1	Cisco	WS-C3750-24TS			
16	BJ_L2_Core_6	BJ_L2_Core_6	Cisco	catalyst356048TS	CPU Performance Check BJ_L2_C...	Low CPU: 4	06/14/2024 06:44:4...

You can create an Intent Dashboard similar to the Section [5.1.6](#).

5.3 Assess and Document Your Network Operational Status

In the last two sections, you learn how to create the base table of an ADT from the built-in system data, such as network devices and their properties. We recommend that you use this easy method to populate your base table if possible. However, the built-in data may not satisfy your need, and you have to query the network operational status from the live network via the CLI command. For this case, you will parse the data from the CLI command results and export the data (variables) to the ADT table. The method to build this use case is called the **Pre-replicated Intent Template**.

In this section, you will be asked to document all down interfaces, including **Administratively Down** or **Down**, and the interface uptime. This report can help you figure out how many ports are not used in your network and future network expansion planning.

This section includes the following main steps:

- [Create Network Intent](#)
- [Add Signature Variables to ADT Base Table](#)
- [Export ADT to CSV file](#)

The Final ADT will be like:

No.	Device	Interface_Name	Status	Input	Hostname	Mgmt IP	Vendor	Model	Software Version
1	US-BOS-SW2	Ethernet1/0	administratively down	never	US-BOS-SW2	10.8.1.242	Cisco	3560E	15.2(HL_20170202)F...
2	BSTX	Serial0/2/0	down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
3	BSTX	Serial0/3/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
4	BSTX	Serial0/3/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
5	BSTX	BRI1/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
6	BSTX	BRI1/0:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
7	BSTX	BRI1/0:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
8	BSTX	BRI1/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
9	BSTX	BRI1/1:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
10	BSTX	BRI1/1:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
11	BSTX	BRI1/2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
12	BSTX	BRI1/2:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
13	BSTX	BRI1/2:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
14	BSTX	BRI1/3	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
15	BSTX	BRI1/3:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
16	BSTX	BRI1/3:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
17	BSTX	BRI1/4	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
18	BSTX	BRI1/4:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
19	BSTX	BRI1/4:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4

5.3.1 Create Network Intent

In this section, you will create a Network Intent to specify the CLI command for the Parser and subsequently define the **Signature Variable** for down Interfaces. The Signature Variables can be imported to the ADT.

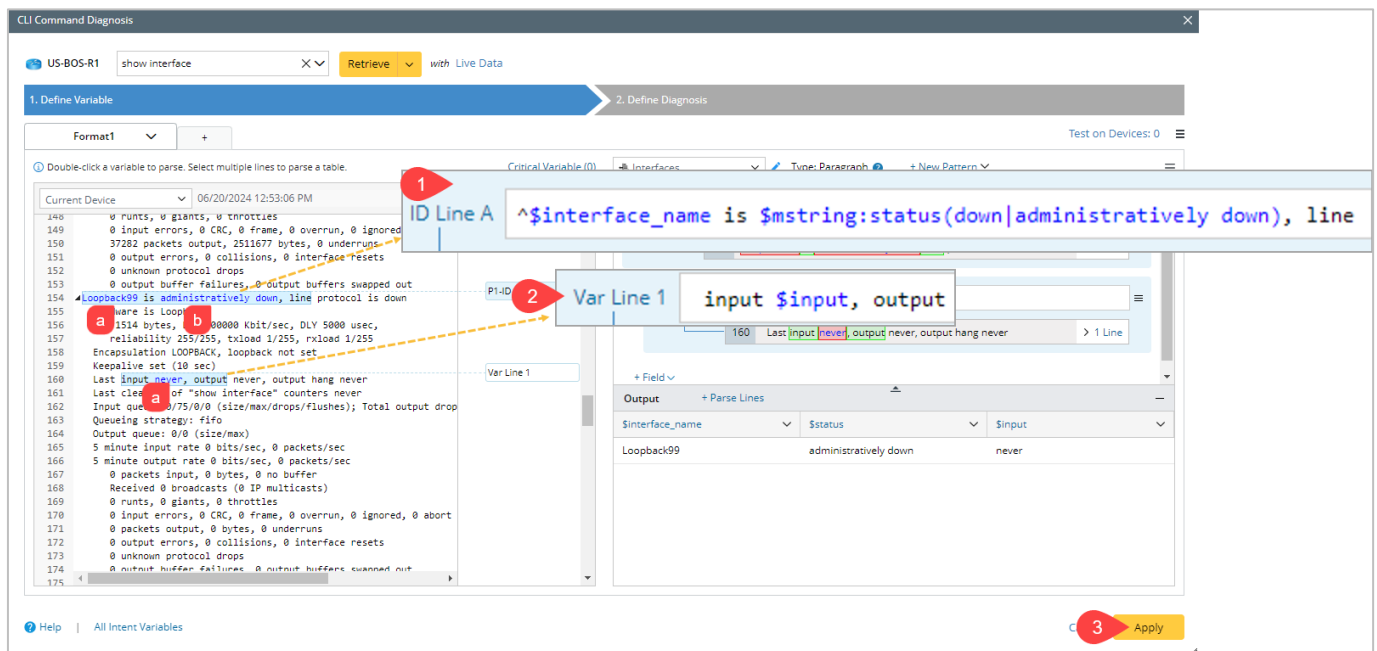
1. From the **Intent Manager**, create a new intent, **Down Interface Report**.
2. Add the target device(s), e.g., device **US-BOS-R1**.
3. Add a **CLI Diagnosis**: enter the command **show interface** and retrieve the data from the **Live Data**.
4. In the **+ New Pattern** dropdown, select **Paragraph** parser.

The screenshot displays the 'Network Intent (Edit Mode)' window. At the top, the intent is named 'Down Interface Report'. Below this, there's a section for 'US-BOS-R1' with a '+ Add CLI Diagnosis' button highlighted by a red circle with the number 3. A yellow dashed arrow points from this button to the 'CLI Command Diagnosis' section below. In this section, the device 'US-BOS-R1' and the command 'show interface' are entered, with a 'Retrieve' button and a 'with Live Data' option. Below this, there are two tabs: '1. Define Variable' and '2. Define Diagnosis'. The '1. Define Variable' tab is active, showing a list of sample data for the command 'show interface'. A yellow callout box labeled 'Sample Data' points to this list. On the right side of the '1. Define Variable' tab, there's a '+ New Pattern' dropdown menu, with the 'Paragraph' option highlighted by a red circle with the number 4. A yellow dashed arrow points from the 'Paragraph' option to the 'Paragraph' button in the 'CLI Command Diagnosis' section.

5.3.1.1 Define the Paragraph Parser

In the sample data, select the text and create the following Variables:

1. Define ID Line A.
 - a) Double-click the interface **Loopback99** in line 154. The Variable **\$var1** is created.
Change the default name **\$var1** to **\$Interface_name**
 - b) In the same line of the sample data, double-click the phrase **administratively down**. The Variable **\$var2 down**, is created in the same ID Line.
Change the Variable **\$var2 down** with **\$mstring:status(down|administratively down), line**
The final **ID Line A** will be: **^\$Interface_name is \$mstring:status(down|administratively down), line**
2. Define **Var Line 1** for interface uptime.
 - a) Double-click the word **never** in line 160. The Variable **Input \$input, output** is created in the Var Line 1.
3. Verify the **Output** and click **Apply**.



4. In the **Confirmation** popup, click **Apply and Continue** to save the parser.
5. Click the pen icon, rename the parser name from **Paragraph1** to **Interfaces**, and then click **OK**.

5.3.1.2 Define Intent Variable for Seed Logic

Since you only want to report the down interfaces (not all interfaces), you can create a compound table (sub-table) to filter out these interfaces:

Intent Variables for Seed Logic

Intent Variable Use Automation Data Table Task Variable

+ Add Compound Variable + Add Compound Table

Intent

US-BOS-R1

Interfaces

US-BOS-R1

Merged Table

Sub Table (Filtered by Value)

Sub Table (Filtered by Condition)

Appended Table

Run with Live Data Save Help

Intent Settings

Intent Variables

Full Settings for Template

Lock Settings

Add Intent Diagnosis Block

Add Diagnosis via Auto Intent

Switch Devices

Add Sub Table (Filtering Row by Condition)

Table Name: down_interfaces

Base Table: Interfaces

Filtering Logic: Only Keep table rows if values match the below condition

A status Contains down

B Please Select

Boolean Expression: A

Calculate


Base Table (Interfaces) New Table (down_interfaces)

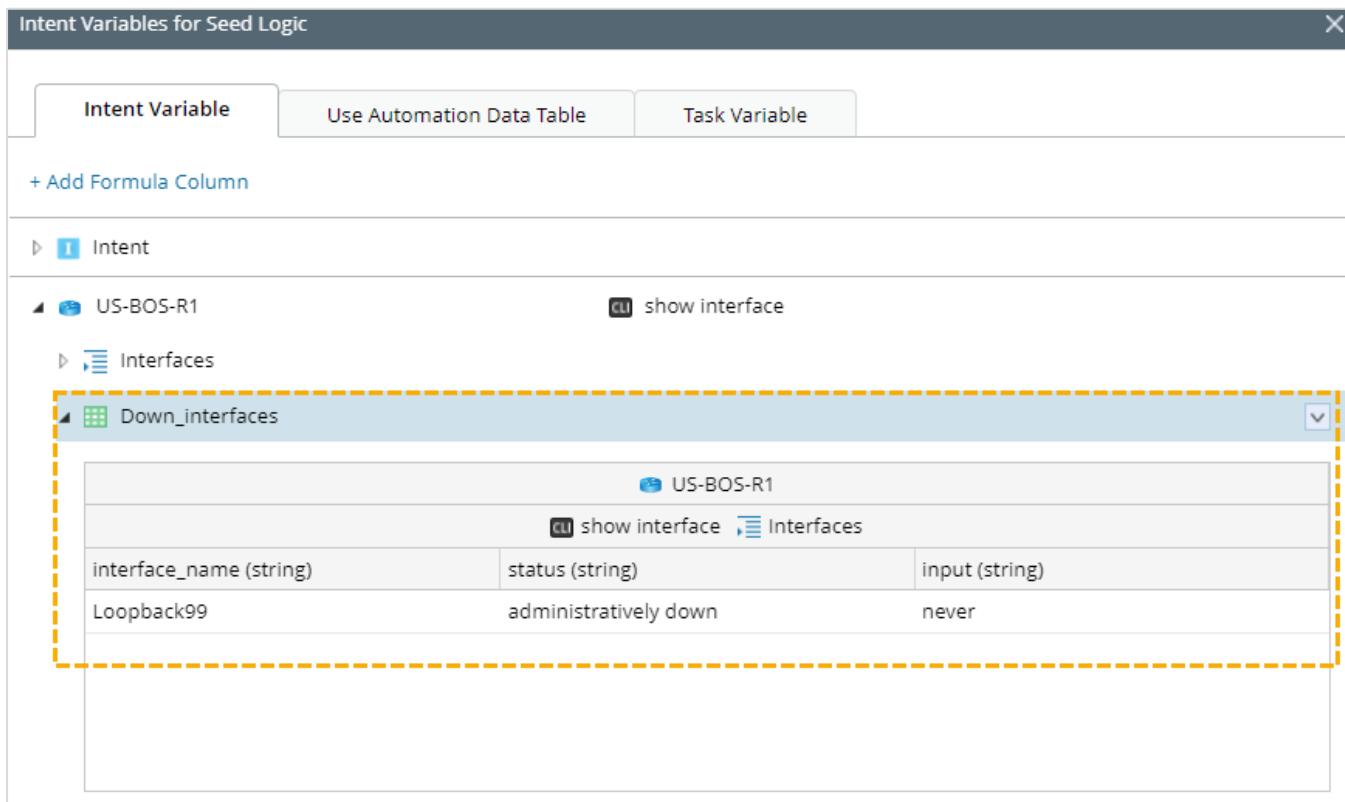
US-BOS-R1

show interface Interfaces

interface_name (string)	status (string)	input (string)
Loopback99	administratively down	never

Help Cancel OK

1. In the **Network Intent (Edit Mode)** window, click  Menu and open **Intent Variables** to define compound Variables.
2. In the **Intent Variables for Seed Logic** window, select **US-BOS-R1**.
3. From the **+ Add Compound Table** dropdown, select the **Sub Table (Filtered by Condition)** option.
4. In the **Add Sub Table (Filtering Row by Condition)** window, enter the table name like **down_interfaces**.
5. Select variable **Interfaces** as a Base Table from the dropdown.
6. Define condition **A** for Variable **status** contains **down**.
7. Click **OK** to save the configuration. You will see the Compound Table is added in the **Intent Variables** tab.
8. Click **Close** to save and close the window.




Intent Variables for Seed Logic

Intent Variable Use Automation Data Table Task Variable



+ Add Formula Column

▶ Intent

▶ US-BOS-R1  show interface


▶ Interfaces

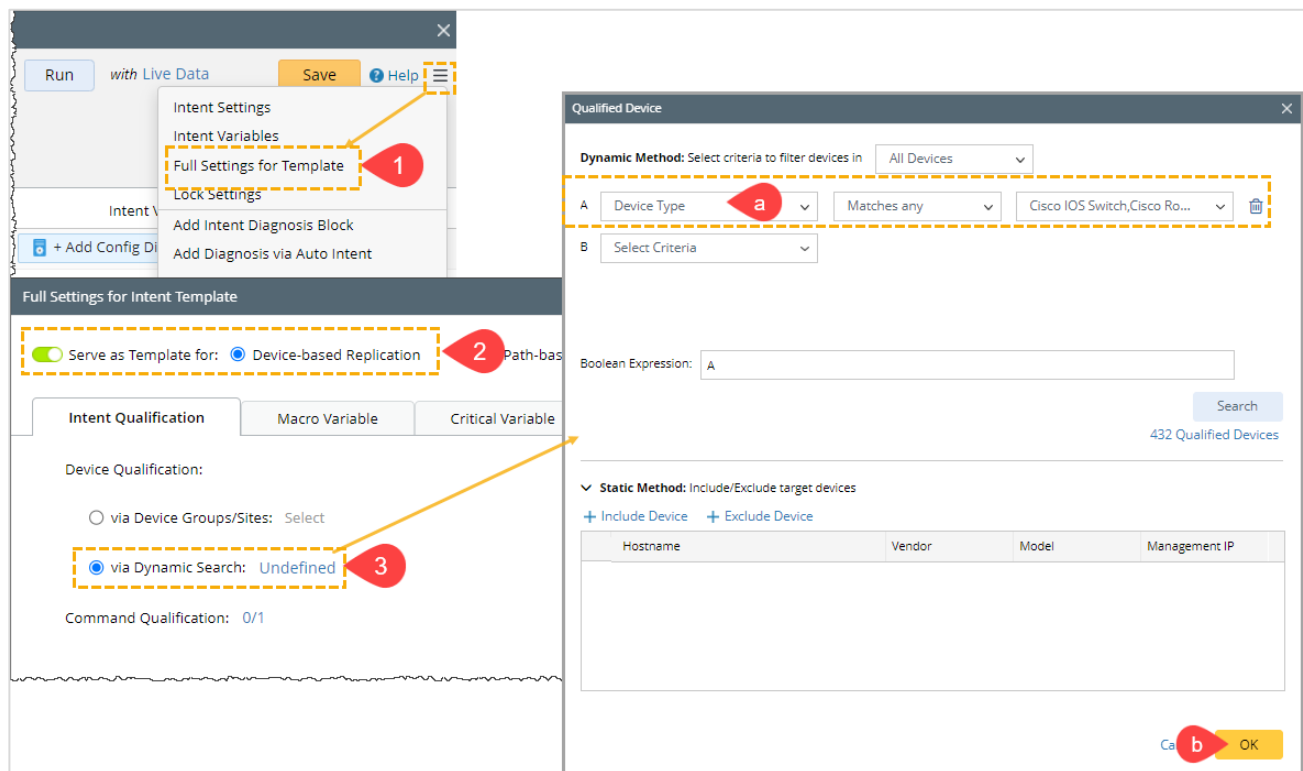
▶ Down_interfaces

US-BOS-R1		
 show interface 		
interface_name (string)	status (string)	input (string)
Loopback99	administratively down	never

5.3.1.3 Create Signature Variables

For any intent variable to be added to an ADT, you have to define it as a **Signature Variable** in the **Full Settings for Intent Template**. To create signature Variables, start by defining the **Intent Qualification** using devices and critical Variables.

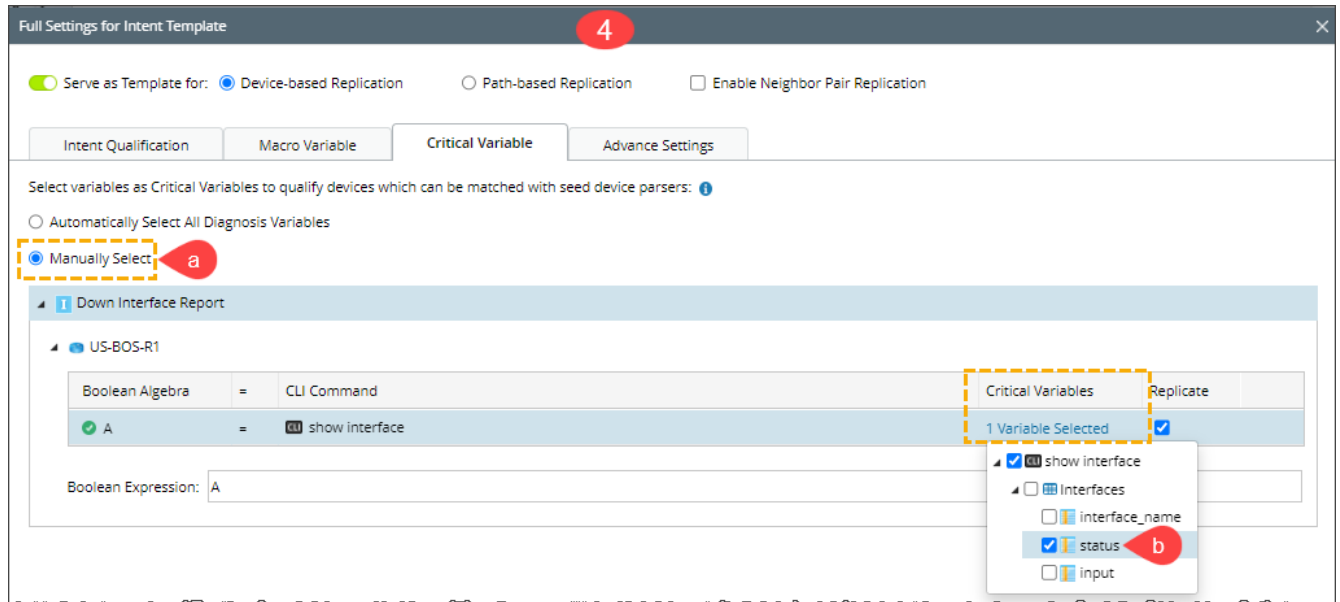
1. In the **Network Intent (Edit Mode)** window, go to  menu, and click **Full Settings for Template** to define device qualification and critical Variables.
2. Enable the toggle button, **Serve as Template for**, and ensure that **Device-based Replication** is selected.
3. Under the **Intent Qualification** tab, click **Undefined** to define the devices via **Dynamic Search**.
 - a) Set the **Device Type** to be **Cisco IOS Switch** and **Cisco Router**.
 - b) Click **OK** to save the configuration and close the window.



4. Define Critical Variable.

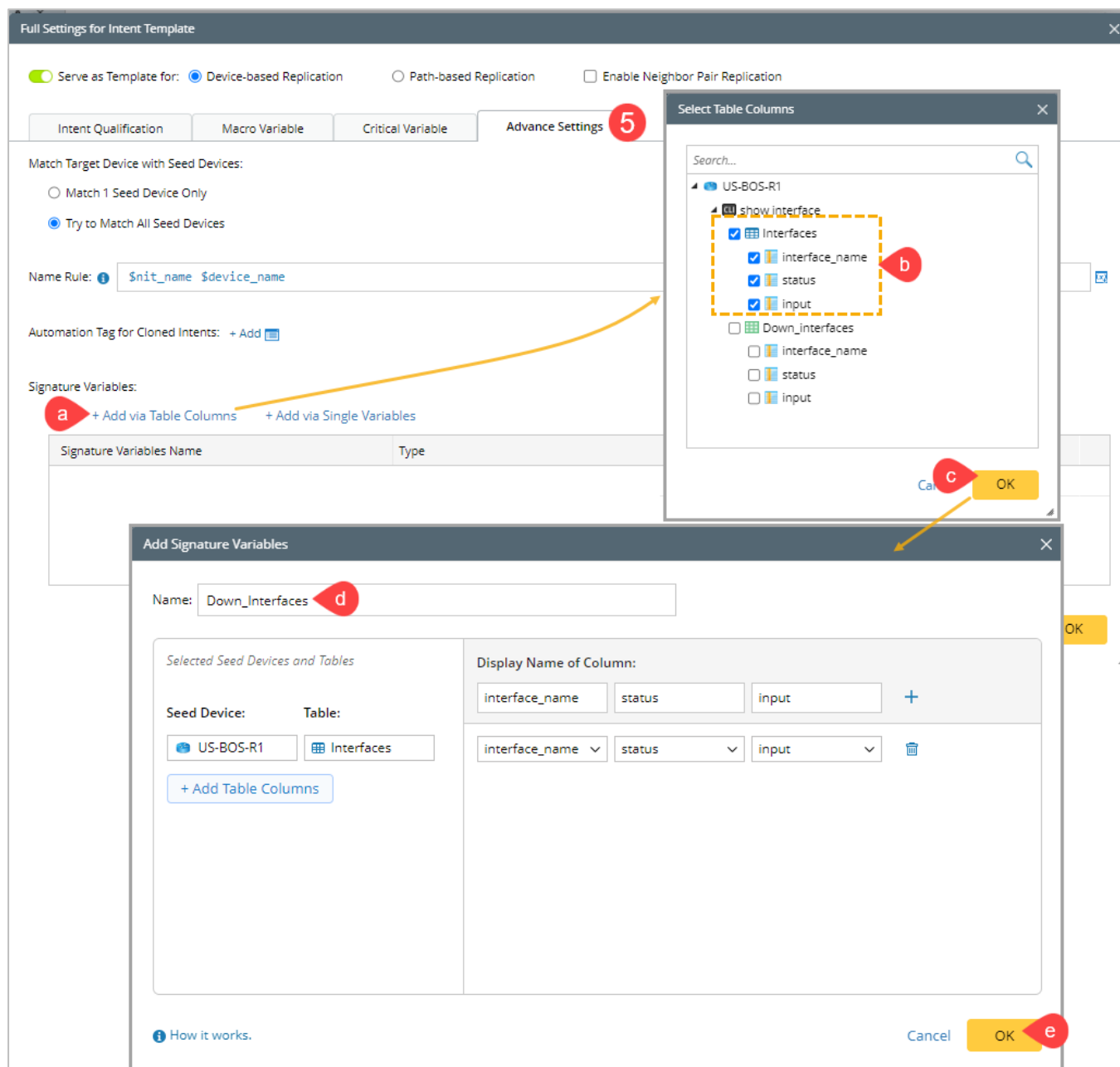
- a) Under the **Critical Variable** tab, click the **Manually Select** option.
- b) Under the **Critical Variable** column, select the Variable **status**.

Note: Select only the Variable **status**; otherwise, the ADT will include empty lines.



5. Create **Signature Variables** in **Advance Settings**.

- a) Under the **Signature Variables** section of the **Advanced Settings** tab, click **+ Add via Table Columns**.
- b) From the **Select Table Columns** window, check all the **Interfaces** Variables checkboxes.
- c) Click **OK** to open the **Add Signature Variable** window.
- d) In the **Add Signature Variables** window, provide the name **Down_interface**.
- e) Click **OK** to save the table columns.



6. Click **OK** to save the Intent Template settings.
7. Click **Save** to save the NI and then close the window.

5.3.2 Add Signature Variables to ADT Base Table

In this section, you will create a new base table in ADT using a **Pre-replicated Network Intent Template** method for the inclusion of Signature Variables.

Note: The Signature Variables can only be used in the base table of ADT and when the method to build the base table is the **Pre-replicated Intent Template**.

A Signature Variable is a specialized type of Variable intended to reveal key Variable values used in parsers for NIT replication. It can hold two types of values: single value and table. This Variable type enables data output to an ADT as row data.

5.3.2.1 Decode Network Intent

Before an intent template can be used to create a base table of an ADT, you must install and decode it in the **IBA Center**:

The screenshot displays the NetBrain web interface, specifically the 'Intent-Based Automation' section. The 'Intent Based Automation Center' is highlighted with a red circle '1'. Below it, the 'Installed Intent Templates' tab is active, and the '+ Add Intent Template' button is highlighted with a red circle '2'. A modal window titled 'Add Intent Template' is open, showing the 'Intent Template' field with a 'Browse' button highlighted by a red circle '3'. Another modal window titled 'Select Intent Template' is open, showing a list of templates under the 'Sachin' category. The 'Down Interface Report' template is highlighted with a red circle '4'.

NetBrain

Search App..

Domain Management

Network | Map-Based Automation | **Intent-Based Automation** | Incident & Change | Misc

Intent

- Intent Manager
- Intent Cluster Manager
- Automation Data Table (ADT)
- Task Variable Manager

Execute Intent

- Intent Based Automation Center**
- Triggered Automation Manager
- Preventive Automation Manager
- Schedule Automation

Dashboard

- Summary Dashboard
- Intent Dashboard
- Universal Dashboard
- Report Manager

Intent Based Automation Center

Installed Intent Templates | Published Intents | Auto Intent | Auto Intent Profile | NetBrain Download

Items: **+ Add Intent Template**

Intent Template Name	Location	Intent Decoding	Deco
0-BGP Demo			

Add Intent Template

Intent Template: **Browse**

Group:

Decoding Settings

Cancel OK

Select Intent Template


Select Intent Template from: ☒ All Intents ☐ Installed Intents

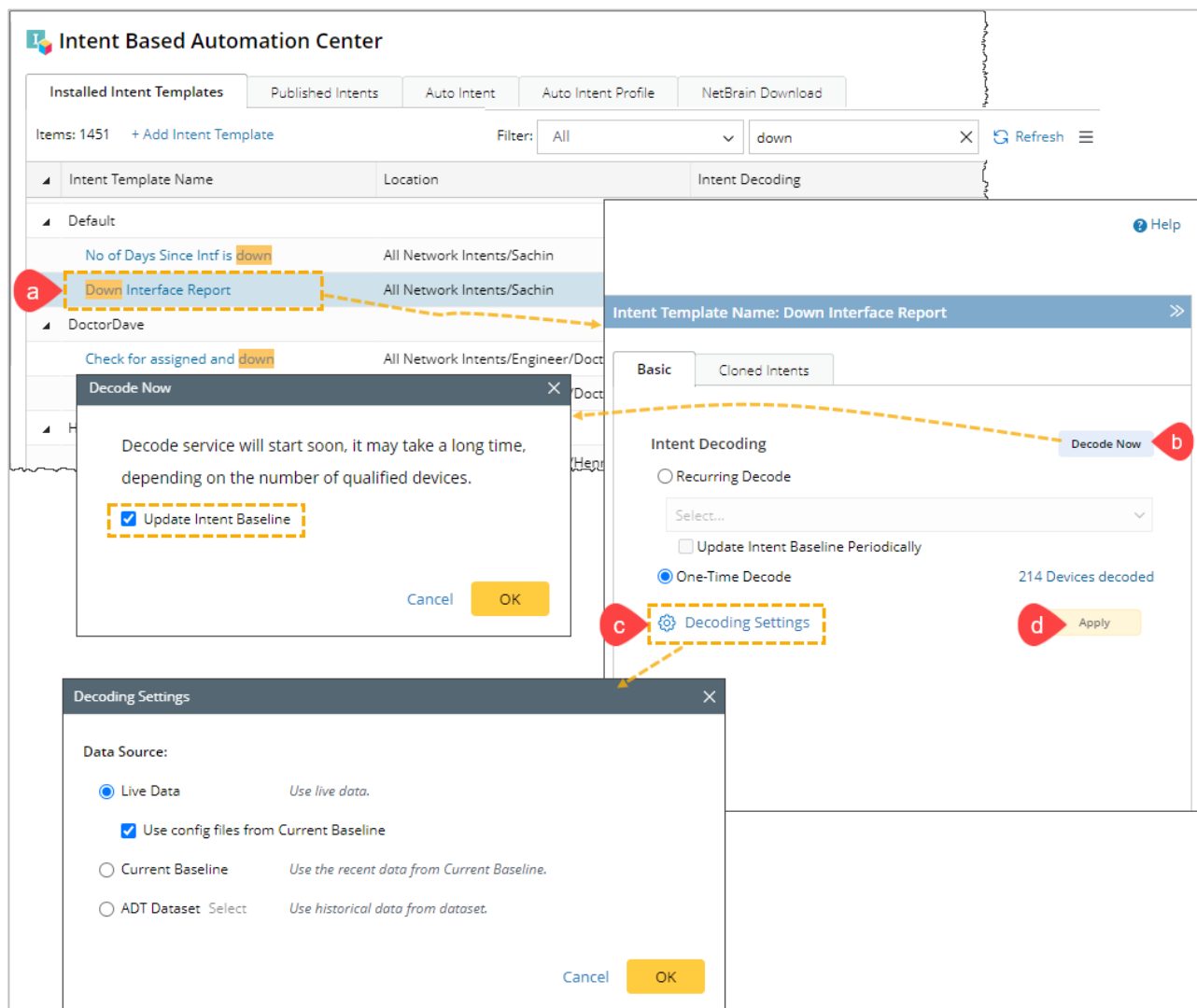
Type: Filter by:

All | 10.10.10.1 (6) | 192.168.29.62 (4) | 3Com (5) | 5 Report (2) | AAA (5) | Acces >>

- All Network Intents
 - Sachin**
 - NIST Compliance
 - Check AWS EC2 Configuration Against Baseline
 - CP1
 - CPU Performance Check
 - Demo-OSPF Router
 - Down Interface Report**

Cancel OK

1. Click the start  menu and select **Intent Based Automation Center** from the **Intent-Based Automation** tab.
2. Click **+ Add Intent Template** to add your intent.
3. From the **Add Intent Template** dialog, click **Browse** to select Intent Template from All Intent.
4. Search for your Intent Template, select it, and then click **OK**. Your Intent Template is installed in the IBA center.
5. Decode Intent Template.



- a) Select your Intent Template.
- b) In the right pane, click **Decode Now**. From the **Decode Now** dialog, tick the **Update Intent Baseline** checkbox and then click **OK**.

The message "Intent decoding request has been sent successfully." Is displayed.

- c) Click **Decoding Settings**, tick the **Use config files from Current Baseline** checkbox, and click **OK**.
- d) Click **Apply** to apply the settings.

It may take some time to decode the devices if there are many devices in the device group. The final results show that 215 devices are decoded.

Intent Based Automation Center

Installed Intent Templates

Published Intents

Auto Intent

Auto Intent Profile

NetBrain Download

Items: 1388

+ Add Intent Template

Filter: All


down interface

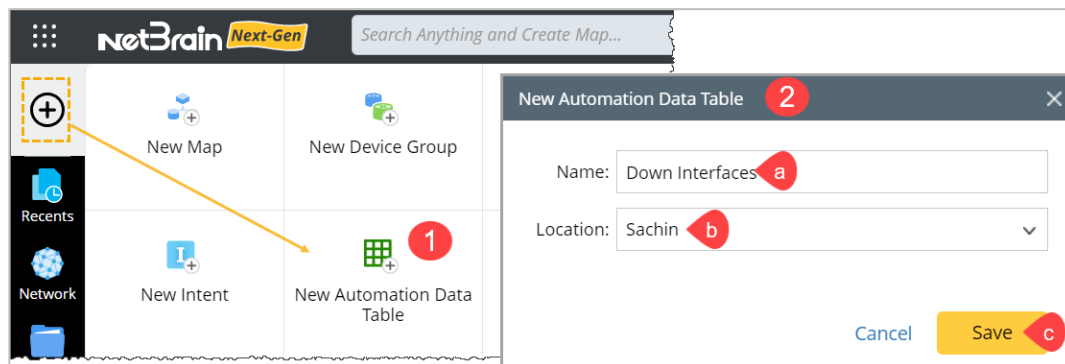
Refresh

Intent Template Name	Location	Intent Dec...	Decoded ...	Auto Inten...	Cloned Int
Default					
Down Interface Report	All Network Intents/Sachin	Last Deco...	215		0
DoctorDave					
Check for down interfaces	All Network Intents/Engineer/Doc...	Last Deco...	152		135

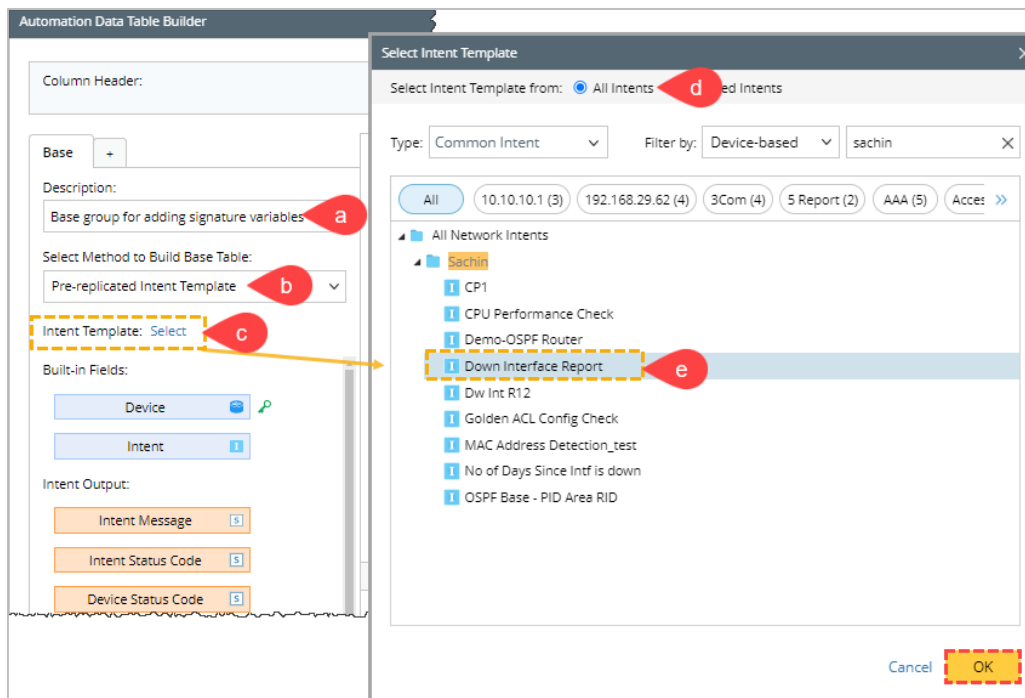
5.3.2.2 Build Base Table

Follow the step-by-step instructions to build the base table with the **Signature Variables**:

1. Click the plus icon  and click the **New Automation Data Table** option.
2. In the **New Automation Data Table** popup:
 - a) Enter ADT name, **Down Interfaces**.
 - b) Select the **Location** you wish to store the ADT.
 - c) Click **Save** and wait a moment for ADT to open.

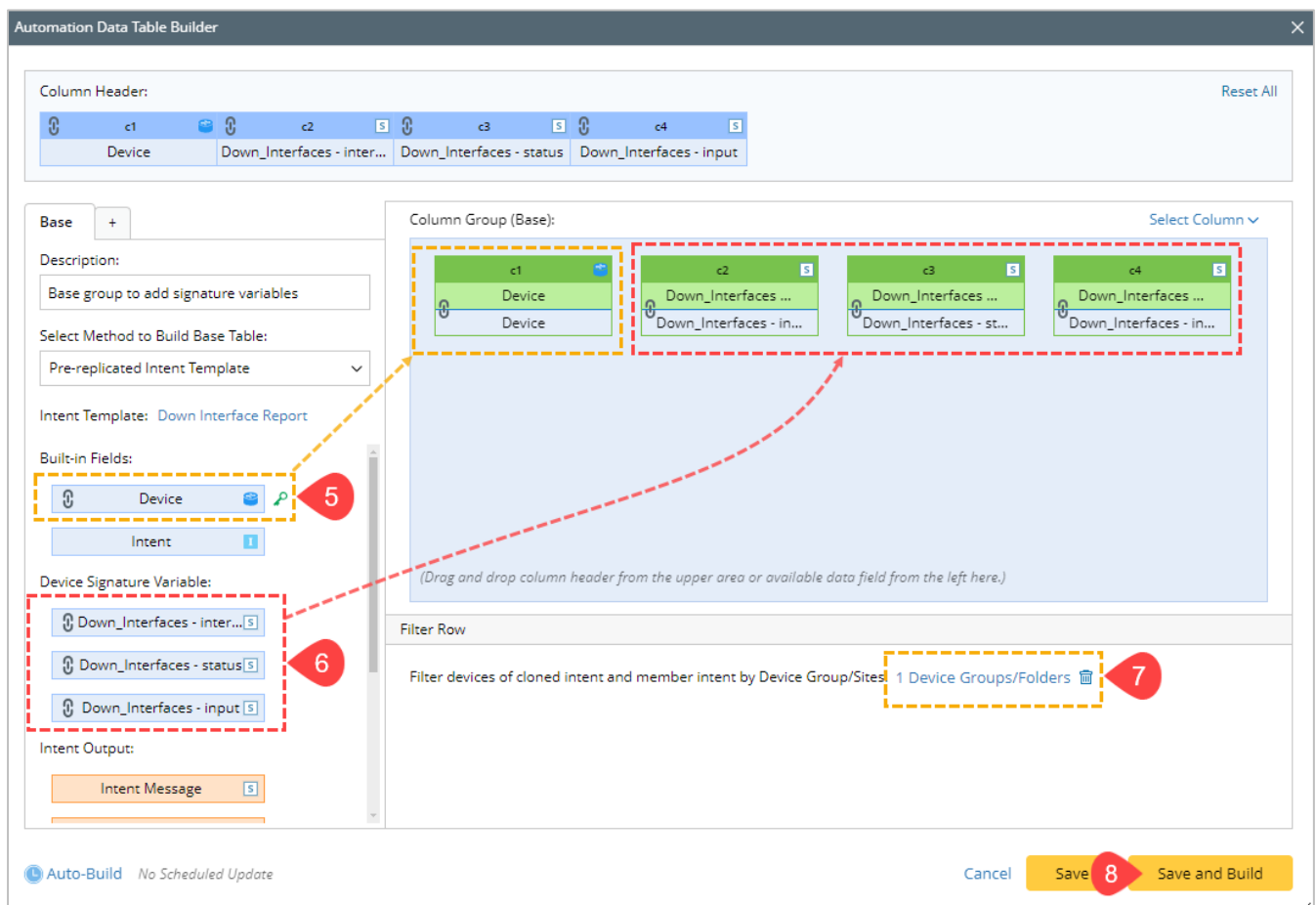


3. Click **Table Builder** to open **Automation Data Table Builder** to define the Base Group.
Base Group tab settings:
4. Under the **Base** tab of Automation Data Table Builder, define the following settings:

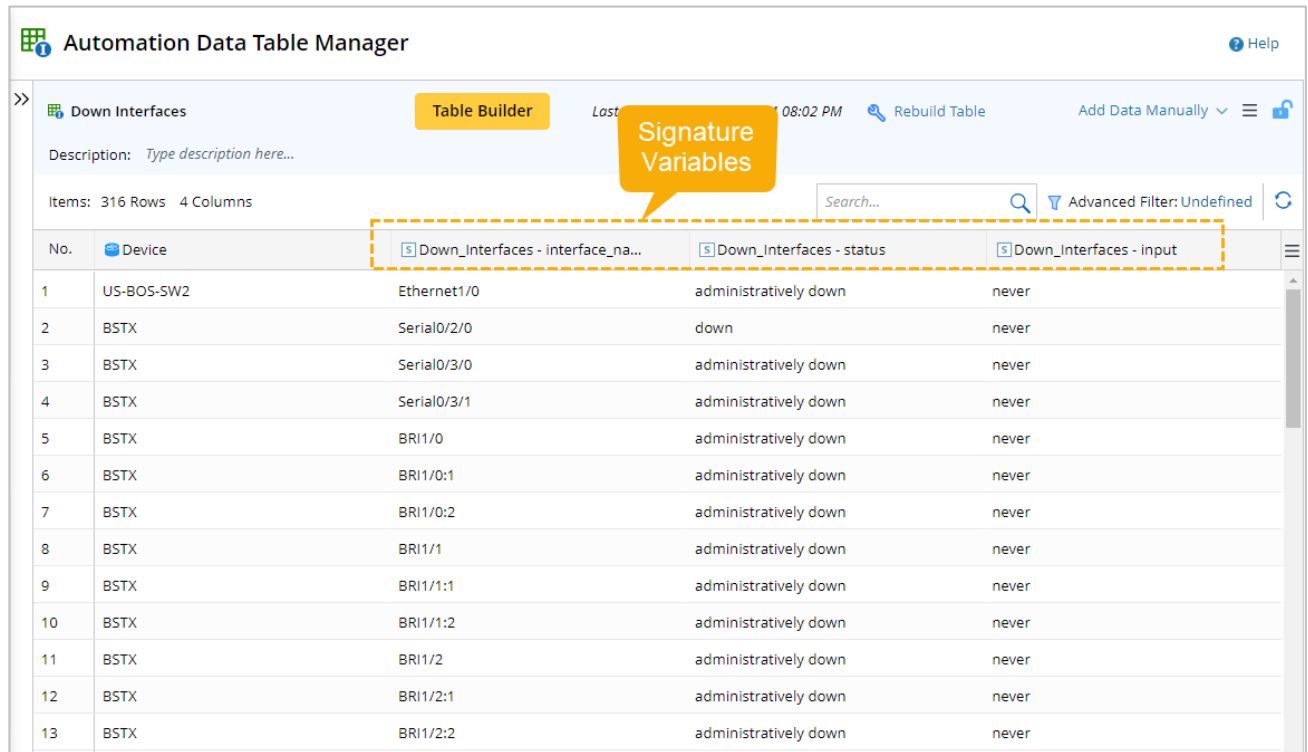


- Input descriptions for the base table to describe its use and function.
 - Make sure the method, **Pre-replicated Intent Template**, is selected.
 - Click the **Select** link to select the created NI (**Down Interface Report**) for building the base table with signature Variables.
 - Click **All Intents** to access your Intent.
 - Select your Intent and click **OK**.
- From the **Built-in Fields** section, drag the **Device** and **Intent** column into the **Column Group (Base)** pane.
 - From the **Device Signature Variable** section, drag all the Signature Variables one by one into the **Column Group (Base)** pane.
 - In the **Filter Row** pane, click **Select** to add a device group.
 - Click **Save and Build** to save all the base table settings.

In the **Build Table** dialog, select the column groups to be built, select a **Log** mode, and then click **Build** to save the changes.



You will see that the columns are populated with the data.



Automation Data Table Manager

Table Builder | Last Updated at: 08:02 PM | Rebuild Table | Add Data Manually

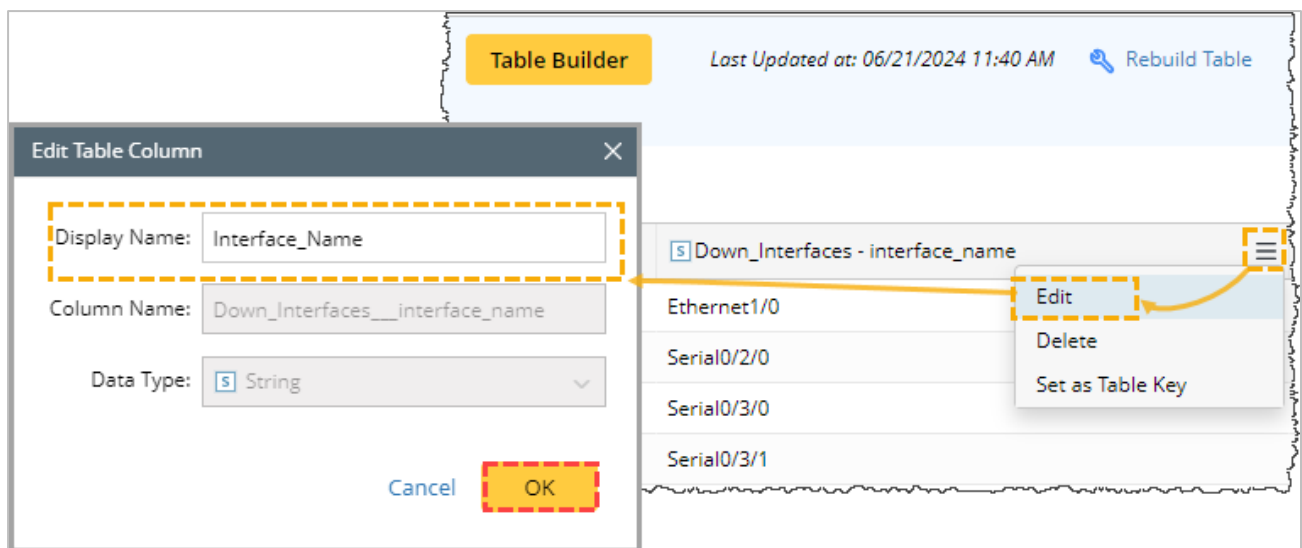
Description: Type description here...

Items: 316 Rows 4 Columns

Search... | Advanced Filter: Undefined


No.	Device	[S] Down_Interfaces - interface_na...	[S] Down_Interfaces - status	[S] Down_Interfaces - input
1	US-BOS-SW2	Ethernet1/0	administratively down	never
2	BSTX	Serial0/2/0	down	never
3	BSTX	Serial0/3/0	administratively down	never
4	BSTX	Serial0/3/1	administratively down	never
5	BSTX	BR11/0	administratively down	never
6	BSTX	BR11/0:1	administratively down	never
7	BSTX	BR11/0:2	administratively down	never
8	BSTX	BR11/1	administratively down	never
9	BSTX	BR11/1:1	administratively down	never
10	BSTX	BR11/1:2	administratively down	never
11	BSTX	BR11/2	administratively down	never
12	BSTX	BR11/2:1	administratively down	never
13	BSTX	BR11/2:2	administratively down	never

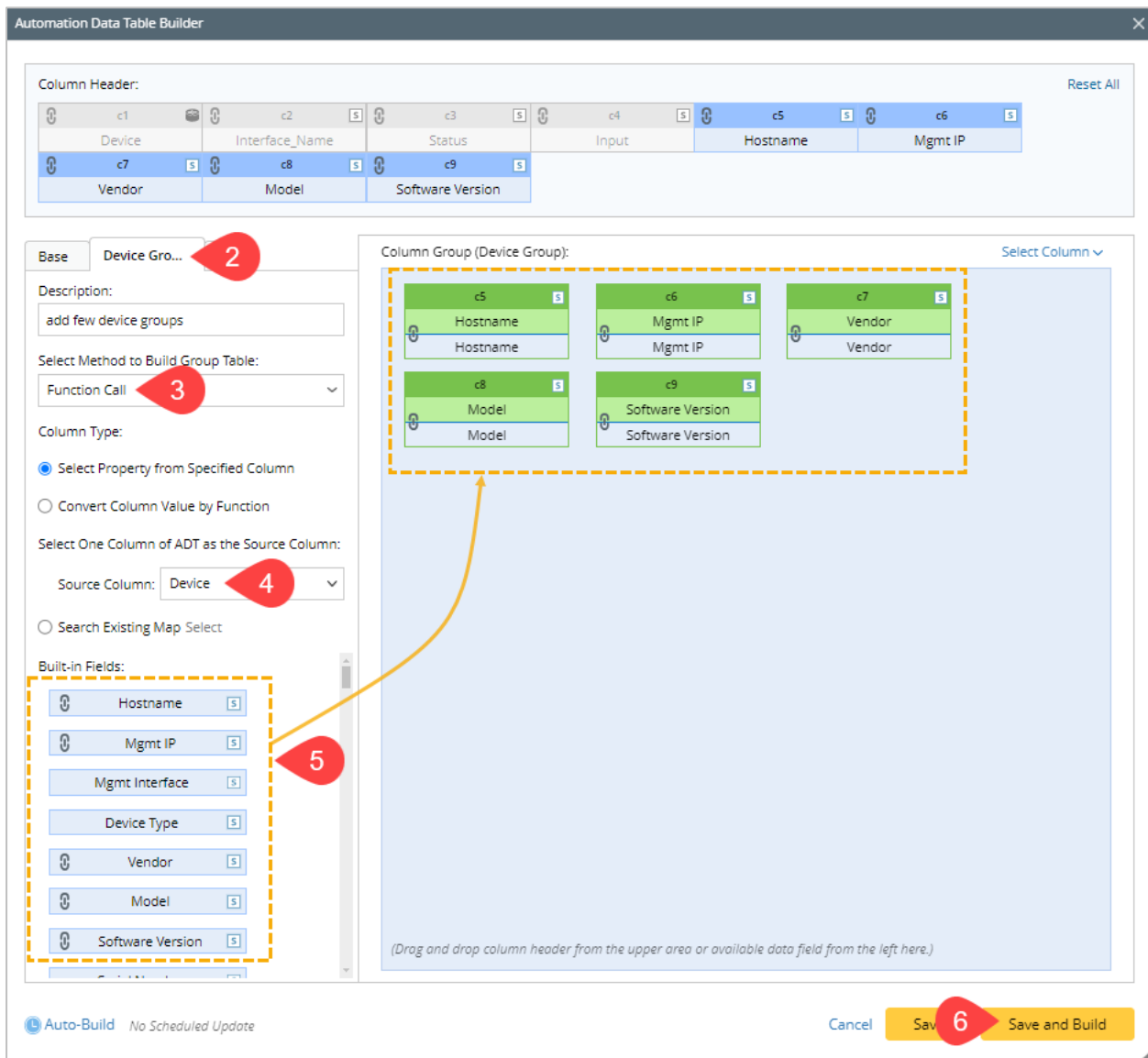
9. Edit the Table Column to display the specific name of the Signature Variables. Repeat this step to rename two more Signature Variables columns.



5.3.2.3 Add Device Property Columns in ADT

Use the function call method to add **Device Property** in ADT.

1. Click **Table Builder** to open the **Automation Data Table Builder**.
2. Click  to add a new tab to add the **Device group**.
3. From the **Select Method to Build Group Table** dropdown, select the **Function Call** method.
4. From the **Source Column** dropdown, select **Device**.
5. From the **Built-in Fields** section, drag and drop **Hostname**, **Mgmt IP**, **Vendor**, **Model**, and **Software Version** into the **Column Group (Device Type)** pane.
6. Click **Save and Build** to add device property columns in ADT.



The screenshot shows the 'Automation Data Table Builder' window. At the top, the 'Column Header' section displays a table with columns c1 through c6. Below this, the 'Base' tab is selected, and the 'Device Group' tab is highlighted with a red circle 2. The 'Description' field contains 'add few device groups'. The 'Select Method to Build Group Table' dropdown is set to 'Function Call' (red circle 3). The 'Column Type' section has 'Select Property from Specified Column' selected. The 'Select One Column of ADT as the Source Column:' section has 'Device' selected in the 'Source Column' dropdown (red circle 4). The 'Built-in Fields' section on the left lists various fields, with 'Hostname', 'Mgmt IP', 'Vendor', 'Model', and 'Software Version' highlighted by a dashed orange box and a red circle 5. An arrow points from this box to the 'Column Group (Device Group)' pane on the right, which contains a dashed orange box with five green cards representing these fields. The 'Save and Build' button at the bottom right is highlighted with a red circle 6.

c1	c2	c3	c4	c5	c6
Device	Interface_Name	Status	Input	Hostname	Mgmt IP
c7	c8	c9			
Vendor	Model	Software Version			

Column Header: Reset All

Base **Device Group** 2

Description: add few device groups

Select Method to Build Group Table: Function Call 3

Column Type:

- ☒ Select Property from Specified Column
- ☐ Convert Column Value by Function

Select One Column of ADT as the Source Column:

Source Column: Device 4

☐ Search Existing Map Select

Built-in Fields:

- ☒ Hostname 5
- ☒ Mgmt IP 5
- ☒ Mgmt Interface 5
- ☒ Device Type 5
- ☒ Vendor 5
- ☒ Model 5
- ☒ Software Version 5

Column Group (Device Group): Select Column v

(Drag and drop column header from the upper area or available data field from the left here.)

Auto-Build No Scheduled Update Cancel Save 6 Save and Build

You will see the device group columns are populated with the data.

Automation Data Table Manager

Help

>>Down Interfaces

Table Builder

Last Updated at: 06/21/2024 08:02 PM

Rebuild Table

Device Group Columns

Add Data Manually

Description: Type description here...

Items: 316 Rows 9 Columns

Search...

Advanced Filter: Undefined

No.	Device	Interface_Name	Status	Input	Hostname	Mgmt IP	Vendor	Model
1	US-BOS-SW2	Ethernet1/0	administratively down	never	US-BOS-SW2	10.8.1.242	Cisco	3560E
2	BSTX	Serial0/2/0	down	never	BSTX	172.24.32.209	Cisco	2811
3	BSTX	Serial0/3/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811
4	BSTX	Serial0/3/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811
5	BSTX	BRI1/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811
6	BSTX	BRI1/0:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811
7	BSTX	BRI1/0:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811
8	BSTX	BRI1/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811
9	BSTX	BRI1/1:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811
10	BSTX	BRI1/1:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811
11	BSTX	BRI1/2	administratively down	never	BSTX	172.24.32.209	Cisco	2811
12	BSTX	BRI1/2:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811
13	BSTX	BRI1/2:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811

5.3.3 Export ADT to CSV file

ADT can be exported as a CSV file for documentation.

The screenshot shows the 'Table Builder' interface. At the top, it says 'Last Updated at: 06/21/2024 08:02 PM' and has a 'Rebuild Table' button. A search bar is present. The table has columns: 'me', 'Status', 'Input', 'Hostname', and 'Mgn'. A dropdown menu is open, showing options: 'Import from System Table', 'Lock Settings', 'Execution log', 'Export', 'Export to CSV Only' (highlighted with a dashed orange box), 'Export Datasets to File', 'Dataset Tag Settings', and 'Table Settings'.

me	Status	Input	Hostname	Mgn
	administratively down	never	US-BOS-SW2	10.8.1...
	down	never	BSTX	172.24...
	administratively down	never	BSTX	172.24...
	administratively down	never	BSTX	172.24.32.209
	administratively down	never	BSTX	172.24.32.209

The exported CSV file will be saved automatically to your computer's default download location, typically the "Downloads" folder (e.g., **C:\Users<your username>\Downloads**).

1	Device	Interface_Name	Status	Input	Hostname	Mgmt IP	Vendor	Model	Software Version
2	US-BOS-S	Ethernet1/0	administratively down	never	US-BOS-SW2	10.8.1.242	Cisco	3560E	15.2(HI_20170202)FLO_DS
3	BSTX	Serial0/2/0	down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
4	BSTX	Serial0/3/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
5	BSTX	Serial0/3/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
6	BSTX	BRI1/0	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
7	BSTX	BRI1/0:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
8	BSTX	BRI1/0:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
9	BSTX	BRI1/1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
10	BSTX	BRI1/1:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
11	BSTX	BRI1/1:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
12	BSTX	BRI1/2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
13	BSTX	BRI1/2:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
14	BSTX	BRI1/2:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
15	BSTX	BRI1/3	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
16	BSTX	BRI1/3:1	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
17	BSTX	BRI1/3:2	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4
18	BSTX	BRI1/4	administratively down	never	BSTX	172.24.32.209	Cisco	2811	12.4(15)T4

PART 2 - Intent Based Automation Usecase Study

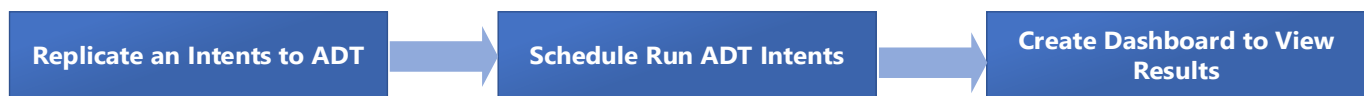
6 Network Assessment Case Study: Security Assessment

Network security is the protection of the underlying networking infrastructure from unauthorized access. The National Institute of Standards and Technology (**NIST**) publishes a set of configuration standards for security, and CVE shares vendor-reported vulnerability and exposure. The intent-based automation provides an ideal platform to enforce network security standards and best practices.

In this section, you will learn how to check NIST compliance against your network devices. You will create Intents to check the configurations such as VTY, AAA, and password policy against the NIST standard. The Base ADT will be created via the method **Devices of Device group** you learned in [Section 5 Prerequisites](#). A Summary Dashboard will be created to show the NIST compliance for your network.

In the second part of this section, you will learn how to create Intent for CVE issues to check device potential security risk. You will create two different Intents to check if a new CVE is affecting certain devices and the other to show the CVEs that may affect a device.

The workflow of the security assessment will be as follows:

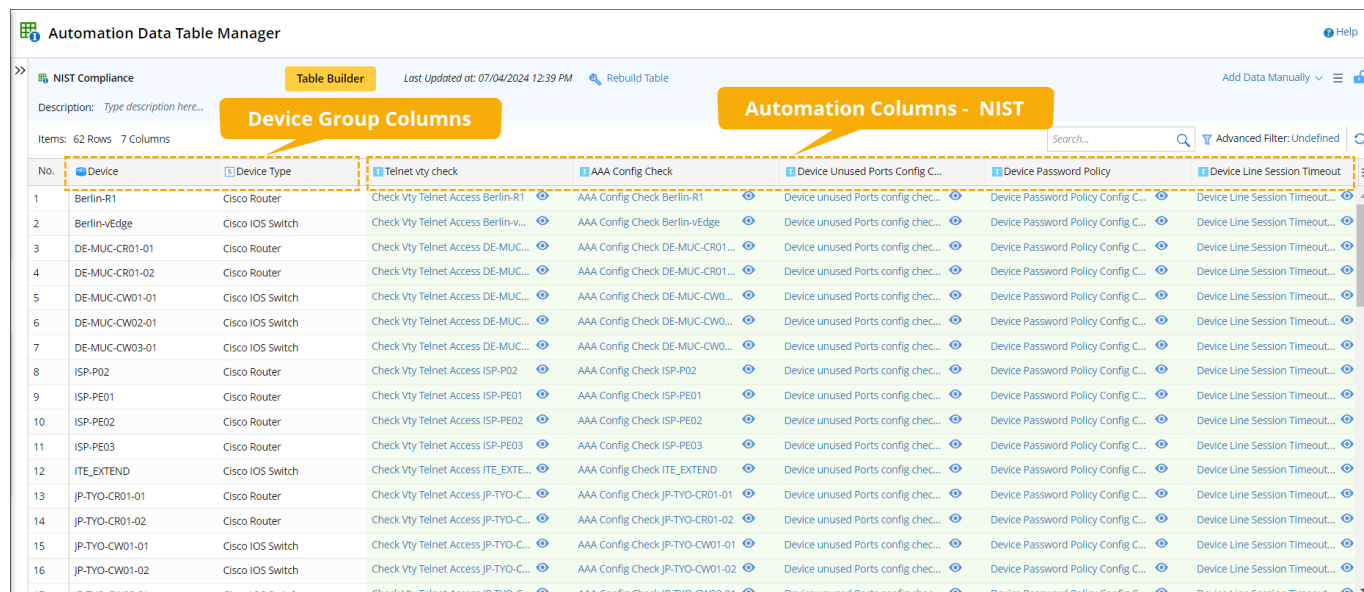


6.1 Check NIST Compliance and Vulnerability

You will create a dashboard to check the NIST Compliance for the following configurations:

1. Devices to allow Telnet (check Line VTY)
2. AAA Configuration
3. Unused ports but no shutdown
4. Device Password Policy (not encrypted, etc.)
5. Line Session Timeout

The final ADT will be:



Automation Data Table Manager

Table Builder Last Updated at: 07/04/2024 12:39 PM Rebuild Table

Description: Type description here...

Items: 62 Rows 7 Columns

Search... Advanced Filter: Undefined

No.	Device	Device Type	Telnet vty check	AAA Config Check	Device Unused Ports Config C...	Device Password Policy	Device Line Session Timeout
1	Berlin-R1	Cisco Router	Check Vty Telnet Access Berlin-R1	AAA Config Check Berlin-R1	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
2	Berlin-vEdge	Cisco IOS Switch	Check Vty Telnet Access Berlin-v...	AAA Config Check Berlin-vEdge	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
3	DE-MUC-CR01-01	Cisco Router	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CR01...	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
4	DE-MUC-CR01-02	Cisco Router	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CR01...	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
5	DE-MUC-CW01-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
6	DE-MUC-CW02-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
7	DE-MUC-CW03-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
8	ISP-P02	Cisco Router	Check Vty Telnet Access ISP-P02	AAA Config Check ISP-P02	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
9	ISP-PE01	Cisco Router	Check Vty Telnet Access ISP-PE01	AAA Config Check ISP-PE01	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
10	ISP-PE02	Cisco Router	Check Vty Telnet Access ISP-PE02	AAA Config Check ISP-PE02	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
11	ISP-PE03	Cisco Router	Check Vty Telnet Access ISP-PE03	AAA Config Check ISP-PE03	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
12	ITE_EXTEND	Cisco IOS Switch	Check Vty Telnet Access ITE_EXTEND	AAA Config Check ITE_EXTEND	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
13	JP-TYO-CR01-01	Cisco Router	Check Vty Telnet Access JP-TYO-C...	AAA Config Check JP-TYO-CR01-01	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
14	JP-TYO-CR01-02	Cisco Router	Check Vty Telnet Access JP-TYO-C...	AAA Config Check JP-TYO-CR01-02	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
15	JP-TYO-CW01-01	Cisco IOS Switch	Check Vty Telnet Access JP-TYO-C...	AAA Config Check JP-TYO-CW01-01	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...
16	JP-TYO-CW01-02	Cisco IOS Switch	Check Vty Telnet Access JP-TYO-C...	AAA Config Check JP-TYO-CW01-02	Device unused Ports config chec...	Device Password Policy Config C...	Device Line Session Timeout...

This section includes the following main steps:


- [Build ADT Base Table](#)
- [Create Network Intent: Check Vty Telnet Access](#)
- [Create NI: AAA Config Check](#)
- [Create NI: Device Unused Ports Config Check](#)
- [Create NI: Device Password Policy Config Check](#)
- [Create NI: Device Line Session Timeout](#)
- [Create Intent and Summary Dashboard](#)

6.1.1 Build ADT Base Table

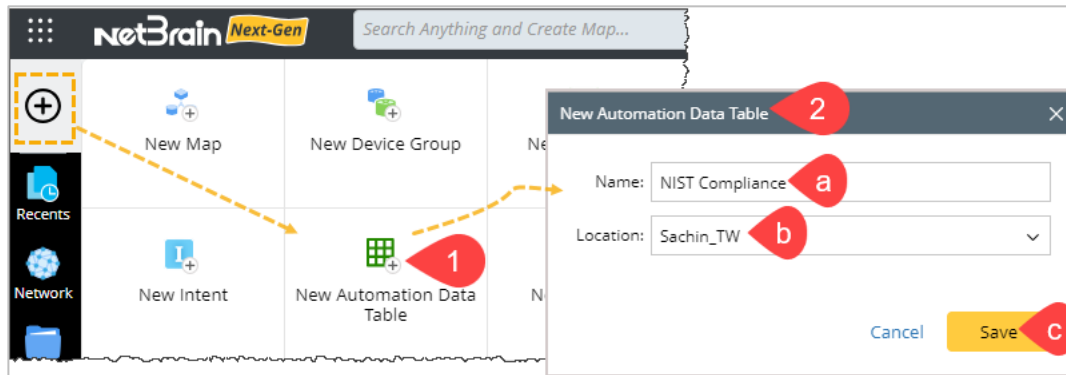
In the first step, you will create an ADT base table using the method **Devices of evic Group**. Make sure the Device Group is created based on your requirements.

6.1.1.1 Build Base ADT

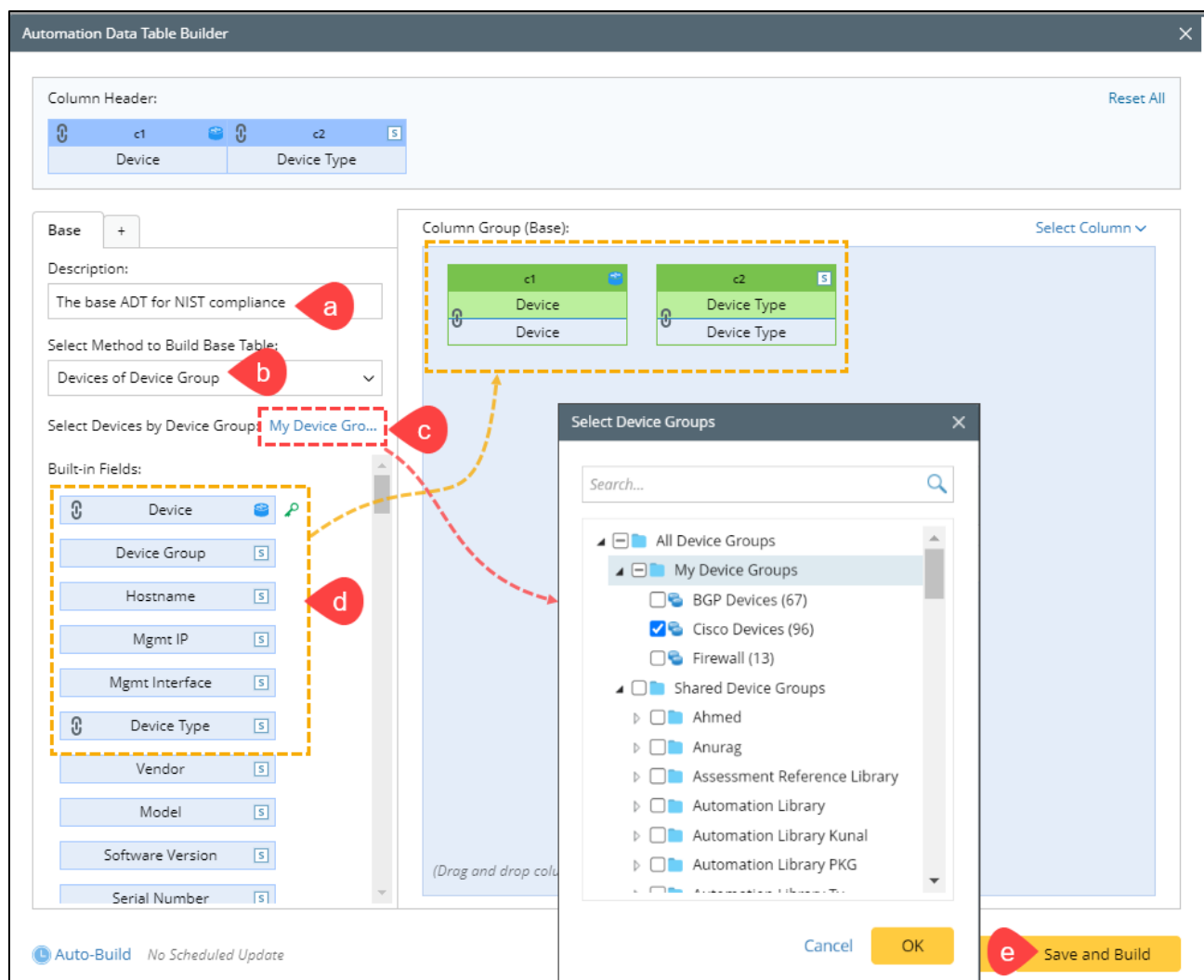
To build the base table with the data in devices, follow the steps below:

1. Click the plus icon  and click **New Automation Data Table**.
2. In the New Automation Data Table popup:
 - a) Enter the ADT Name.

- b) Select the **Location** you wish to store the ADT.
- c) Click **Save** and wait a moment for ADT to open.



3. Click **Table Builder** to open **Automation Data Table Builder** to define the Base Group.
4. Base Group tab settings.
5. Under the **Base** tab of Automation Data Table Builder, define the following settings:



- Input **Description** for the base table to describe its use and function.
- Select **Method**, the **Devices of Device Group** from the dropdown to build the base table.
- Click the **Select** link to select the created device group for building the base table with the devices in the device group, and then click **OK**.
- Within the **Built-in Fields** section, choose each column and move it into the **Column Group (base)** pane.

Build an ADT table with the following columns: **Device** and **Device Type**.

- Click **Save and Build**. The **Build** Table dialog appears, define the settings as per your preferences and then click **Build** to save all the settings.

The Base ADT will be like:

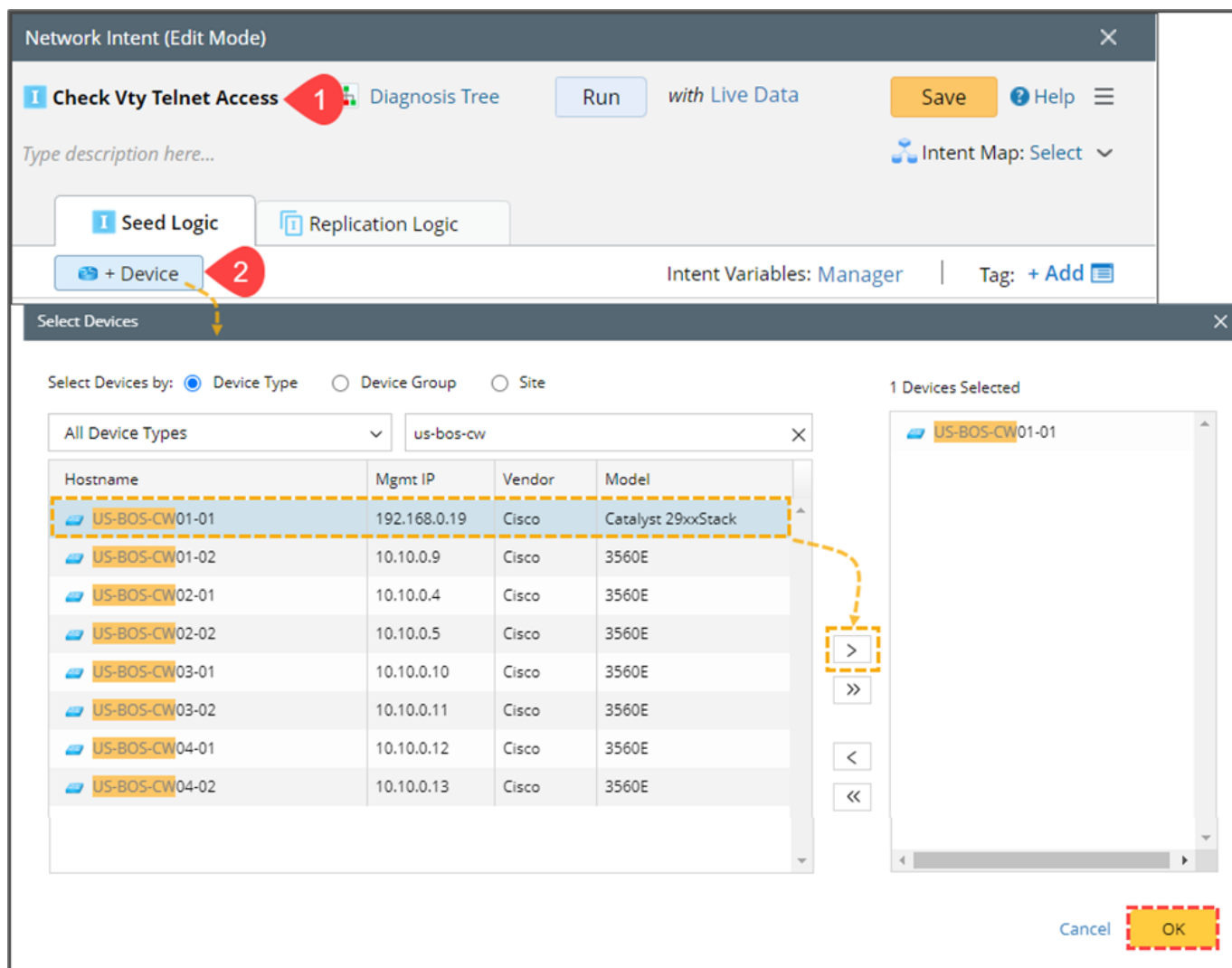
Automation Data Table Manager		
<div> <div>>></div> <div> <div>NIST Compliance</div> <div>Table Builder</div> <div>Last Updated at: 07/04/2024 12:39 PM</div> <div>Rebuild Table</div> </div> </div>		
Description: <i>Type description here...</i>		
Items: 62 Rows 2 Columns		
No.	Device	Device Type
1	Berlin-R1	Cisco Router
2	Berlin-vEdge	Cisco IOS Switch
3	DE-MUC-CR01-01	Cisco Router
4	DE-MUC-CR01-02	Cisco Router
5	DE-MUC-CW01-01	Cisco IOS Switch
6	DE-MUC-CW02-01	Cisco IOS Switch
7	DE-MUC-CW03-01	Cisco IOS Switch
8	ISP-P02	Cisco Router
9	ISP-PE01	Cisco Router
10	ISP-PE02	Cisco Router
11	ISP-PE03	Cisco Router
12	ITE_EXTEND	Cisco IOS Switch
13	JP-TYO-CR01-01	Cisco Router

6.1.2 Create Network Intent: Check Vty Telnet Access

In this section, you will create a Network Intent to check the VTY configurations (**Check Vty Telnet Access**) from the **Intent Manager** and use the NI in the **Intent Replication Wizard** to replicate the Intent to all the devices in the ADT table. The intent will check whether the telnet access is disabled and the SSH access is enabled in the VTY configurations.

Follow the step-by-step instructions to create an Intent.

1. Create a new intent from the **Intent Manager** and set the intent name (title) as **Check Vty Telnet Access**.
2. Add a new target device, such as device **US-BOS-R1**.



6.1.2.1 Add Config Diagnosis and Parse the VTY Configurations

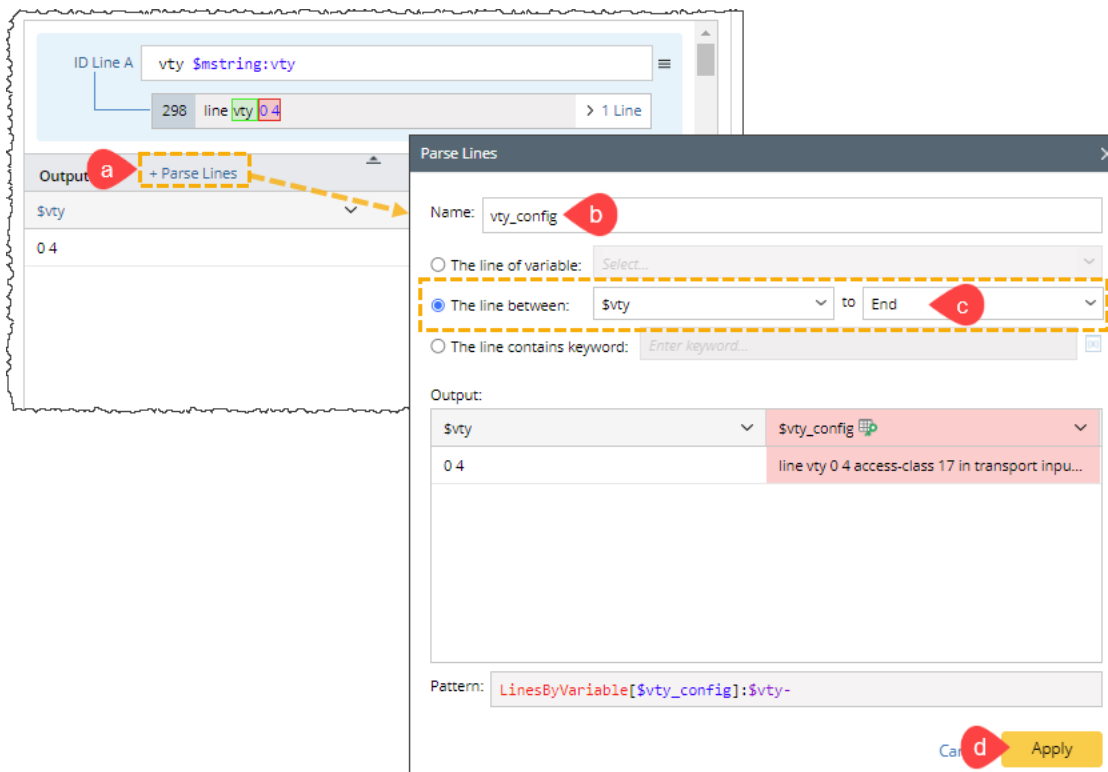
You will parse the VTY configurations with a Config Diagnosis:

1. Click **+ Add Config Diagnosis** to open the Configuration Diagnosis window.
2. Click **Retrieve** to retrieve the data from the **Live Data**.
3. The source data is displayed in the **Define Variable** pane. You can edit this data depending on the use case.
4. In the **+ New Pattern** dropdown, select the **Paragraph** parser to parse VTY configurations. The Paragraph parser is used since there can be multiple VTY lines.
5. Define **ID Line A** for the VTY configurations:
6. In the configurations, find the configurations of VTY. Double-click the vty number, for example, **0 4** of the sample data. The Variable in the ID Line A **vty \$mstring:vty** will be created.

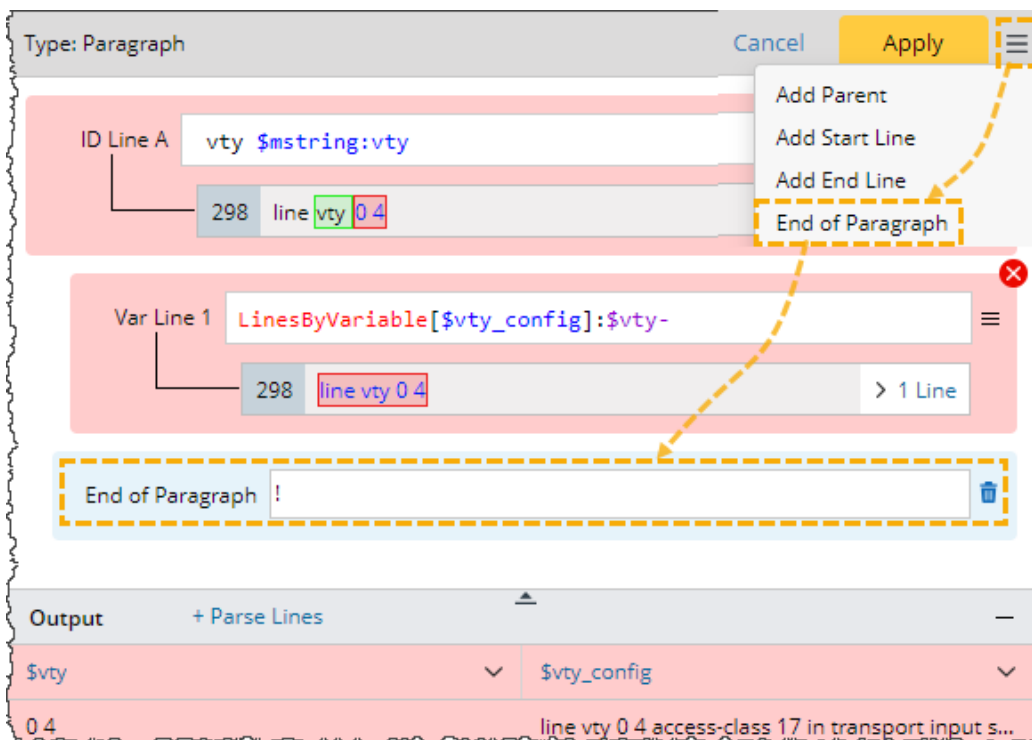
Check the Output.



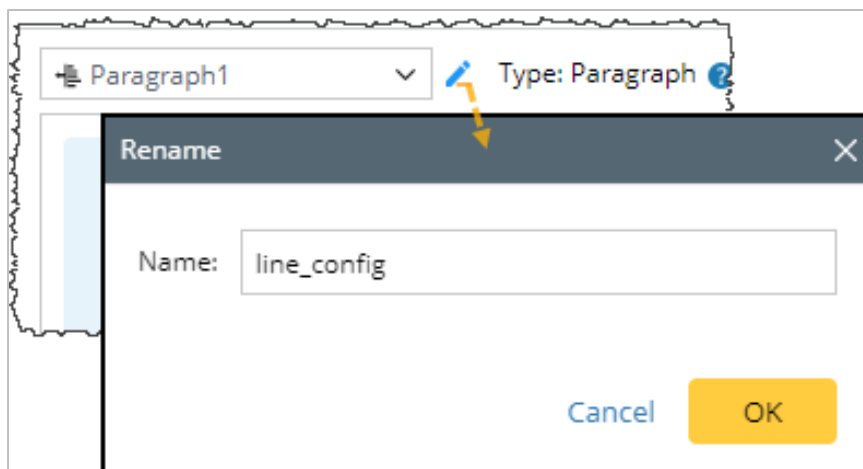
7. Define the configurations of a vty line as a variable. Here, we will use the function **LineByVariables**, which can parse multiple lines into one variable:
 - a) Click **+ Parse Lines**.
 - b) Enter Variable name, **vty_config**.
 - c) From **The line between**, parse the line between **\$vty** to the **End**.
 - d) Check the **Pattern** and click **Apply**.



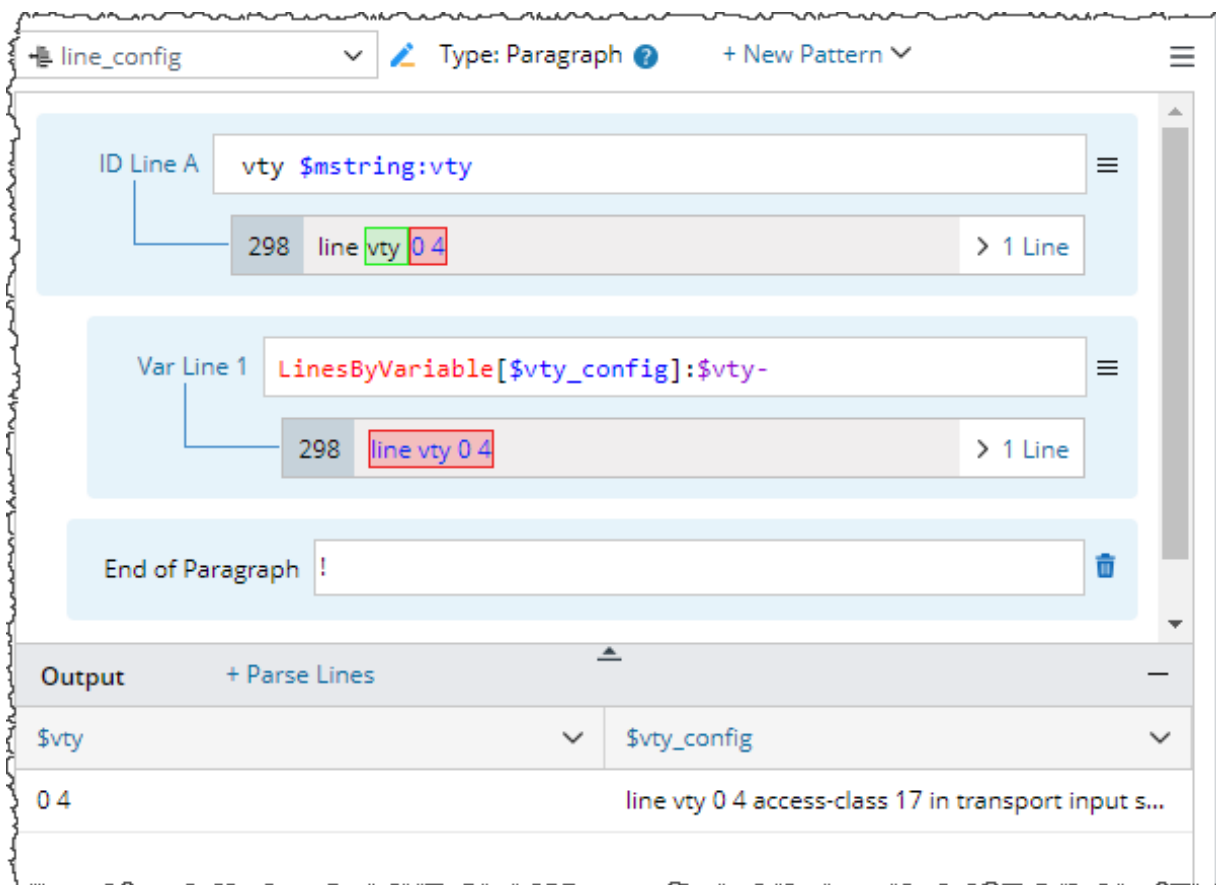
- e) To parse the whole vty configuration line, modify the pattern to ***LinesByVariable[\$vty_config]:\$vty-***
- f) Add the **End of Paragraph** using an exclamation mark (!) to parse the whole vty configuration line.



8. Verify the **Output** and click **Apply**.
9. In the **Confirmation** popup, click **Apply and Continue** to save the parser.
10. Click the pen icon, rename the parser name from **Paragraph1** to **line_config**, and then click **OK**.



The final Variables window will be:



6.1.2.2 Define Intent Variable for Seed Logic

In the diagnosis, you will loop through each VTY line (each row of the table **\$line_config**) and check whether the telnet access is allowed for each line. However, you do not want to create multiple alerts if the multiple VTY lines allow the telnet access. To create only one alert per device,

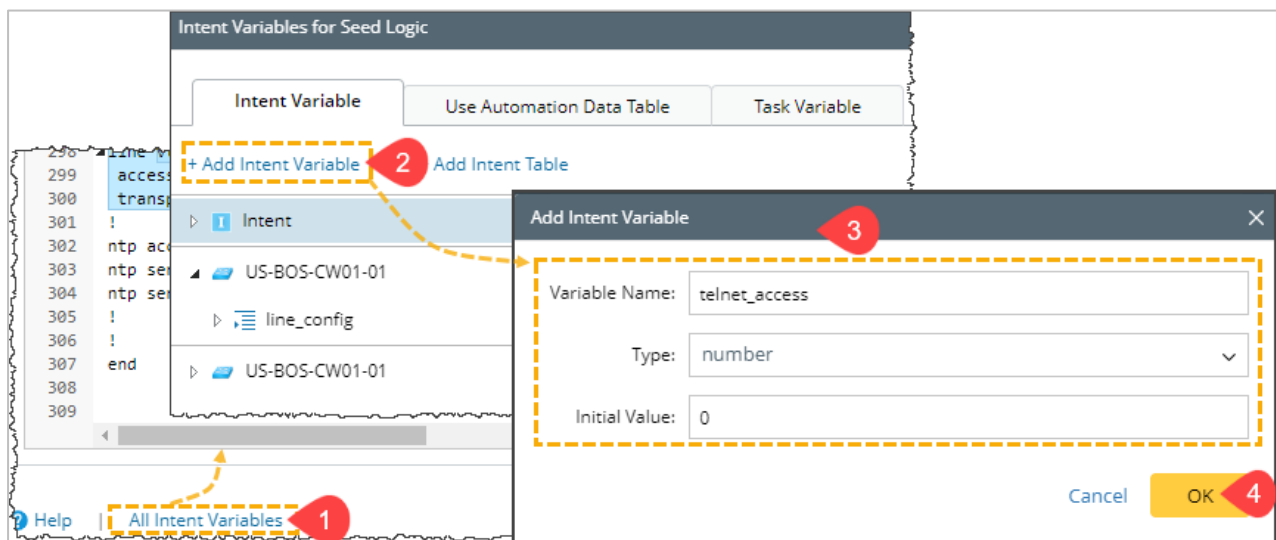
- you can define an intent variable, **\$telnet_access**, with the default value 0.
- Set this variable as **1** if the telnet access if the VTY line allows telnet access while you loop through the table.
- Then, outside the loop, check this intent variable and create an alert.

To add an intent variable:

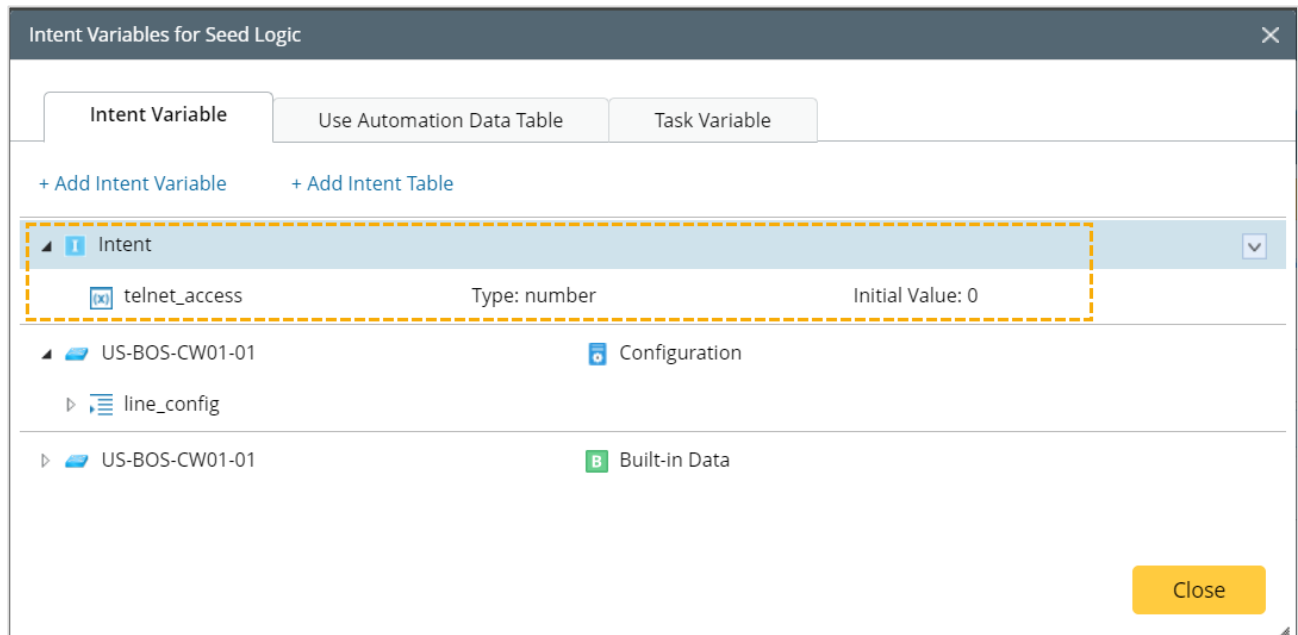
1. Click **All Intent Variables** in the bottom right corner.
2. Click the **Intent** section and then click **+ Add Intent Variable**.
3. In the **Add Intent Variable** popup fields, enter the following values:

Filed	Value
Variable Name	<i>telnet_access</i>
Type	number
Initial Value	0

4. Click **OK** to save the Intent Variable.



- Click **Close** to exit from the **Intent Variable for Seed Logic** window. The Intent Variable you just added is displayed.



6.1.2.3 Define Diagnosis

As explained earlier, we will define two diagnostics: **Check VTY telnet access** to loop through VTY lines and set the intent variable **\$telnet_access**, and **Create Alert** to set the status code according to the value of **\$telnet_access**.

Diagnosis 1: Check VTY telnet access

- Go to the **Define Diagnosis** section to define the diagnosis logic.
- Click **Add Diagnosis** to define conditions and Intent output message.
- Enter the diagnosis name, e.g., **Check VTY telnet access** and select an Anchor **\$vty_config** from the dropdown.
- Tick the **Loop Table Rows** checkbox and select the Table Variable (**line_config**) and the Table Key (**vty_config**) from the dropdown.

5. Define **If** condition as detailed in the image:

- a) Variable **vty_config** Contains **All**.
- b) Variable **vty_config** Contains **telnet**
- c) Variable **vty_config** Does not contain **ssh**
- d) Boolean Expression: **A or B or C**.

The screenshot shows the '2. Define Diagnosis' configuration window. It includes a header with a blue arrow icon and the title '2. Define Diagnosis'. Below the header, there are two buttons: 'Add Note' and 'Add Diagnosis'. The 'Add Diagnosis' button is highlighted with a red circle and the number 2. The main form area is divided into several sections. The first section is for the diagnosis name and anchor, with a red circle and the number 3 pointing to the 'Name' field. The second section is for the table configuration, with a red circle and the number 4 pointing to the 'Table Key' dropdown. The third section is the 'If' condition, with a red circle and the number 5 pointing to the 'Current' dropdown. The 'If' condition is expanded, showing three rows: A, B, and C. Each row has a variable dropdown, a comparison operator dropdown, and a value dropdown. Row A has 'vty_config', 'Contains', and 'All'. Row B has 'vty_config', 'Contains', and 'telnet'. Row C has 'vty_config', 'Does not contain', and 'ssh'. There is also a 'D' section for a Boolean Expression, which is set to 'A or B or C'. The 'Boolean Expression' field is highlighted with a red circle and the number d.

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automatic

Name: Check VTY telnet access Anchor: line_config.\$vty_con... ▾

Type description of the diagnosis...

☒ Loop Table Rows line_config ▾ Table Key: vty_config ▾

▽ If

A US-BOS-C... Current ▾

a vty_config ▾ Contains ▾ All ▾

B US-BOS-C... Current ▾

b vty_config ▾ Contains ▾ telnet ▾

C US-BOS-C... Current ▾

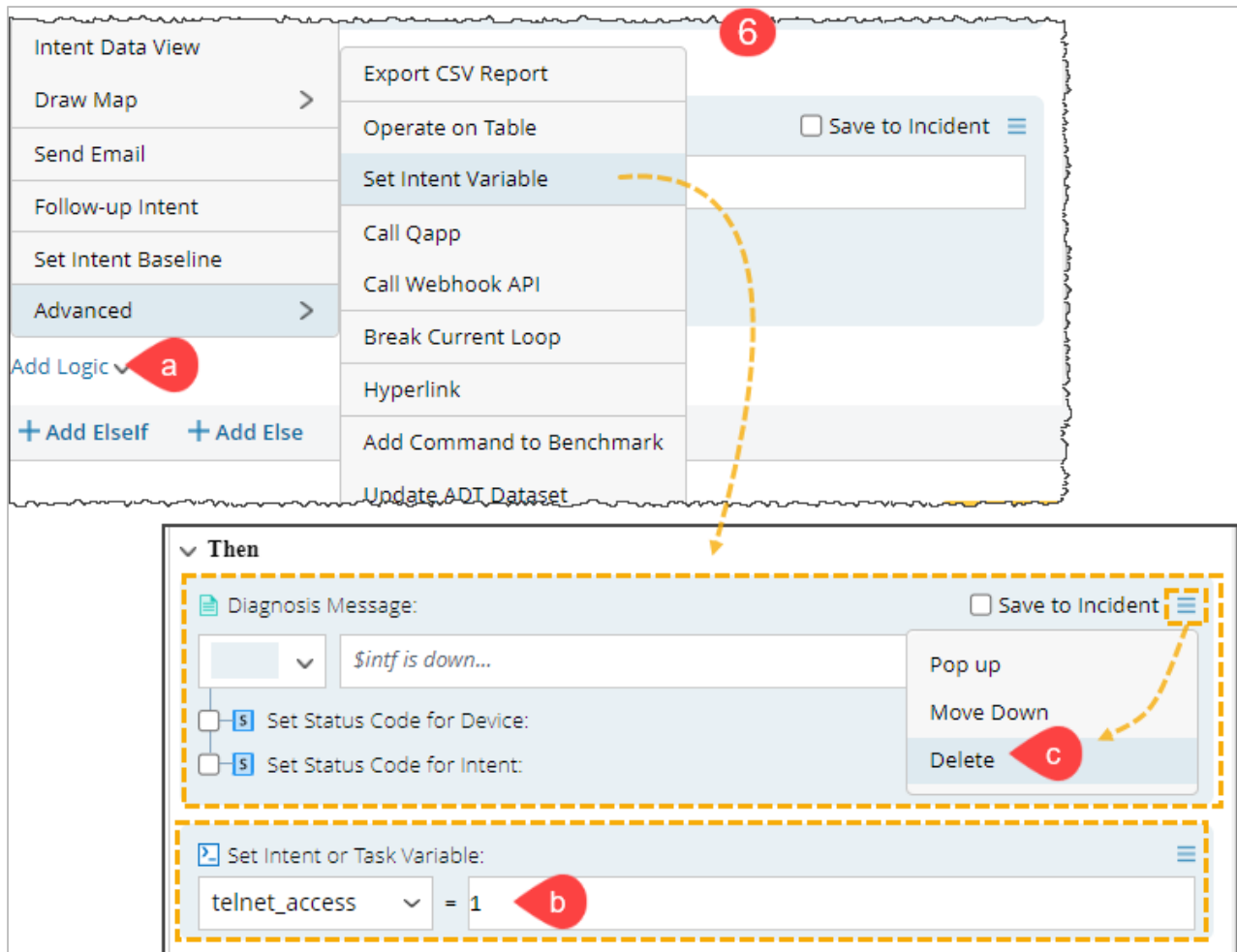
c vty_config ▾ Does not contain ▾ ssh ▾

D Select Variable ▾

d Boolean Expression: A or B or C

6. Set Intent Variable.

- In the **Add Logic** dropdown, select Advanced > Set Intent Variable.
- In the **Set Intent or Task Variable** dropdown, select **telnet_access** as Intent Variable and enter the value **1**.
- Delete the **Diagnosis Message** section.



Diagnosis 2: Create Alert

- Click **Add Diagnosis** again to define conditions and Intent output message.
- Enter the diagnosis name, e.g., **Create alert**.
- Define the **If** condition as detailed in the image: Intent Variable **telnet_access** Equals **1**.

2. Define Diagnosis

Can also click a variable on the left to add automation.

1 Add Diagnosis

Name: **2** Create alert Anchor: Select Variable

Type description of the diagnosis...

☐ Loop Table Rows

3 If

A Current

telnet_access Equals Example: 1

B Select Variable

4. Define Intent output: Enter a message under the **Then** and **Else** output areas to appear as the result of the diagnosis.

4

a Then

Diagnosis Message: ☐ Save to Incident

b ☒ Set Status Code for Device: Error \$this_device Telnet is enabled

b ☒ Set Status Code for Intent: Error \$this_device Telnet is enabled

Add Logic

c Else

Diagnosis Message: ☐ Save to Incident

d ☒ Set Status Code for Device: Success \$this_device Telnet is disabled

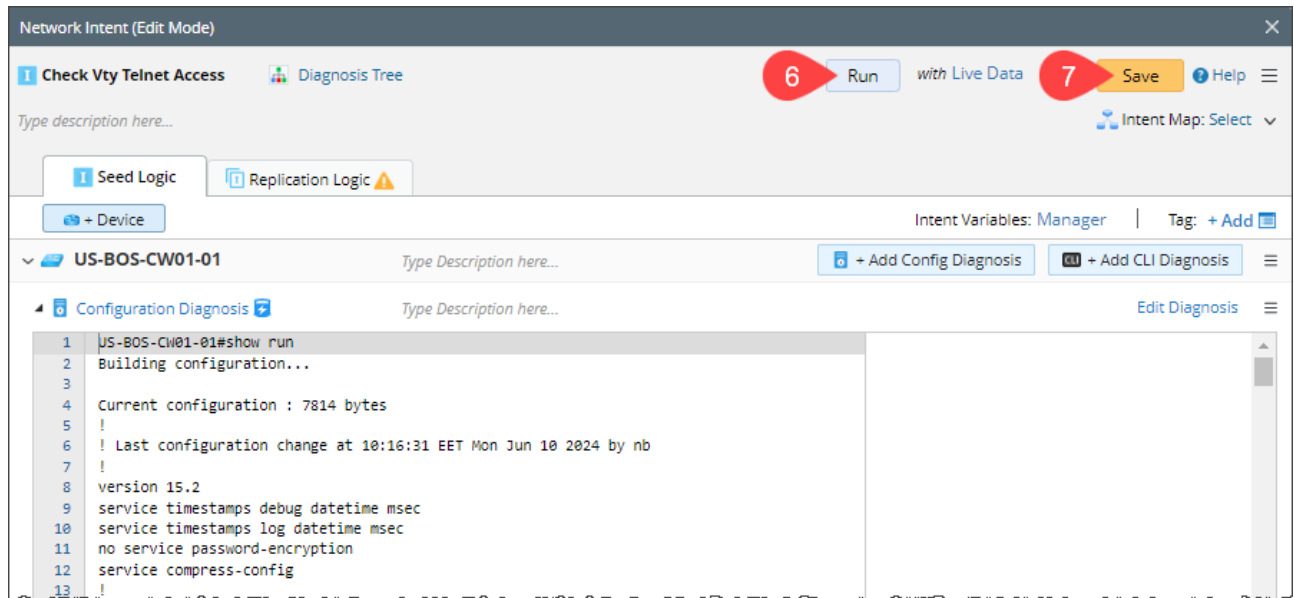
d ☒ Set Status Code for Intent: Success \$this_device Telnet is disabled

Add Logic

+ Add Elself


Cancel **5** Apply

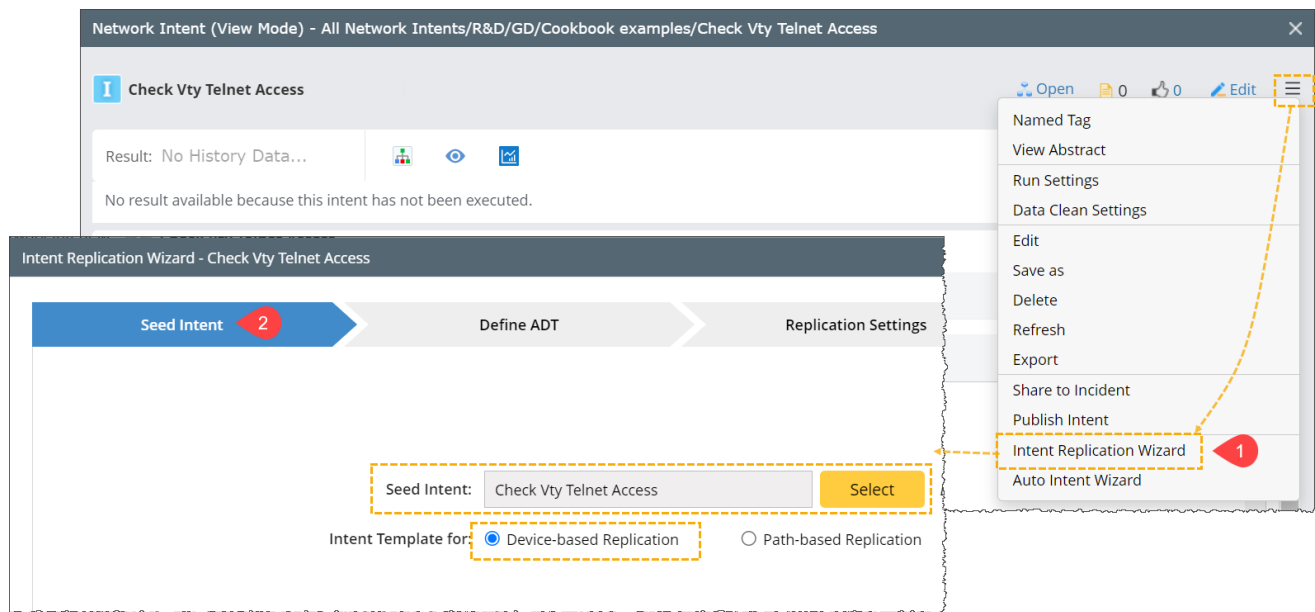
- Click **Apply** to save all the diagnosis settings and then close the **Configuration Diagnosis** window.
- In the **Network Intent (Edit Mode) window**, click **Run** to execute the Intent and check whether it is working as per your configuration settings.
- Click **Save** to save the Network Intent successfully and then close the window.



6.1.2.4 Use Intent Replication Wizard

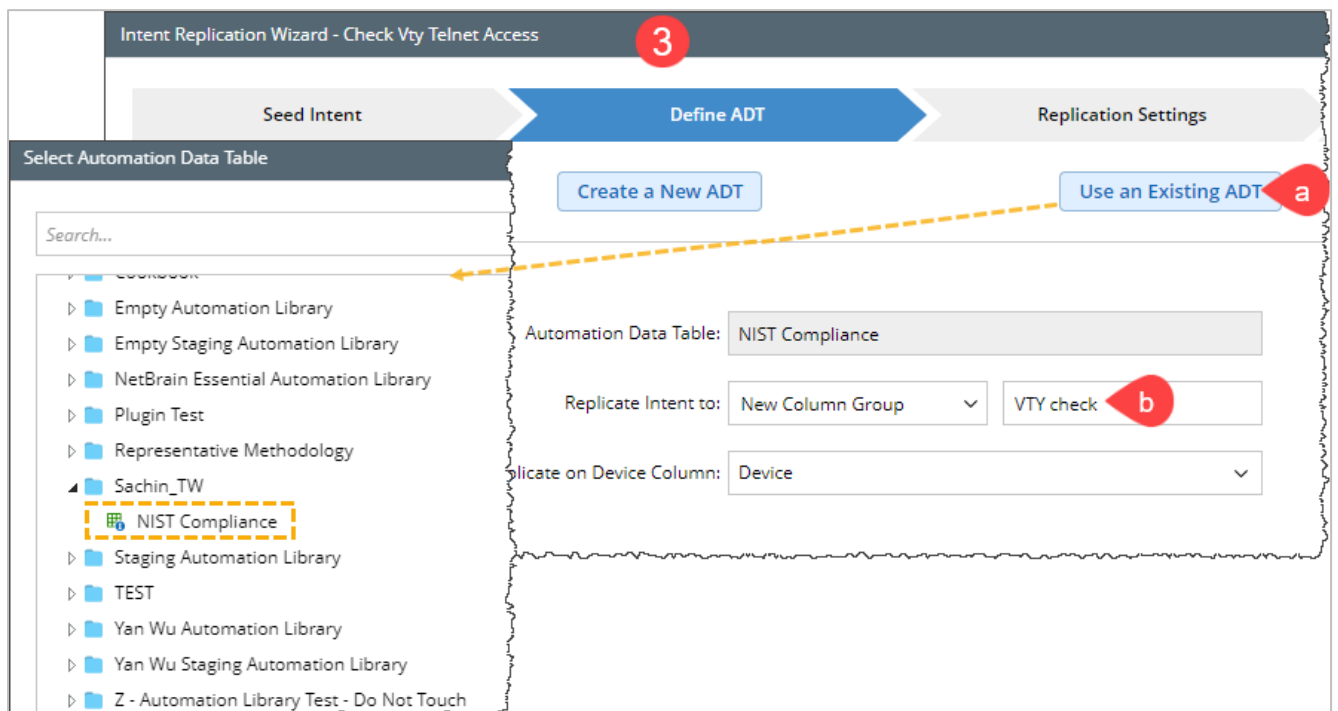
In this section, you will use the **Intent Replication Wizard** to add the Automation Column to the ADT that you have created, **NIST Compliance**.

- In the Network Intent (View Mode) window, go to the  menu and open the Intent Replication Wizard.
- In the **Seed Intent** tab, check for your NI (**Check Vty Telnet Access**) and then click **Next** to go to the **Define ADT** tab.



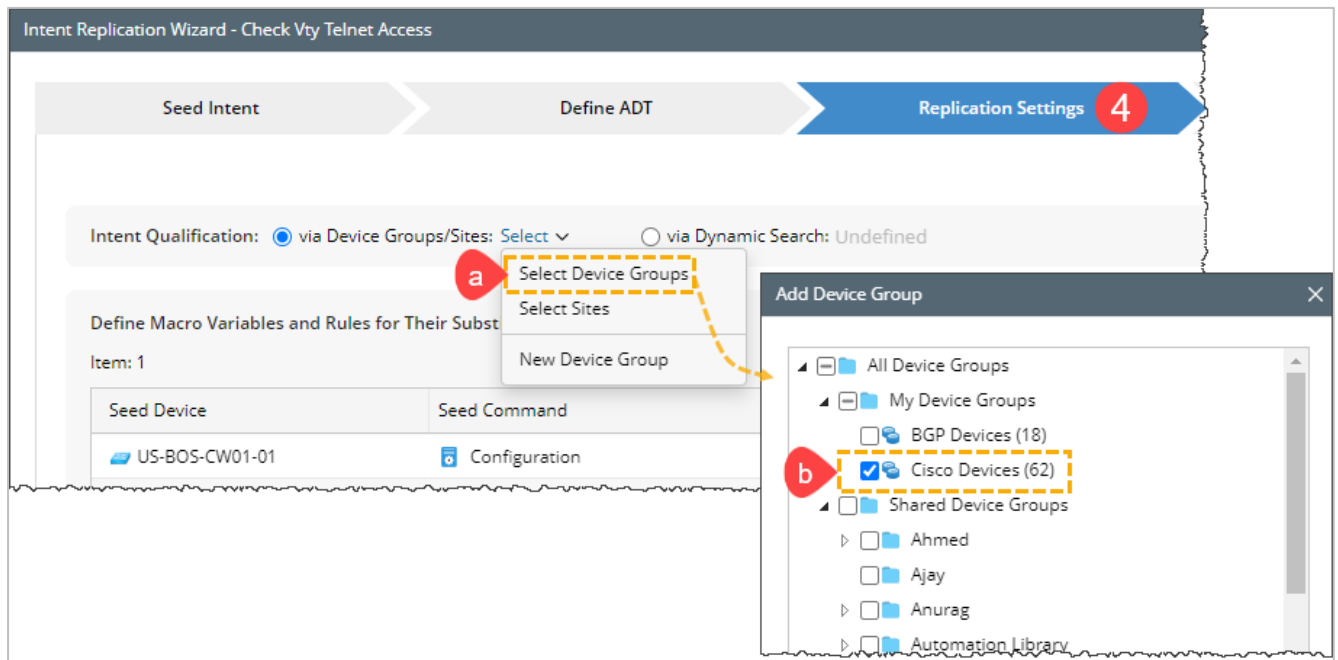
3. Define ADT:

- In the **Define ADT** tab, click **Use an Existing ADT** option and select the ADT from your folder.
- In the **Replicate Intent to** section, enter the Intent group as an Intent column group name.
- Click **Next** to go to the **Replicate Intent** tab.



4. Replication Settings:

- In the **Intent Qualification** section, click the **Select** link to add Device Group. Use the device group that you have used during ADT creation.
- Add the device group and then click **OK**.
- Click **Next** to go to the **Replicate Intent** tab.



5. Replicate Intent:

- In the **ADT Columns** section, you can see the **Column Name** as **Replicated Intent**. You can change this name to **Telnet vty check**.
- Click **Save and Replicate** to save all the settings.
You can see the **Open Output ADT** option after the successful submission of the replication request.
- Click **Open Output ADT** to check the replicated Intent column in the ADT Manager.

Intent Replication Wizard - Check Vty Telnet Access

Seed Intent
Define ADT
Replication Settings
Replicate Intent 5

ADT Columns:
Additional Columns

Column Data	Column Name	Tag
Replicated Intent	Telnet vty check a	0 tags

Save and Replicate b

Replication Request submitted at: 07/02/2024 01:31 c
Open Output ADT

Selection Mode: Device-based Replication, ADT: NIST Compliance, 0 Macro Variables.

Previous
Finish

The table will populate with devices and the replicated Intents (**Telnet vty check**). Review the new Intent column.

NIST Compliance
Table Builder
Last Updated at: 07/02/2024 01:3...
Rebuild Table
Add Data Manually

Description: Type description here...

Items: 62 Rows 3 Columns
Search...
Advanced Filter: Undefined

No.	Device	Device Type	Telnet vty check
1	Berlin-R1	Cisco Router	Check Vty Telnet Access Berlin-R1
2	Berlin-vEdge	Cisco IOS Switch	Check Vty Telnet Access Berlin-vEd...
3	DE-MUC-CR01-01	Cisco Router	Check Vty Telnet Access DE-MUC-C...
4	DE-MUC-CR01-02	Cisco Router	Check Vty Telnet Access DE-MUC-C...
5	DE-MUC-CW01-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-C...
6	DE-MUC-CW02-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-C...
7	DE-MUC-CW03-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-C...
8	ISP-P02	Cisco Router	Check Vty Telnet Access ISP-P02
9	ISP-PE01	Cisco Router	Check Vty Telnet Access ISP-PE01

6.1.2.5 Run Intent Once and Rebuild Table

Select the Intent column you added in the last section and **Run** once Intents of this column. Select the **Live Data** as a Data Source.

The screenshot shows the 'Table Builder' interface with a table containing columns for 'Device Type', 'Telnet vty...', and 'tails'. A red circle with the number '1' highlights the 'Run' button in the 'Telnet vty...' column. A dashed arrow points from this button to a dialog box titled 'Run Intents Once - Telnet vty check'. This dialog box has a 'Data Source' set to 'Live Data' and an option 'Only Run Intents if' with a dropdown menu. Below this, there is a 'Notification' dialog box with the message 'The intents in the current ADT column are executed once now.' and an 'OK' button highlighted with a red circle and the number '3'. A dashed arrow points from the 'OK' button in the notification dialog to the 'OK' button in the 'Run Intents Once' dialog, which is highlighted with a red circle and the number '2'.

Wait for the system to finish executing all intents and **Rebuild Table** to populate the intent results.

The screenshot shows the 'Table Builder' interface with a table containing columns for 'No.', 'Device', 'Device Type', 'Telnet vty check', and 'AAA Config Check'. A red circle with the number '4' highlights the 'Rebuild Table' button in the top right corner. A dashed arrow points from this button to a dialog box titled 'Rebuild Table'. This dialog box has a 'Build the column groups' dropdown menu set to 'All' (highlighted with a red circle and the number '5'), a 'Log' section with 'Production Mode' selected, and a 'Build' button highlighted with a red circle and the number '6'. The background table shows rows of data, including 'Berlin-R1' and 'DE-I'.

6.1.3 Create NI: AAA Config Check

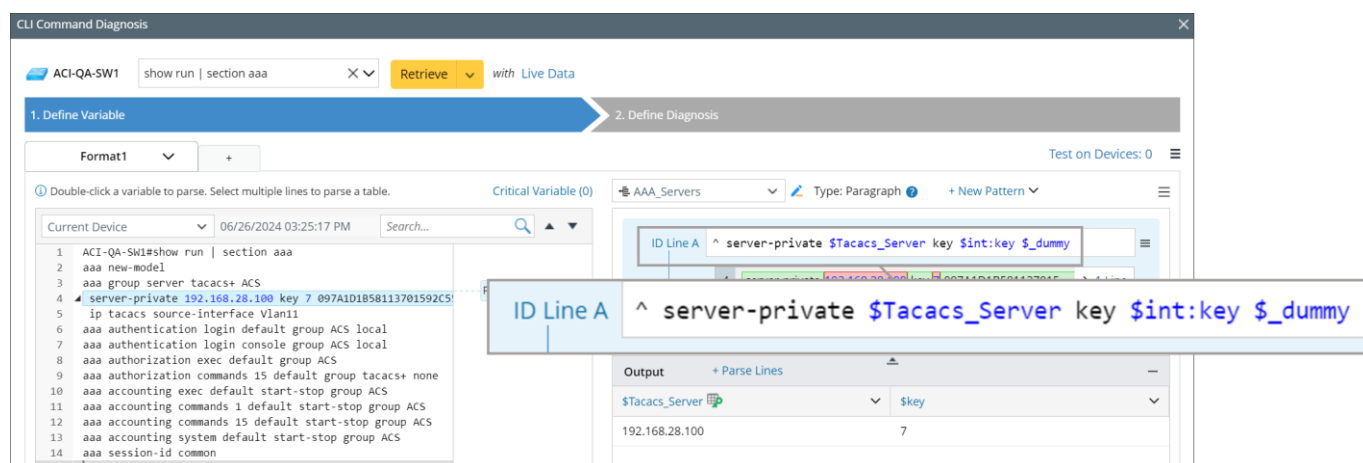
You can follow the same flow to create, replicate and run the intents for other types of configurations against NIST standards. In this and the next few sections, we will skip the detailed steps of instructions but highlight differences and new concepts. You can refer to the screenshots for each step, accompanied by brief explanations.

In this section, you will create AAA configuration check Intent for your Network devices.

6.1.3.1 Define Variables with Visual Parser

1. Use the **+ Add CLI Diagnosis** option to retrieve sample data. Alternatively, you can use the **+ Add Configuration Diagnosis** option, which does not require the use of CLI commands.
2. If you use **+Add CLI diagnosis**, then use the command, **show run | section aaa** to retrieve the sample data.
3. Define the Variable with **Paragraph** parser for **Server IP** and its **Key** value.

The ID line pattern is: **^ server-private \$Tacacs_Server key \$int:key \$_dummy**



4. Define another Variable using Variable Operator **LinesByKeyword** to extract all the data related to the Variable. The function **LinesByKeyword** is identical to **LinesByVariable** but uses keywords instead of variables to combine lines.
 - a) Click **+ Parse Lines** in the **Output** pane, and the **Parser Lines** window appears.
 - b) Set Variable name as **AAA_New_Model**.
 - c) Select The line contains keyword option, and enter the keyword, *aaa new-model*.
 - d) Click **Apply** to close the **Parser Lines** window.

Parse Lines

Name: AAA_New_Model

b

☐ The line of variable:

Select...

☐ The line between:

Select...

to

Select...

☒ The line contains keyword:

aaa new-model

c

OK

Output:

\$AAA_New_Model=
aaa new-model

Pattern:

LinesByKeyword[\$AAA_New_Model]:aaa new-model

Cancel

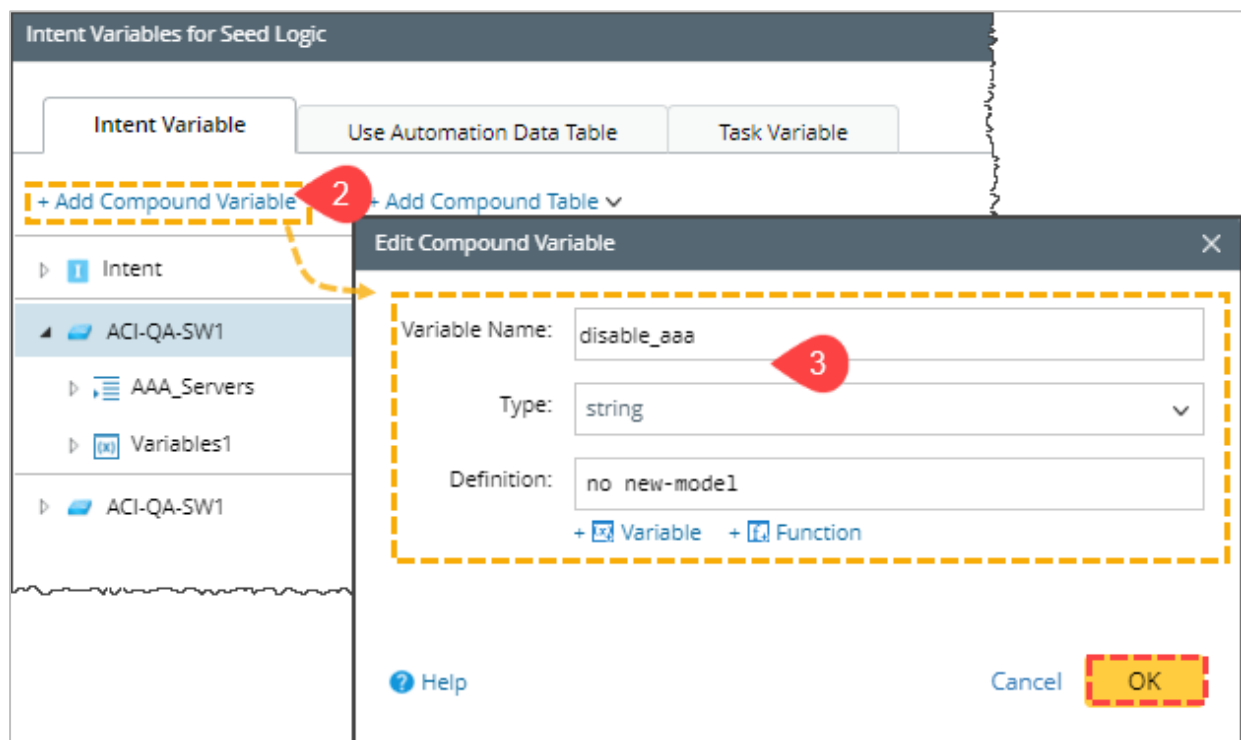
d

Apply

6.1.3.2 Define Compound Variable

In the diagnosis, we will check whether whether AAA new model is configured. If the variable **AAA_New_Model** is equal to **"no new-model"** the intent will raise an alert. Instead of repeating the string **"no new-model"** in many places of intent definition (if condition and status codes), you can create a compound variable so that it can be referenced through the intent:

1. Click **All the Variables** and select the device.
2. Click **+Add Compound Variable** to open edit Compound Variable.
3. In the Edit Compound Variable popup, define the Variable Name and its Definition.



6.1.3.3 Define Diagnosis

Define Diagnosis to set the **If** condition for the Variables:

2. Define Diagnosis

Add Note Add Diagnosis 1 Can also click a variable on the left to add automation.

Name: AAA_Config 2 Anchor: \$AAA_New_Model

Type description of the diagnosis...

☐ Loop Table Rows

▼ If 3

A ACI-QA-S... Current

AAA_New_Model Equals disable_aaa

B ACI-QA-S... Current

AAA_Servers Is empty

C Select Variable

Boolean Expression: A or B

Enter a message under the **Then** and **Else** output areas to appear as the result of the diagnosis:

▼ Then 1

Diagnosis Message: Save to Incident

\$this_device AAA is disabled

☒ Set Status Code for Device:

Error \$this_device AAA is disabled

☒ Set Status Code for Intent:

Error \$this_device AAA is disabled

Export to CSV Report: Define 2

CSV Name: AAA config check

Add Logic ▼

▼ Else 3 Delete

Diagnosis Message: Save to Incident

\$this_device AAA is enabled

☒ Set Status Code for Device:

Success \$this_device AAA is enabled

☒ Set Status Code for Intent:

Success \$this_device AAA is enabled

6.1.3.4 Replicate and Run Intent

Now, use the Intent Replication Wizard to add the Automation Column (**AAA Config Check**) to the existing ADT table, **NIST Compliance**.

No.	Device	Device Type	Telnet vty check	AAA Config Check
1	Berlin-R1	Cisco Router	Check Vty Telnet Access Berlin-R1	AAA Config Check Berlin-R1
2	Berlin-vEdge	Cisco IOS Switch	Check Vty Telnet Access Berlin-vEdge	AAA Config Check Berlin-vEdge
3	DE-MUC-CR01-01	Cisco Router	Check Vty Telnet Access DE-MUC-CR01-01	AAA Config Check DE-MUC-CR01-01
4	DE-MUC-CR01-02	Cisco Router	Check Vty Telnet Access DE-MUC-CR01-02	AAA Config Check DE-MUC-CR01-02
5	DE-MUC-CW01-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-CW01-01	AAA Config Check DE-MUC-CW01-01
6	DE-MUC-CW02-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-CW02-01	AAA Config Check DE-MUC-CW02-01
7	DE-MUC-CW03-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC-CW03-01	AAA Config Check DE-MUC-CW03-01
8	ISP-P02	Cisco Router	Check Vty Telnet Access ISP-P02	AAA Config Check ISP-P02
9	ISP-PE01	Cisco Router	Check Vty Telnet Access ISP-PE01	AAA Config Check ISP-PE01
10	ISP-PE02	Cisco Router	Check Vty Telnet Access ISP-PE02	AAA Config Check ISP-PE02
11	ISP-PE03	Cisco Router	Check Vty Telnet Access ISP-PE03	AAA Config Check ISP-PE03
12	ITE_EXTEND	Cisco IOS Switch	Check Vty Telnet Access ITE_EXTEND	AAA Config Check ITE_EXTEND
13	JP-TYO-CR01-01	Cisco Router	Check Vty Telnet Access JP-TYO-CR01-01	AAA Config Check JP-TYO-CR01-01
14	JP-TYO-CR01-02	Cisco Router	Check Vty Telnet Access JP-TYO-CR01-02	AAA Config Check JP-TYO-CR01-02
15	JP-TYO-CW01-01	Cisco IOS Switch	Check Vty Telnet Access JP-TYO-CW01-01	AAA Config Check JP-TYO-CW01-01

Run the intent columns once and rebuild the table.

Run Intents Once - Telnet vty check

Data Source: [Live Data](#)

☐ Only Run Intents if [Intent Device](#) belongs to [Select](#)

Notification

The intents in the current ADT column are executed once now.

[View Details](#) [OK](#)

NIST Compliance

Table Builder

Last Updated at: 07/03/2024 10:44 PM

Rebuild Table

4

Description: Type description here...

Items: 62 Rows 7 Columns

No.	Device	Device Type	Telnet vty check	AAA Config Check
1	Berlin-R1	Cisco Router	Check Vty Telnet Access Berlin-R1	AAA Config Check Berlin-R1
2	Berlin-R1			Check Berlin-vEdge
3	DE-MUC-CR01			Check DE-MUC-CR01...
4	DE-MUC-CR01			Check DE-MUC-CR01...
5	DE-MUC-CW01			Check DE-MUC-CW0...
6	DE-MUC-CW01			Check DE-MUC-CW0...

Rebuild Table

Build the column groups: All

5

Log: ☒ Production Mode Only show major execution process log
☐ Debug Mode Show all the detailed log

Cancel

Build

6

NetBrain R11.1b | 187

6.1.4 Create NI: Device Unused Ports Config Check

In this section, you will repeat the same flow to create an Intent **Device Unused Ports Config Check** to check the unused ports.

6.1.4.1 Define Variables with Visual Parser

Add a CLI Diagnosis and enter the command **show run | sec interface** to retrieve sample data. Use the **Paragraph** Parser pattern.

Line	Pattern
ID Line A	<code>^interface \$intf</code>
Var Line 1	<code>LinesByVariable[\$intf_config]:\$intf-</code>

CLI Command Diagnosis

US-BOS-R1

show run | sec interface

Retrieve with Live Data

1. Define Variable

2. Define Diagnosis

Format1

Test on Devices: 0

Double-click a variable to parse. Select multiple lines to parse a table.

Critical Variable (0)

Int_config

Type: Paragraph

New Pattern

Current Device

06/27/2024 12:14:28 PM

Search...

1 US-BOS-R1#show run | sec interface

2 match interface input

3 collect interface output

4 interface Loopback0

5 description This is my Boston Sites

6 ip address 10.10.10.10 255.255.255.255

7 ip ospf 1 area 0

8 interface Loopback1

9 description This is my Boston Site

10 ip address 8.8.8.8 255.255.255.255

11 ip ospf 1 area 0

12 interface Loopback99

13 no ip address

14 shutdown

15 interface Tunnel0

16 description TEST-Kunal - Remove Description later

17 bandwidth 10000

18 ip address 10.99.1.1 255.255.255.0

19 no ip redirects

20 ip pim sparse-mode

21 ip nat inside

22 ip nhrp map multicast dynamic

23 ip nhrp network-id 23500

24 ip nhrp redirect

25 ip virtual-reassembly in

26 ip ospf network point-to-multipoint

27 ip ospf 1 area 51

28 tunnel source Ethernet0/2

29

P1-ID Line A

Var Line 1

P2-ID Line A

Var Line 1

P3-ID Line A

Var Line 1

P4-ID Line A

Var Line 1

ID Line A

^interface \$intf

4 interface Loopback0

> 9 Lines

Var Line 1

LinesByVariable[\$intf_config]:\$intf-

4 interface Loopback0

> 9 Lines

+ Field

Output

+ Parse Lines

\$intf

\$intf_config

Loopback0

interface Loopback0 description This is my B...

Loopback1

interface Loopback1 description This is my B...

Loopback99

interface Loopback99 no ip address shutdown

Tunnel0

interface Tunnel0 description TEST-Kunal - R...

Tunnel10

interface Tunnel10 ip address 172.16.253.10 ...

Ethernet0/0

interface Ethernet0/0 description to 192 MG...

Ethernet0/11

interface Ethernet0/11 description to 192 MG...

Help

All Intent Variables

Cancel

Apply

6.1.4.2 Define Intent Variable for Seed Logic

Define an intent variable, **\$Alert**, to indicate whether to raise an alert and set the default value to **YES**.

Intent Variables for Seed Logic

Intent Variable Use Automation Data Table Task Variable

+ Add Intent Variable + Add Intent Table

1 2

3 4 5

Cancel OK

6.1.4.3 Define Diagnosis

Define a diagnosis, **Check_intf_conf**, to determine if the unused ports are not shutdown. If so, create an alert for this interface and set the intent variable to **YES**.

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: check_intf_conf Anchor: Int_config.\$intf_config

Type description of the diagnosis...

☒ Loop Table Rows Int_config Table Key: intf

1. Define the **If** condition for Variable:
 - a) Variable **intf_config** does not contain **shutdown**.
 - b) Variable **intf_config** does not contain **switchport**.
 - c) Variable **intf_config** does not contain **address**.
 - d) Variable **intf_config** contains no ip address.
 - e) Variable **intf_config** contains no switchport.
 - f) Variable **intf_config** does not contain **spanning-tree**.
 - g) Variable **intf_config** does not contain **vrf forwarding**.
 - h) Boolean Expression: A and B and F and G and (C or D or E).

If

1

A US-BOS-R1 Current

intf_config Does not contain shutdown

B US-BOS-R1 Current

intf_config Does not contain switchport

C US-BOS-R1 Current

intf_config Does not contain address

D US-BOS-R1 Current

intf_config Contains no ip address

E US-BOS-R1 Current

intf_config Contains no switchport

F US-BOS-R1 Current

intf_config Does not contain spanning-tree

G US-BOS-R1 Current

intf_config Does not contain vrf forwarding

H Select Variable

Boolean Expression: A and B and F and G and (C or D or E)

2. Define Intent Output message. To Set the Intent Variable, follow the steps outlined in section 6.1.2.3.

Then

Diagnosis Message: ☐ Save to Incident

☒ Set Status Code for Device:

☒ Set Status Code for Intent:

☒ Set Intent or Task Variable:

Now, add another diagnosis to check whether this device has unused showdown interfaces and create the status code on the device level.

2. Define Diagnosis

Can also click a variable on the left to add automa

Name: Anchor:

☐ Loop Table Rows

If

	Current
A	<input type="text" value="Alert"/> <input type="text" value="Equals"/> <input type="text" value="YES"/>
B	<input type="text" value="Select Variable"/>

Then 4

Diagnosis Message:

\$this_device has unused port but no shutdown

☐ Save to Incident

Set Status Code for Device:

Error

\$this_device has unused port but no shutdown

Set Status Code for Intent:

Error

\$this_device has unused port but no shutdown

Add Logic

Else 5

Diagnosis Message:

\$this_device doesn't have unused port but no shutdown

☐ Save to Incident

Set Status Code for Device:

Success

\$this_device doesn't have unused port but no shutdown

Set Status Code for Intent:

Success

\$this_device doesn't have unused port but no shutdown

Add Logic

+ Add Elself

Delete

6.1.4.4 Replicate and Run Intent

Replicate this intent and add the automation column (**Device Unused Ports**) to the ADT table, **NIST Compliance**. Then, run the replicated intents once and rebuild the table.

NIST Compliance		Table Builder	Last Updated at: 07/02/2024 08:28 PM		Rebuild Table	Add Data Manually			
Description: Type description here...									
Items: 62 Rows 5 Columns		Search...		Advanced Filter: Undefined					
No.	Device	Device Type	Telnet vty check	AAA Config Check	Device Unused Ports Config C...				
1	Berlin-R1	Cisco Router	Check Vty Telnet Access Berlin-R1	AAA Config Check Berlin-R1	Device unused Ports config ...				
2	Berlin-vEdge	Cisco IOS Switch	Check Vty Telnet Access Berlin-v...	AAA Config Check Berlin-vEdge	Device unused Ports config ...				
3	DE-MUC-CR01-01	Cisco Router	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CR01...	Device unused Ports config ...				
4	DE-MUC-CR01-02	Cisco Router	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CR01...	Device unused Ports config ...				
5	DE-MUC-CW01-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config ...				
6	DE-MUC-CW02-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config ...				
7	DE-MUC-CW03-01	Cisco IOS Switch	Check Vty Telnet Access DE-MUC...	AAA Config Check DE-MUC-CW0...	Device unused Ports config ...				
8	ISP-P02	Cisco Router	Check Vty Telnet Access ISP-P02	AAA Config Check ISP-P02	Device unused Ports config ...				
9	ISP-PE01	Cisco Router	Check Vty Telnet Access ISP-PE01	AAA Config Check ISP-PE01	Device unused Ports config ...				
10	ISP-PE02	Cisco Router	Check Vty Telnet Access ISP-PE02	AAA Config Check ISP-PE02	Device unused Ports config ...				
11	ISP-PE03	Cisco Router	Check Vty Telnet Access ISP-PE03	AAA Config Check ISP-PE03	Device unused Ports config ...				

192 | NetBrain R11.1b

6.1.5 Create NI: Device Password Policy Config Check

In this section, you will practice creating an intent to check the **Device Password Policy Config**. You will use the command ***show run | section username*** to retrieve the sample data and define a **Paragraph** Parser to find the username and password level.

Line	Pattern
ID Line A	^username
Var Line 1	^username \$string:username
Var Line 2	password \$int:pwd_level

In the diagnosis, you will check whether the ***password level*** is less than **7**.

The screenshot shows the '2. Define Diagnosis' configuration window. At the top, there are buttons for 'Add Note' and 'Add Diagnosis' (callout 1). Below these, the 'Name' field is set to 'pwd_level_check' (callout 2) and the 'Anchor' field is set to 'pwd_conf.\$username' (callout 3). A text area for 'Type description of the diagnosis...' is present. Below the text area, the 'Loop Table Rows' checkbox is checked, and the table is set to 'pwd_conf' with a 'Table Key' of 'username' (callout 4). Under the 'If' section, there are two conditions. Condition A is 'CA-TOR-SW2' with a 'Current' dropdown (callout 5). Condition B is 'Select Variable'. The main condition is 'pwd_level' (callout 5) with a 'Less than' operator and the value '7'.

Define the Diagnosis message:

Then

Diagnosis Message:

\$this_device username \$username password level is less than 7

Save to Incident

Set Status Code for Device:

Set Status Code for Intent:

Set Intent or Task Variable:

Alert

=

"YES"

Add Logic

Else

Diagnosis Message:

\$this_device Password policy meet security requirements

Save to Incident

Set Status Code for Device:

Set Status Code for Intent:

Add Logic

6.1.6 Create NI: Device Line Session Timeout

In this section, you will practice creating an intent to check the **Line Session Timeout**.

Use the **Config Diagnosis** and define the **Paragraph** Parser pattern as follows:

Line	Pattern
Start Line	^line
End Line	!
ID Line A	^line \$linetype \$int:port
Var Line 1	LinesByVariable[\$configlet]:\$linetype

The screenshot shows the Configuration Diagnosis tool interface. On the left, a list of configuration lines is displayed, with line 188 highlighted. On the right, the 'Define Diagnosis' section is active, showing the configuration of a 'Paragraph' parser pattern. The pattern is defined with the following fields:

- Start Line:** ^line
- End Line:** !
- ID Line A:** ^line \$linetype \$int:port
- Var Line 1:** LinesByVariable[\$configlet]:\$linetype

The 'Output' section shows the results of the parser pattern, with columns for \$linetype, \$port, and \$configlet. The output is as follows:

\$linetype	\$port	\$configlet
con	0	line con 0
aux	0	line aux 0
vty	0	line vty 0 4

In the diagnosis, you check whether the device configuration contains **exec-timeout**.

Define the Diagnosis message:

6.1.7 Create Intent and Summary Dashboard

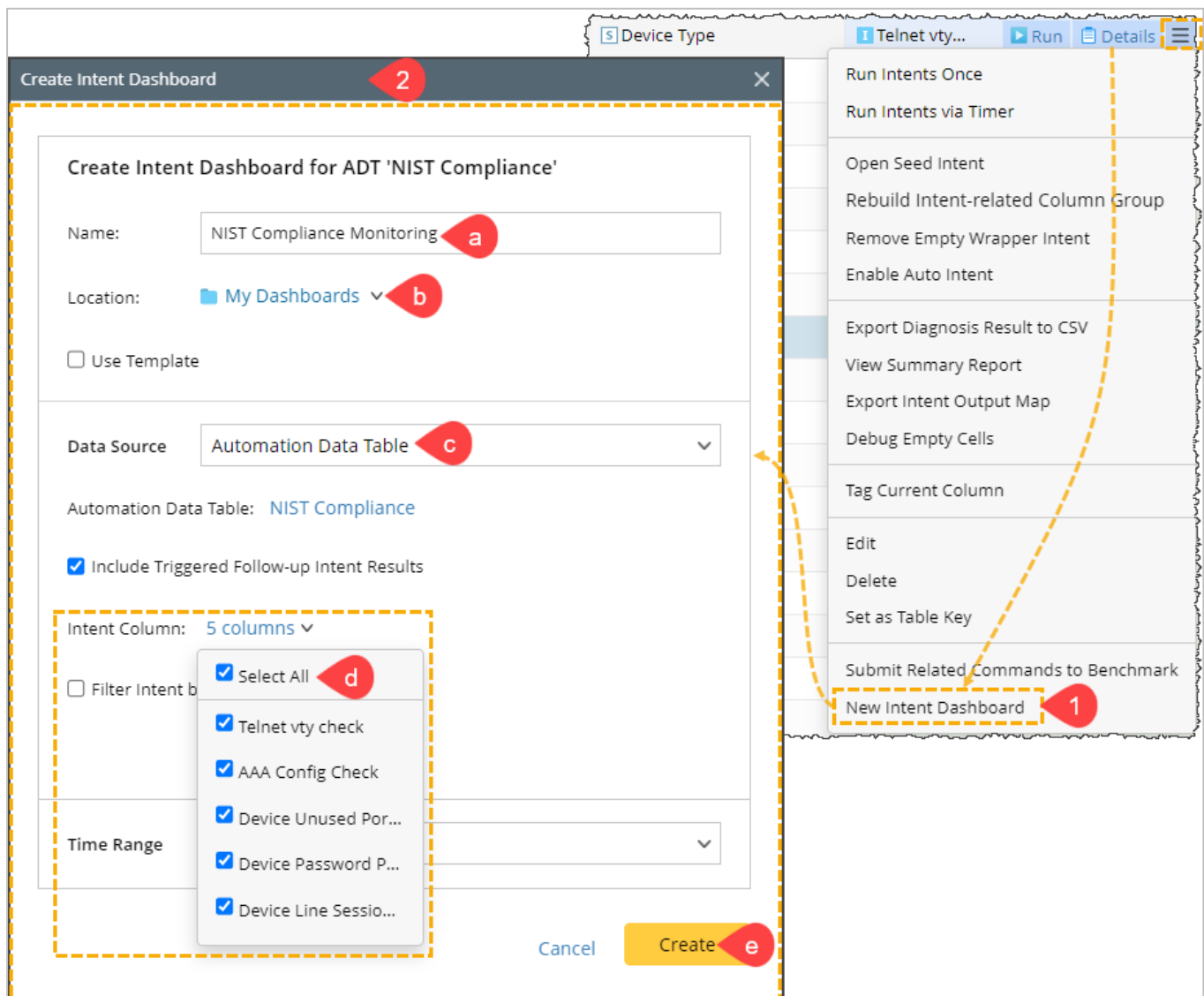
You can view the Intent results through two different dashboards:

- **Intent Dashboard** to view the individual Intent results and
- **Summary Dashboard** to view the consolidated Intent results in a single view.

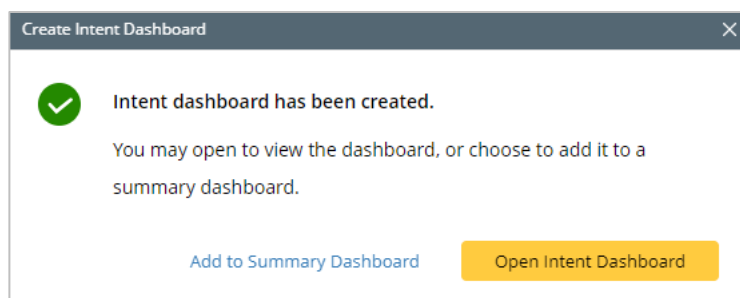
6.1.7.1 Intent Dashboard

The Intent Dashboard monitors specific network issues, providing detailed information and displaying the results. You can save frequently used dashboards as templates.

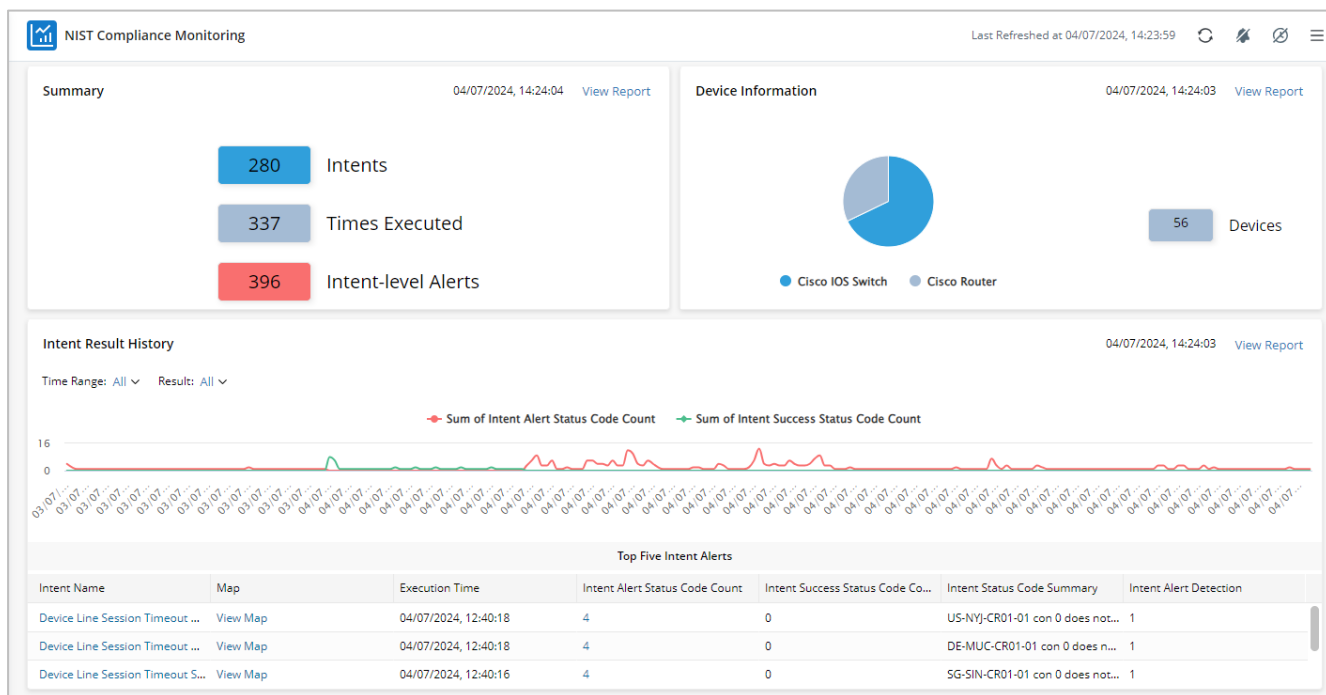
To create an Intent Dashboard directly from the ADT automation column menu, follow these steps:



1. Hover over the ADT automation column, go to the menu and then click **New Intent Dashboard**.
2. In the **Create Intent Dashboard** window, define the following:
 - a) Enter the Dashboard Name, **NIST Compliance Monitoring**.
 - b) Select the **Location** to save the Intent Dashboard.
 - c) Data Source: By default, Automation Data Table is selected.
 - d) **Intent Column**: Click the **Intent Column** dropdown and check the **Select All** checkbox to select all Intents, allowing the Intent dashboard to be created for all Intents at once
 - e) Click **Create**.
3. In the **Create Intent Dashboard** dialog popup, click **Open Intent Dashboard**.




4. The Intent Dashboard will be like:

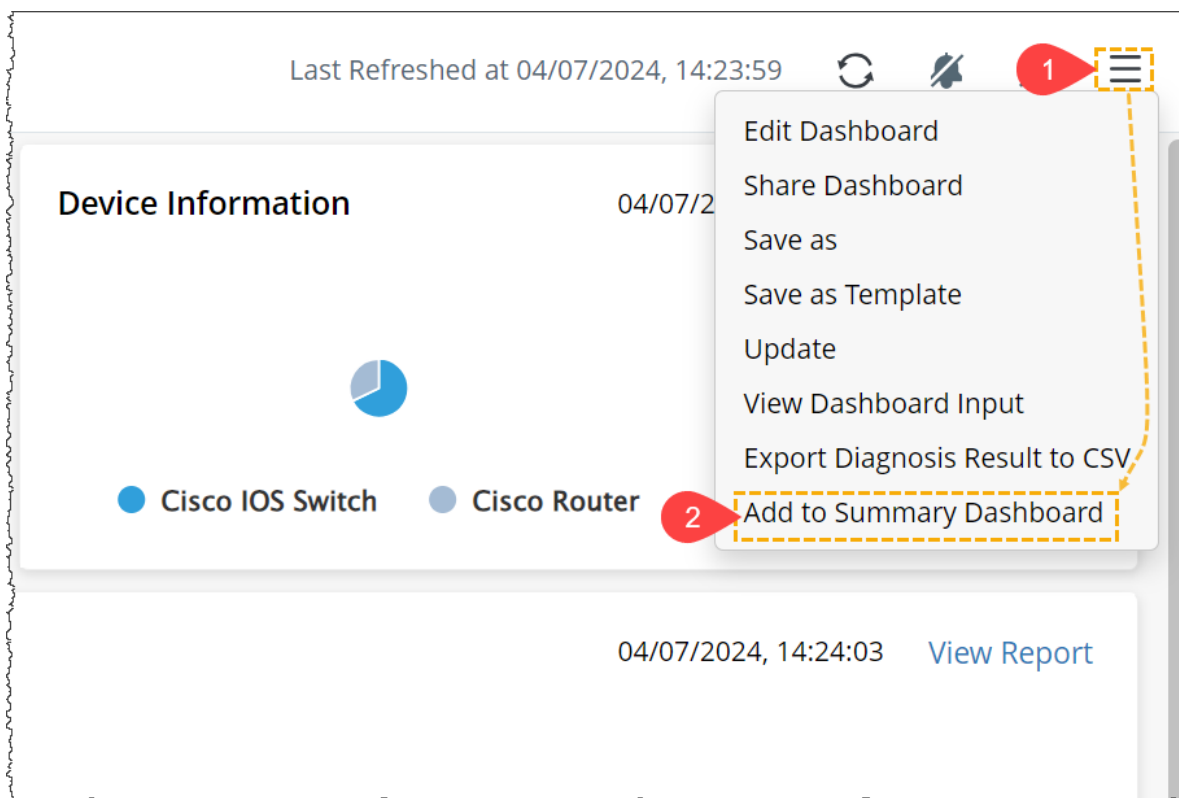


6.1.7.2 Summary Dashboard

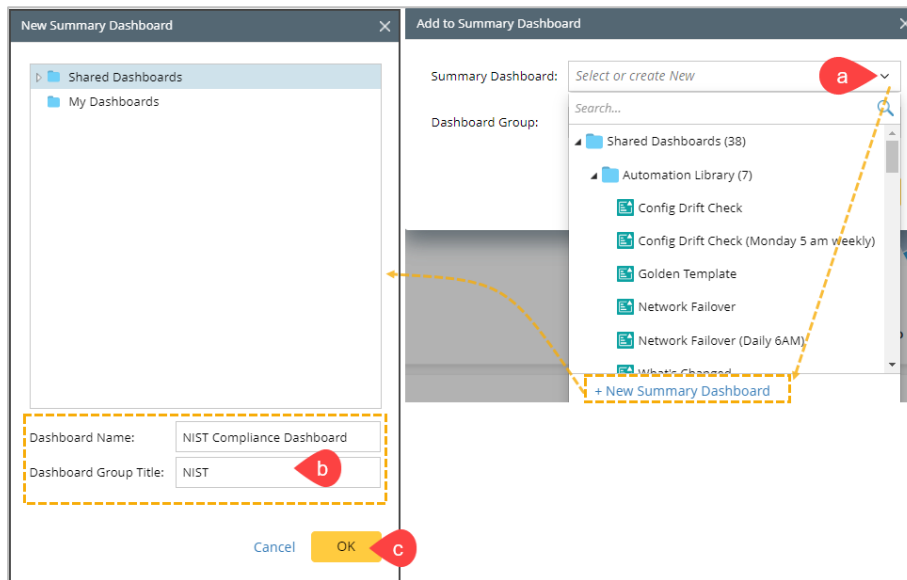
The summary dashboard provides an overview displaying results from multiple Intent dashboards of the entire network or a set of network devices. With Summary Dashboard, you can group Intent Dashboards into widgets based on diagnosis purpose and display results by device, site or device groups. You can use the summary dashboard to monitor critical information across thousands of devices and discover the root cause for issues in one view.

Follow the step-by-step instructions to create a Summary Dashboard:

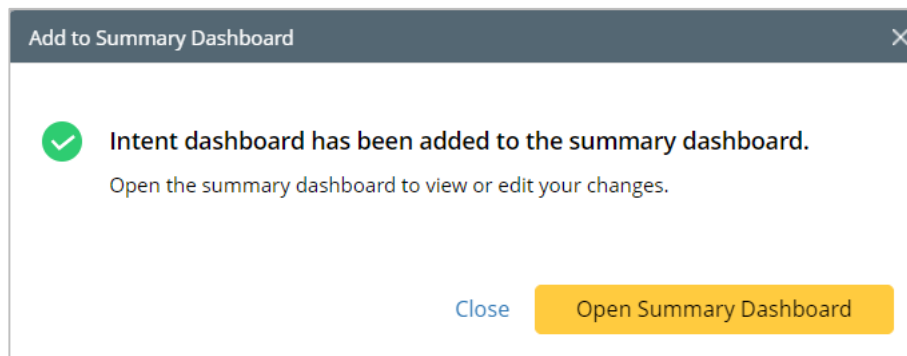
1. Click  to open the menu located at the top-right corner of the dashboard window.
2. Select **Add to Summary Dashboard** to open the corresponding window for creating a summary dashboard.



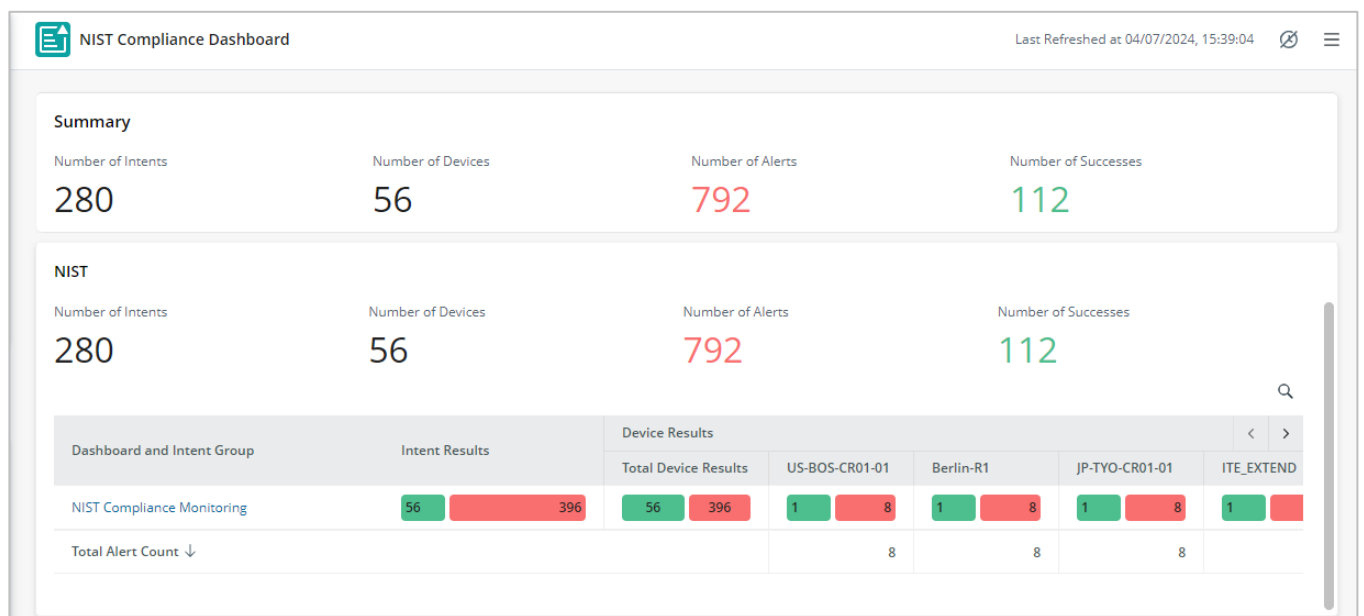
3. In the **Add to Summary Dashboard** window, let us create the new summary dashboard and group as follows:
 - a) **Summary Dashboard:** open the dropdown menu and select **+New Summary Dashboard** to pop up its dialogue.
 - b) Enter the dashboard basic details like **name**, **group title** and **location** of the summary dashboard to save.
 - c) Click **OK** to save and create the summary dashboard.



4. In the Add to Summary Dashboard dialog popup, click Open Summary Dashboard.



The Summary Dashboard will be like:



You can access the created Summary Dashboard, as shown in the image.

NetBrain Next-Gen

Search Anything and Create Map...

Summay Dashboard > Shared Dashboards

Help

Dashboard Template

Search Dashboard and Double-click to Open...

Summary Intent and Universal

Triggered Automation Dashboard (1)

Shared Dashboards (39)

Automation Library (7)

Automation Library PKG11 (1)

Automation Library Ty (1)

Minghui Qiu (1)

NetBrain Essential Automation Libra...

Sophia Wang (2)

Xinyu Zhang (2)

Yan Wu (2)

Yan Wu Automation Library (4)

Yihong Liu (2)

Yueqin (Amy) Li (2)

Zelin Deng (3)

My Dashboards (0)

Name	Modified Date
Automation Library	11/04/2024, 07:03:54
NetBrain Essential Automati...	17/06/2024, 21:22:24
Automation Library Ty	18/06/2024, 21:24:36
Xinyu Zhang	25/06/2024, 22:09:57
Yihong Liu	25/06/2024, 22:29:09
Yan Wu	25/06/2024, 23:21:00
Minghui Qiu	25/06/2024, 23:31:36
Zelin Deng	26/06/2024, 00:23:54
Yueqin (Amy) Li	26/06/2024, 00:56:28
Sophia Wang	26/06/2024, 02:27:30
Yan Wu Automation Library	27/06/2024, 00:10:17
Automation Library PKG11	27/06/2024, 04:25:19
Golden Template Check	02/05/2024, 14:21:12
What's Changed	21/06/2024, 00:54:59
Summary	01/06/2024, 00:14:35
Network_Failover	30/04/2024, 18:42:11
checkSSH	25/06/2024, 23:32:21
Michelle	27/06/2024, 23:50:57
SA Triage Dashboard	27/06/2024, 22:19:26
NIST Compliance Dashboard	04/07/2024, 15:37:21

6.2 CVE Security Advisory

CVE (Common Vulnerabilities and Exposures) classifies the vulnerabilities according to the different threat levels. It is critical to understand the severe CVEs that may affect your network devices and create a plan to address them, usually upgrading the system to a certain version that has fixed the CVE.

Each major network vendor publishes their CVEs. For example, Cisco has security advisories at <https://sec.cloudapps.cisco.com/security/center/publicationListing.x>.

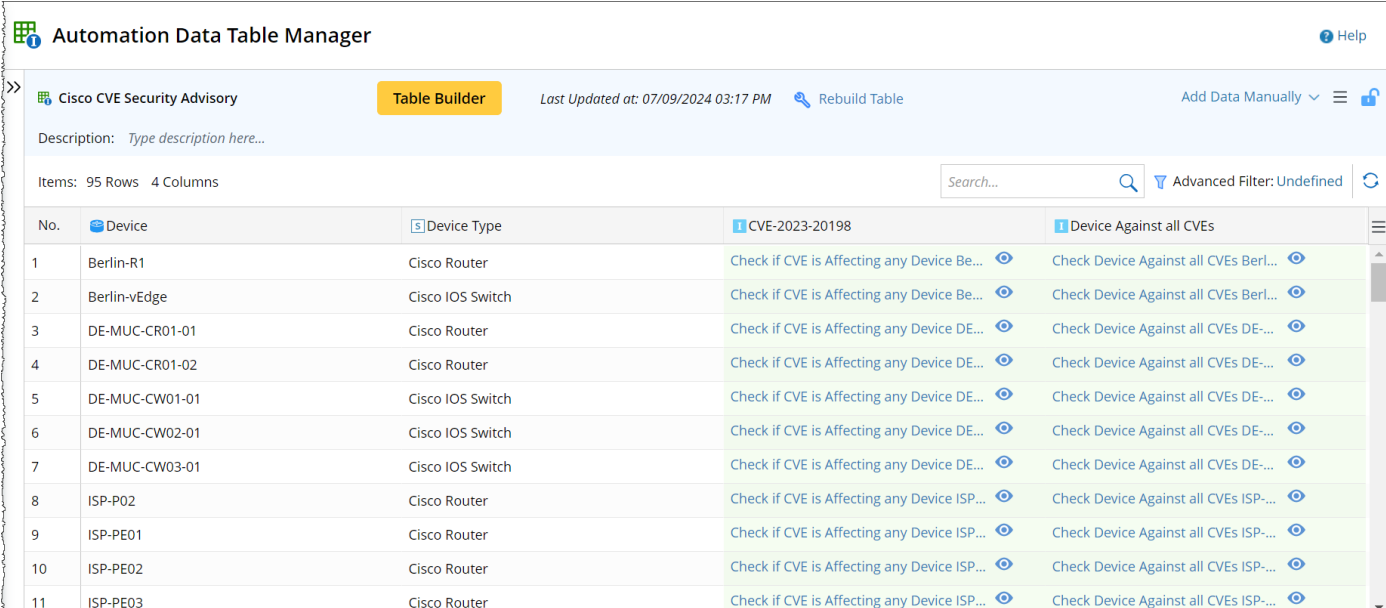
In this section, we will create intents for the vendor-specific CVEs:

1. Given a CVE (for example, the newfound one), find how many devices are affected.
2. Given a device, find how many CVEs affect it.

This section includes the following main steps:

- [Create CVE table](#)
- [Build Base ADT](#)
- [Create an Intent to Check if a CVE is Affecting any Device](#)
- [Create Intent to Check Device Against all CVEs](#)
- [Create Intent and Summary Dashboard](#)

The Final ADT will be:



The screenshot shows the 'Automation Data Table Manager' interface. At the top, it says 'Cisco CVE Security Advisory' with a 'Table Builder' button. Below this, there's a description field and a search bar. The table has 95 rows and 4 columns. The columns are: No., Device, Device Type, and two columns for CVE-2023-20198. The table lists various devices and their types, along with the CVE status for each.

No.	Device	Device Type	CVE-2023-20198	Device Against all CVEs
1	Berlin-R1	Cisco Router	Check if CVE is Affecting any Device Be...	Check Device Against all CVEs Berl...
2	Berlin-vEdge	Cisco IOS Switch	Check if CVE is Affecting any Device Be...	Check Device Against all CVEs Berl...
3	DE-MUC-CR01-01	Cisco Router	Check if CVE is Affecting any Device DE...	Check Device Against all CVEs DE...
4	DE-MUC-CR01-02	Cisco Router	Check if CVE is Affecting any Device DE...	Check Device Against all CVEs DE...
5	DE-MUC-CW01-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE...	Check Device Against all CVEs DE...
6	DE-MUC-CW02-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE...	Check Device Against all CVEs DE...
7	DE-MUC-CW03-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE...	Check Device Against all CVEs DE...
8	ISP-P02	Cisco Router	Check if CVE is Affecting any Device ISP...	Check Device Against all CVEs ISP...
9	ISP-PE01	Cisco Router	Check if CVE is Affecting any Device ISP...	Check Device Against all CVEs ISP...
10	ISP-PE02	Cisco Router	Check if CVE is Affecting any Device ISP...	Check Device Against all CVEs ISP...
11	ISP-PE03	Cisco Router	Check if CVE is Affecting any Device ISP...	Check Device Against all CVEs ISP...

6.2.1 Prerequisites

Please ensure that the following prerequisites are met:

1. Create the Cisco Device Group according to your requirements. Refer to the section [5](#).
2. Set up a CVE table that includes at least the CVE_ID and affected versions.


6.2.1.1 Create CVE table

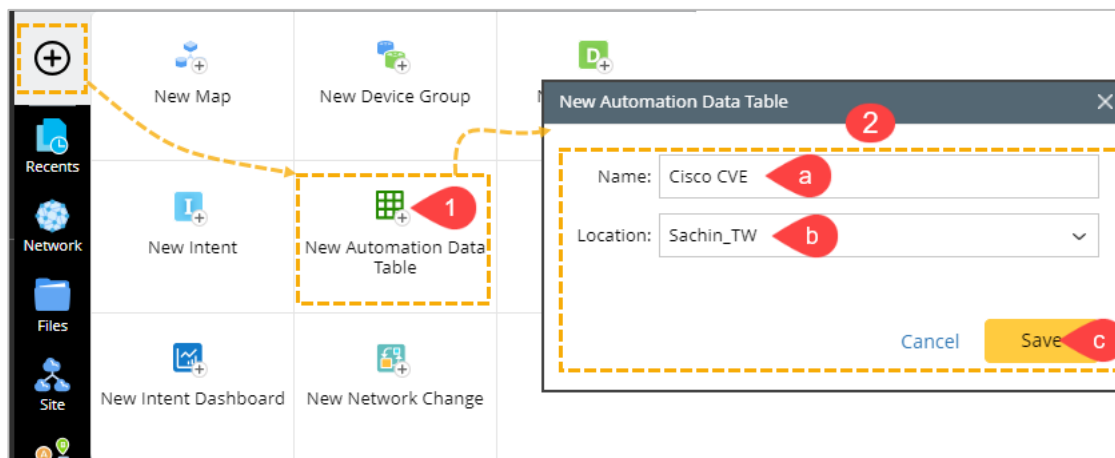
The CVE table is necessary for configuring the Intent Template Settings to define diagnoses for the available CVEs.

You can manually create an ADT table to add columns for CVE_ID, CVE_URL, and Affected_Versions. Alternatively, you can use the CSV in the ADT table builder via the Imported CSV method.

Note: this exercise is for your practice only. NetBrain will maintain and update the CVE tables and you can download the table from your system.

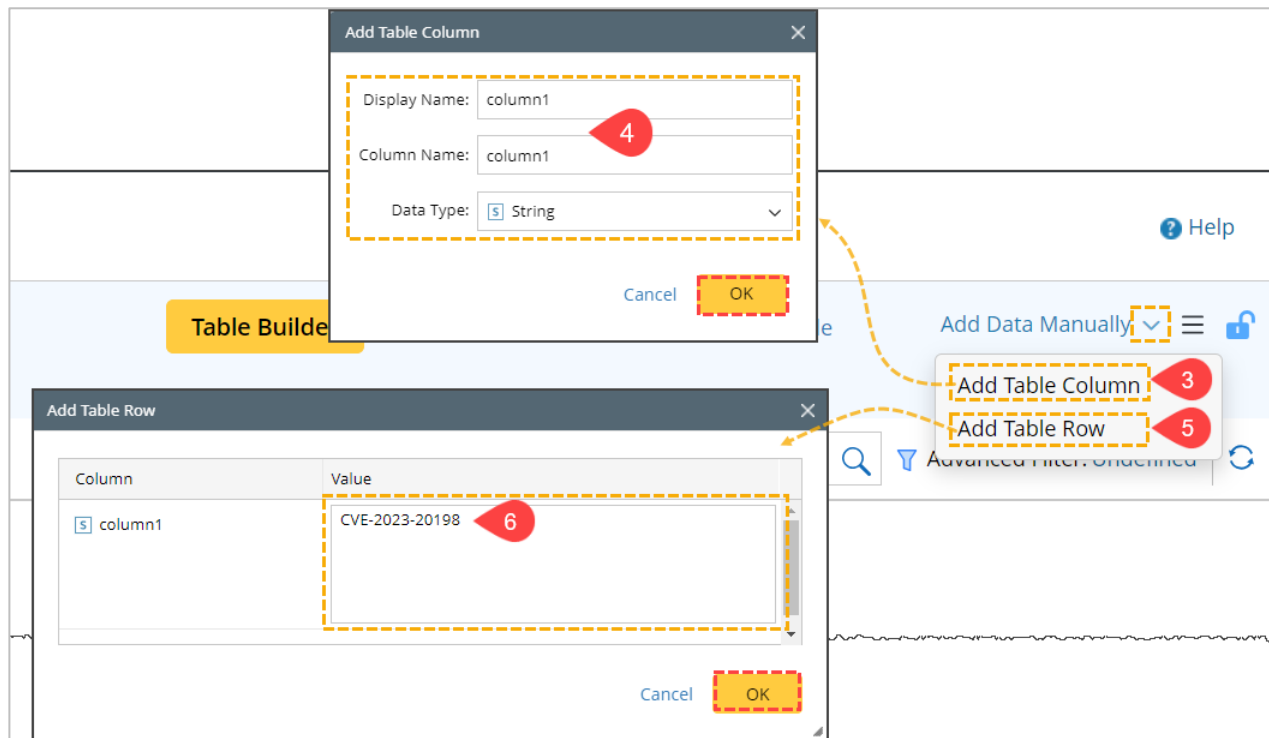
Follow the step-by-step instructions to manually create the ADT table.

1. Click the plus icon  and click **New Automation Data Table**.
2. In the **New Automation Data Table** popup:
 - a) Enter the ADT Name, e.g., **Cisco CVE**.
 - b) Select the **Location** you wish to store the ADT.
 - c) Click **Save** and wait a moment for ADT to open.



3. In the **Automation Data Table Manager**, Click the **Add Data Manually** dropdown, and then click **Add Table Column**.
4. In the Add Table Column popup, define the following values and click **OK**.:
 - a) Enter Display Name, CVE_ID

- b) Enter Column Name, CVE_ID.
- c) Select Data Type as String
5. Click the **Add Data Manually** dropdown and click **Add Table Row**.
6. In the **Add Table Row** popup, enter the **Column Value** as your CVE ID number and then click **OK**.



Similarly, create additional columns and rows for **CVE_URL** and **Affected Version**. The **Cisco CVE** ADT will be:

Automation Data Table Manager

Search...

Cisco CVE **Table Builder** Last Updated at: 07/... [Rebuild Table](#) [Add Data Manually](#)

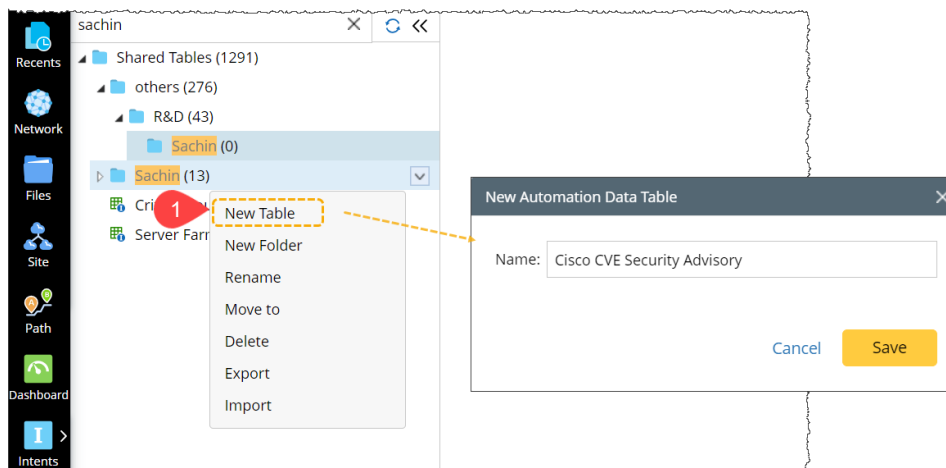
Description: *Type description here...*

Items: 4 Rows 3 Columns [Advanced Filter: Undefined](#)

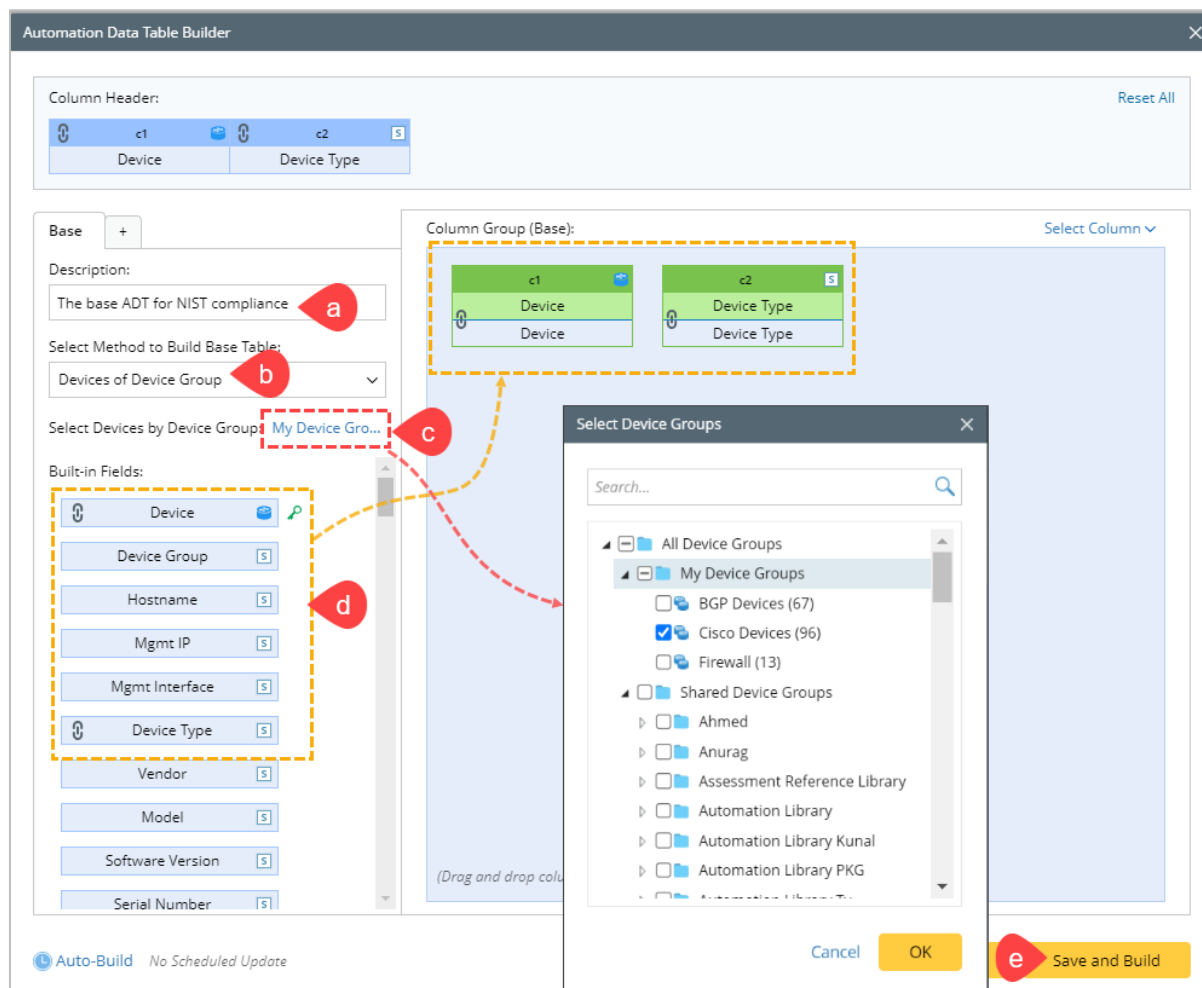
No.	CVE_ID	CVE URL	Affected Version
1	CVE-2023-20048	https://sec.cloudapps.cisco.com/security/center/co...	9.8(3)26
2	CVE-2023-20198	https://sec.cloudapps.cisco.com/security/center/co...	17.6.5
3	CVE-2017-12240	https://sec.cloudapps.cisco.com/security/center/co...	15.1(04)M09
4	CVE-2018-0171	https://sec.cloudapps.cisco.com/security/center/co...	15.2(05)E

6.2.2 Build Base ADT

1. Create a new ADT, Cisco CVE Security Advisory.



2. Build the base group with the method **Devices of Device Group**. Add the following columns: **Device** and **Device Type**.



The Base ADT will be:

Automation Data Table Manager

Search...

- Automation Library PMS11 (7)
- Build-Auto Template by Plugin (2)
- Build-Auto Template by Plugin PMS (2)
- Build-Auto Template by PluginPMS (2)
- Cookbook (14)
- Empty Automation Library (10)
- Empty Staging Automation Library (20)
- NetBrain Essential Automation Library L...
- Plugin Test (16)
- Representative Methodology (2)
- Sachin_TW (5)
 - CA test
 - Cisco CVE
 - Cisco CVE Security Advisory
 - NIST Compliance
 - Test Data
- Staging Automation Library (40)
- TSP (2)
- Yen Wu Automation Library (40)
- Yen Wu Staging Automation Library (20)
- Z - Automation Library Test - Do Not Test...
- BGP Config Change Diagnosis
- Check OSPF Neighbor
- Golden_config_check_jumper
- Ping Target

Cisco CVE Security Advisory
Table Builder
Last Updated at: 07/08/2024 11:27 AM

Description: Type description here...

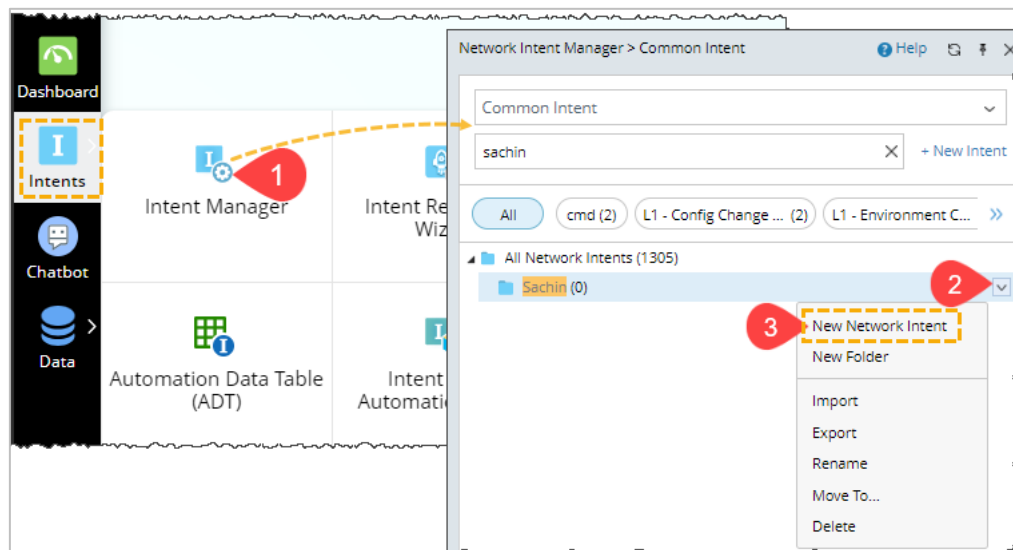
Items: 95 Rows 2 Columns

No.	Device	Device Type
1	Berlin-R1	Cisco Router
2	Berlin-vEdge	Cisco IOS Switch
3	DE-MUC-CR01-01	Cisco Router
4	DE-MUC-CR01-02	Cisco Router
5	DE-MUC-CW01-01	Cisco IOS Switch
6	DE-MUC-CW02-01	Cisco IOS Switch
7	DE-MUC-CW03-01	Cisco IOS Switch
8	ISP-P02	Cisco Router
9	ISP-PE01	Cisco Router
10	ISP-PE02	Cisco Router
11	ISP-PE03	Cisco Router
12	ITE_EXTEND	Cisco IOS Switch
13	JP-TYO-CR01-01	Cisco Router
14	JP-TYO-CR01-02	Cisco Router
15	JP-TYO-CW01-01	Cisco IOS Switch
16	JP-TYO-CW01-02	Cisco IOS Switch

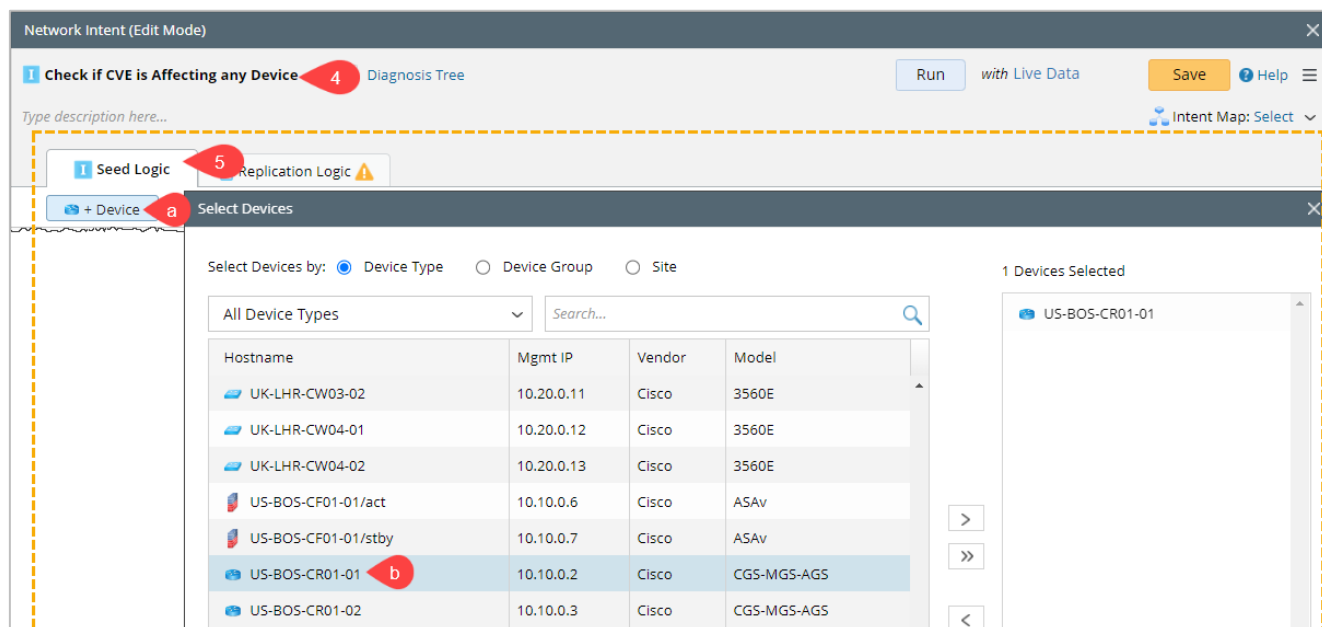
In the next sections, you will create two separate Intents to **Check if CVE is Affecting a Device** and another to **Check Device Against all CVEs**.

6.2.3 Create an Intent to Check if a CVE is Affecting any Device

From **Intent Manager**, create a new intent, **Check if CVE is Affecting a Device**.



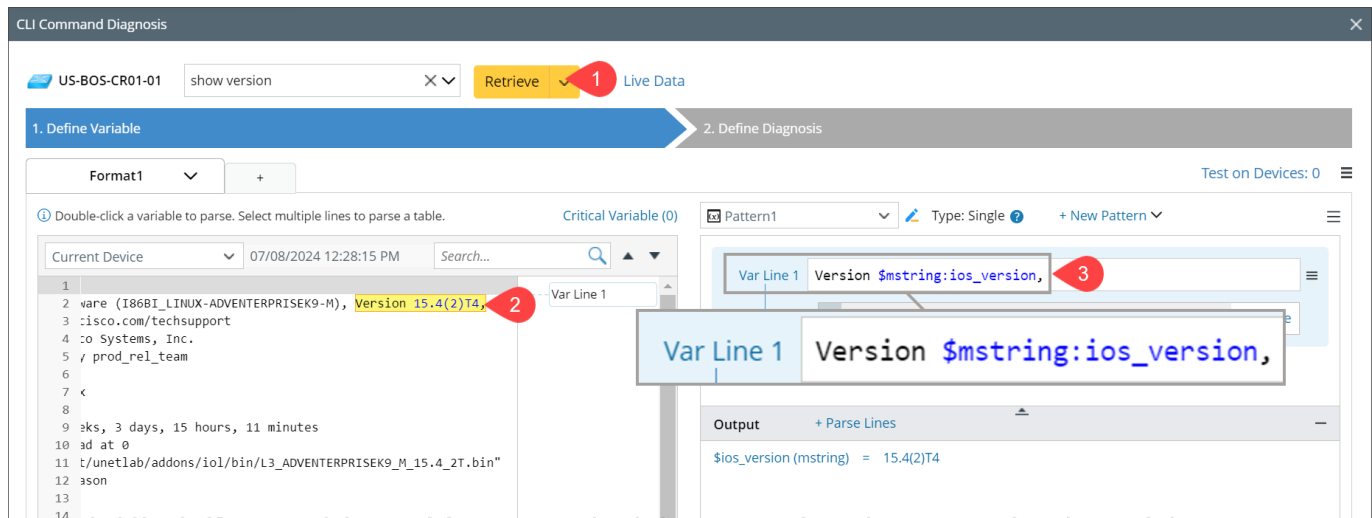
Select a Cisco device as the seed device.



6.2.3.1 Define Variables with Visual Parser

Add a **CLI Diagnosis**. Enter the command, **show version**, and then click **Retrieve** to retrieve the data from the **Live Data**.

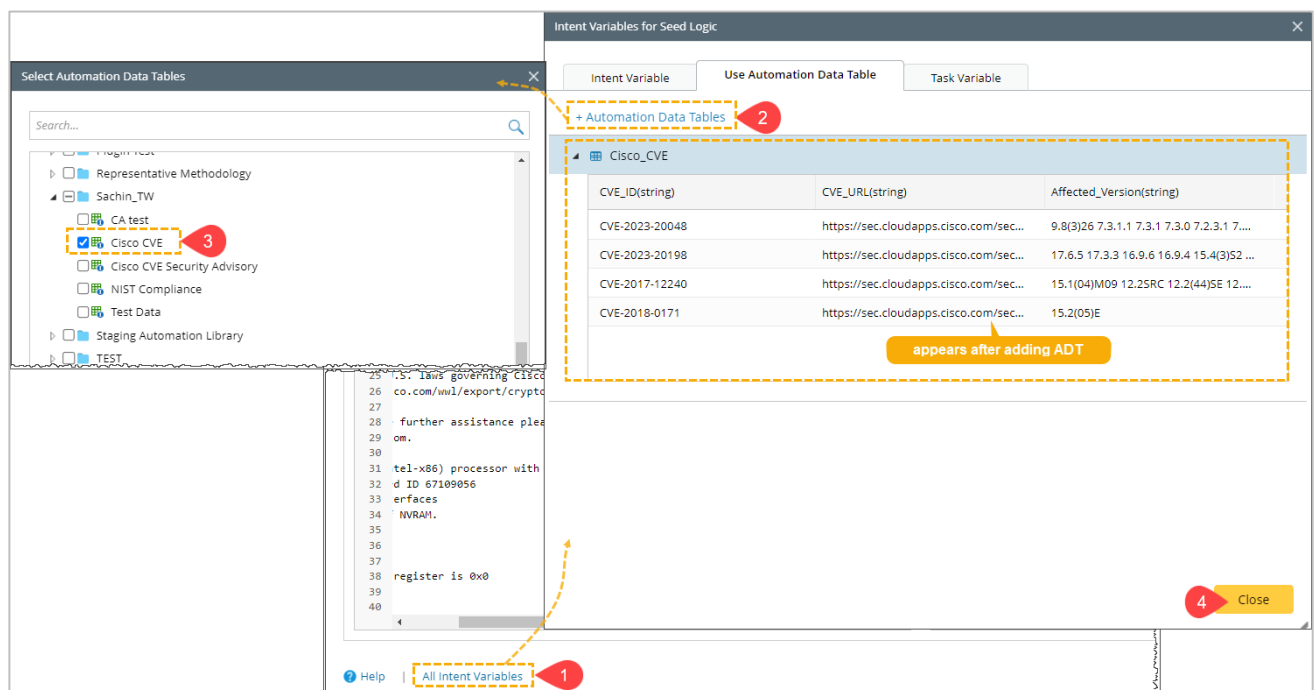
Define a single parser for the software version: **Version** `$mstring:ios_version`,



6.2.3.2 Define Intent Variable for Seed Logic

In this section, you will use the ADT **Cisco CVE** you created as a prerequisite. The columns in this ADT will be used as the Variables to define the diagnosis.

1. Click **All Intent Variables** in the bottom right corner.
2. Go to the **Use Automation Data Table** tab, click **+ Automation Data Table**, and select the **Cisco CVE**.
3. From the **Select Automation Data Table** window, select the **Cisco CVE** ADT and then click **OK**.
4. Click Close to exit from the Intent Variable for Seed Logic window.



6.2.3.3 Define Diagnosis

In this section, you will define diagnostics: to **Check if any CVE is affecting a device**. This diagnosis is usually useful if a new CVE is published and you want to check whether it affects your network device.

The diagnosis logic is simple: we will loop through the CVE table to find the matched CVE and check whether the affected software version includes the version of this device:

1. Go to the **Define Diagnosis** section.
2. Click **Add Diagnosis** to define conditions and Intent output message.
3. Enter the diagnosis name, e.g., **CVE-2023-20198** and select an Anchor ***\$asa_version*** from the dropdown.
4. Tick the **Loop Table Rows** checkbox and select the Table Variable (***Cisco_CVE***) and the Table Key (***CVE_ID*** and ***Affected_version***) from the dropdown.
5. Define **If** condition as detailed in the image:
 - a) Variable ***CVE_ID*** Equals **CVE-2023-20198**.
 - b) Variable ***Affected_Version*** Equals ***asa_version***.
 - c) The Boolean Expression will be **A and B**.

6. Define Intent Output message for **If** condition.

▼ Then 6

Diagnosis Message: ☐ Save to Incident

a \$this_device software version has the CVE-2023-20198 vulnerability. Please update to the latest version.

b ☒ Set Status Code for Device: \$this_device software version has the CVE-2023-20198 vulnerability. Please update to the latest version.

c ☒ Set Status Code for Intent: \$this_device software version has the CVE-2023-20198 vulnerability. Please update to the latest version.

Add Logic ▼

7. Define **ElseIf** condition, i.e., Variable **CVE_ID** Equals **CVE-2023-20198**.

8. Define Intent Output message for **ElseIf** condition.

▼ ElseIf 7

A Current

CVE_ID Equals CVE-2023-20198

B Select Variable

▼ Then 8

Diagnosis Message: ☐ Save to Incident

\$this_device software version has resolved the CVE-2023-20198 vulnerability


☒ Set Status Code for Device: \$this_device software version has resolved the CVE-2023-20198 vulnerability

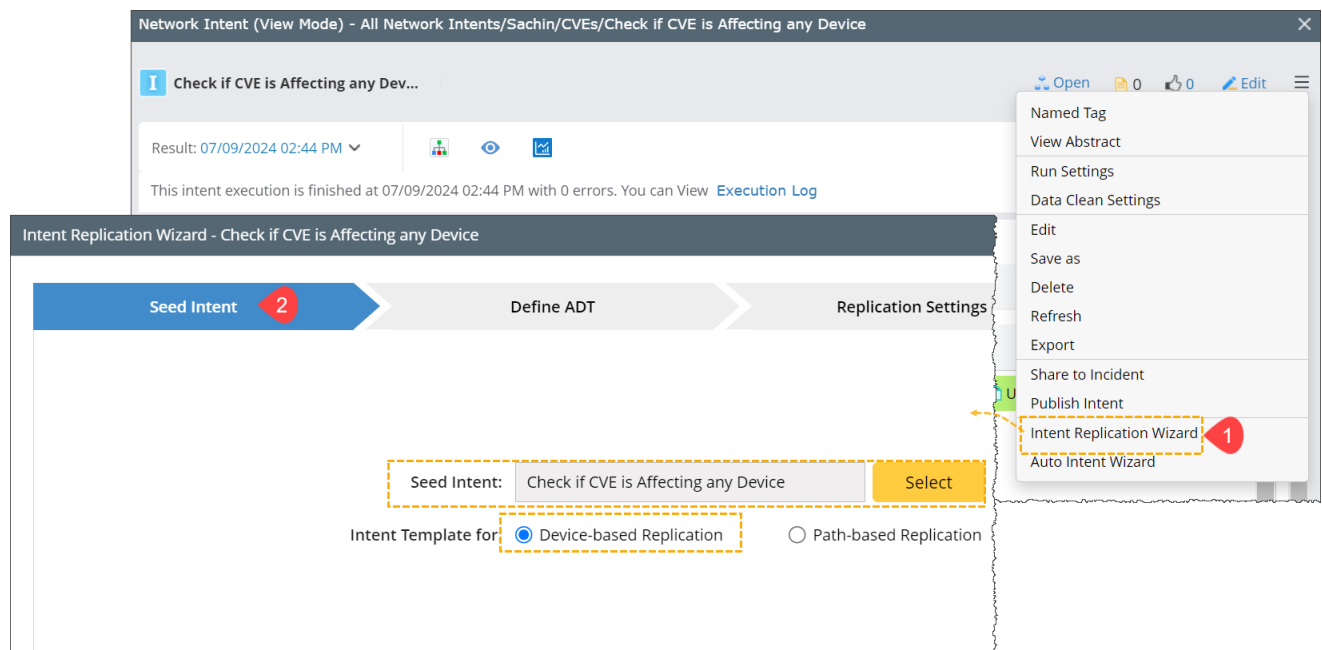
☒ Set Status Code for Intent: \$this_device software version has resolved the CVE-2023-20198 vulnerability

9. Click **Apply** to save all the diagnosis settings and run the Intent. Check whether it is working as per your configuration settings.

6.2.3.4 Use Intent Replication Wizard

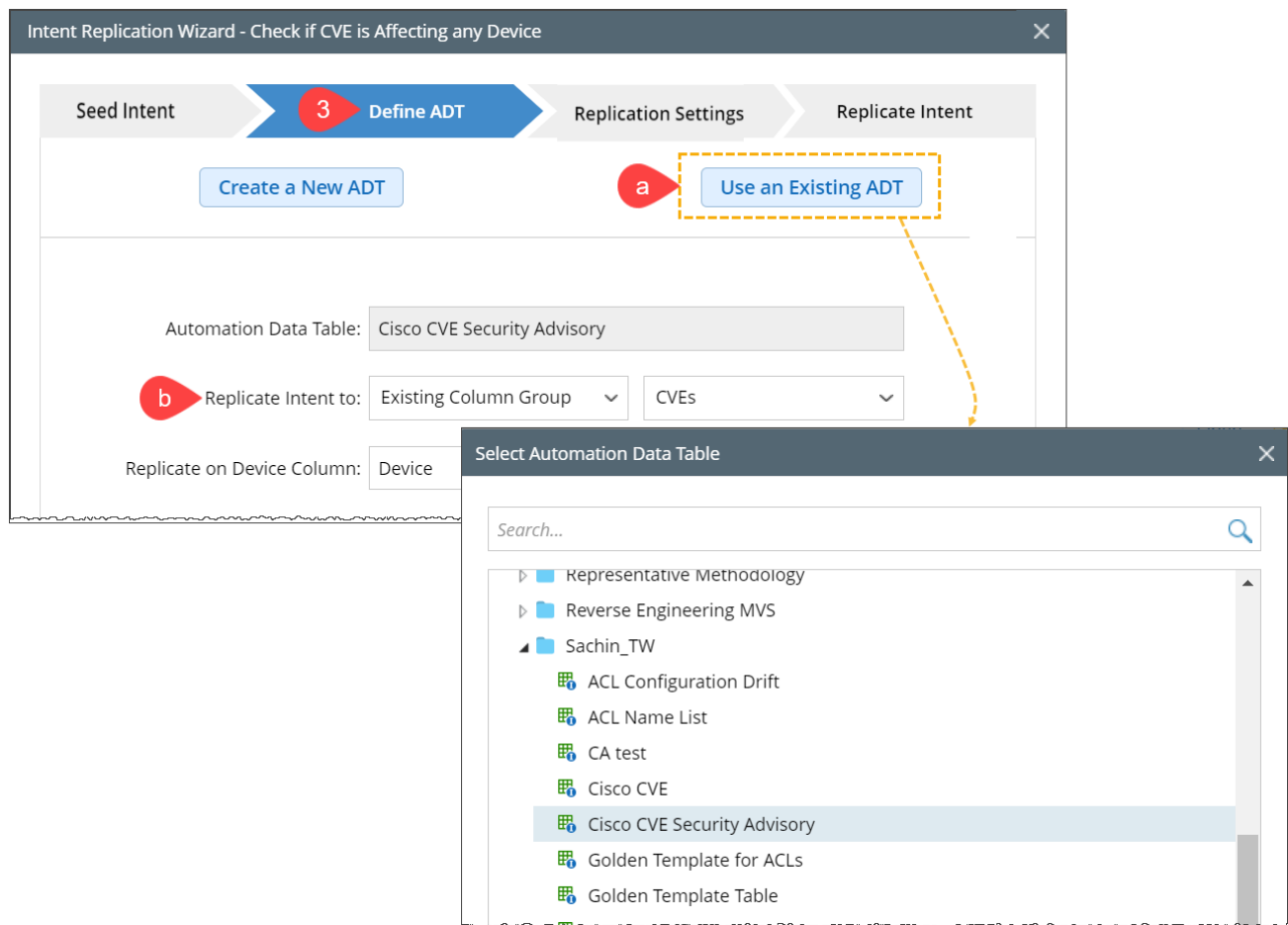
In this section, you will use the **Intent Replication Wizard** to add the Automation Column in the ADT table. You will use the existing ADT that you have created, **Cisco CVE Security Advisory**.

1. In the **Network Intent (View Mode)** window, go to the  menu and open the **Intent Replication Wizard**.
2. In the **Seed Intent** tab, check for your NI (**Cisco CVE Security Advisory**) and then click **Next** to go to the **Define ADT** tab.



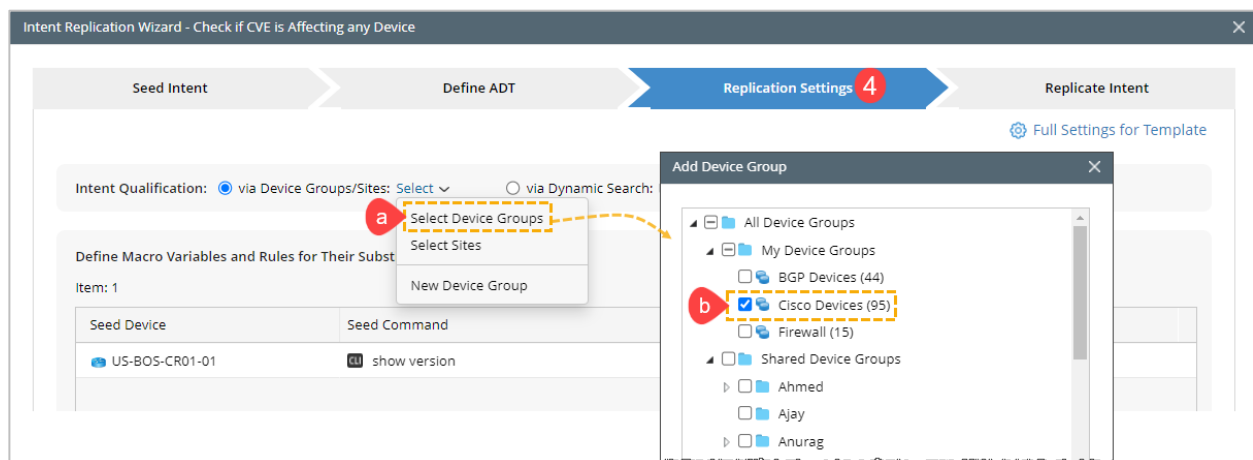
3. Define ADT:

Use an Existing ADT and select **Cisco CVE Security Advisory**. Enter a meaningful Intent group name here.



4. Replication Settings:

- In the **Intent Qualification** section, click the **Select** link to add Device Group. Use the device group that you have used during ADT creation.
- Add the device group and then click **OK**.
- Click **Next** to go to the **Replicate Intent** tab.



5. Replicate Intent:

- In the **ADT Columns** section, you can see the **Column Name** as **Replicated Intent**. You can change this name to **CVE-2023-20198**.
- Click **Save and Replicate** to save all the settings. You can see the **Open Output ADT** option after the successful submission of the replication request.
- Click **Open Output ADT** to check the replicated Intent column in the ADT Manager.

Intent Replication Wizard - Check if CVE is Affecting any Device

Seed Intent > Define ADT > Replication Settings > **5 Replicate Intent**

ADT Columns: Additional Columns ▾

Column Data	Column Name	Tag
1 Replicated Intent	CVE-2023-20198	0 tags

Replication Request submitted at: 08/13/2024 05:34 PM

Selection Mode: Device-based Replication, ADT: Cisco CVE Security Advisory, 0 Macro Variables.

Previous Finish

6. Review the populated Intent column.

Automation Data Table Manager

Cisco CVE Security Advisory Table Builder Last Updated at: 07/09/2024 03:17 PM Rebuild Table Add Data Manually ▾

Description: Type description here...

Items: 95 Rows 3 Columns Search... Advanced Filter: Undefined

No.	Device	Device Type	CVE-2023-20198
1	Berlin-R1	Cisco Router	Check if CVE is Affecting any Device Berlin-R1
2	Berlin-vEdge	Cisco IOS Switch	Check if CVE is Affecting any Device Berlin-vE...
3	DE-MUC-CR01-01	Cisco Router	Check if CVE is Affecting any Device DE-MUC-...
4	DE-MUC-CR01-02	Cisco Router	Check if CVE is Affecting any Device DE-MUC-...
5	DE-MUC-CW01-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE-MUC-...
6	DE-MUC-CW02-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE-MUC-...
7	DE-MUC-CW03-01	Cisco IOS Switch	Check if CVE is Affecting any Device DE-MUC-...
8	ISP-P02	Cisco Router	Check if CVE is Affecting any Device ISP-P02
9	ISP-PE01	Cisco Router	Check if CVE is Affecting any Device ISP-PE01

6.2.3.5 Run Intent Once and Rebuild Table

Run the replicated intents once and rebuild the table. Check the intent status code.

The screenshot shows the 'Table Builder' interface. At the top, there's a header with 'Table Builder', 'Last Updated at: 07/09/2024 03:00 PM', and a 'Rebuild Table' button. Below the header, there's a search bar and an 'Advanced Filter: Undefined' button. The main table has columns for 'Device Type' and 'Intent'. The 'Device Type' column contains 'Cisco Router'. The 'Intent' column contains 'CVE-2023-20198'. A 'Run' button is highlighted with a red circle and a dashed arrow pointing to it. Below the table, there's a 'Run Intents Once - CVE-2023-20198' dialog. The dialog has a 'Data Source: Live Data' section and a 'Only Run Intents if' section with a dropdown menu set to 'Intent Device'. Below this, there's a 'Notification' dialog with the text 'The intents in the current ADT column are executed once now.' and a 'View Details' link. The 'Notification' dialog has 'Cancel' and 'OK' buttons. The 'Run' button in the table has a red circle with the number '1' next to it. The 'OK' button in the 'Run Intents Once' dialog has a red circle with the number '2' next to it. The 'OK' button in the 'Notification' dialog has a red circle with the number '3' next to it.

The screenshot shows the 'Automation Data Table Manager' interface. At the top, there's a header with 'Automation Data Table Manager', 'Cisco CVE Security Advisory', 'Table Builder', 'Last Updated at: 08/11/2024 09:36 PM', and a 'Rebuild Table' button. Below the header, there's a 'Description: Type description here...' field. The main table has columns for 'No.', 'Intent', and 'Status'. The 'Intent' column contains 'CVE-2023-20198'. The 'Status' column contains 'Check if CVE is Affecting'. A 'Rebuild Table' dialog is open, showing 'Build the column groups: All' and 'Log: Production Mode' (selected). The dialog also has 'Only show major execution process log' and 'Show all the detailed log' options. The 'Rebuild Table' button in the header has a red circle with the number '1' next to it. The 'Build' button in the dialog has a red circle with the number '2' next to it.

6.2.4 Create Intent to Check Device Against all CVEs

In this section, you will learn how to create an Intent to **Check Device Against all CVEs**. You can follow the same steps as 6.2.3 with the only following difference while defining diagnosis:

In the diagnosis, you will loop through all CVEs and check whether the affected software versions of this CVE contain the software version of this device:

2. Define Diagnosis

Add Note Add Diagnosis 1 Can also click a variable on the left to add automation.

Name: Check device for all CVEs 2 Anchor: \$ios_ver

Type description of the diagnosis...

☒ Loop Table Rows 3 Cisco_CVE Table Key: Affected_Version

If

A 4 Current US-BOS-C... Current

Affected_Version Equals asa_ver

B Select Variable

If so, raise an alert; if not, create a green info:

Then 5

Diagnosis Message: ☐ Save to Incident

☒ Set Status Code for Device: Error \$this_device software version has \$CVE_ID. Please update.

☒ Set Status Code for Intent: Error \$this_device software version has \$CVE_ID. Please update.

Add Logic

Else 6

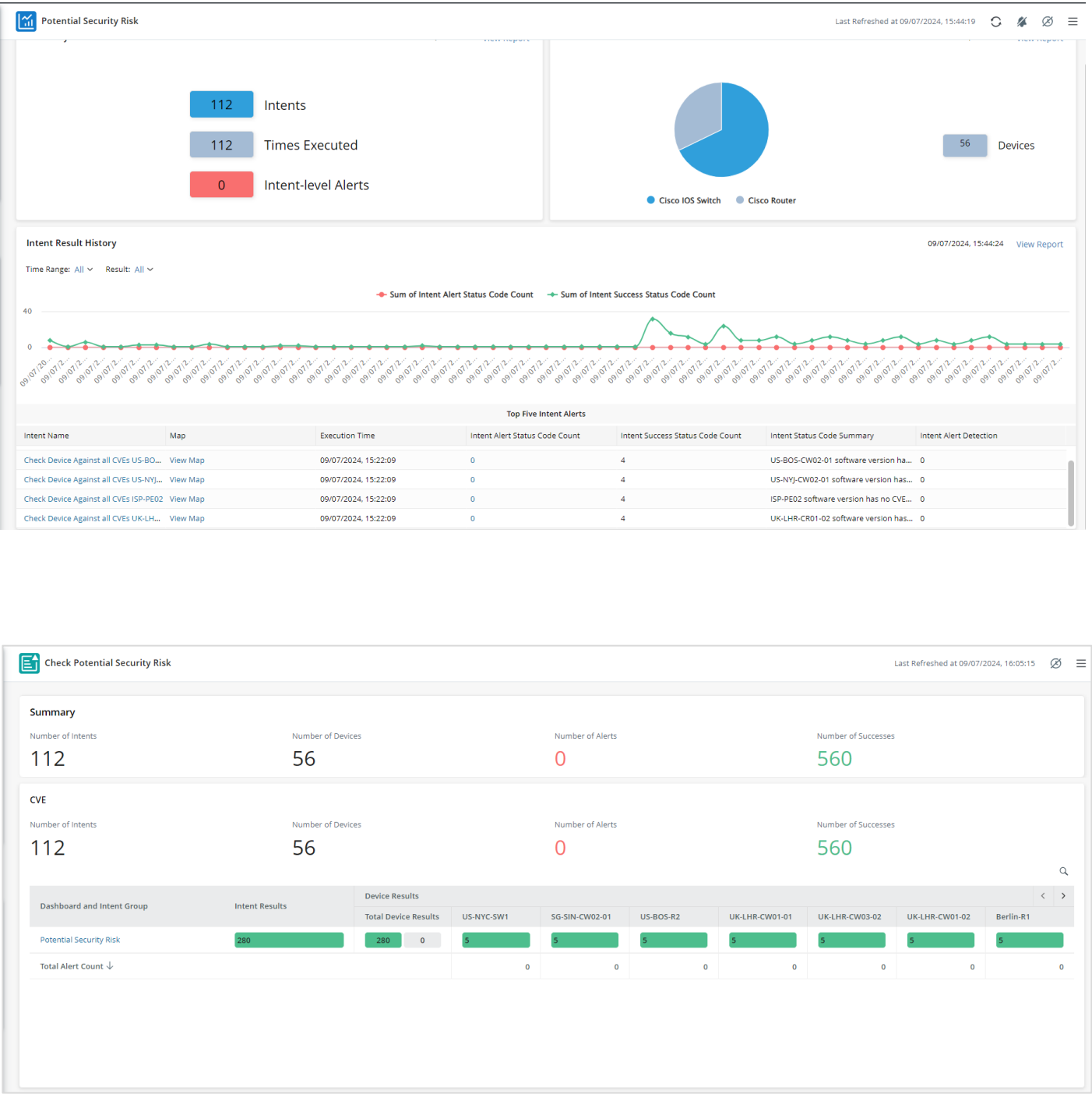
Diagnosis Message: ☐ Save to Incident

☒ Set Status Code for Device: Success \$this_device software version has no \$CVE_ID.

☒ Set Status Code for Intent: Success \$this_device software version has no \$CVE_ID.

6.2.5 Create Intent and Summary Dashboard

Now, you can create the Intent Dashboard and Summary Dashboard to view the results. Follow the same steps as 6.1.7 to create both dashboards and the same results as follows:



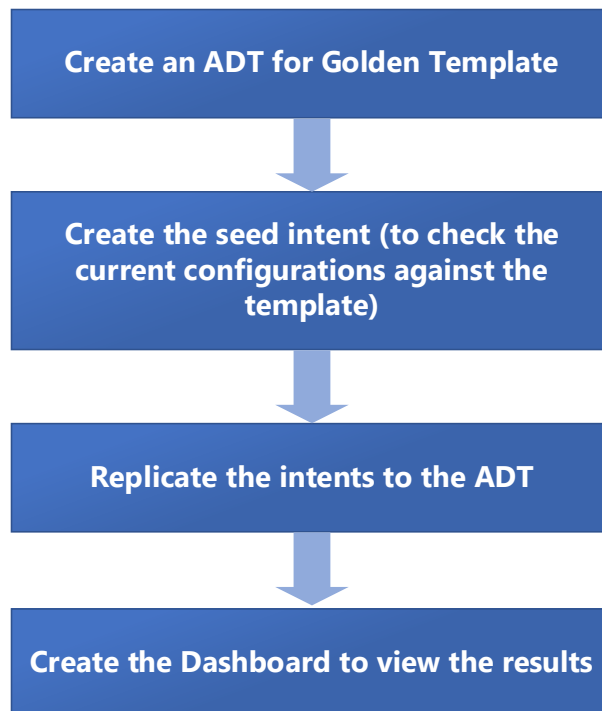
7 Network Assessment Case Study: Configuration Drift

Every network has certain configurations that should not be changed, sometimes called the **Golden Config** or **Golden Template**. The drift from these golden configs can lead to the network outage or performance downgrade. However, the prevention of configuration drift is a complex task since every network has its uniqueness.

In this chapter, you will apply what you have learned in previous chapters to create the automations to find out the configuration drift against the golden template across your whole network. These automations can be scheduled to run for continuous network assessment or run while troubleshooting or making the network change to ensure that the configuration drift does not cause the incident or the network change does not lead to breaking the golden config.

We will use two common examples: the standard ACL and NTP configurations. In the last section, we will demonstrate that the same principle can also be applied to the public cloud configurations.

The flow to check the configuration drift is as follows:



You can also create a CSV report for the configuration drift in the seed intent. After replicating the intent to the ADT, you can create a summary CSV report from the ADT.

7.1 Check Configuration Drift Against the Golden Template

In this session, we will walk you through the flow to check the Golden Template using the standard **ACL** as an example.

The final ADT will be:

Automation Data Table Manager								
ACL Configuration Drift								
Description: Type description here...								
Items: 95 Rows 9 Columns								
No.	Device	Mgmt IP	Vendor	Model	Software Version	ACL Configuration Drift	Intent Status Code	Last Execution Time
1	Berlin-R1	172.16.8.60	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	Berlin-R1: acl_1 golden comp	07/16/2024 06:14:31 ...
2	Berlin-vEdge	192.168.0.1	Cisco	WS-C4500X-32	03.04.04.SG	Check ACL against Golden Templ...	Berlin-vEdge: acl_1 golden cr	07/16/2024 06:14:27 ...
3	DE-MUC-CR01-01	10.20.1.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	DE-MUC-CR01-01: acl_1 gold	07/16/2024 06:14:27 ...
4	DE-MUC-CR01-02	10.20.1.3	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	DE-MUC-CR01-02: acl_1 gold	07/16/2024 06:14:31 ...
5	DE-MUC-CW01-01	10.20.1.4	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW01-01: acl_1 golk	07/16/2024 06:14:31 ...
6	DE-MUC-CW02-01	10.20.1.5	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW02-01: acl_1 golk	07/16/2024 06:14:27 ...
7	DE-MUC-CW03-01	10.20.1.6	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW03-01: acl_1 golk	07/16/2024 06:14:27 ...
8	ISP-PE02	4.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	ISP-PE02: acl_1 golden compli	07/16/2024 06:14:31 ...
9	ISP-PE01	1.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	ISP-PE01: acl_1 golden comp	07/16/2024 06:14:27 ...
10	ISP-PE02	2.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	ISP-PE02: acl_1 golden comp	07/16/2024 06:14:27 ...
11	ISP-PE03	3.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	ISP-PE03: acl_1 golden comp	07/16/2024 06:14:31 ...
12	ITE_EXTEND	192.168.30.207	Cisco	WS-C3560X-48P	15.2(4)E7	Check ACL against Golden Templ...	ITE_EXTEND: acl_1 golden co	07/16/2024 06:14:31 ...
13	JP-TYO-CR01-01	10.30.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	JP-TYO-CR01-01: acl_1 golder	07/16/2024 06:14:31 ...
14	JP-TYO-CR01-02	10.30.0.3	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	JP-TYO-CR01-02: acl_1 golder	07/16/2024 06:14:26 ...
15	JP-TYO-CW01-01	10.30.0.4	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	JP-TYO-CW01-01: acl_1 g	07/16/2024 06:14:26 PM
16	JP-TYO-CW01-02	10.30.0.5	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	JP-TYO-CW01-02: acl_1 g	07/16/2024 06:14:27 PM
17	JP-TYO-CW02-01	10.30.0.6	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	JP-TYO-CW02-01: acl_1 g	07/16/2024 06:14:27 PM
18	JP-TYO-CW03-01	10.30.0.7	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	JP-TYO-CW03-01: acl_1 g	07/16/2024 06:14:25 PM


The summary CSV report will be:

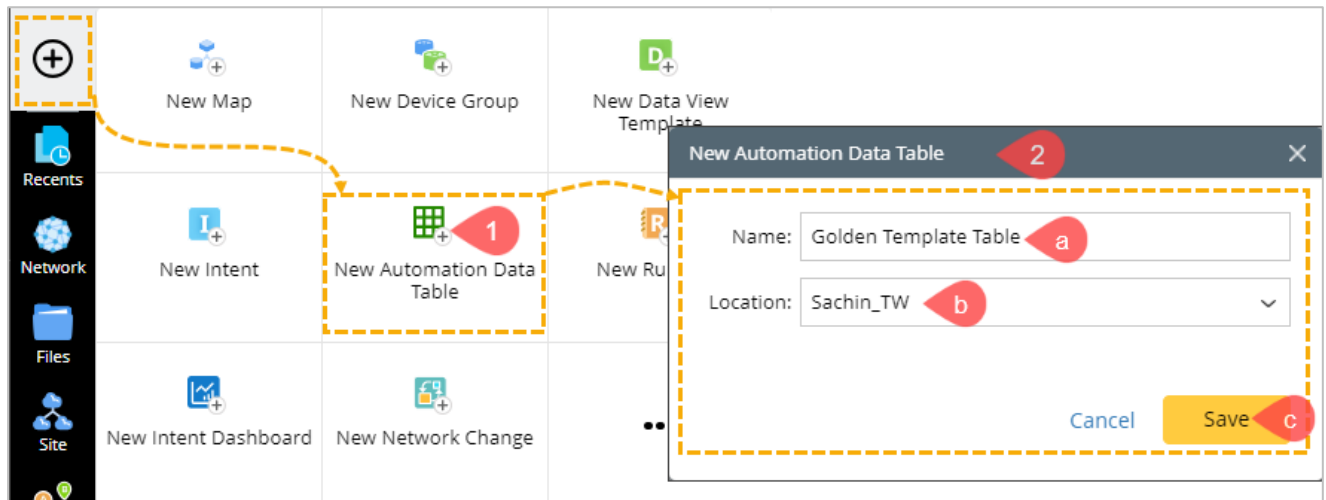
	A	B	C	D	E	F	G	H	I
1	Device	Golden Config	Current Config	Matched	Missing Lines	Extra Lines	Vendor	Model	Software Version
2	US-NYC-SW1			TRUE			Cisco	3560E	15.2(CML_NIGHTLY_20180510)FLO_DSGS7
3	UK-LHR-CW02-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
4	US-BOS-SW3			TRUE			Cisco	3560E	15.2(HI_20170202)FLO_DSGS7
5	US-NYJ-CR01-01		access-list 1	FALSE	access-list 1		Cisco	CGS-MGS-AGS	15.4(2)T4
6	SG-SIN-CW01-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
7	JP-TYO-CW01-02		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
8	SG-SIN-CW03-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
9	US-BOS-CW01-02	access-list 1	access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
10	SG-SIN-CW02-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
11	US-BOS-CW02-02	access-list 1	access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
12	JP-TYO-CW02-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
13	UK-LHR-CW01-01			TRUE			Cisco	Catalyst 38xx Stack	15.2(20170809:194209)
14	US-BOS-SW2			TRUE			Cisco	3560E	15.2(HI_20170202)FLO_DSGS7
15	US-NYJ-CW02-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
16	ISP-PE01			TRUE			Cisco	CGS-MGS-AGS	15.4(2)T4
17	DE-MUC-CW01-01		access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
18	US-BOS-CW04-01	access-list 1	access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
19	US-BOS-SW4			TRUE			Cisco	3560E	15.2(HI_20170202)FLO_DSGS7
20	US-BOS-CW04-02	access-list 1	access-list 1	FALSE	access-list 1		Cisco	3560E	15.2(20170809:194209)
21	Berlin-vEdge			TRUE			Cisco	WS-C4500X-32	03.04.04.SG
22	US-BOS-CR01-02		access-list 1	FALSE	access-list 1		Cisco	CGS-MGS-AGS	15.4(2)T4
23	DE-MUC-CR01-02		access-list 1	FALSE	access-list 1		Cisco	CGS-MGS-AGS	15.4(2)T4

7.1.1 Prerequisites

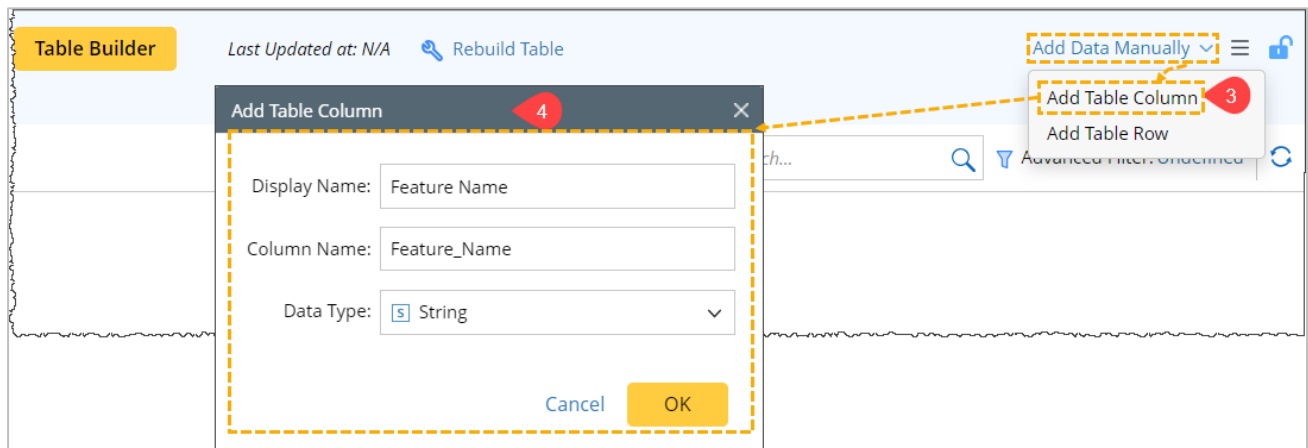
You can manually create an ADT to store your Golden Template for the standard ACLs. If you are not sure what configurations you can use as the standard ACLs, please refer to Section 7.2 on how the NetBrain system can help you identify these configurations.

To create a Golden Template table manually,

1. Click the plus icon  and click **New Automation Data Table**.
2. In the New Automation Data Table popup:
 - a) Enter the ADT Name, e.g., **Golden Template Table**.
 - b) Select the **Location** you wish to store the ADT.
 - c) Click **Save** and wait a moment for ADT to open.



3. In the Automation Data Table Manager, click the Add Data Manually dropdown, and then select the Add Table Column.
4. In the **Add Table Column** popup, define the **Display Name** as **Feature Name**, keep the data type as the default value (**String**) and click **OK**:



Similarly, create three more columns: **Golden Configuration**, **Site**, and **Rep Device**. The golden configuration of an access list can be different for the different sites.

5. Click the **Add Data Manually** dropdown and click **Add Table Row**.
6. In the **Add Table Row** popup, enter the column values for all columns, then click **OK**.

Note: This data is for illustration purposes only; you should choose the best golden configuration for your network.

- a) Feature Name: ***acl_1***
- b) Golden Configuration:
access-list 1 permit 192.168.0.0 0.0.255.255
access-list 1 permit 10.0.0.0 0.255.255.255
- c) Site: ***My Network\NA\US-BOS*** (you can add more sites).
- d) Rep Device: ***US-BOS-CW*** (its configuration of ACL 1 is used as a golden configuration.)

The screenshot shows the 'Table Builder' interface with a 'Last Updated at: N/A' status and a 'Rebuild Table' button. A dropdown menu 'Add Data Manually' is open, showing 'Add Table Column' and 'Add Table Row' (highlighted with a red circle 5). The 'Add Table Row' popup (highlighted with a red circle 6) contains a table with the following data:

Column	Value
Feature Name	acl_1 (a)
Golden Configuration	access-list 1 permit 192.168.0.0 0.0.255.255 access-list 1 permit 10.0.0.0 0.255.255.255 (b)
Site	My Network\NA\US-BOS (c)
Rep Device	US-BOS-CW (d)

At the bottom of the popup are 'Cancel' and 'OK' buttons. The background interface shows a table with a 'Rep Device' column.

7. Similarly, add another row for the **acl_10** feature.

Add Table Row

Column	Value
<div>Feature Name</div>	<div>acl_10</div>
<div>Golden Configuration</div>	<div>access-list 10 permit 8.8.8.8 access-list 10 permit 8.8.4.4</div>
<div>Site</div>	<div>My Network\NAUS-BOS</div>
<div>Rep Device</div>	<div>US-BOS-CW</div>

Cancel

OK

The final Golden Template Table will be:

Automation Data Table Manager

Golden Template Table

Table Builder

Last Updated at: N/A

Rebuild Table

Add Data Manually

Description: Type description here...

Items: 2 Rows 4 Columns

Search...

Advanced Filter: Undefined

No.	Feature Name	Golden Configuration	Site	Rep Device
1	acl_1	access-list 1 permit 192.168.0.0 (My Network\NAUS-BOS	US-BOS-CW
2	acl_10	access-list 10 permit 8.8.8.8	My Network\NAUS-BOS	US-BOS-CW

7.1.2 Create Intent to Check ACL Against Golden Template

In this section, you will create a seed Intent (**Check ACL Against Golden Template**) to check the configuration drift of ACLs against those you just defined in 7.1.1. Later, you will use the **Intent Replication Wizard** to replicate to all devices you want to check the configuration drift.

From the **Intent Manager**, create a new Intent, **Check ACL against Golden Template**, and choose **US-BOS-CW-03-01** as the seed device.

7.1.2.1 Define Variables with Visual Parser

In this section, you will learn how to parse the Variable for ACL list configuration using Variable Operation, **LinesByKeyword**.

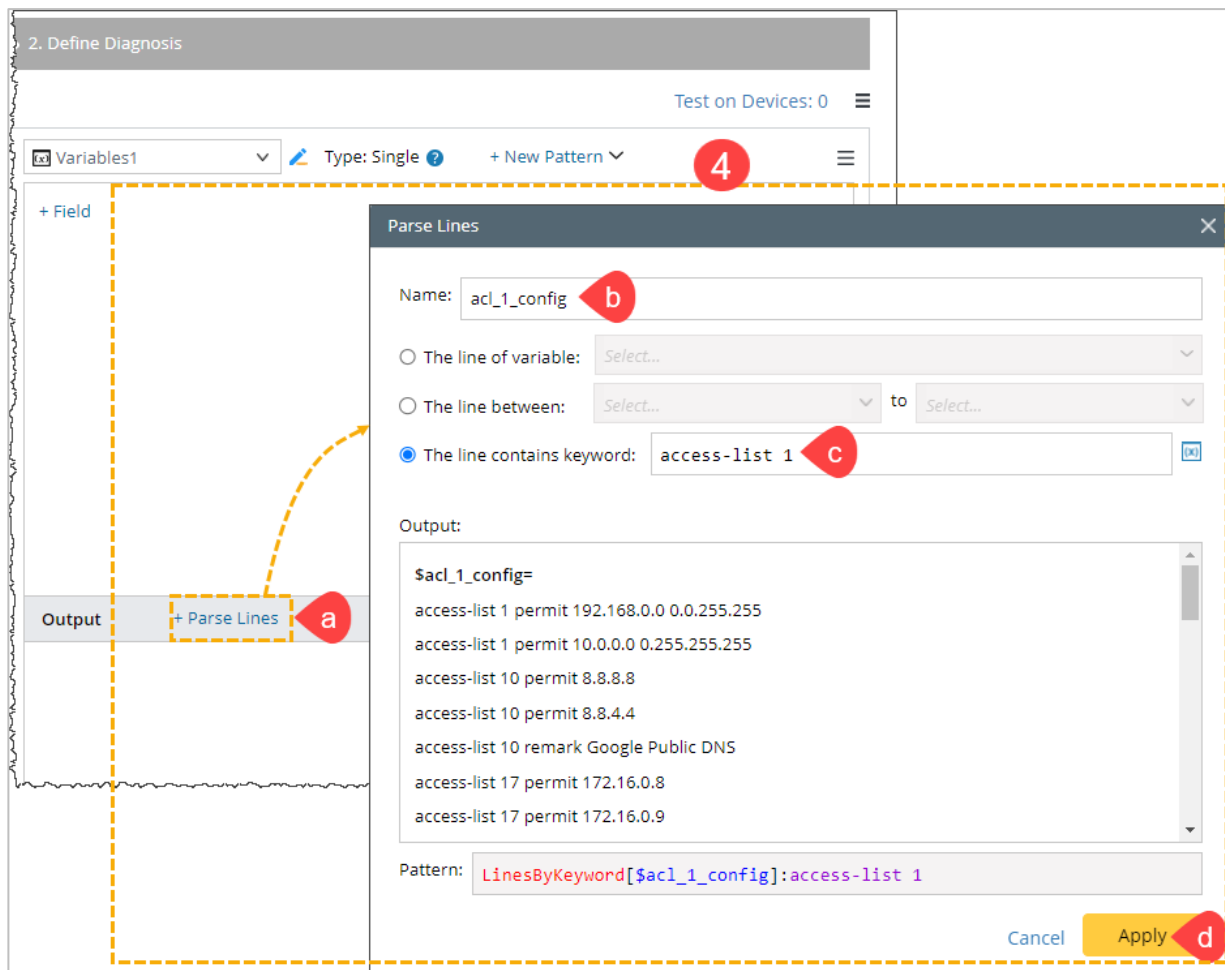
1. Click **+ Add Config Diagnosis** to open the Configuration Diagnosis window.
2. Click **Retrieve** to retrieve the data from the **Live Data**.

The source data is displayed in the **Define Variable** pane. You can edit this data depending on the use case.

3. In the **+ New Pattern** dropdown, select **Single Variable** parser.

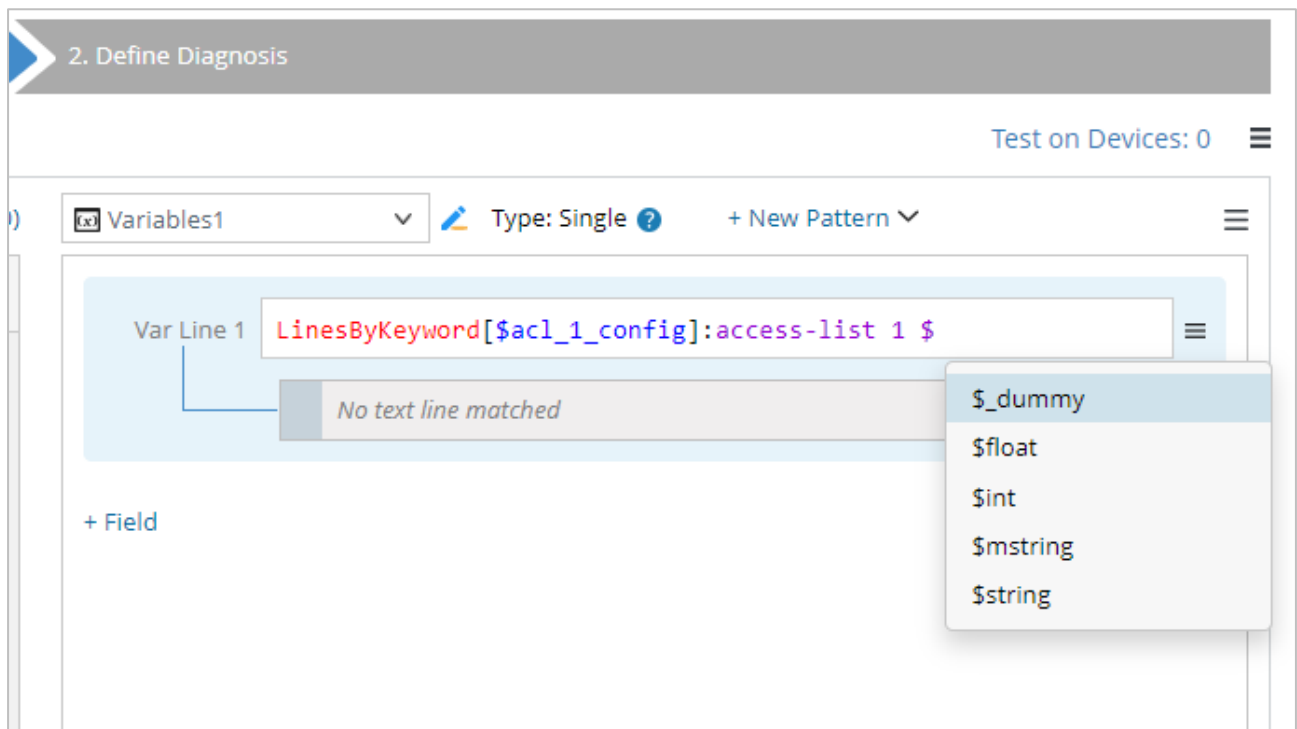
The screenshot displays two overlapping windows from the NetBrain interface. The top window, titled 'Network Intent (Edit Mode)', shows an intent named 'Check ACL against Golden Template'. It includes buttons for 'Run', 'with Live Data', 'Save', and 'Help'. A red circle with the number '1' points to the '+ Add Config Diagnosis' button. The bottom window, titled 'Configuration Diagnosis', shows the 'Define Variable' pane with a list of ACL configurations. A red circle with the number '2' points to the 'Retrieve' button, and a red circle with the number '3' points to the 'Single Variable' option in the '+ New Pattern' dropdown. A yellow callout labeled 'Sample data' points to the ACL configuration text in the 'Define Variable' pane.

4. Define **Var Line 1** to retrieve the configurations of **access-list 1**, which may include multiple lines starting with the keyword **access-list 1**. For this type of text, you can use the function **LinesByKeyword** to extract all data:
 - a) Click **+ Parse Lines** in the **Output** pane, and the **Parser Lines** window appears.
 - b) Set the Variable name as **acl_1_config**.
 - c) Select **The line contains keyword** option, and enter the keyword, **access-list 1**.
 - d) Click **Apply** to close the **Parser Lines** window. The result includes the configurations of other access lists, such as access-list 10, since these lines also include the keyword **access-list 1**.



- e) To get the exact match for the access-list 1, you can modify the **Var Line 1** by adding a space after 1 (which excludes the numbers 10-19) and add the **\$_dummy** (to match any text after the keyword) at the end of the pattern.

The final Var Line 1 will be: **LinesByKeyword[\$acl_1_config]:access-list 1 \$_dummy**

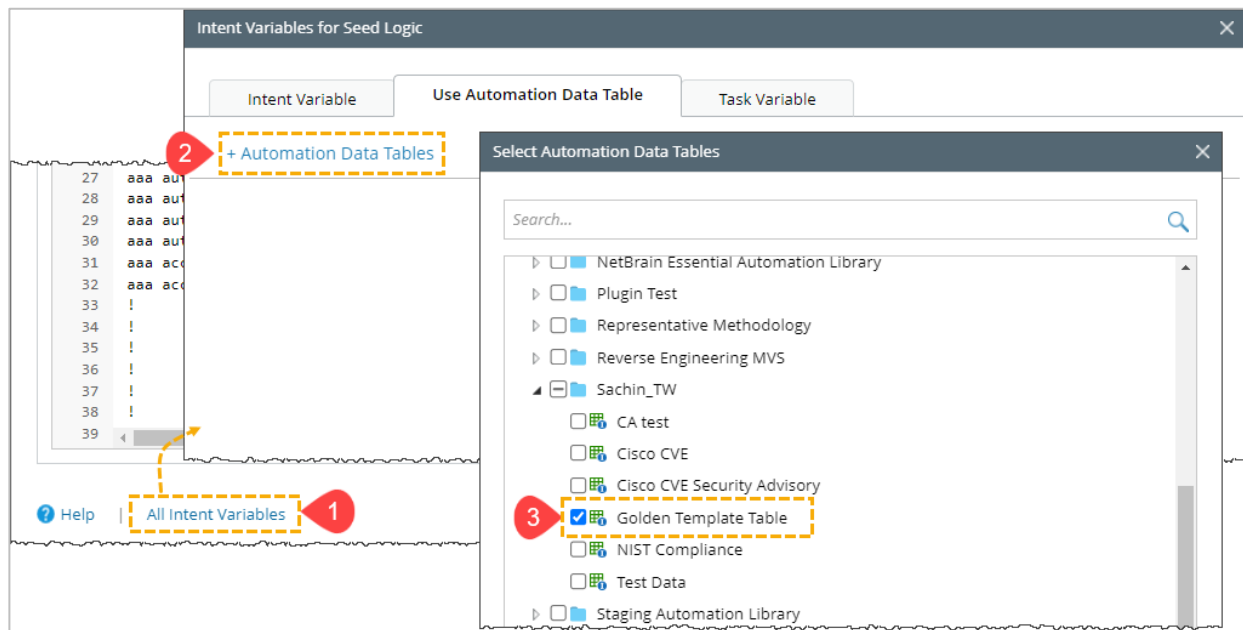


- f) Similarly, create **Var Line 2** for the **access list 10**: **LinesByKeyword[\$acl_10_config]:access-list 10 \$_dummy**.
- g) Click **Apply** to save the parser settings.

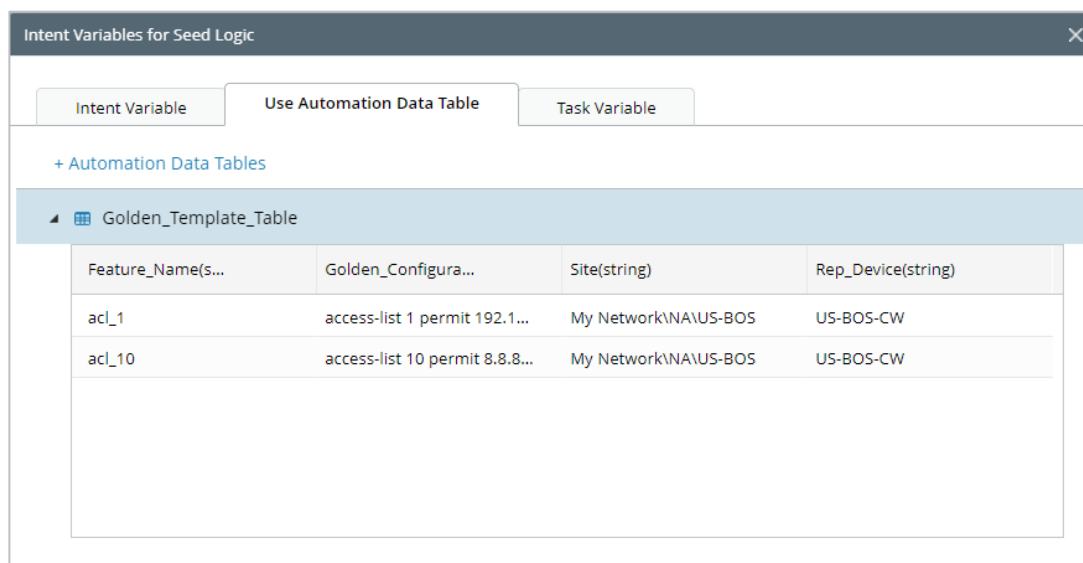
7.1.2.2 Add ADT Table as Intent Variables

You have parsed the configurations of the access lists. In the diagnosis, you will compare them against the golden configurations defined in the ADT. In order for an intent to refer to the ADT elements, you need to add the ADT table as **Intent Variables**:

1. Click **All Intent Variables** in the bottom-right corner.
2. Go to the Use Automation Data Table tab and click + Automation Data Table to select ADT.
3. From the **Select Automation Data Table** window, select the **Golden Template Table** ADT and then click **OK**.




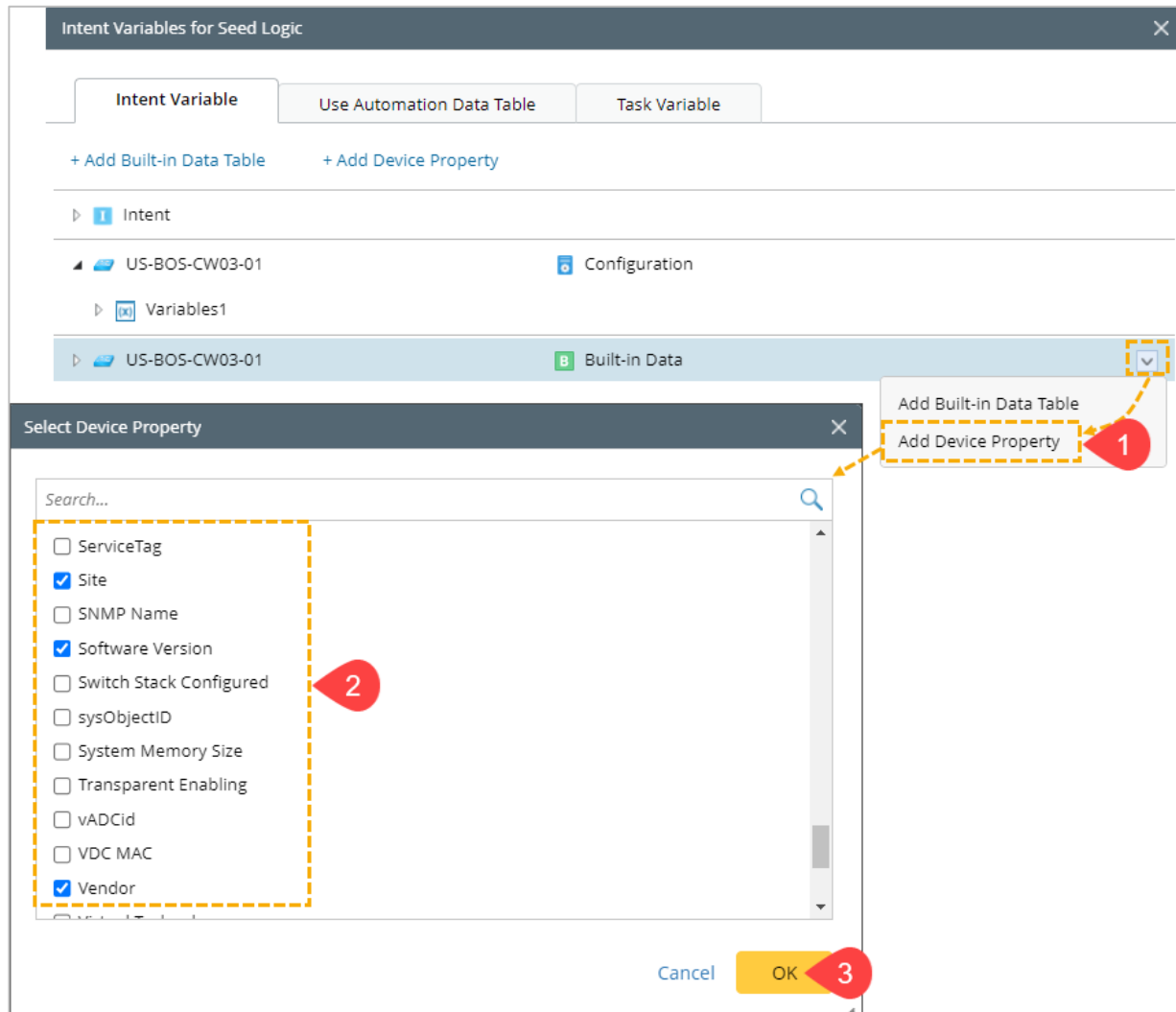
The data of the ADT table can be used in the intent diagnosis.



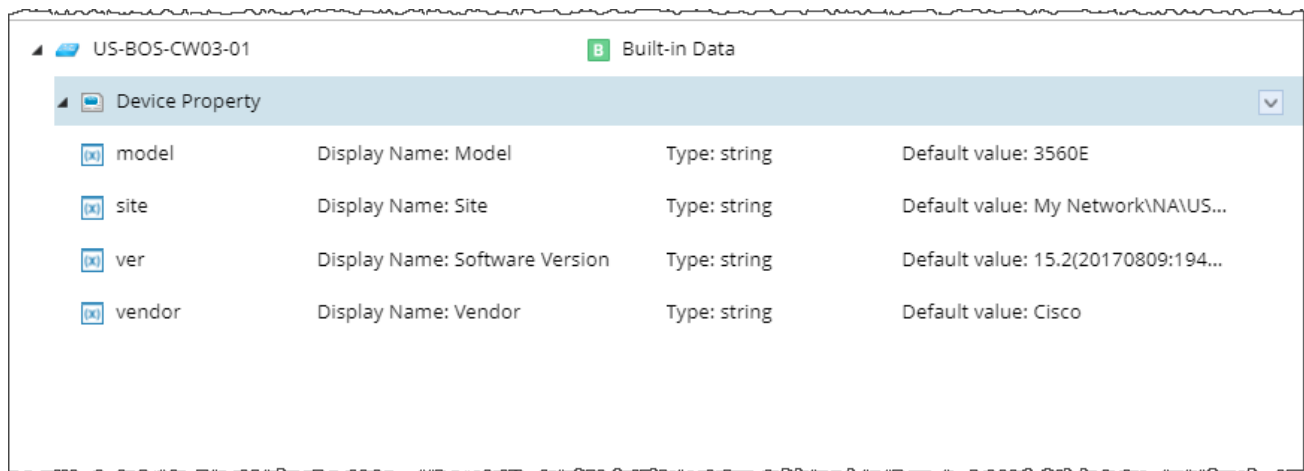
7.1.2.3 Add Built-in Device Properties

Similarly, you can add the **Built-in Device Properties** to the Intent Variable so that the intent can use these variables. For example, adding the model and software version to a report can be useful.

1. Click  of the Device **US-BOS-CW03-01**, and click **Add Device Property** from dropdown.
2. From the **Select Device Property** window, select the properties you wish to add. For example, add the **Site**, **Software Version**, **Vendor**, and **Model**.
3. Click **OK** to add device properties.



The Device properties list will be:



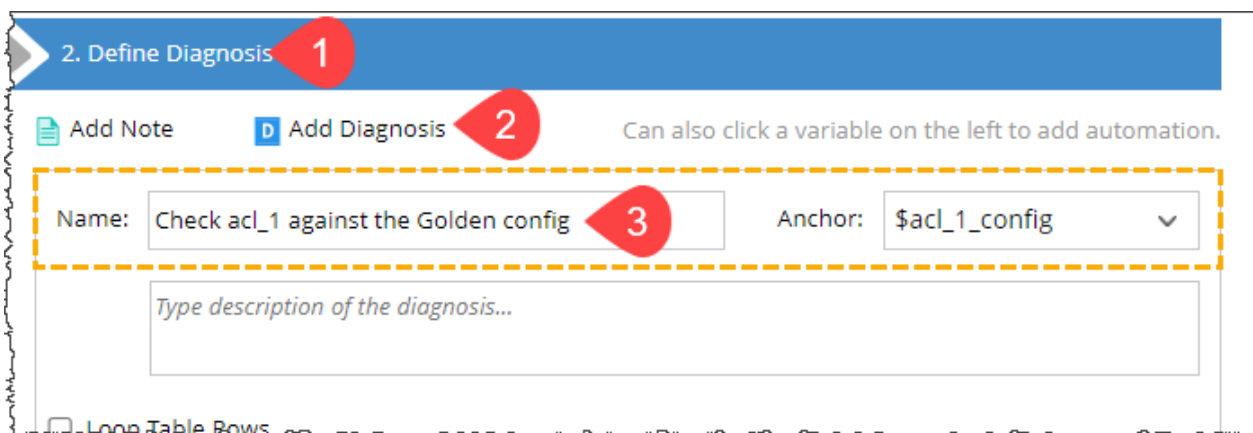
US-BOS-CW03-01		8 Built-in Data	
Device Property			
model	Display Name: Model	Type: string	Default value: 3560E
site	Display Name: Site	Type: string	Default value: My Network\NA\US...
ver	Display Name: Software Version	Type: string	Default value: 15.2(20170809:194...
vendor	Display Name: Vendor	Type: string	Default value: Cisco

7.1.2.4 Define Diagnosis

In the diagnosis, you will compare the configurations you defined in 7.1 against the golden configurations defined in the ADT. You will use two useful functions:

- **Match Pattern (MP)**, which compares two strings line by line and reports the differences, including the changed, added and removed lines.
- **Get_Table_Cell**, which will retrieve a cell value from an ADT table.

You will create a diagnosis per access list. We will use access-list 1 as an example and leave access-list 10 for your homework:



2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: Check acl_1 against the Golden config Anchor: \$acl_1_config

Type description of the diagnosis...

Loop Table Rows

Define **If** Condition as follows:

☐ Loop Table Rows

▼ **If**

A US-BOS-CW03-... Current ▼

acl_1_config ▼ MP(Rule1) ▼ Get_Table_Cell(Golden_T ▼

B

Get_Table_Cell(Golden_T ▼ Is not empty ▼

C Select Variable ▼

Boolean Expression: A and B

Define **If** condition **A**:

- Select Variable **acl_1_config**.
- Select **Match Pattern** from the dropdown, uncheck the **Ignore Order of Lines** checkbox (the order of configurations matters for the access list), and then click **OK**. You can leave the rule name as the default or change the name if you have multiple match pattern rules in one diagnosis.

☐ Loop Table Rows

▼ **If**

A US-BOS-CW03-... Current ▼

a acl_1_config ▼ Equals ▼ ▼

B Select Variable ▼ ▼

☐ Save to Incident

Match Pattern

Rule Name: Rule1

Compare Paragraph by: Exact Match ▼

Current string and pattern have identical matched lines

☐ Ignore Order of lines

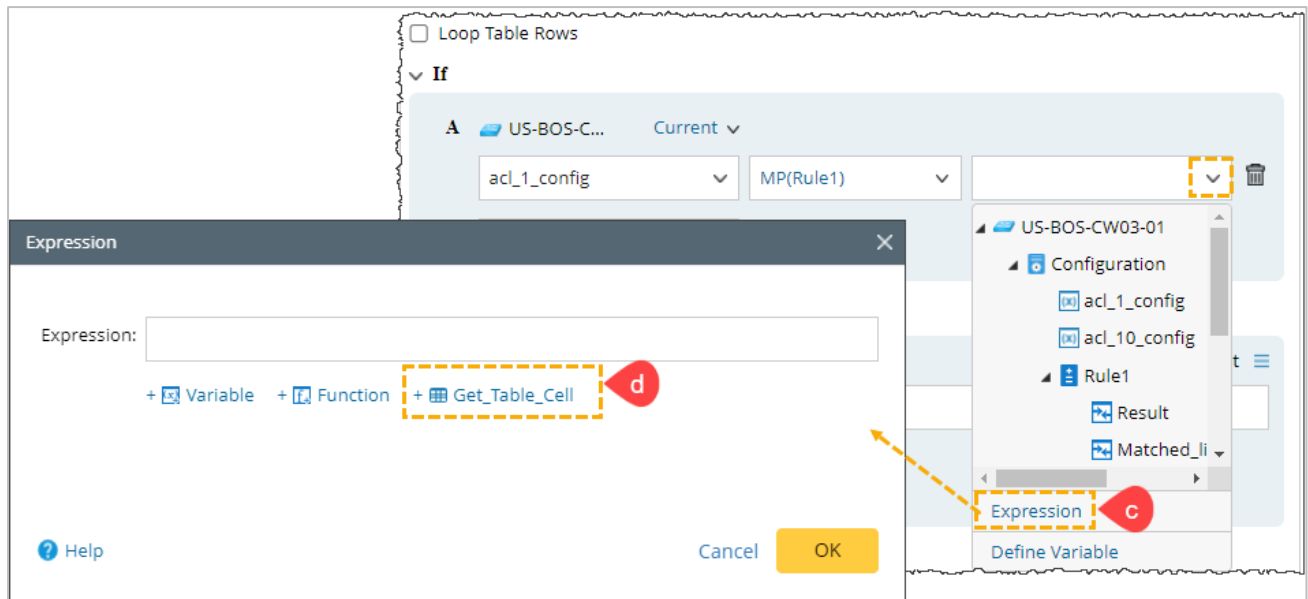
Compare Each Line By: ☒ Equal ☐ Contain

The line is identical to an expanded pattern line

[Learn more about Match Pattern rule](#) Cancel OK

b Match pattern

- c) To retrieve the golden configuration from the ADT table, select **Expression** from the dropdown.
- d) In the **Expression** popup, click the + **Get_Table_Cell** function.



- e) In the **Get Table Cell** window, select the ADT table, column, and condition to retrieve a table cell:
 - i. Select the table **Golden_Template_Table**.
 - ii. Select the Column **Golden_Configuration**.
 - iii. Define Row Matching Condition:
 - A: **Golden_Configuration** Equals "**acl_1**".
 - B: **Site** Contains **site**
 - Boolean Expression: **A and B**

The Final Expression will be:

Get_Table_Cell (Golden_Template_Table, Golden_Configuration, <Condition: A and B>)

The screenshot shows the 'Get Table Cell' dialog box. It has a title bar with a close button (X) and a red circle with the letter 'e'. The main area is titled 'Retrieve Cell Value from:'. It contains a 'Select Table:' dropdown menu with 'Golden_Template_Table' selected, annotated with a red circle 'i'. Below it is a 'From Column:' dropdown menu with 'Golden_Configuration' selected, annotated with a red circle 'ii'. A dashed box labeled 'From the Row Matching Condition:' contains three rows of conditions, annotated with a red circle 'iii':

	Field	Operator	Value	Action
A	Golden_Configuration	Equals	"acl_1"	Trash icon
B	Site	Contains	site	Trash icon
C	Select Variable			

Below the conditions is a 'Boolean Expression:' field containing 'A and B'. At the bottom, the 'Expression:' field shows the full function call: 'Get_Table_Cell (Golden_Template_Table, Golden_Configuration, <Condition: A and B>'. There are 'Cancel' and 'OK' buttons at the bottom right, with 'OK' highlighted by a red dashed box.

4. Define **If** Condition **B**.

Here you want to test whether the golden-config is defined or not. If it is not defined, then we should not create any message. Follow the same instructions of step 1 to use the function **Get_Table_Cell** to retrieve the golden template for the **access-list 1** and check that it is not empty.

The final expression will be:

Get_Table_Cell (Golden_Template_Table, Golden_Configuration, <Condition: A and B>) is not empty

5. Define Intent Output message.

Enter a message under the **Then** and **Else** output areas to appear as the result of the diagnosis.

5

Then **a**

Diagnosis Message: ☐ Save to Incident

☒ ☐ Set Status Code for Device:

☒ ☐ Set Status Code for Intent:

Add Logic

Else **b**

Diagnosis Message: ☐ Save to Incident

☒ ☐ Set Status Code for Device:

☒ ☐ Set Status Code for Intent:

Delete

You need to set the unmatched lines rule to check for missing IP addresses by comparing them with the Golden configuration table.


The diagnosis message will be: ***\$this_device: acl_1 golden compliance failed. Rule \$Rule1.Result, Missing Lines \$Rule1.Unmatched_lines.***

7.1.3 Modify Intent to Create CSV File

Often, you want to export the results into a CSV file. In this section, you will learn how to do this. Since we want to export the golden config to the CSV file, we will create an intent variable to hold the value first.

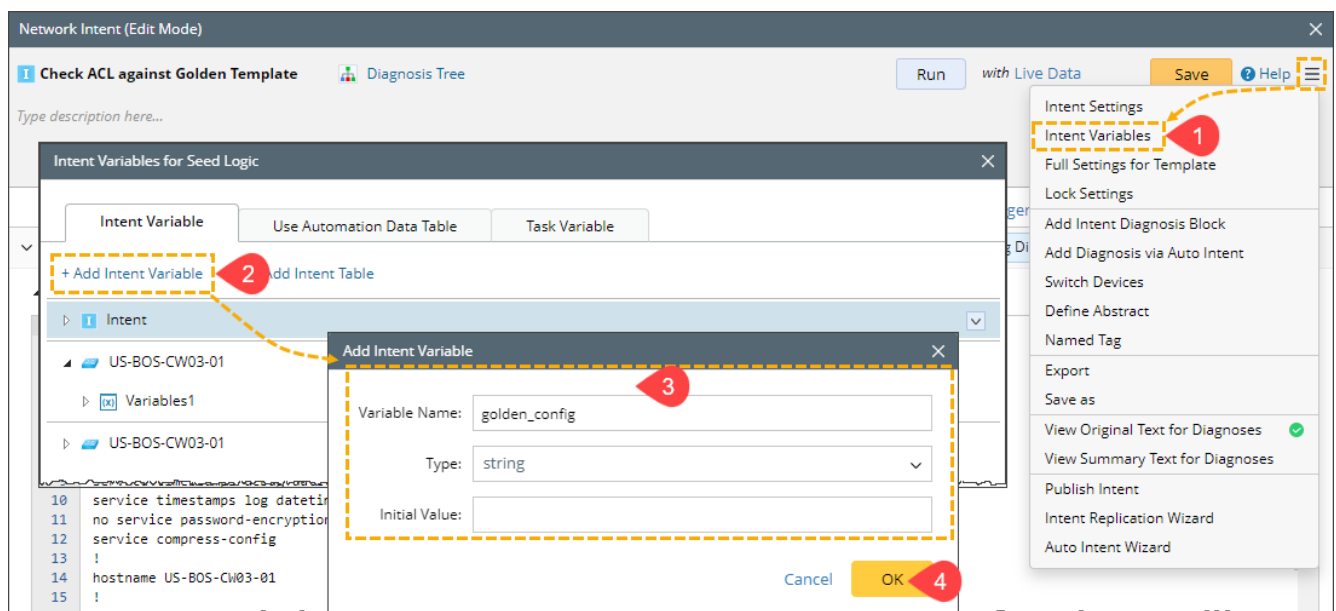
7.1.3.1 Add Intent Variable

To add an Intent variable:

1. In the **Network Intent (Edit Mode)** window, go to the  menu and click **Intent Variables**.
2. In the Intent Variables for Seed Logic window, click the **Intent** section and then click **+ Add Intent Variable**.
3. In the **Add Intent Variable** popup fields, enter the following values:

Fields	Value
Variable Name	golden_config
Type	string
Initial Value	no value

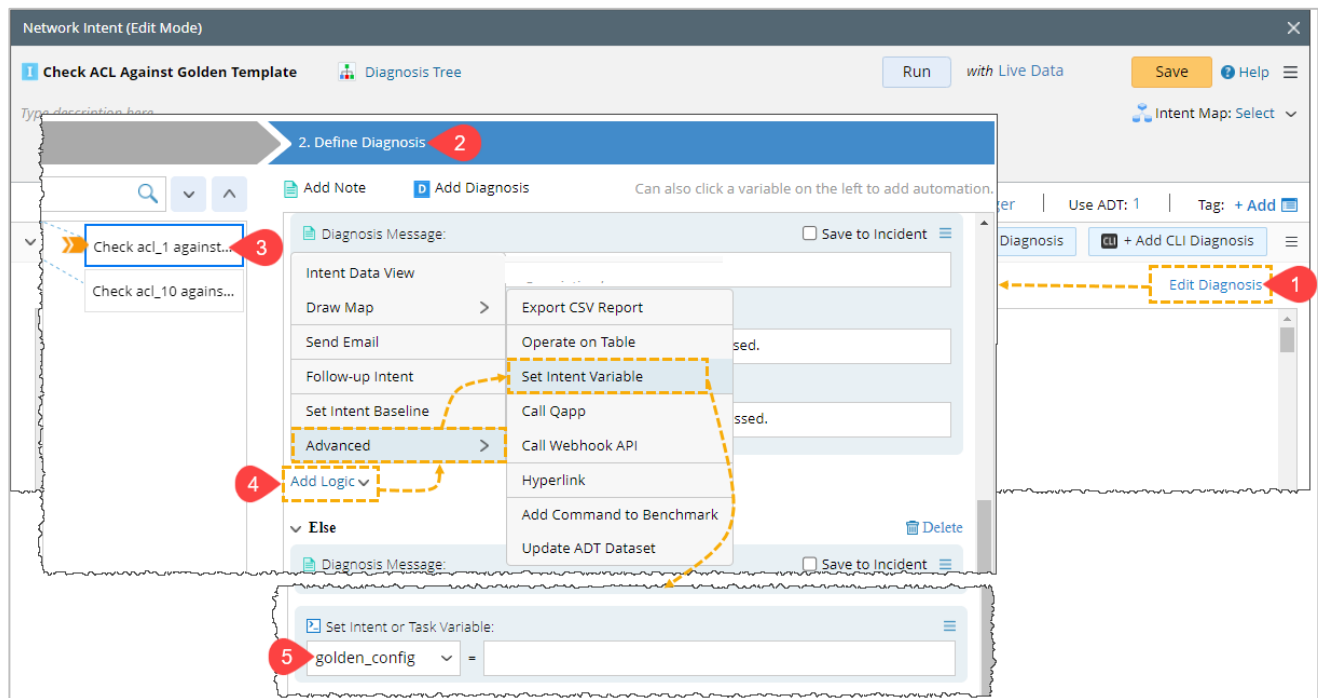
4. Click **OK** to save the Intent Variable.
5. Click **Close** to exit from the Intent Variable for Seed Logic window.



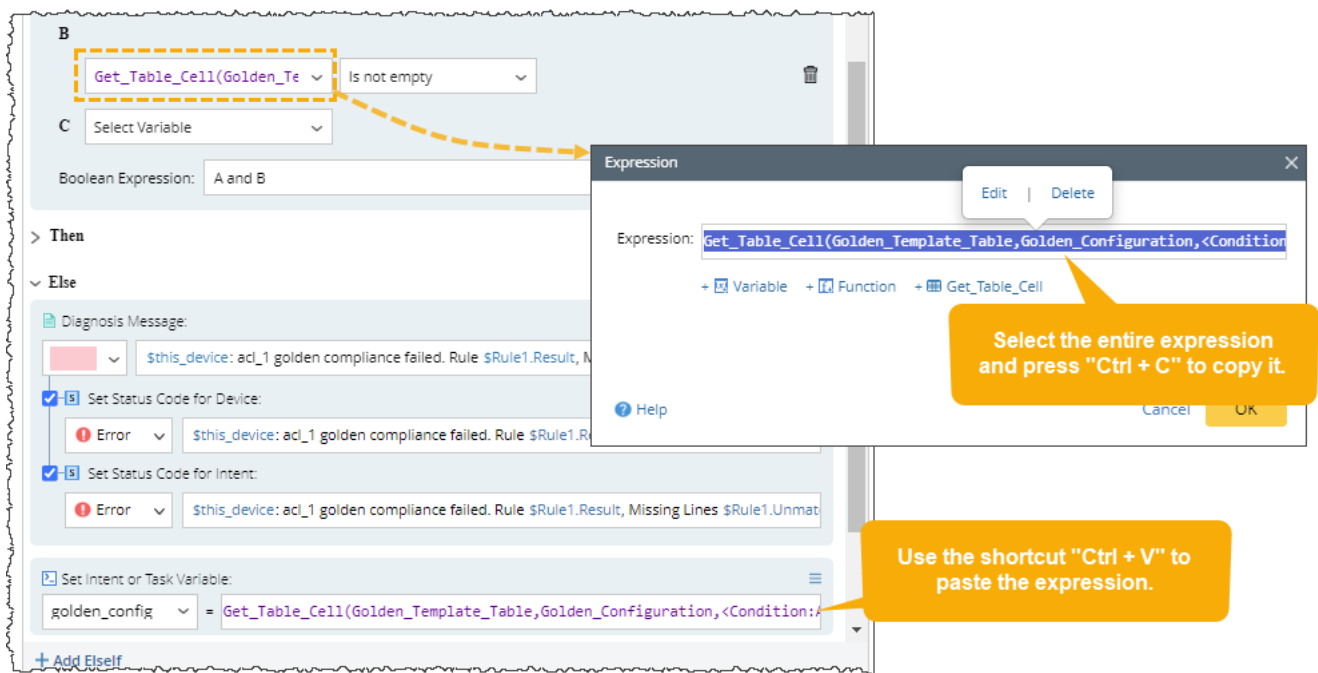
7.1.3.2 Set Intent Variable in Diagnosis

To set the value of an Intent variable,

1. Click **Edit Diagnosis** to set the intent variable under the **Define Diagnosis** tab.
2. Click **Define Diagnosis** to access your created diagnosis.
3. Open the **Check acl_1 against the Golden config** diagnosis to set the Intent Variable.
4. In the **Add Logic** dropdown, select **Advanced > Set Intent Variable**.
5. In the **Set Intent or Task Variable** dropdown, select **Golden_config**.



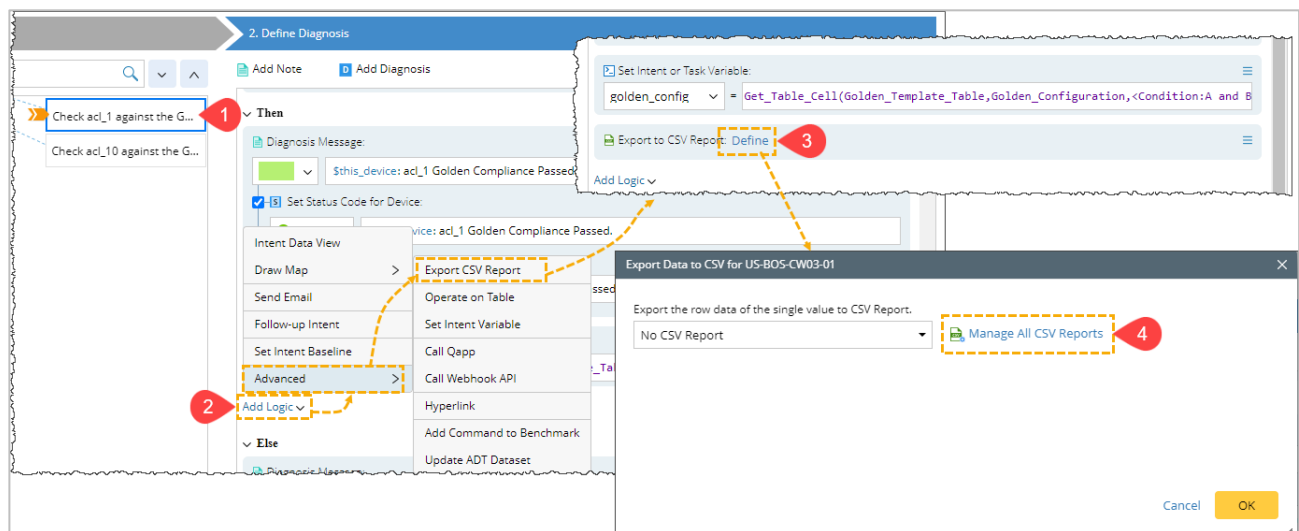
6. To add the variable value, you will use the **Get_Table_Cell** expression. Since you already have the function definition earlier, you can copy and paste it here (using the shortcuts Ctrl + C and Ctrl + V) instead of going through the UI: **Get_Table_Cell (Golden_Template_Table, Golden_Configuration, <Condition: A and B>).**



7.1.3.3 Add Logic to Export CSV File

In this section, you will learn how to export the following columns to a CSV file: **Device, Golden Config, Current Config, Matched, Missing Lines, Extra Lines, Vendor, Model, and Software Version**:

1. Open the **Check acl_1 against the Golden config** diagnosis.
2. In the **Add Logic** dropdown, select **Advanced > Export CSV Report**.
3. Click **Define** to open **Export Data to CSV for <device_name>**.
4. Click **Manage All CSV Reports** to open the Intent Settings window.



5. Add a CSV file.
 - a) Enter the CSV Name, e.g., *ACL Configuration Drift*.
 - b) Enter Columns name separated by commas, e.g., Device, Vendor, Model, Software Version, Golden Config, Device Config, Matched, Missing Lines, Extra Lines.
 - c) Select the Save CSV Reports to Files checkbox.
 - d) Click the **Browse** button to select the **Location** to save the CSV files.
 - e) Click **OK** to save the settings.

Intent Settings

Intent Map and Data View Execution Settings Embedded Incident **CSV Report Files** Follow-up Intent

Define the CSV report files

+ Add CSV 5

* CSV Name: ACL Configuration Drift_acl1 a

* Columns: Device, Golden Config, Current Config, Matched, Missing Lines, Extra Lines, Vendor, Model, Software Version b

☒ Save CSV Report to Files c

* Location: Private Browse d

* CSV File: ACL Configuration Drift_acl1

The CSV file name will be reset as file name + intent name when this intent is used as intent template/intent cluster.

Cancel OK e

6. Select CSV Report name ACL Configuration Drift from the dropdown.

Export Data to CSV for US-BOS-CW03-01

Export the row data of the single value to CSV Report.

ACL Configuration Drift_acl1 6 Manage All CSV Reports

Define the column mapping from the single value to the selected CSV Report. 7

Device	Golden Config	Current Config	Matched	Missing Lines	Extra Lines	Vendor	Model	Software Version
\$this_device	golden_config	acl_1_config	Result	Unmatched_lir	Unused_patter	vendor	model	ver

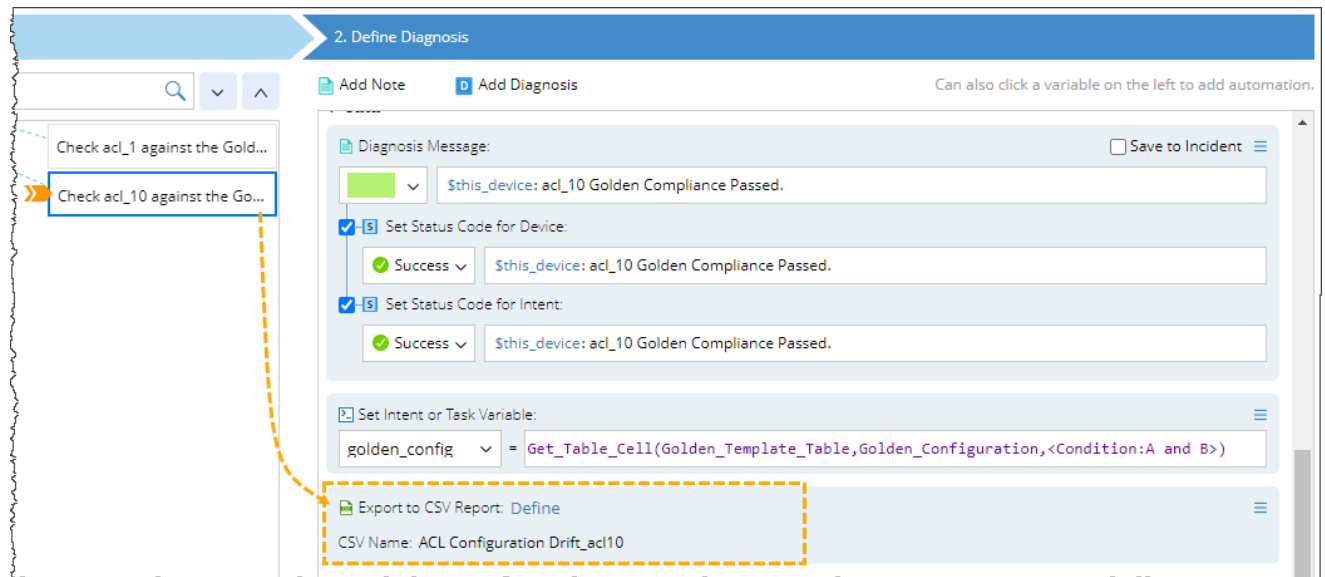
Cancel 8 OK

7. Define the column value from the dropdown to the selected CSV report.

Column	Value
Device	<i>\$this_device</i>
Golden Config	<i>golden_config</i>
Current Config	<i>acl_1_config</i>
Matched	<i>Result</i>
Missing Lines	<i>Unmatched_lines</i>
Extra Lines	<i>Unused_pattern_lines</i>
Vendor	<i>vendor</i>
Model	<i>model</i>
Software Version	<i>ver</i>

8. Click **OK** to save the setting.

The **Export CSV Report** will be added to the Diagnosis.



7.1.3.4 Run the Intent and Check CSV Report

Run the Intent. Click **CSV Report** from the dropdown menu and switch the tab to see the results.

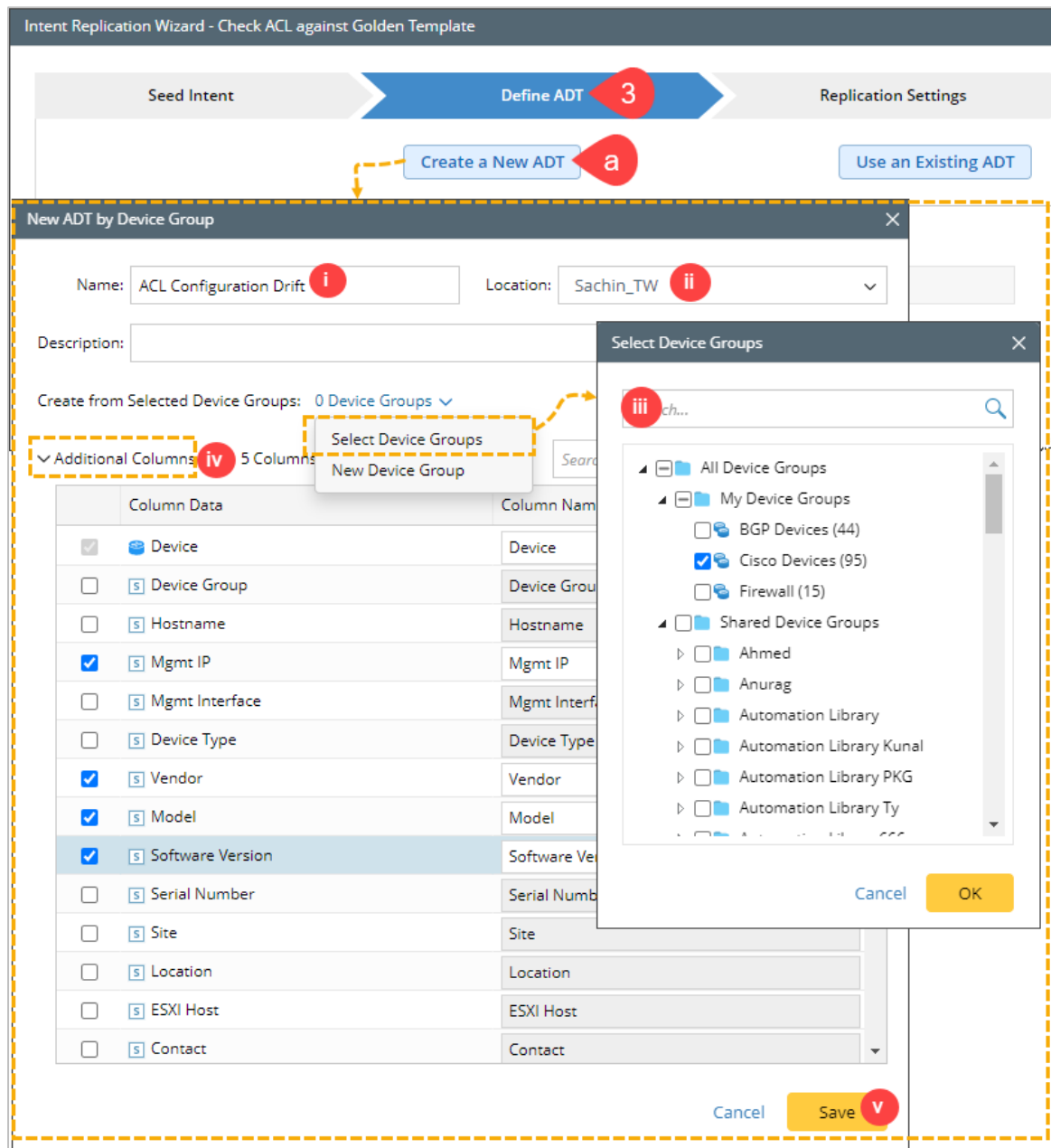
The screenshot displays the NetBrain interface. The top window, titled 'Network Intent (View Mode) - All Network Intents/Sachin/Check ACL against Golden Template', shows the execution of the 'Check ACL against Golden Template' intent. The result indicates that the intent execution is finished at 07/16/2024 04:20 PM with 0 errors. A dropdown menu is open, showing the 'CSV Report' option highlighted. Below the menu, the 'Configuration Diagnosis' section shows 2 diagnoses for the device 'US-BOS-CW03-01'. The first diagnosis is 'US-BOS-CW03-01: acl_1 golden compliance failed. Rule False, Missing Lines access-list 1 permit 192.168.0.0 0.0.255.255'. The second diagnosis is 'US-BOS-CW03-01: acl_1 golden compliance fail...'. The bottom window, titled 'CSV Report', shows the 'ACL Configuration Drift' tab selected. It displays a table with 1 row and 9 columns: Device, Golden Config, Current Config, Matched, Missing Lines, Extra Lines, Vendor, Model, and Software Version. The table contains one row for the device 'US-BOS-CW03-01'.

Device	Golden Config	Current Config	Matched	Missing Lines	Extra Lines	Vendor	Model	Software Version
US-BOS-CW03-01	access-list 10 permit 8.8.8.8 ...	access-list 10 permit 8.8.8.8 ...	False	access-list 10 permit 8.8.8.8 ...		Cisco	3560E	15.2(20170809:194209)

7.1.4 Use Intent Replication Wizard

In this section, you will use the **Intent Replication Wizard** to replicate the intent we just created and the replicated intents to a new ADT table.

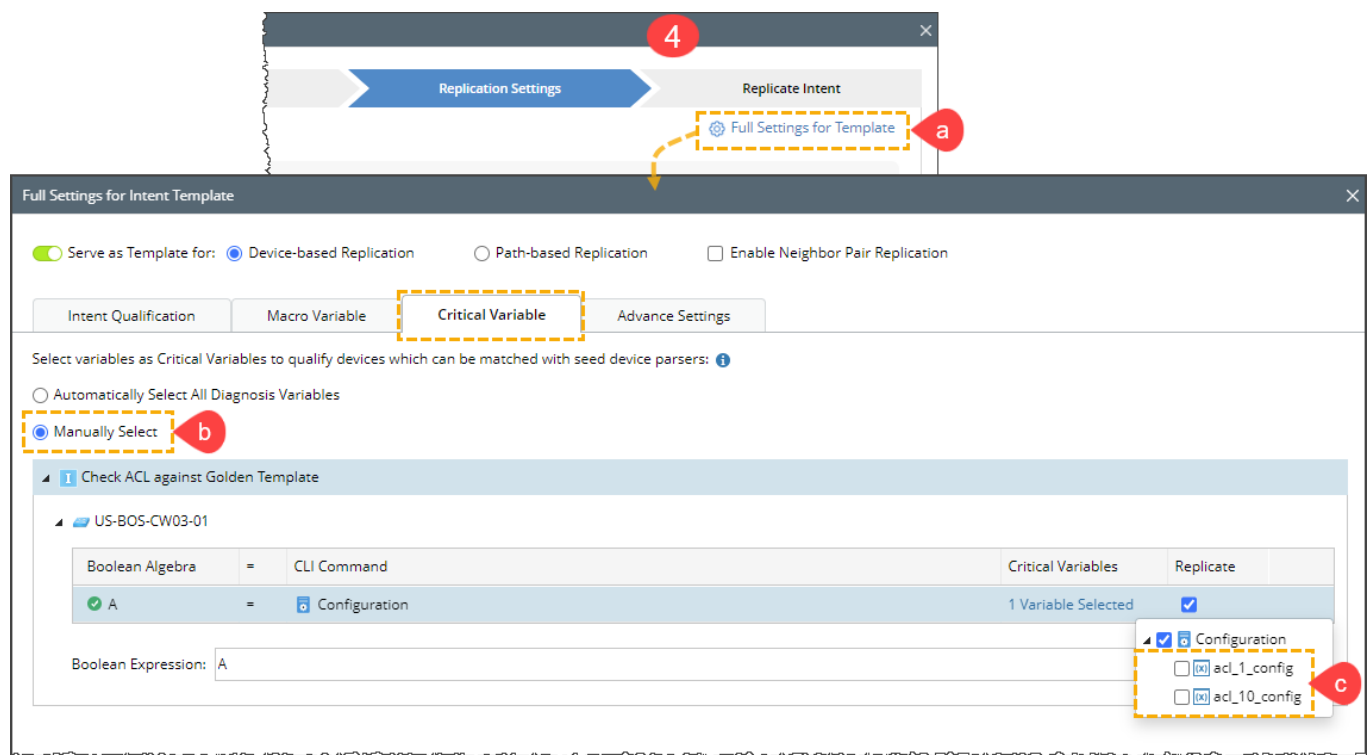
1. Select **Intent Replication Wizard** from the menu.
2. The seed intent is automatically set.
3. Select a device group and create a new ADT. Add the built-in device properties as the additional columns.



4. In the **Replication Settings**, you select the same device group as the intent qualification.

In this step, you should also change the **Critical Variable** setting. By default, the system automatically selects all variables in the **If** conditions as critical variables. In this case, the variable **\$acl_1** is set as the critical variable. However, we want to create an alert if the **access_list 1** is configured, and so we want to remove **\$acl_1** from the critical variables:

- Click the **Full Intent Template Setting** link.
- Click the **Manually Select** option to see the **Critical Variable** section.
- In the **Critical Variable** section, uncheck **acl_1_config** and **acl_10_config** Variables except **Configuration**.
- Click **OK** to save Intent Template Settings.



5. In the Replicate Intent, add additional columns such as **Intent Status Code** and **Last Execution Time**.

Intent Replication Wizard - Check ACL against Golden Template

Seed Intent Define ADT Replication Settings Replicate Intent 5

ADT Columns:

Column Data	Column Name	Tag
1 Replicated Intent	ACL Configuration Drift a	0 tags
5 Intent Status Code	Intent Status Code	
1 Last Execution Time	Last Execution Time	

Additional Columns b

- ☒ Replicated Intent
- ☐ Intent Message
- ☒ Intent Status Code
- ☐ Device Status Code
- ☐ Intent Devices
- ☐ Intent Map
- ☐ Intent CLI Comma...
- ☒ Last Execution Time

Save and Replicate c

Replication Request submitted at: 07/16/2024 05:39 P d

Open Output ADT

Selection Mode: Device-based Replication, ADT: ACL Configuration Drift, 0 Macro Variables.

Previous Finish

The table will now be populated with devices and the replicated Intents (**ACL Configuration Drift**).

Automation Data Table Manager

ACL Configuration Drift Table Builder Last Updated at: 08/07/2024 03:46 PM Rebuild Table

Description: Type description here...

Items: 95 Rows 8 Columns

Replicated Intent

No.	Device	Mgmt IP	Vendor	Model	Software Version	ACL Configuration Drift	Intent Status Code	Last Execution Time
1	Berlin-R1	172.16.8.60	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
2	Berlin-vEdge	192.168.0.1	Cisco	WS-C4500X-32	03.04.04.5G	Check ACL against Golden Templ...		
3	DE-MUC-CR01-02	10.20.1.3	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
4	DE-MUC-CW01-01	10.20.1.4	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		
5	DE-MUC-CW02-01	10.20.1.5	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		
6	DE-MUC-CW03-01	10.20.1.6	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		
7	ISP-P02	4.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
8	ISP-PE01	1.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
9	ISP-PE02	2.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
10	ISP-PE03	3.0.0.2	Cisco	CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...		
11	ITE_EXTEND	192.168.30.207	Cisco	WS-C3560X-48P	15.2(4)E7	Check ACL against Golden Templ...		
12	JP-TYO-CR01-01	192.168.20.1	Cisco	Catalyst 4500 Virtual Sw...	15.4(2)T4	Check ACL against Golden Templ...		
13	JP-TYO-CW01-01	10.30.0.4	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		
14	JP-TYO-CW01-02	10.30.0.5	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		
15	JP-TYO-CW02-01	10.30.0.6	Cisco	3560E	15.2(20170809:194209)	Check ACL against Golden Templ...		

7.1.5 Run Intent Once and Rebuild Table

Run all intents in the ADT once and rebuild the table. Review the intent results (the column **Intent Status Codes**).

Automation Data Table Manager

Help

>>

ACL Configuration Drift

Table Builder

Last Updated at: 08/07/2024 03:46 PM

Rebuild Table

Add Data Manually

Description: *Type description here...*

Items: 95 Rows 8 Columns


Search...

Advanced Filter: Undefined

Model	Software Version	ACL Configuration Drift	Intent Status Code	Last Execution Time
CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	Berlin-R1: acl_1 golden cor	08/07/2024 03:47:0...
WS-C4500X-32	03.04.04.SG	Check ACL against Golden Templ...	Berlin-vEdge: acl_1 golden	07/16/2024 06:14:2...
CGS-MGS-AGS	15.4(2)T4	Check ACL against Golden Templ...	DE-MUC-CR01-02: acl_1 gc	07/16/2024 06:50:3...
3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW01-01: acl_1 g	07/16/2024 06:50:2...
3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW02-01: acl_1 g	07/16/2024 06:50:3...
3560E	15.2(20170809:194209)	Check ACL against Golden Templ...	DE-MUC-CW03-01: acl_1 g	08/07/2024 03:47:0...

7.1.6 View Summary Report

You can create a summary report from the ADT, which combines all CSV reports from all replicated intents in the column:

1. In the Intent column, click  menu and open **View Summary Report**.
2. Select the time frame as per your preference and click **Create** to generate the report.
3. Click the **Export** to export the results into a CSV file.

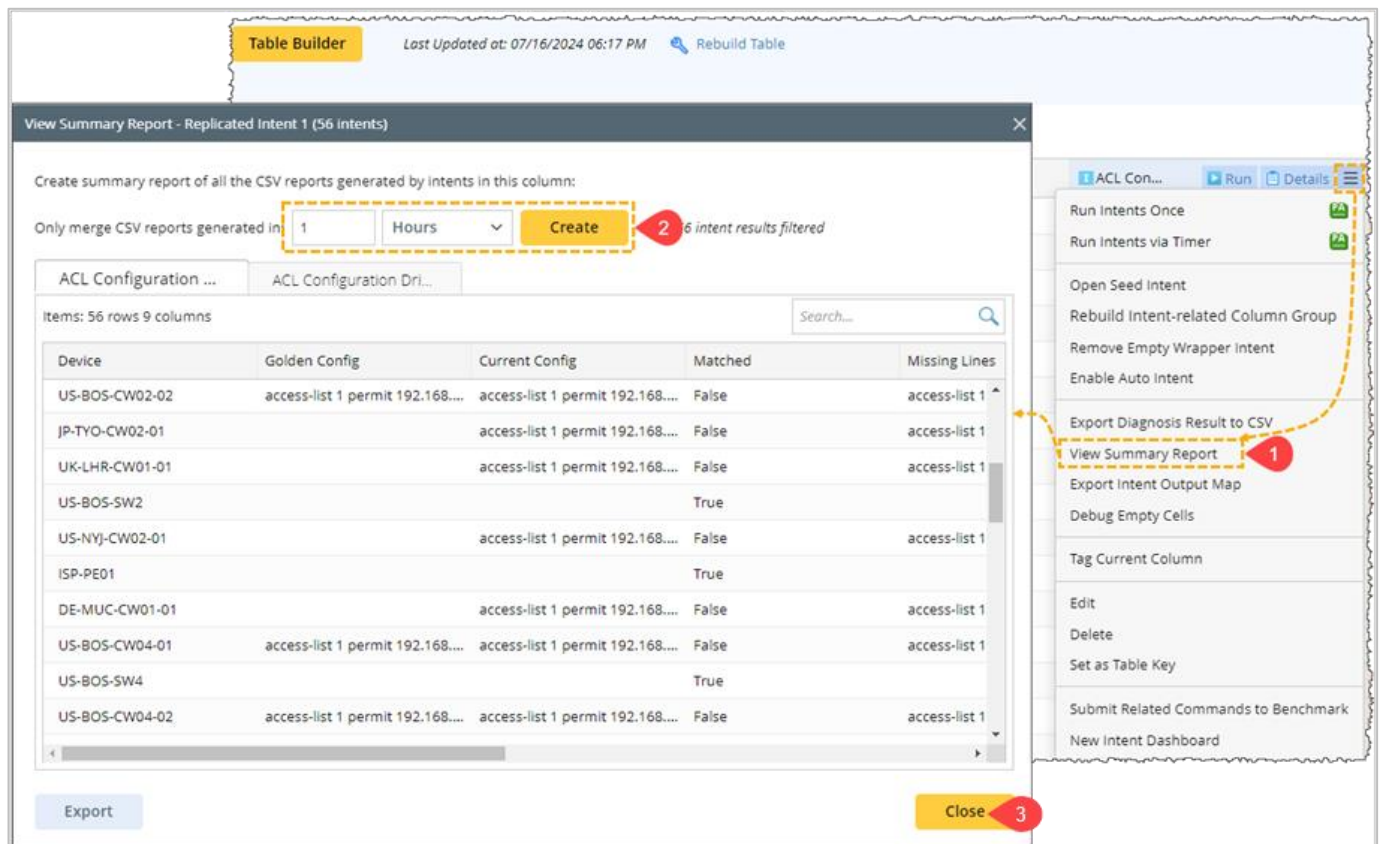


Table Builder Last Updated at: 07/16/2024 06:17 PM Rebuild Table

View Summary Report - Replicated Intent 1 (56 intents)

Create summary report of all the CSV reports generated by intents in this column:

Only merge CSV reports generated in: 1 Hours Create 2 5 intent results filtered

ACL Configuration ... ACL Configuration Dri...

Items: 56 rows 9 columns Search...

Device	Golden Config	Current Config	Matched	Missing Lines
US-BOS-CW02-02	access-list 1 permit 192.168....	access-list 1 permit 192.168....	False	access-list 1
JP-TYO-CW02-01		access-list 1 permit 192.168....	False	access-list 1
UK-LHR-CW01-01		access-list 1 permit 192.168....	False	access-list 1
US-BOS-SW2			True	
US-NYJ-CW02-01		access-list 1 permit 192.168....	False	access-list 1
ISP-PE01			True	
DE-MUC-CW01-01		access-list 1 permit 192.168....	False	access-list 1
US-BOS-CW04-01	access-list 1 permit 192.168....	access-list 1 permit 192.168....	False	access-list 1
US-BOS-SW4			True	
US-BOS-CW04-02	access-list 1 permit 192.168....	access-list 1 permit 192.168....	False	access-list 1

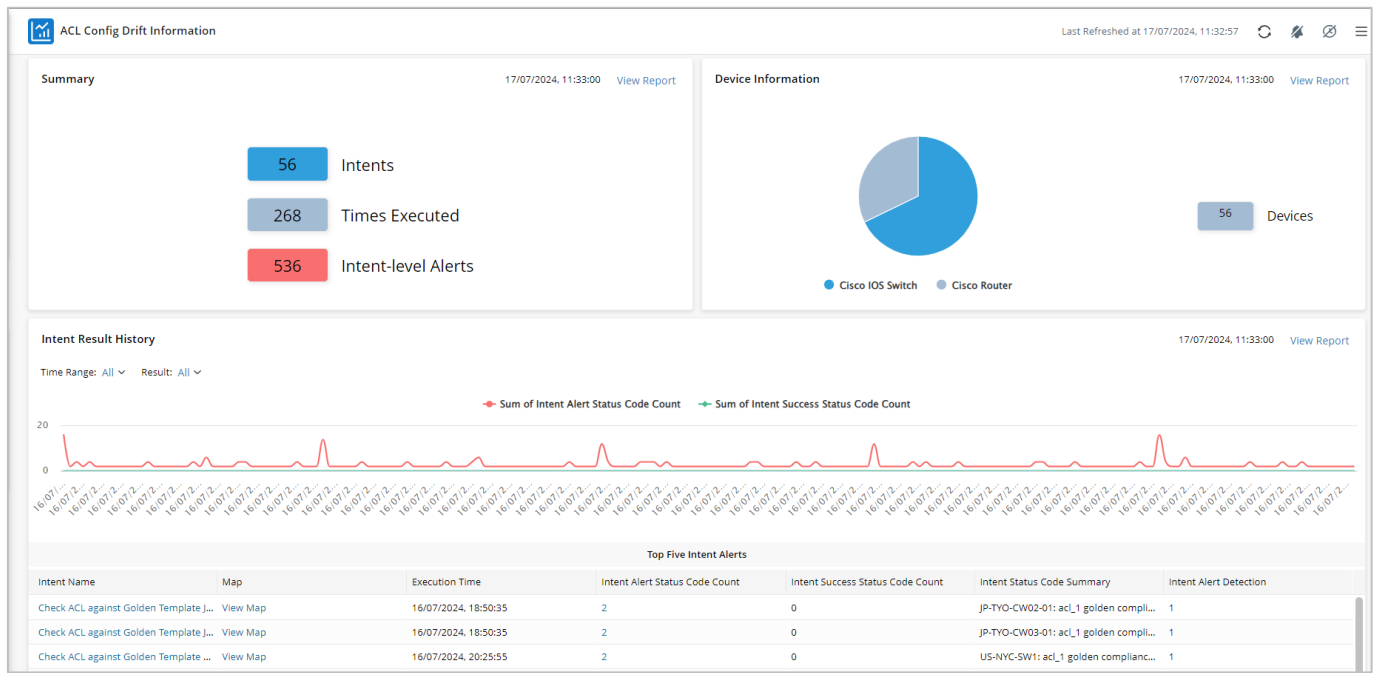
Export Close 3

ACL Con... Run Details

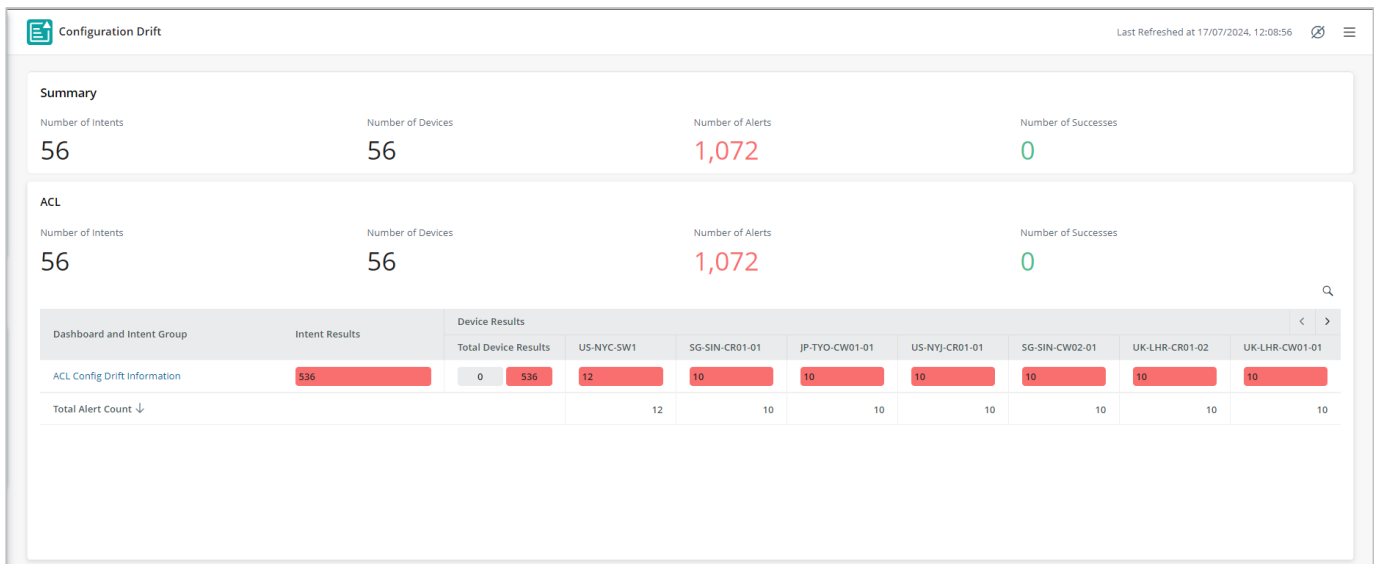
- Run Intents Once
- Run Intents via Timer
- Open Seed Intent
- Rebuild Intent-related Column Group
- Remove Empty Wrapper Intent
- Enable Auto Intent
- Export Diagnosis Result to CSV
- View Summary Report 1
- Export Intent Output Map
- Debug Empty Cells
- Tag Current Column
- Edit
- Delete
- Set as Table Key
- Submit Related Commands to Benchmark
- New Intent Dashboard

7.1.7 Create Intent and Summary Dashboard

Create an intent dashboard to better view the result.



You can also create a new summary dashboard for this intent dashboard:



7.2 Create Golden Template

In this section, you will learn how to use intent-based automation to analyze ACL configurations of your network to find the Golden Template. First, you will create a summary report for all ACL configurations, from which you will create a pivot table to identify the common Golden configuration (if an ACL has the same configurations for all devices in one or multiple sites, it is a good candidate for the Golden Config). This template will be used to compare and verify the ACL configurations of other network devices.

If you know the Golden Config of your network, you can skip this section.

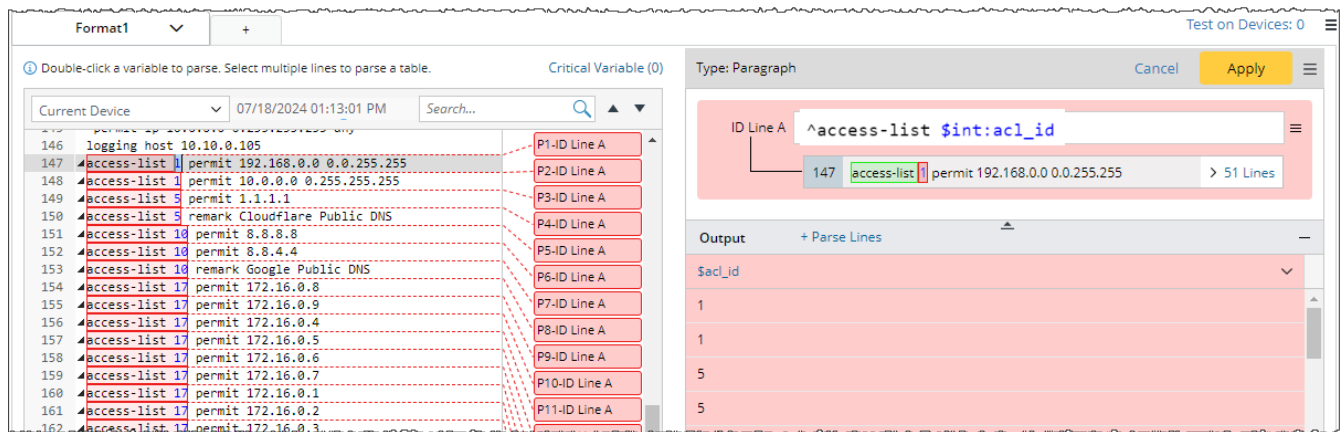
7.2.1 Create Intent to Export ACL Configuration to a CSV File

First, you create an intent to export all ACL configurations into a CSV file. This intent will then be replicated to all devices with the **Intent Replication Wizard**. From the replicated Intent, you can create a summary report for all ACL configurations across your network.

Create a Network Intent, **ACL Name List**, from the **Intent Manager**, and select a target device with the ACL configurations (e.g., **US-BOS-CW-04-02**).

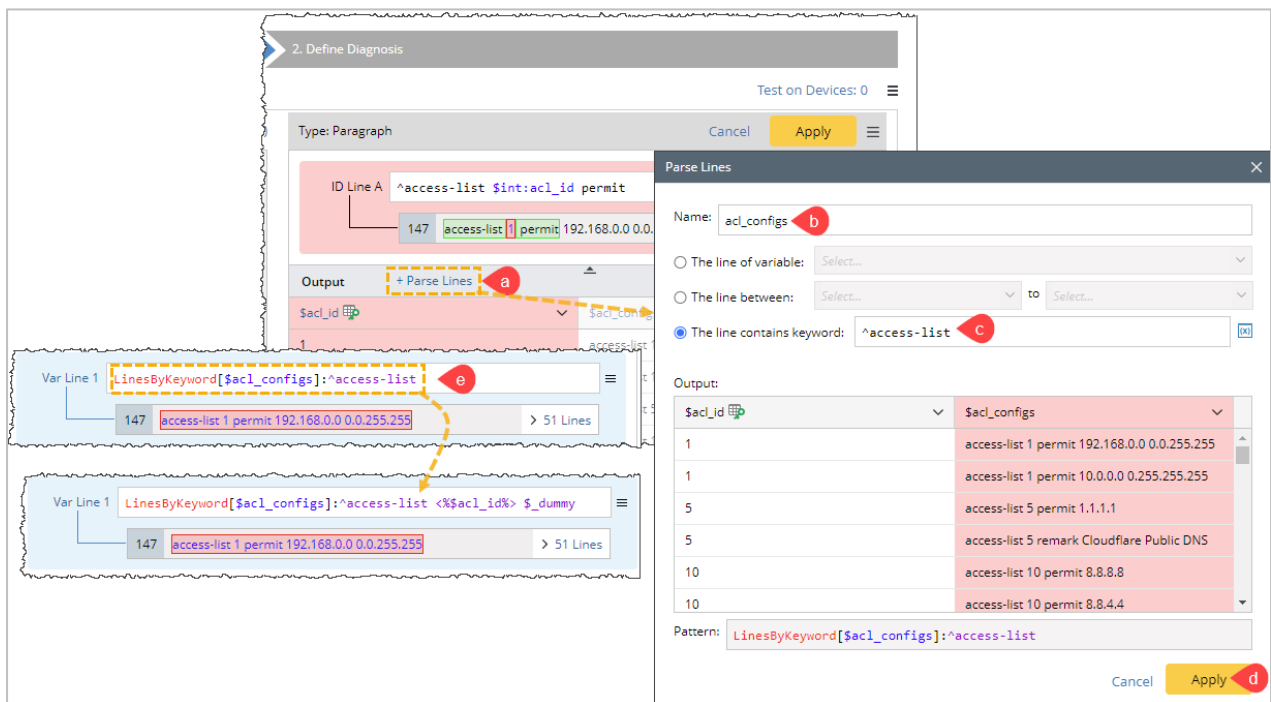
Add a **Config Diagnosis**, **Retrieve** data, and define the Variable using the **Paragraph** pattern:

- ID Line: **`^access-list $int:acl_id`**



- Define the variable **`$acl_config`** using the Variable Operator **LinesByKeyword** to extract all configurations related to an ACL.
 - a) Click **+ Parse Lines** in the **Output** pane, and the **Parser Lines** window appears.
 - b) Set the Variable name as **`acl_configs`**.
 - c) Select **The line contains keyword** option, and enter the keyword, **`^access-list`**.


- d) Check the pattern and click **Apply** to close the **Parser Lines** window. The Var Line 1 pattern will be ***LinesByKeyword[\$acl_configs]:^access-list***.
- e) Modify the pattern to ***LinesByKeyword[\$acl_configs]:^access-list <%%\$acl_id%> \$_dummy***.



7.2.1.1 Add Formula Column in the Parser Table

The table has two columns: **acl_id** and **acl_configs**. For others to better understand this table, you may want to add a new column, **acl_name**, with the value to be **"ACL " + \$acl_id**, such as **ACL 1**.

You can add a **Formula Column** to table variables, which converts the original variables into different formats or values:

1. Click **All Intent Variables** in the bottom-right corner.
2. Expand the **US-BOS-CW04-01 Configuration** section and then expand **acl_table**.
3. Click  and open **Add Formula Column** from the dropdown.
4. In the **Add Formula Column** window, define the following fields:
 - Name: **acl_name**
 - Type: **string**
 - Definition: **"ACL " + \$acl_id** (click **\$** to open pop up to **acl_id** variable)

Intent Variables for Seed Logic

Intent Variable | Use Automation Data Table | Task Variable

+ Add Formula Column

1 Intent

2 US-BOS-CW04-02 Configuration

3 acl_table

acl_id (int)	acl_configs (string)
1	access-list 1 permit 192.168.0.0
1	
5	
5	
10	
10	
10	

4 Add Formula Column

Name: acl_name

Type: string

Initial Value:

Definition: "ACL " + \$acl_id

+ Variable + Function

Cancel OK

1 All Intent Variables

The Formula column will be:

acl_id (int)	acl_configs (string)	acl_name (string)
1	access-list 1 permit 192.168.0.0 0.0.255.255	ACL 1
1	access-list 1 permit 10.0.0.0 0.255.255.255	ACL 1
5	access-list 5 permit 1.1.1.1	ACL 5
5	access-list 5 remark Cloudflare Public DNS	ACL 5
10	access-list 10 permit 8.8.8.8	ACL 10
10	access-list 10 permit 8.8.4.4	ACL 10
10	access-list 10 remark Google Public DNS	ACL 10

7.2.1.2 Define Diagnosis to Export ACL Configurations

Create a new diagnosis to export the **acl_table** to a **CSV file**.

Follow the same step in Section 7.1.3.3. Besides this table, you may also add the built-in device properties and export them.

Export Data to CSV for US-BOS-CW04-02

Export the row data of the "acl_table" to CSV Report.

Network Feature

Manage All CSV Reports

Define the column mapping from the "acl_table" to the selected CSV Report.

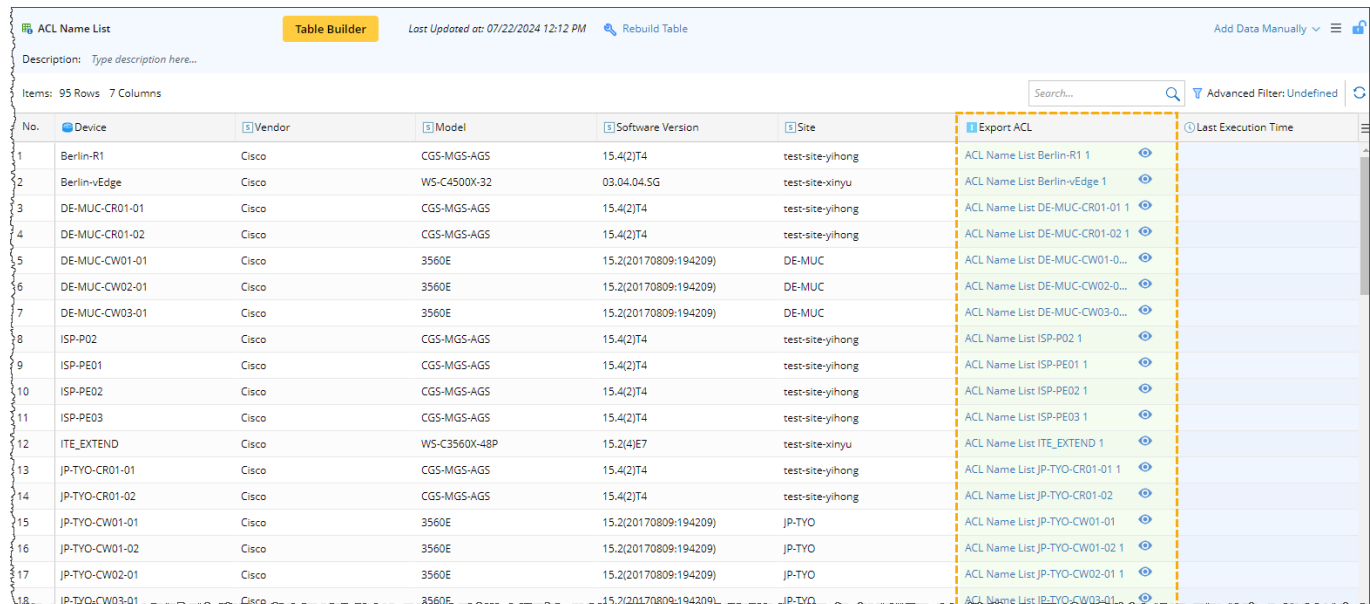
Device	Device Type	Feature Tag	Feature Name	Configuration	Model	Version	Site
\$this_device	subTypeName	ACL	acl_name	acl_configs	model	ver	site

Cancel

OK

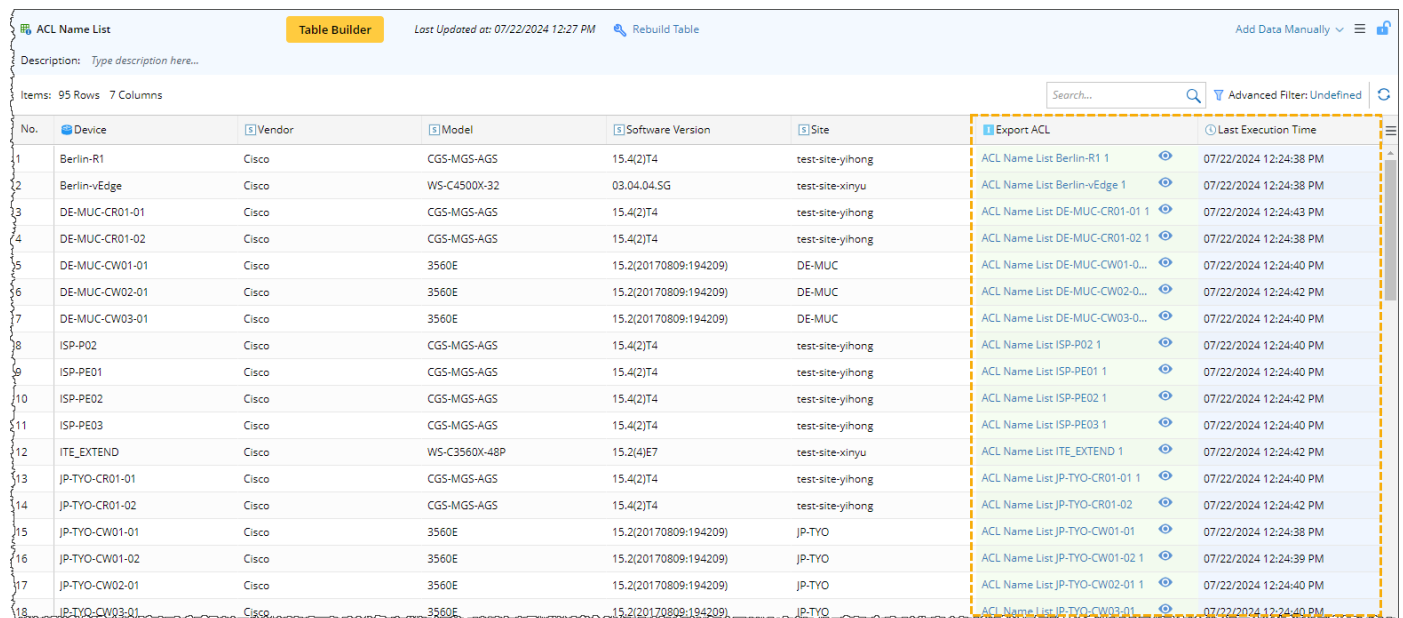
7.2.2 Create ACL Report for the Whole Network Devices

Replicate the intents to all Cisco IOS devices via the **Intent Replication Wizard**. You will create a new ADT like this:



No.	Device	Vendor	Model	Software Version	Site	Export ACL	Last Execution Time
1	Berlin-R1	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List Berlin-R1 1	
2	Berlin-vEdge	Cisco	WS-C4500X-32	03.04.04.5G	test-site-xinyu	ACL Name List Berlin-vEdge 1	
3	DE-MUC-CR01-01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List DE-MUC-CR01-01 1	
4	DE-MUC-CR01-02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List DE-MUC-CR01-02 1	
5	DE-MUC-CW01-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW01-0...	
6	DE-MUC-CW02-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW02-0...	
7	DE-MUC-CW03-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW03-0...	
8	ISP-P02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-P02 1	
9	ISP-PE01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE01 1	
10	ISP-PE02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE02 1	
11	ISP-PE03	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE03 1	
12	ITE_EXTEND	Cisco	WS-C3560X-48P	15.2(4)E7	test-site-xinyu	ACL Name List ITE_EXTEND 1	
13	JP-TYO-CR01-01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List JP-TYO-CR01-01 1	
14	JP-TYO-CR01-02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List JP-TYO-CR01-02	
15	JP-TYO-CW01-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW01-01	
16	JP-TYO-CW01-02	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW01-02 1	
17	JP-TYO-CW02-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW02-01 1	
18	JP-TYO-CW03-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW03-01	

Run the replicated intents once and rebuild the table:



No.	Device	Vendor	Model	Software Version	Site	Export ACL	Last Execution Time
1	Berlin-R1	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List Berlin-R1 1	07/22/2024 12:24:38 PM
2	Berlin-vEdge	Cisco	WS-C4500X-32	03.04.04.5G	test-site-xinyu	ACL Name List Berlin-vEdge 1	07/22/2024 12:24:38 PM
3	DE-MUC-CR01-01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List DE-MUC-CR01-01 1	07/22/2024 12:24:43 PM
4	DE-MUC-CR01-02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List DE-MUC-CR01-02 1	07/22/2024 12:24:38 PM
5	DE-MUC-CW01-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW01-0...	07/22/2024 12:24:40 PM
6	DE-MUC-CW02-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW02-0...	07/22/2024 12:24:42 PM
7	DE-MUC-CW03-01	Cisco	3560E	15.2(20170809:194209)	DE-MUC	ACL Name List DE-MUC-CW03-0...	07/22/2024 12:24:40 PM
8	ISP-P02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-P02 1	07/22/2024 12:24:40 PM
9	ISP-PE01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE01 1	07/22/2024 12:24:40 PM
10	ISP-PE02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE02 1	07/22/2024 12:24:42 PM
11	ISP-PE03	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List ISP-PE03 1	07/22/2024 12:24:40 PM
12	ITE_EXTEND	Cisco	WS-C3560X-48P	15.2(4)E7	test-site-xinyu	ACL Name List ITE_EXTEND 1	07/22/2024 12:24:42 PM
13	JP-TYO-CR01-01	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List JP-TYO-CR01-01 1	07/22/2024 12:24:40 PM
14	JP-TYO-CR01-02	Cisco	CGS-MGS-AGS	15.4(2)T4	test-site-yihong	ACL Name List JP-TYO-CR01-02	07/22/2024 12:24:43 PM
15	JP-TYO-CW01-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW01-01	07/22/2024 12:24:38 PM
16	JP-TYO-CW01-02	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW01-02 1	07/22/2024 12:24:39 PM
17	JP-TYO-CW02-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW02-01 1	07/22/2024 12:24:40 PM
18	JP-TYO-CW03-01	Cisco	3560E	15.2(20170809:194209)	JP-TYO	ACL Name List JP-TYO-CW03-01	07/22/2024 12:24:40 PM

View the summary report and export the results into a CSV file. The CSV file, **Network Feature_ACL Name List** will be downloaded in your computer's download folder.

View Summary Report - Export ACL (56 intents)

Create summary report of all the CSV reports generated by intents in this column:

Only merge CSV reports generated in: 1 Hours **Create** 2 56 intent results filtered

Network Feature

Items: 2234 rows 8 columns

Device	Device Type	Feature Tag	Feature Name
DE-MUC-CR01-01	Cisco Router	ACL	ACL 1
DE-MUC-CR01-01	Cisco Router	ACL	ACL 1
DE-MUC-CR01-01	Cisco Router	ACL	ACL 2
DE-MUC-CR01-01	Cisco Router	ACL	ACL 5
DE-MUC-CR01-01	Cisco Router	ACL	ACL 5
DE-MUC-CR01-01	Cisco Router	ACL	ACL 10

Export 3 **Close** 4

Export Reports

Select CSV Reports:

- ☒ CSV Name Seed Intent Template Name
- ☒ Network Feature ACL Name List

Export as: ☐ Excel ☒ CSV **Export** 2

Downloads > csvReport (1)

Name	Type	Compressed size	Password pr...	Size
Network Feature_ACL Name List	Microsoft Excel Comma Separ...	14 KB	No	

The following is a sample CSV file:

	A	B	C	D	E	F	G	H
1	Device	Device Type	Featu	Feature	Configuration	Model	Version	Site
2	DE-MUC-CR01-01	Cisco Router	ACL	ACL 1	access-list 1 permit 192.168.0.0	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
3	DE-MUC-CR01-01	Cisco Router	ACL	ACL 1	access-list 1 permit 10.0.0.0.0.2	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
4	DE-MUC-CR01-01	Cisco Router	ACL	ACL 2	access-list 2 permit 192.168.201	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
5	DE-MUC-CR01-01	Cisco Router	ACL	ACL 5	access-list 5 permit 1.1.1.1	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
6	DE-MUC-CR01-01	Cisco Router	ACL	ACL 5	access-list 5 remark Cloudflare	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
7	DE-MUC-CR01-01	Cisco Router	ACL	ACL 10	access-list 10 permit 8.8.8.8	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
8	DE-MUC-CR01-01	Cisco Router	ACL	ACL 10	access-list 10 permit 8.8.4.4	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
9	DE-MUC-CR01-01	Cisco Router	ACL	ACL 10	access-list 10 remark Google Pu	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
10	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.8	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
11	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.9	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
12	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.4	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
13	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.5	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
14	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.6	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
15	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.7	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
16	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.1	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
17	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.2	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
18	DE-MUC-CR01-01	Cisco Router	ACL	ACL 17	access-list 17 permit 172.16.0.3	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
19	DE-MUC-CR01-01	Cisco Router	ACL	ACL 20	access-list 20 permit 129.6.15.2	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
20	DE-MUC-CR01-01	Cisco Router	ACL	ACL 20	access-list 20 permit 129.6.15.2	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong
21	DE-MUC-CR01-01	Cisco Router	ACL	ACL 20	access-list 20 permit 129.6.15.2	CGS-MGS-AGS	15.4(2)T4	My Network\test-container-site-yihong\test-site-yihong

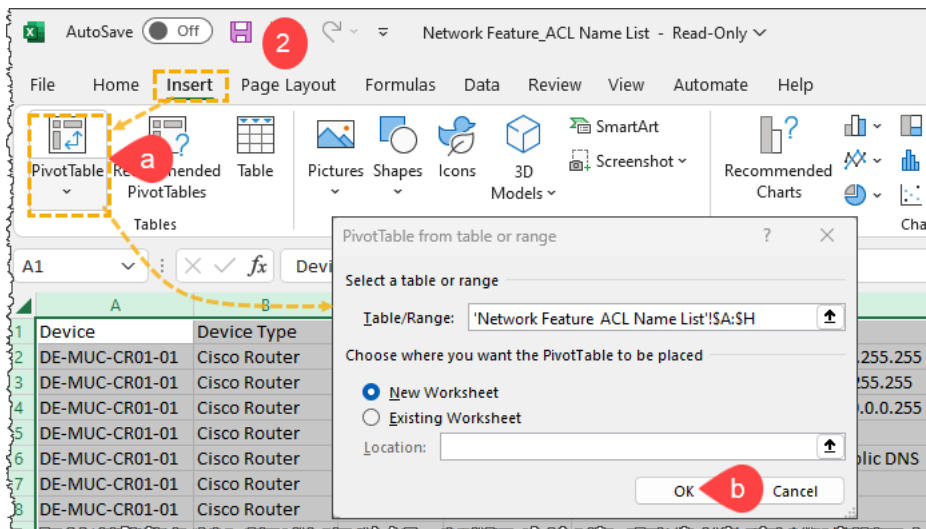
7.2.3 Analyze the Report to Find the Golden Config


In this section, you will analyze the Access Control List (ACL) report to identify the common ACLs that are configured by many devices on one or multiple sites. This analysis will help in determining which ACLs can be standardized into a **Golden Template Table**.

Note: the following is just one way to analyze the report. If you are familiar with Excel, you may find a better way to do this.

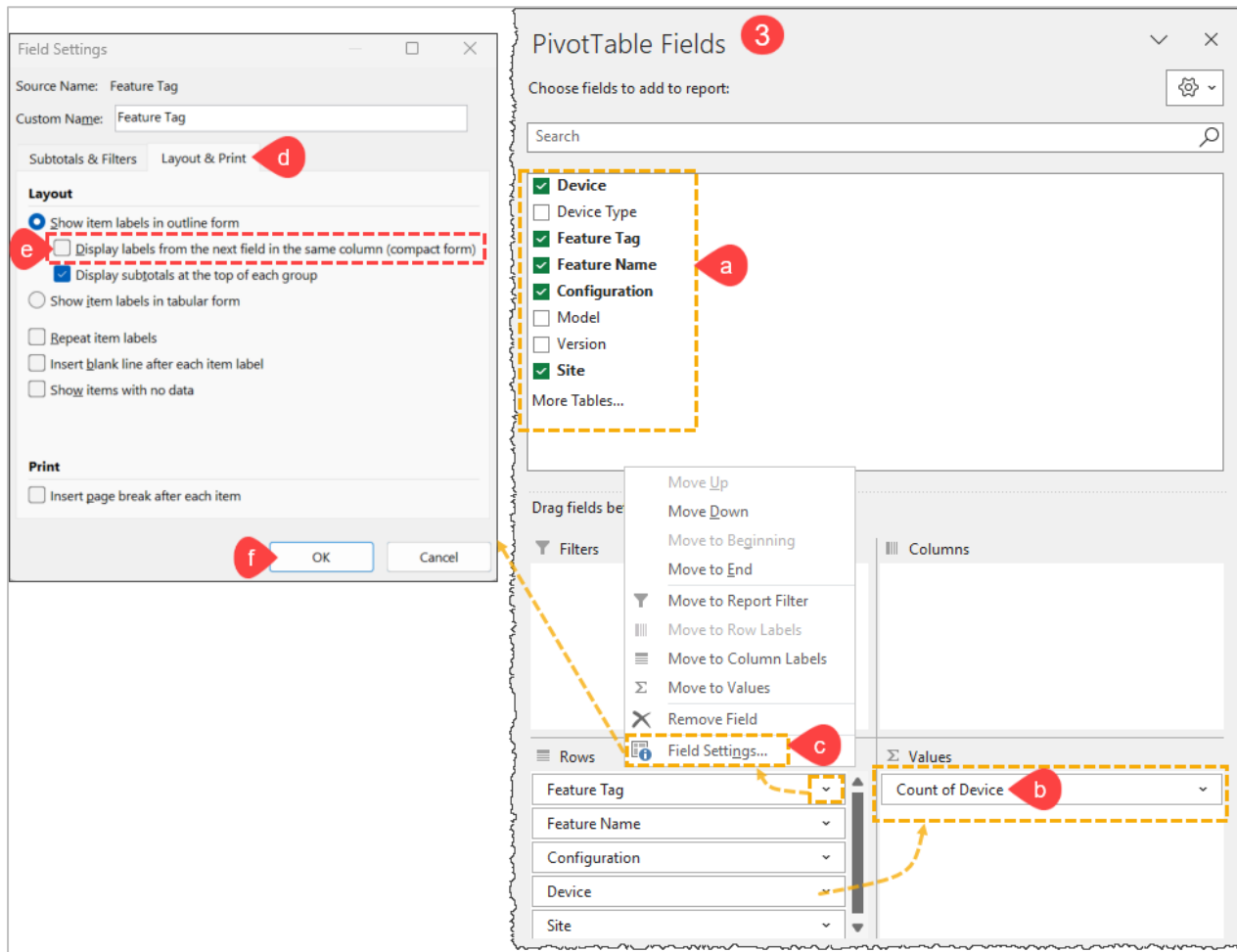
Follow the step-by-step instructions to analyze the report using the **Pivot** table.

1. Open the **Network Feature_ACL Name List** CSV file from your download folder.
2. Create a Pivot Table.
 - a) Select the whole table, go to **Insert**, and click the **PivotTable** option.
 - b) Click **OK** in the **PivotTable from the table or range** popup.



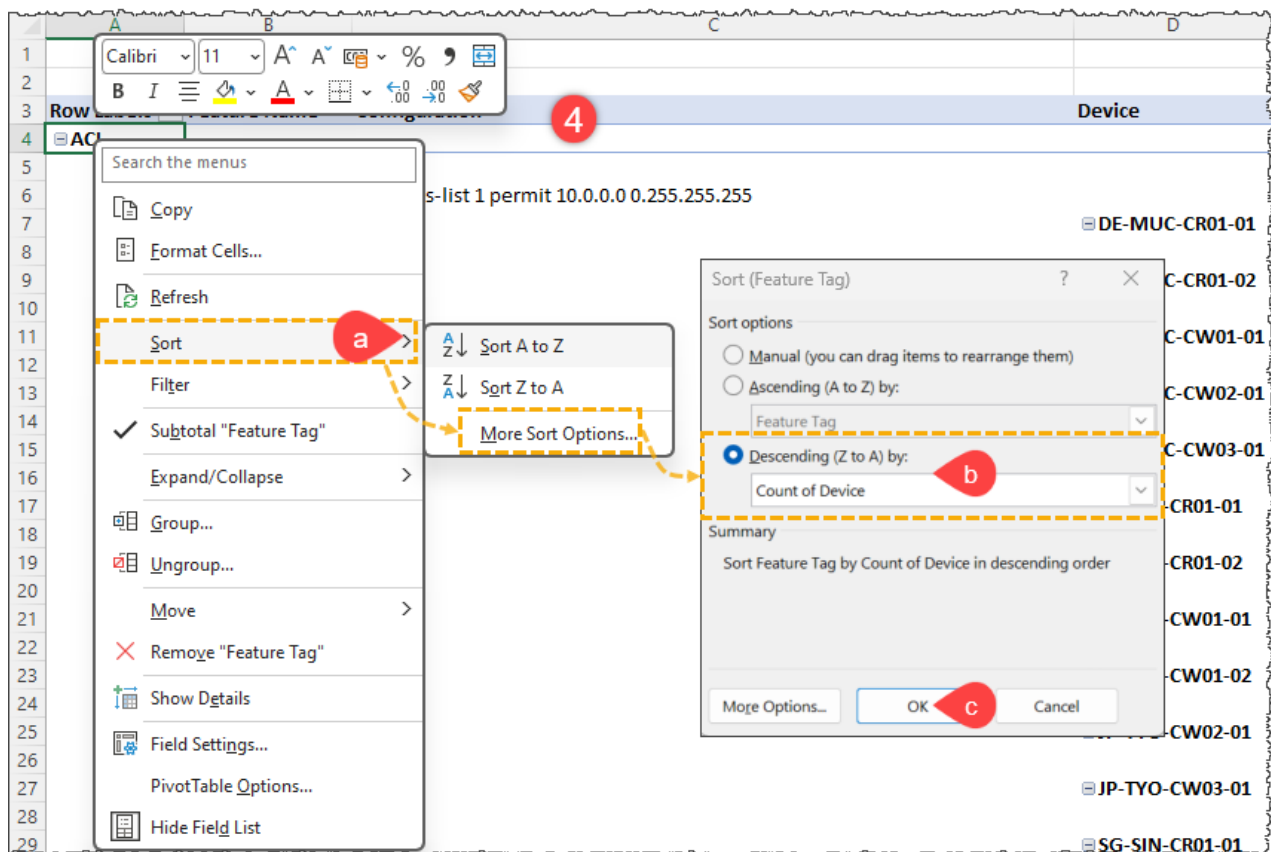
3. Define a PivotTable Fields.
 - a) Choose the fields to add to the report: **Feature Tag**, **Feature Name**, **Configuration**, **Device**, and **Site**.
 - b) Drag the **Device** field into the **Values** column to create another field **Count of Device**.
 - c) Click  menu at the right side of the **Feature Tag** field and select the **Field Settings**.
 - d) In the **Field Settings** window, go to the **Layout & Print** tab.
 - e) Uncheck the option **Display labels** from the next field in the same column (compact from).
 - f) Click **OK** to apply the field settings.

NOTE: Apply the Field Settings to all the fields.

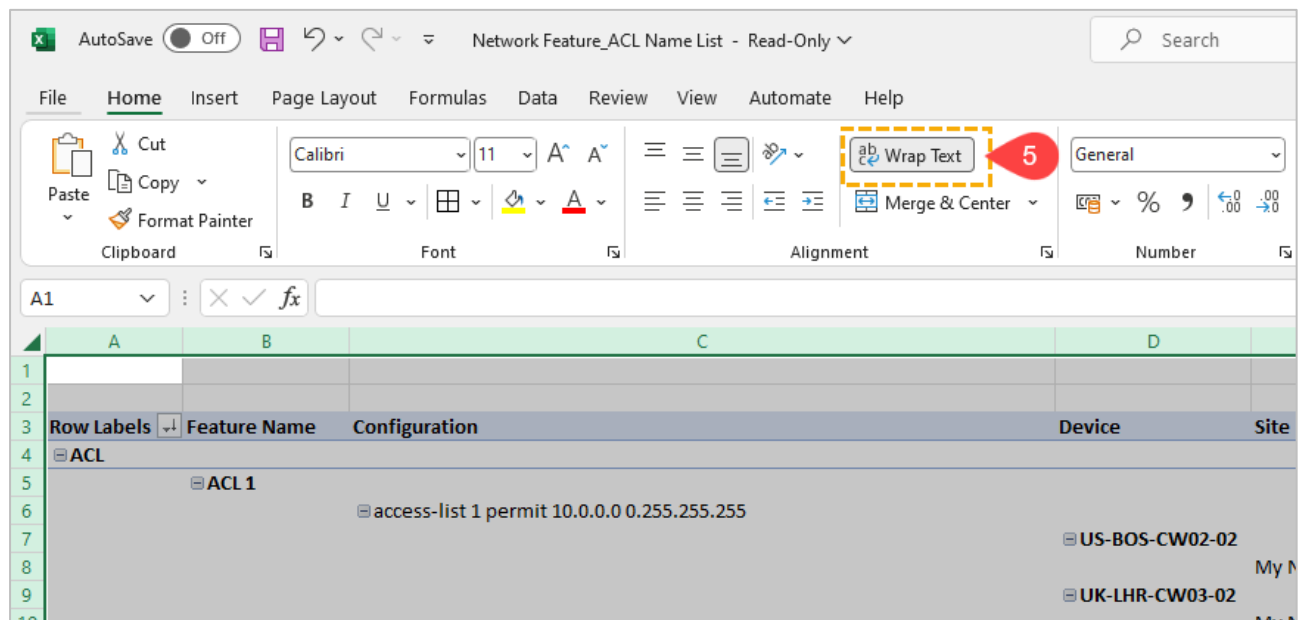


4. Sort the Fields.

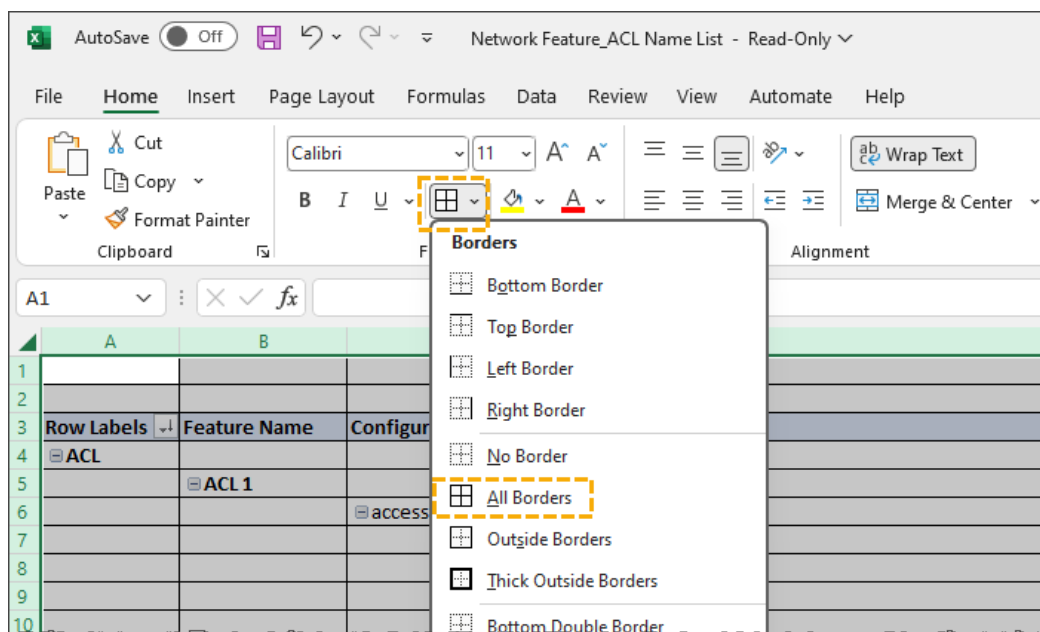
- a) Right-click the field name and select **Sort > More Sort Options**.
- b) In the Sort popup, select Descending (Z to A) by Count of Device.
- c) Click **OK** to apply settings.



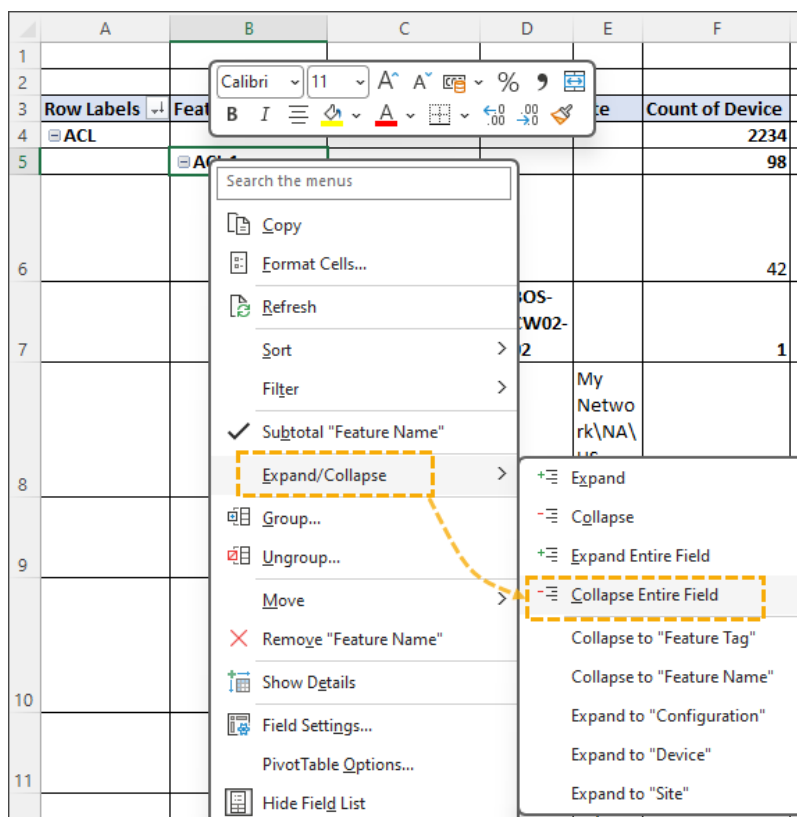
5. Wrap the Wrap Text in the entire sheet to view the Configuration easily.



- To add Borders to the Pivot Table, select the whole table, go to **Home**, and click **Borders** to select **All Borders** from the dropdown menu.



- To find ACL Golden Configuration, right-click the Feature Name and click **Expand/Collapse > Collapse Entire Field**.



The following is the final Pivot Table with all ACL configurations:

3	Row Labels	Feature Name	Configuration	Device	Site	Count of Device
4	ACL					2234
5		ACL 1				98
6		ACL 10				126
7		ACL 100				20
8		ACL 101				192
9		ACL 111				3
10		ACL 120				1
11		ACL 130				1
12		ACL 17				378
13		ACL 190				169
14		ACL 191				1
15		ACL 192				254
16		ACL 193				128
17		ACL 194				128
18		ACL 195				134
19		ACL 196				24
20		ACL 198				3
21		ACL 199				9
22		ACL 2				4
23		ACL 20				252
24		ACL 25				212
25		ACL 44				3
26		ACL 5				85
27		ACL 50				3
28		ACL 51				3
29		ACL 52				3
31	Grand Total					2234

8. Analyze each ACL Configuration.

The ACLs having the same configurations in many devices across multiple sites are a good candidate to be added to the Golden Template Table. For example, **ACL192** has the same configuration for devices in 8 sites. Therefore, the configuration of ACL 192 shown below can be used as the Golden Template for devices within these 8 sites.

Row Labels	Feature Name	Configuration	Site	Device	Count of Device
ACL					2234
	ACL 1				98
	ACL 10				126
	ACL 100				20
	ACL 101				192
	ACL 111				3
	ACL 120				1
	ACL 130				1
	ACL 17				378
	ACL 190				169
	ACL 191				1
	ACL 192				254
		access-list 192 permit tcp any			44
			My Network\test-container-site-yihong\test-site-		12
			My Network\NA\US-BOS		8
			My Network\EMEA\UK-LHR		8
			My Network\APAC\JP-TYO		4
			My Network\NA\US-NYJ		4
			My Network\APAC\SG-SIN		3
			My Network\EMEA\DE-MUC		3
			My Network\test-site-jun		2

9. The ACL has many different types of configurations and each configuration is only configured in one or few devices is not a good candidate. For example, **ACL 25** has various types of configurations over different sites.

ACL 25				212
	access-list 25 permit 134.47.169.0 0.0.0.255			1
		My Network\test-container-site-yihong\test-site-yihong		1
	access-list 25 permit 151.94.180.0 0.0.0.255			1
		My Network\test-container-site-yihong\test-site-yihong		1
	access-list 25 permit 150.57.176.0 0.0.0.255			1
		My Network\NA\US-BOS		1
	access-list 25 deny 16.6.158.27			1
		My Network\NA\US-NYJ		1
	access-list 25 permit 134.27.168.0 0.0.0.255			1
		My Network\EMEA\UK-LHR		1
	access-list 25 deny 167.10.192.47			1
		My Network\APAC\JP-TYO		1
	access-list 25 permit 144.38.170.0 0.0.0.255			1
		My Network\test-container-site-yihong\test-site-yihong		1
	access-list 25 deny 167.11.191.48			1
		My Network\APAC\JP-TYO		1
	access-list 25 permit 151.104.180.0 0.0.0.255			1
	access-list 25 deny 167.12.192.49			1

You can add the good candidates to the Golden Template Table manually and repeat the steps of 7.1 to check the configuration drift for these ACLs.

7.3 Check NTP Server Against Golden Template

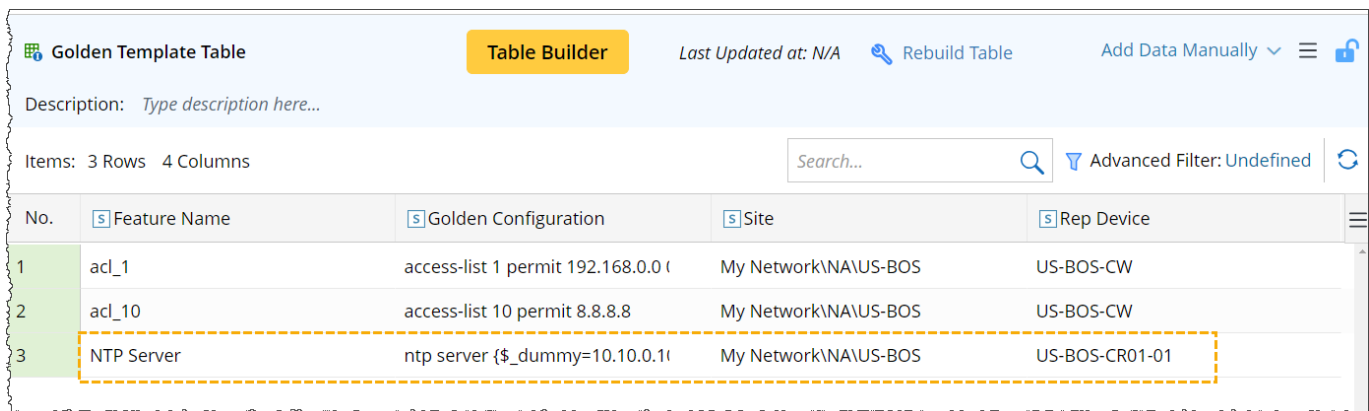
In this session, you will check the **NTP Server** configurations against the golden template. You will go through the same step as Section 7.1 and reuse the ADTs of that section.

7.3.1 NTP Server Golden Template

Open the **Golden Template Table** you created in Section 7.1.1 and manually add a row for the NTP Golden template. The golden configuration can be:

ntp server {\$_dummy=10.10.0.101|10.10.0.102} prefer version 3

Here, we use a special variable, ***\$_dummy***, which has the value ***10.10.0.101|10.10.0.102***, which means if a device configuration, ***ntp server 10.10.0.101 prefer version 3***, or ***ntp server 10.10.0.102 prefer version 3***, will match this golden configuration.

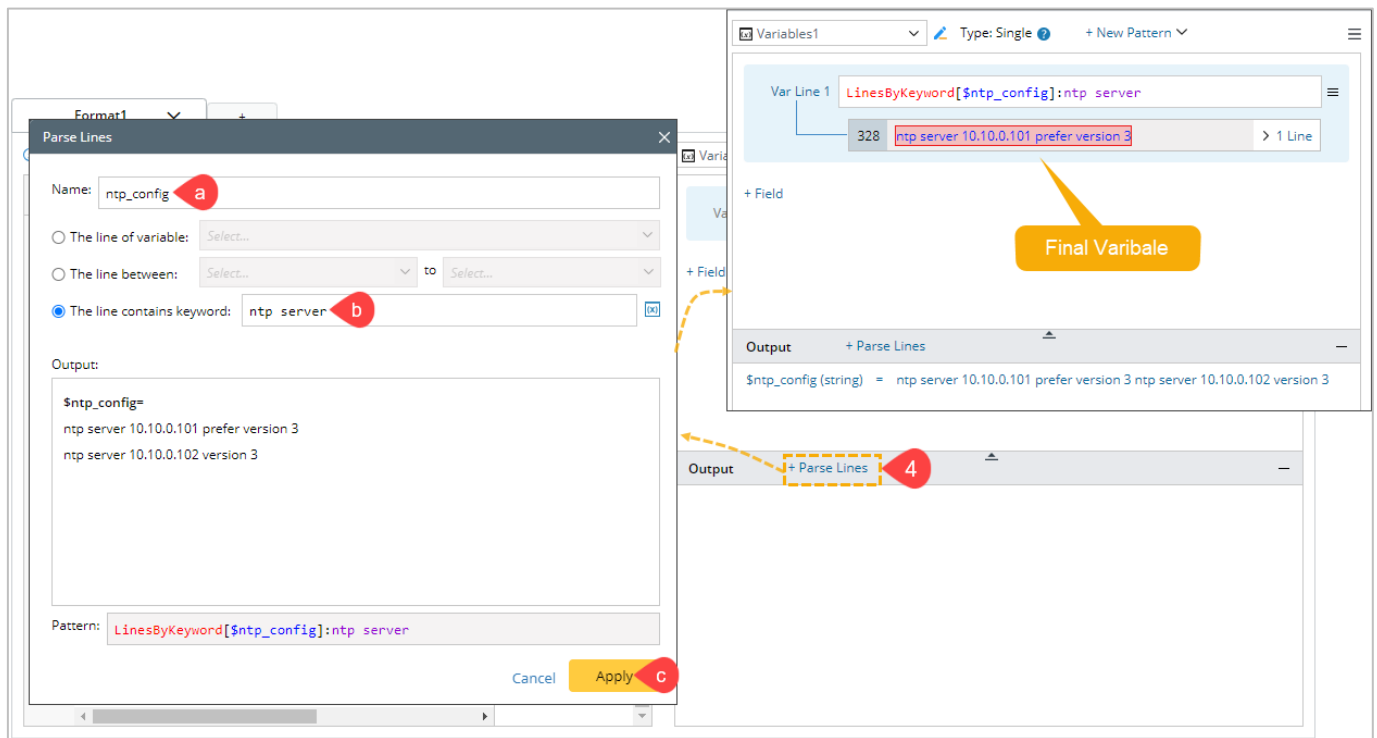


No.	Feature Name	Golden Configuration	Site	Rep Device
1	acl_1	access-list 1 permit 192.168.0.0	My Network\NA\US-BOS	US-BOS-CW
2	acl_10	access-list 10 permit 8.8.8.8	My Network\NA\US-BOS	US-BOS-CW
3	NTP Server	ntp server {\$_dummy=10.10.0.101 10.10.0.102} prefer version 3	My Network\NA\US-BOS	US-BOS-CR01-01

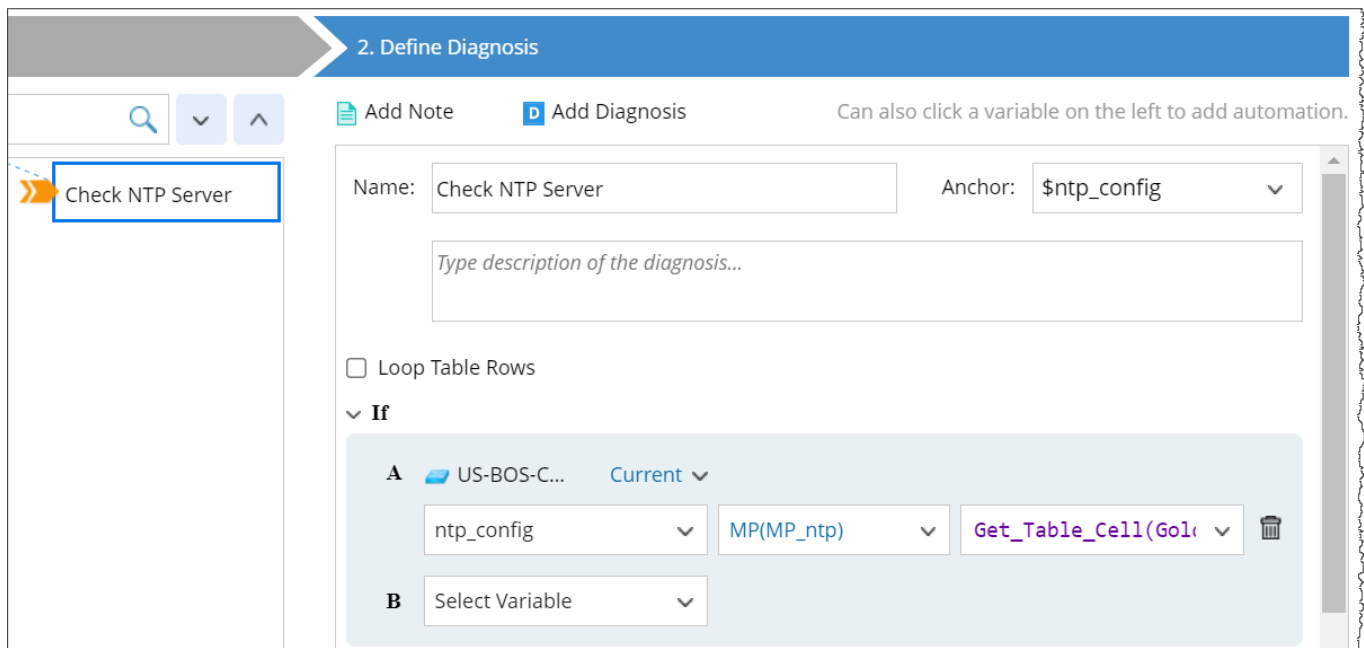
7.3.2 Create Intent to Check NTP Server Against Golden Template

Create a new Network Intent, **Check NTP Server Against Golden Template** from the **Intent Manager**. Select the seed device, e.g., **US-BOS-CR01-01**. Add a **Config Diagnosis**, and use the function **LinesByKeyword** to parse the NTP configurations. Follow the same steps in Section 7.2.1.

LinesByKeyword[\$ntp_config]:ntp server



Repeat the steps of 7.1.2.4 to create the diagnosis, certainly with a different variable and feature name.



▼ Then

📄 Diagnosis Message: ❏ Save to Incident ⋮

👉

`$this_device: NTP Server config complies the golden config.`

✔️ [S] Set Status Code for Device:

✔️ Success

`$this_device: NTP Server config complies the golden config.`

✔️ [S] Set Status Code for Intent:

✔️ Success

`$this_device: NTP Server config complies the golden config.`

Add Logic ▼

▼ Else 🗑 Delete

📄 Diagnosis Message: ❏ Save to Incident ⋮

👉

`$this_device: NTP Server config does not comply the golden config. Rule $MP_ntp.Result, Missing Lines $MP_ntp.Unmatched_lines`

✔️ [S] Set Status Code for Device:

❗ Error

`$this_device: NTP Server config does not comply the golden config. Rule $MP_ntp.Result, Missing Lines $MP_ntp.Unmatched_lines`

✔️ [S] Set Status Code for Intent:

❗ Error

`$this_device: NTP Server config does not comply the golden config. Rule $MP_ntp.Result, Missing Lines $MP_ntp.Unmatched_lines`

Add Logic ▼

+ Add Elself

Cancel

Apply

258 | NetBrain R11.1b

7.3.3 Create the Summary Report and Dashboard

You can repeat the steps from Section 7.1.7 to create a summary report.

View Summary Report - NTP Server Config Check (53 intents)

Create summary report of all the CSV reports generated by intents in this column:

Only merge CSV reports generated in: 1 Hours Create 53 of 53 intent results filtered

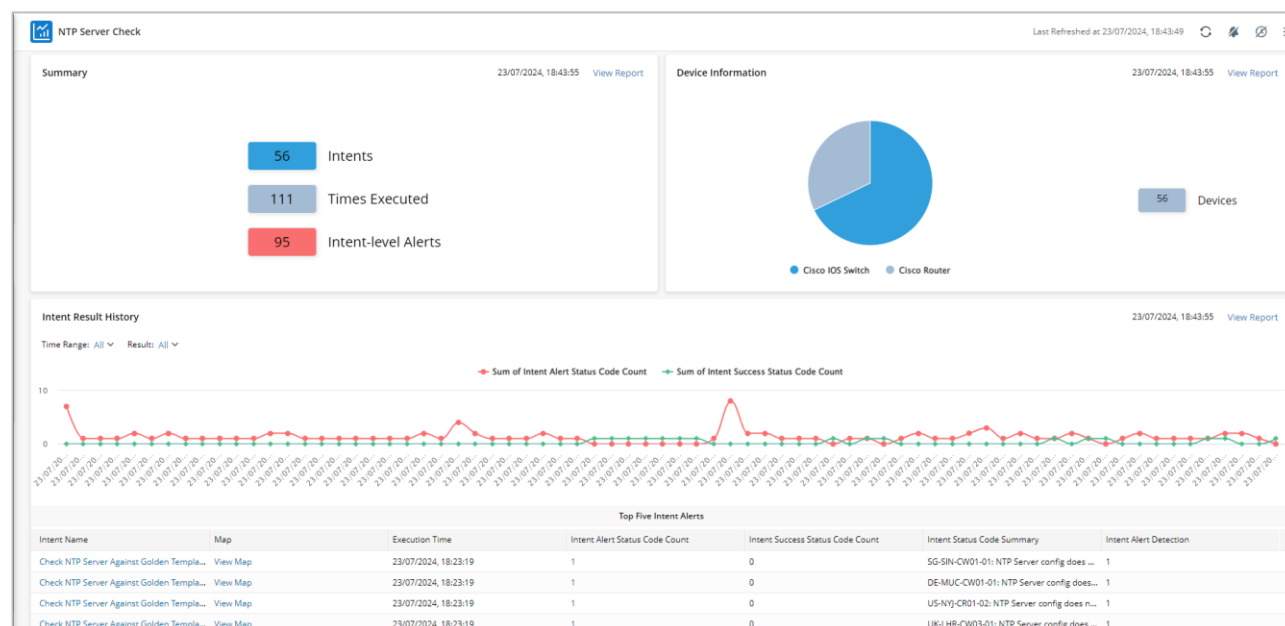
NTP Server Config ...

Items: 45 rows 8 columns Search...

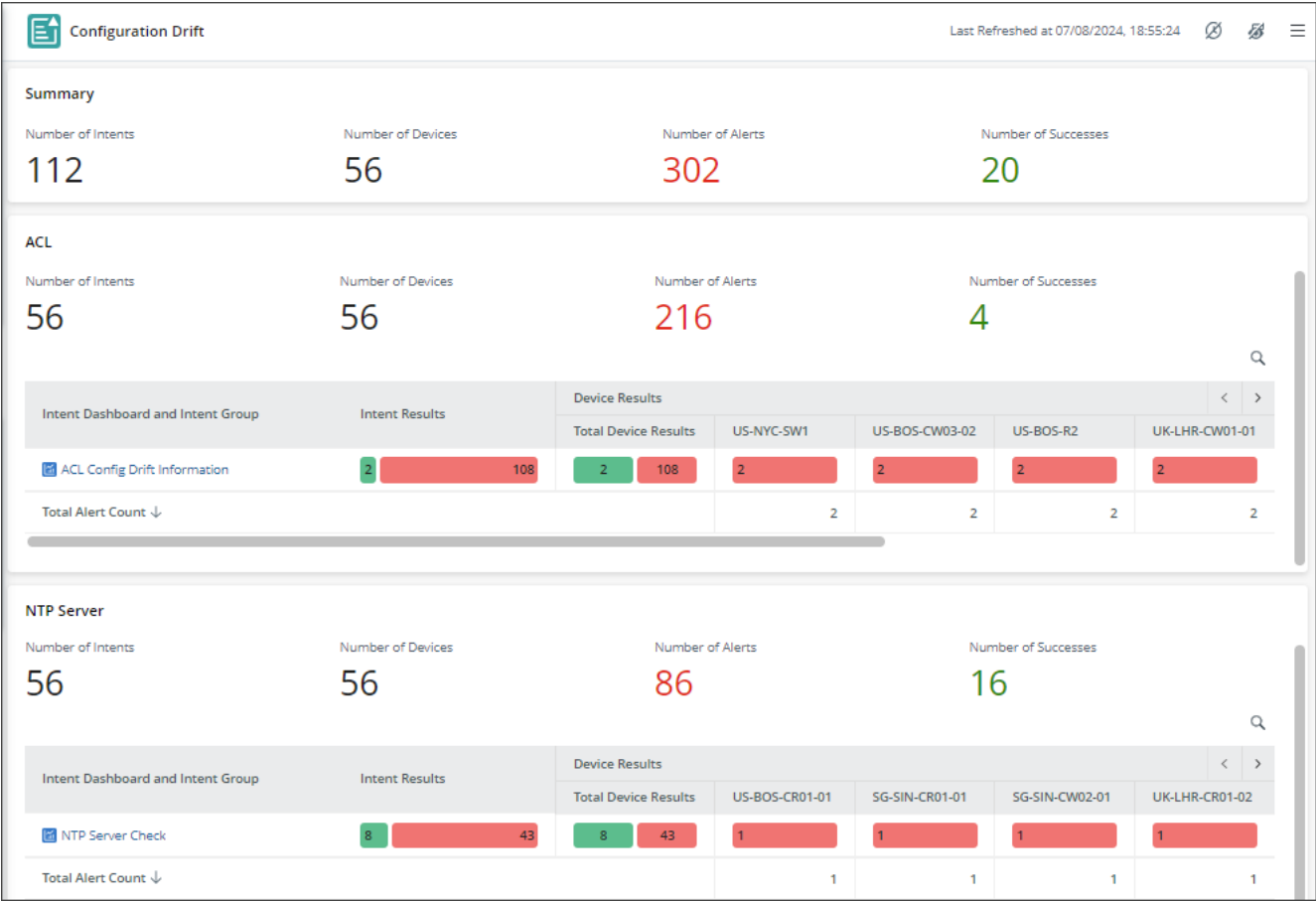
Device	Site	Software Version	Golden Config
US-BOS-CF01-01/stby	My Network\NA\US-BOS	9.5(2)T4	ntp server (\$_dummy=10.10.10.10)
US-BOS-CW03-02	My Network\NA\US-BOS	15.2(20170809:194209)	ntp server (\$_dummy=10.10.10.10)
US-BOS-CW04-02	My Network\NA\US-BOS	15.2(20170809:194209)	ntp server (\$_dummy=10.10.10.10)
US-BOS-SW1	My Network\test-site-jun	15.2(HI_20170202)FLO_DSGS7	
US-BOS-CW03-01	My Network\NA\US-BOS	15.2(20170809:194209)	ntp server (\$_dummy=10.10.10.10)
US-NYJ-CW02-02	My Network\NA\US-NYJ	15.2(20170809:194209)	
US-BOS-CW04-01	My Network\NA\US-BOS	15.2(20170809:194209)	ntp server (\$_dummy=10.10.10.10)
US-NYJ-CW02-01	My Network\NA\US-NYJ	15.2(20170809:194209)	
UK-LHR-CR01-01	My Network\test-container-s...	15.4(2)T4	
US-BOS-CW02-01	My Network\NA\US-BOS	15.2(20170809:194209)	ntp server (\$_dummy=10.10.10.10)

Export Close

Create an Intent Dashboard from the ADT:



Add the dashboard you just created to the existing Summary Dashboard, **Configuration Drift**, created in Section 7.1.7:



7.4 Check Configuration Settings in Public Cloud

Intent-based automation can also be used to check the configuration drift against the standard configurations or best practices in the public cloud. For this use case, you will retrieve the data using **API** and compare this configuration data with the recommended one.

You can follow the same flow of Section 7.1. The difference is that you retrieve the data via API instead of the CLI. The data returned by API is usually structured data that can be easily saved as a **Json table**.

We will use the AWS EC2 configuration as an example.

An example of the final ADT is like:

AWS EC2 Configuration Settings			
Table Builder		Last Updated at: 07/25/2024 05:22 PM	Rebuild Table
Description: Type description here...			
Items: 118 Rows 3 Columns		Search...	Advanced Filter: Undefined
No.	Device	Check EC2 Config	Intent Status Code
1	&2*`n/(&(i-094cb7dd86455412c)	Check AWS EC2 Configuration Against Baseline &2*`n/(&(i-094cb7...	On this VM &2*`n/(&(i-094cb7dd86455412c) instance type, availa...
2	(i-00aefbb9ecd7b69de)	Check AWS EC2 Configuration Against Baseline (i-00aefbb9ecd7b69...	On this VM (i-00aefbb9ecd7b69de) instance type, availability zone...
3	(i-03a18e36322f36e25)	Check AWS EC2 Configuration Against Baseline (i-03a18e36322f36e...	On this VM (i-03a18e36322f36e25) instance type, availability zone...
4	(i-04b6d5d8acdd36c9d)	Check AWS EC2 Configuration Against Baseline (i-04b6d5d8acdd36...	On this VM (i-04b6d5d8acdd36c9d) instance type, availability zon...
5	(i-05363037671558dd8)	Check AWS EC2 Configuration Against Baseline (i-05363037671558...	On this VM (i-05363037671558dd8) instance type, availability zon...
6	(i-0bdd3caa55517ba3c)	Check AWS EC2 Configuration Against Baseline (i-0bdd3caa55517b...	On this VM (i-0bdd3caa55517ba3c) instance type, availability zone...
7	(i-0bf20d0e607b910e9)	Check AWS EC2 Configuration Against Baseline (i-0bf20d0e607b91...	On this VM (i-0bf20d0e607b910e9) instance type, availability zone...
8	5325-to-7925-sharing-Subnet-ec2	Check AWS EC2 Configuration Against Baseline 5325-to-7925-shari...	On this VM 5325-to-7925-sharing-Subnet-ec2-1(i-02ce07443072f33...
9	5325-to-7925-sharing-TGW-ec2-1(i	Check AWS EC2 Configuration Against Baseline 5325-to-7925-shari...	On this VM 5325-to-7925-sharing-TGW-ec2-1(i-06c23d9c39f1b129a...
10	5325To7925-VPC2-instance(i-0c985	Check AWS EC2 Configuration Against Baseline 5325To7925-VPC2-i...	On this VM 5325To7925-VPC2-instance(i-0c985b246b1801c1e) ins...
11	7925-vpc1-ec2(i-06bbadaec14656c	Check AWS EC2 Configuration Against Baseline 7925-vpc1-ec2(i-06...	On this VM 7925-vpc1-ec2(i-06bbadaec14656c48) instance type, a...
12	ASAv10(i-0fa494c7d572e5478)	Check AWS EC2 Configuration Against Baseline ASAv10(i-0fa494c7d...	On this VM ASAv10(i-0fa494c7d572e5478) instance type, availabili...
13	ASAv10-NACL-appliance-ACL-Test-n	Check AWS EC2 Configuration Against Baseline ASAv10-NACL-appli...	On this VM ASAv10-NACL-appliance-ACL-Test-notcommonlyused(i-...
14	ASAv10-duplicateip-ec2-103.6(i-0b	Check AWS EC2 Configuration Against Baseline ASAv10-duplicateip...	On this VM ASAv10-duplicateip-ec2-103.6(i-0b264b6506edaae5b) i...
15	AWS-Firewall-Lab-Instance-01(i-02t	Check AWS EC2 Configuration Against Baseline AWS-Firewall-Lab-In...	On this VM AWS-Firewall-Lab-Instance-01(i-02bd82556acef3e05) i...

7.4.1 Prerequisites

To prepare for this exercise, create a Device Group for all the **AWS EC2 Instances** by the dynamic criteria: **Device Type** matches any **AWS EC2 Instance**.

The screenshot shows the NetBrain Next-Gen interface. On the left, the 'Device Group Properties' panel is visible. On the right, the 'Dynamic Search Device' dialog is open. The dialog has a 'Search Scope' set to 'All Devices'. Under 'Device Criteria', criterion A is 'Device Type' which 'Matches any' 'AWS EC2 Instance'. The 'Boolean Expression' is 'A'. A 'Search' button is at the bottom right. Below the dialog, a table shows search results for EC2 instances.

Hostname	Vendor	Model	Management IP
&2*`\\n/&(-094cb7dd8645541...	Amazon	EC2 Instance	172.16.101.75
(i-00aefbb9ecd7b69de)	Amazon	EC2 Instance	172.31.11.208
(i-03a18e36322f36e25)	Amazon	EC2 Instance	172.26.5.5
(i-04b6d5d8acdd36c9d)	Amazon	EC2 Instance	172.26.4.11
(i-05363037671558dd8)	Amazon	EC2 Instance	10.30.1.192

Also, you will need a script to retrieve the data from the AWS. NetBrain Automation Library provides these scripts. You can ask the Netbrain support team if your request is not covered in the library. For now, you can use the following script:

```
""  
  
Begin Declare Input Parameters
```

```
[  
  
]
```

```
End Declare
```

```
For sample
```

```
[  
  
  {"name": "$param1"},  
  {"name": "$param2"}  
  
]
```

```
""
```

```
import json
```

```
import datetime
```

```

import ast

def BuildParameters(context, device_name, params):
    node_props = GetDeviceProperties(context, device_name, {'techName': 'Amazon AWS', 'paramType': 'SDN', 'params': ['nbPathValue', 'RegionName'] })
    dn = node_props['params']['nbPathValue'].rpartition('/ec2/')[2]
    dev_id = node_props['params']['nbPathValue'].split('/')[4]

    apiServerId = node_props['params']['apiServerId']
    RegionName = node_props['params']['RegionName']
    rtn_params = [{ 'devName': device_name, 'dn': dn, 'apiServerId': apiServerId, 'RegionName': RegionName, 'dev_id': dev_id}]
    return rtn_params

def RetrieveData(rtn_params):
    if isinstance(rtn_params, str):
        rtn_params = json.loads(rtn_params)

    param = rtn_params
    dn = rtn_params['dn'] if 'dn' in rtn_params else ""
    rtn_res = []
    param['region_name'] = rtn_params['RegionName']
    param['resource_type'] = 'ec2'
    param['func_name'] = 'describe_instances'
    param['field_name'] = 'Reservations'
    param['func_param'] = {'InstanceIds': [param['dev_id']]}
    #raise NameError(param)
    res = get_aws_resource_data(param)
    data = json.loads(res)
    return json.dumps(data[0]['Instances'], indent=4)

```

7.4.2 Create Intent to Check AWS EC2 Configuration

Create a Network Intent, **Check AWS EC2 Configuration** from the **Intent Manager**. Select an AWS EC2 instance as the seed device, e.g., **7925-vpc1-ec2(i-06bbadaec14656c48)**.

The screenshot shows the 'Network Intent (Edit Mode)' window in the NetBrain Intent Manager. The intent is named 'Check AWS EC2 Configuration Against Baseline'. The 'Seed Logic' tab is active, and the '+ Device' button is highlighted with a red circle and the number 2. The 'Select Devices' dialog is open, showing a list of devices. The 'Device Type' radio button is selected. The search filter is '7925'. The table lists various AWS resources, with '7925-vpc1-ec2(i-06bbadaec14656c48)' highlighted in blue and marked with a red circle and the number 3. The '1 Devices Selected' panel on the right shows the selected device. The 'OK' button is highlighted in yellow.

Network Intent (Edit Mode)

1 Check AWS EC2 Configuration Against Baseline 1 Diagnosis Tree Run with Live Data Save Help

Type description here...

1 Seed Logic Replication Logic

+ Device 2 Intent Variables: Manager Tag: + Add

Select Devices

Select Devices by: Device Type Device Group Site

All Device Types 7925

Hostname	Mgmt IP	Vendor	Model
5325-to-7925-sharing-Subnet-ec2-1(...)	10.55.2.100	Amazon	EC2 Instance
5325-to-7925-sharing-TGW-ec2-1(i-0...	172.31.3.38	Amazon	EC2 Instance
5325To7925-VPC2-Instance(i-0c985b...	172.31.3.87	Amazon	EC2 Instance
7925-vpc1-ec2(i-06bbadaec14656c48)	172.31.36.178	Amazon	EC2 Instance
7925CA-to-5325NV(vgw-05f0992976...		Amazon	AWS Virtual Private Gat...
7925VPC1(vpc-9d91ebe7)		Amazon	VPC Router
7925VPC2(vpc-0b4a89e5304d086fa)		Amazon	VPC Router
EC2CreatedInSharedSubnetBy0701...	172.29.0.158	Amazon	EC2 Instance
diff-region-vpc-peer-7925-do-not-de...	172.32.1.6	Amazon	EC2 Instance
unattached-eni-7925-to-5325(en-Of...		Amazon	AWS Unattached Netwo...
vpc-peer-diff-region-7925-main(vpc-...		Amazon	VPC Router


1 Devices Selected

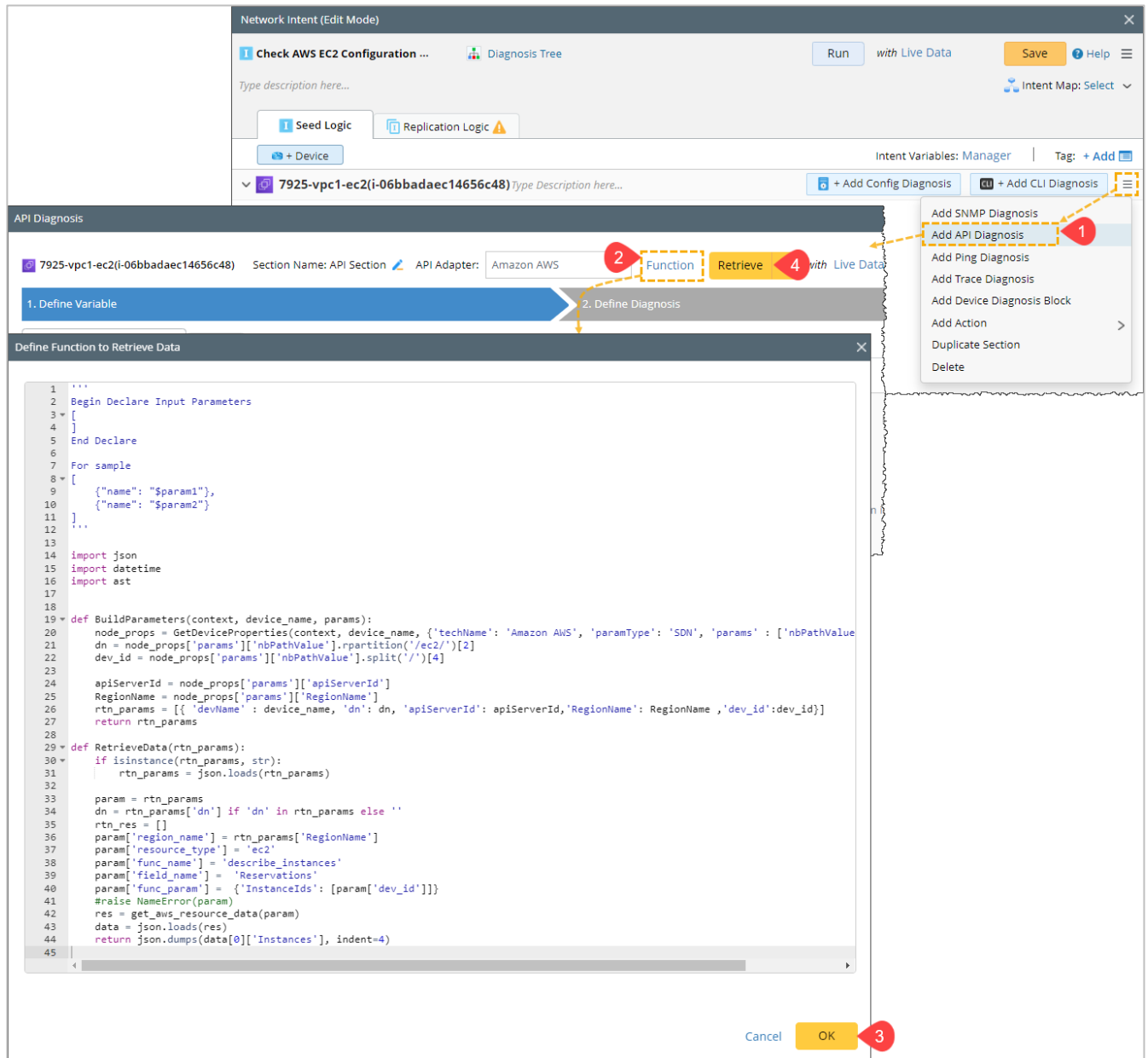
7925-vpc1-ec2(i-06bbadaec14656c...

Cancel OK

7.4.2.1 Add API Diagnosis

In this section, you will retrieve the EC2 configuration settings via the API and parse the data:

1. Click  menu and open **Add API Diagnosis** from dropdown.
2. In the **API Diagnosis** window, click **Function** to define how to retrieve data. Copy and paste the script in the prerequisites section.
3. Click **OK** to save the Function.
4. Click **Retrieve** to retrieve API sample data with **Live Data** source.



Network Intent (Edit Mode)

1 Check AWS EC2 Configuration ... Diagnosis Tree Run with Live Data Save Help

Type description here...

Seed Logic Replication Logic

+ Device

Intent Variables: Manager Tag: + Add

7925-vpc1-ec2(i-06bbadaec14656c48) Type Description here...

+ Add Config Diagnosis + Add CLI Diagnosis

API Diagnosis

7925-vpc1-ec2(i-06bbadaec14656c48) Section Name: API Section API Adapter: Amazon AWS Function Retrieve with Live Data

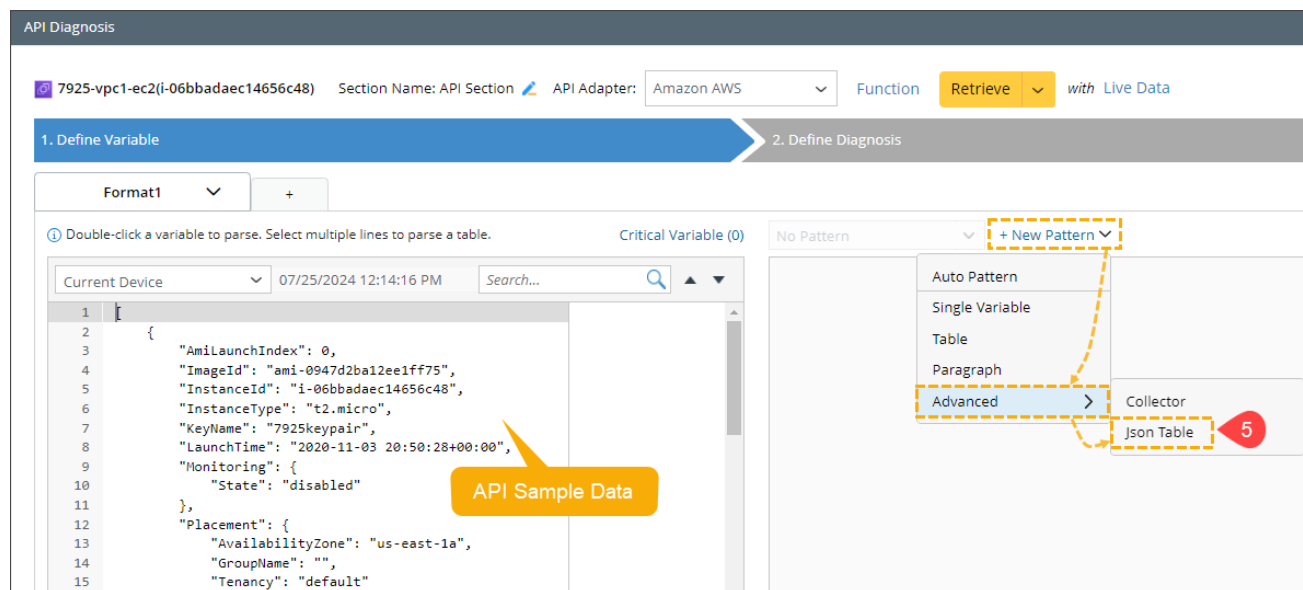
1. Define Variable 2. Define Diagnosis

Define Function to Retrieve Data

```
1 '''
2 Begin Declare Input Parameters
3 [
4 ]
5 End Declare
6
7 For sample
8 [
9   {"name": "$param1"},
10  {"name": "$param2"}
11 ]
12 '''
13
14 import json
15 import datetime
16 import ast
17
18
19 def BuildParameters(context, device_name, params):
20     node_props = GetDeviceProperties(context, device_name, {'techName': 'Amazon AWS', 'paramType': 'SDN', 'params': ['nbPathValue']})
21     dn = node_props['params']['nbPathValue'].rpartition('/ec2/')[2]
22     dev_id = node_props['params']['nbPathValue'].split('/')[4]
23
24     apiServerId = node_props['params']['apiServerId']
25     RegionName = node_props['params']['RegionName']
26     rtn_params = [{'devName': device_name, 'dn': dn, 'apiServerId': apiServerId, 'RegionName': RegionName, 'dev_id': dev_id}]
27     return rtn_params
28
29 def RetrieveData(rtn_params):
30     if isinstance(rtn_params, str):
31         rtn_params = json.loads(rtn_params)
32
33     param = rtn_params
34     dn = rtn_params['dn'] if 'dn' in rtn_params else ''
35     rtn_res = []
36     param['region_name'] = rtn_params['RegionName']
37     param['resource_type'] = 'ec2'
38     param['func_name'] = 'describe_instances'
39     param['field_name'] = 'Reservations'
40     param['func_param'] = {'InstanceIds': [param['dev_id']]}
41     #raise NameError(param)
42     res = get_aws_resource_data(param)
43     data = json.loads(res)
44     return json.dumps(data[0]['Instances'], indent=4)
45
```

Cancel OK

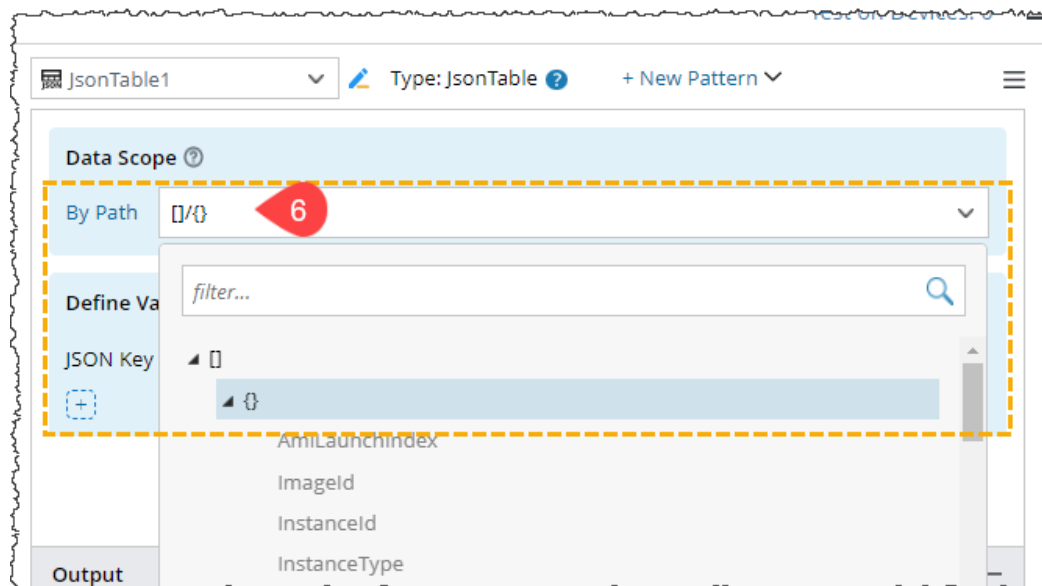
5. In the the **+ New Pattern** dropdown, click **Advanced** > **Json Table**. The data returned from the AWS API is the Json formatted.



6. Define Data Scope.


In the By Path dropdown, select **[]** (Square Brackets) and then **{}** (Curly Brackets).


The By Path value will be: **[]/{}]**

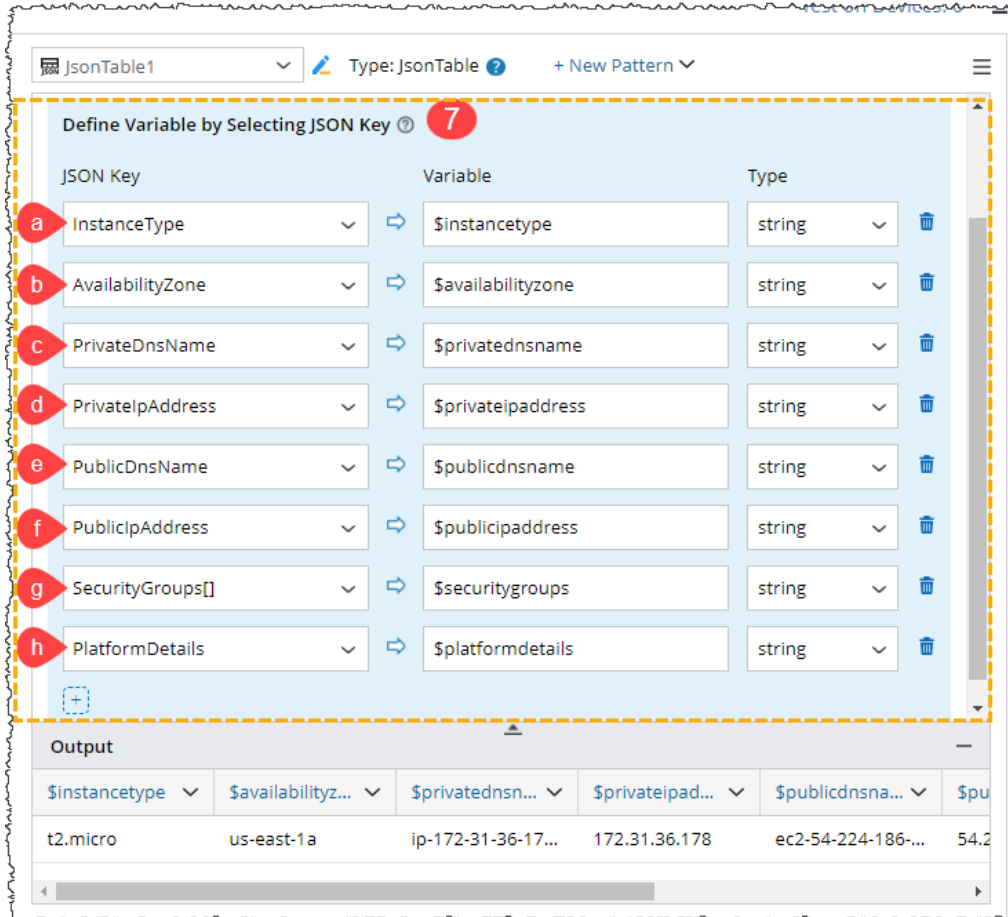


NOTE: You can define some important keys as shown below. In this example, only a few keys are selected, but you can choose the keys according to your specific use case.

Define the Variables by Selecting **JSON Key**.

7. To add the JSON Key, click  to add the **InstanceType** JSON Key from the dropdown.

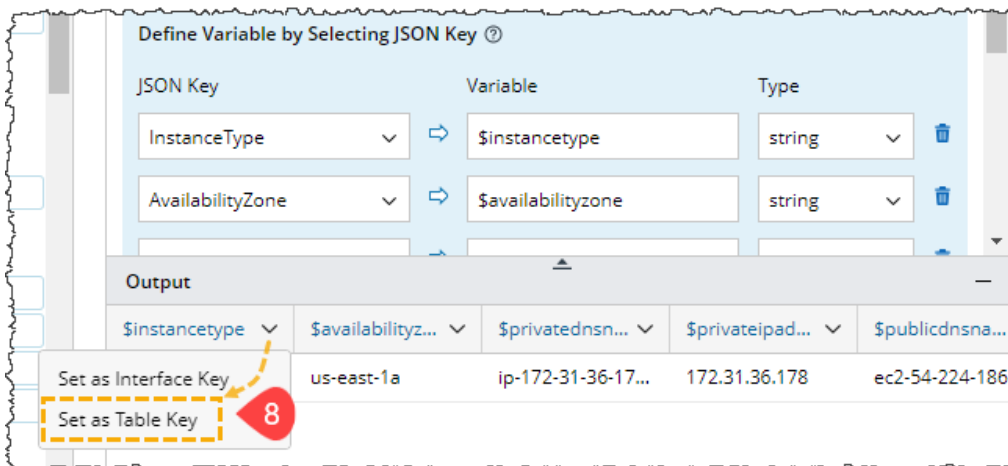
Click  and add AvailabilityZone, PrivateDnsName, PrivateIpAddress, PublicDnsName, PublicIpAddress, SecurityGroups[], and PlatformDetails.



JSON Key	Variable	Type
InstanceType	\$instancetype	string
AvailabilityZone	\$availabilityzone	string
PrivateDnsName	\$privatednsname	string
PrivateIpAddress	\$privateipaddress	string
PublicDnsName	\$publicdnsname	string
PublicIpAddress	\$publicipaddress	string
SecurityGroups[]	\$securitygroups	string
PlatformDetails	\$platformdetails	string

\$instancetype	\$availability...	\$privatednsn...	\$privateipad...	\$publicdnsna...	\$pu
t2.micro	us-east-1a	ip-172-31-36-17...	172.31.36.178	ec2-54-224-186-...	54.2

8. In the Output section, set the **\$instancetype** as the table key.

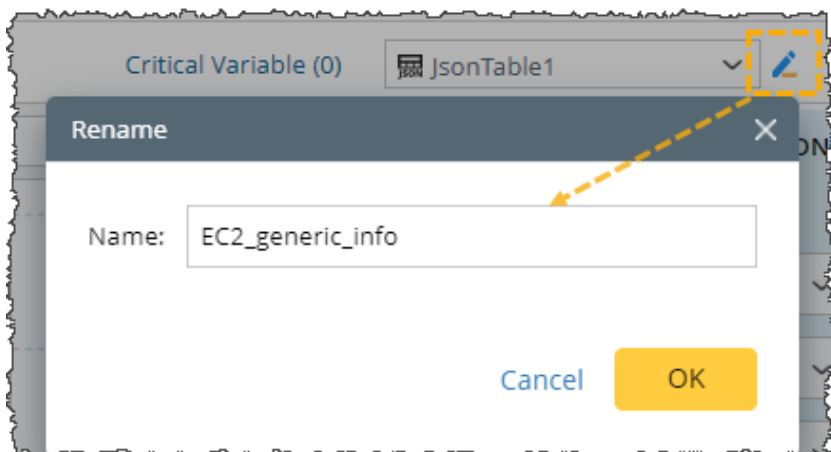


JSON Key	Variable	Type
InstanceType	\$instancetype	string
AvailabilityZone	\$availabilityzone	string

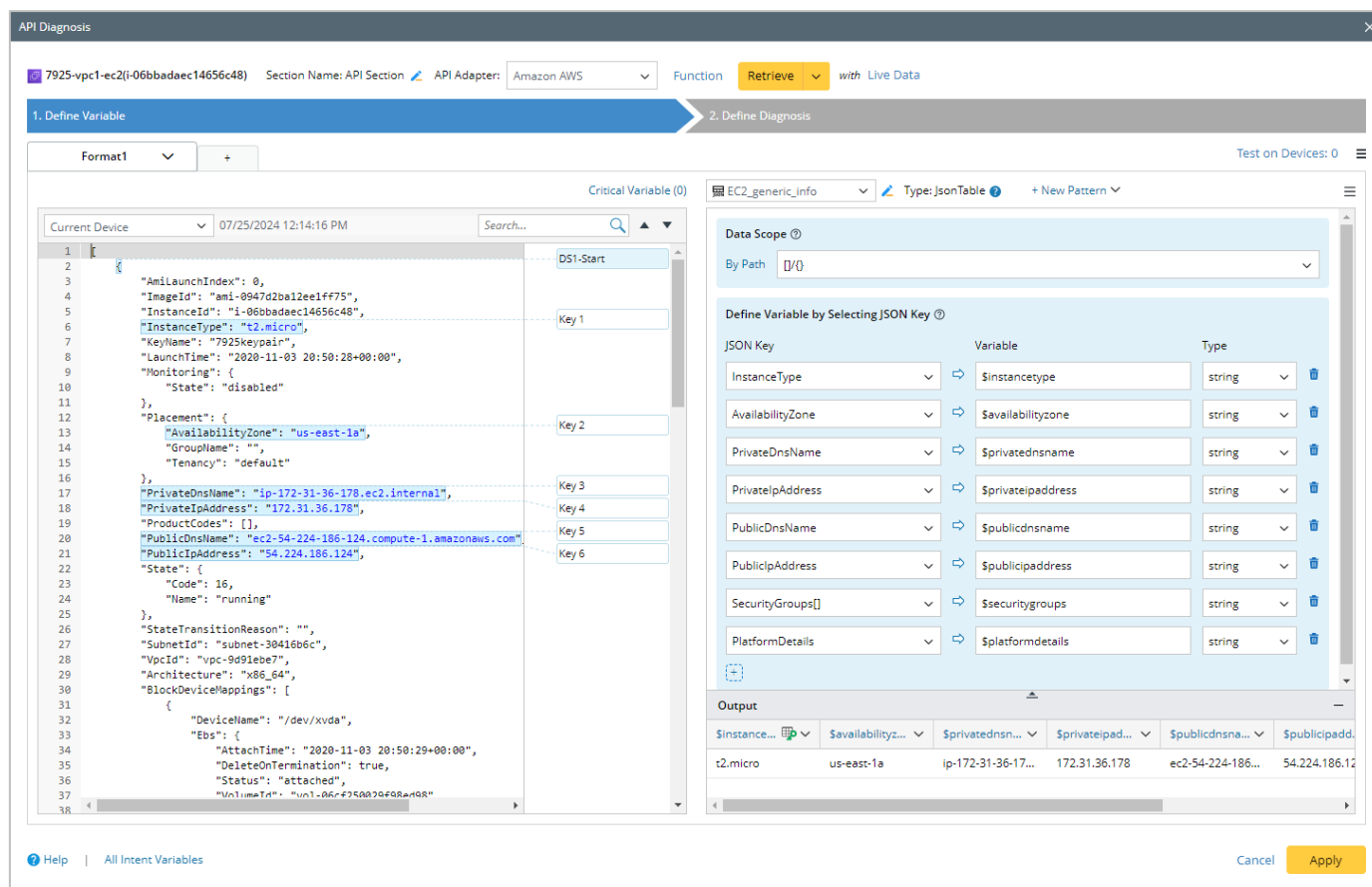
\$instancetype	\$availability...	\$privatednsn...	\$privateipad...	\$publicdnsna...
us-east-1a	ip-172-31-36-17...	172.31.36.178	ec2-54-224-186	

Set as Interface Key
Set as Table Key

9. Click the **pen** icon to rename **JsonTable1** to **EC2_Generic_Info**.



The whole **API Diagnosis** window will be:



7.4.2.2 Define Diagnosis

Defining the diagnosis from the API data is the same as the CLI or Config Diagnosis:


1. Go to the **Define Diagnosis** section to define the diagnosis logic.
2. Click **Add Diagnosis** to define conditions and Intent output message.
3. Enter the diagnosis name, e.g., **Recourse Change Attributes**.
4. Tick the **Loop Table Rows** checkbox and select the Table Variable (**EC2_generic_info**) and Table Key (**instancetype**).


The screenshot shows the '2. Define Diagnosis' section of a configuration tool. It features a blue header bar with the title '2. Define Diagnosis' and a red callout '1'. Below the header, there are two buttons: 'Add Note' and 'Add Diagnosis', with a red callout '2' pointing to the latter. A text input field for 'Name' contains 'Resource Change Attributes' and is marked with a red callout '3'. To its right is an 'Anchor' dropdown menu. Below the name field is a text area with the placeholder 'Type description of the diagnosis...'. At the bottom, there is a section for 'Loop Table Rows' which includes a checked checkbox, a dropdown menu showing 'EC2_generic_info', and another dropdown menu showing 'instancetype' as the 'Table Key'. A red callout '4' points to the 'instancetype' dropdown. Below this section is an 'If' condition dropdown.

5. Define the **If** condition for each **JSON Key Variable** as detailed in the image. For all the Variables, compare the **Current** configuration with the **Baseline** configuration.

The Boolean Expression is **A or B or C or D or E or F or G or H**.

2. Define Diagnosis

 Add Note

 Add Diagnosis

Can also click a variable on the left to add automation.

☒ Loop Table Rows

















 EC2_generic_info ▾

Table Key: Please Select... ▾



▾ If

5

- A  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- a instancetype ▾ Does not equal ▾ instancetype ▾ 
- B  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- b availabilityzone ▾ Does not equal ▾ availabilityzone ▾ 
- C  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- c privatednsname ▾ Does not equal ▾ privatednsname ▾ 
- D  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- d privateipaddress ▾ Does not equal ▾ privateipaddress ▾ 
- E  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- e publicdnsname ▾ Does not equal ▾ publicdnsname ▾ 
- F  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- f publicipaddress ▾ Does not equal ▾ publicipaddress ▾ 
- G  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- g securitygroups ▾ Does not equal ▾ securitygroups ▾ 
- H  7925-vpc1-ec2(i-... Current ▾ Baseline ▾
- h platformdetails ▾ Does not equal ▾ platformdetails ▾ 
- I Select Variable ▾

i Boolean Expression: A and B and C and D and E and F and G and H

+ Add Elself

+ Add Else

Cancel

Apply

6. Define Intent Output message.

Enter a message under the **Then** and **Else** output areas to appear as the result of the diagnosis.

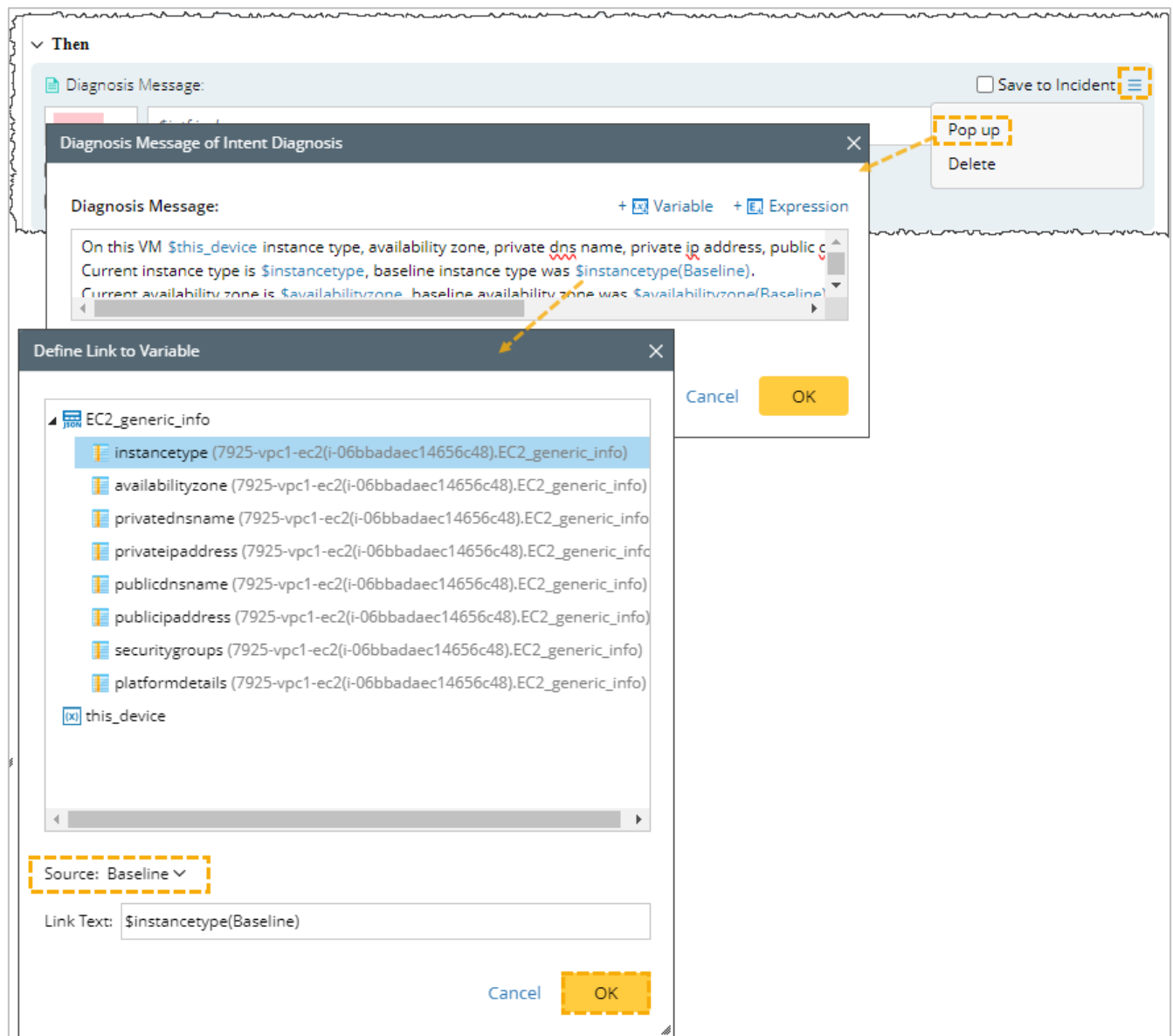
The screenshot shows the configuration interface for a diagnosis. It is divided into two main sections: **Then** and **Else**.

- Then Section (a):**
 - Color:** A red color is selected (b).
 - Message:** A message is entered in the text field (c): "On this VM \$this_device instance type, availability zone, private dns name, private ip address, public dns name, public ip address, security groups, and platform details have not changed."
 - Status Code for Device:** Set to "Error" (d).
 - Status Code for Intent:** Set to "Error" (d).
- Else Section (c):**
 - Color:** A green color is selected (d).
 - Message:** A message is entered in the text field (c): "On this VM \$this_device instance type, availability zone, private dns name, private ip address, public dns name, public ip address, security groups, and platform details have not changed."
 - Status Code for Device:** Set to "Success" (d).
 - Status Code for Intent:** Set to "Success" (d).

At the bottom right, there are "Cancel" and "Apply" buttons. The "Apply" button is highlighted with a red circle and the number 7.

Then: define the Red color, message and status in case **If** condition is satisfied as follows:

On this VM `$this_device` instance type, availability zone, private dns name, private ip address, public dns name, public ip address, security groups, and platform details.
Current instance type is `$instancetype`, baseline instance type was `$instancetype(Baseline)`.
Current availability zone is `$availabilityzone`, baseline availability zone was `$availabilityzone(Baseline)`.
Current private DNS is `$privatednsname`, baseline DNS was `$privatednsname(Baseline)`.
Current private IP is `$privateipaddress`, baseline private IP was `$privateipaddress(Baseline)`.
Current public DNS is `$publicdnsname`, baseline public DNS was `$publicdnsname(Baseline)`.
Current public IP is `$publicipaddress`, baseline public IP was `$publicipaddress(Baseline)`.
Current security groups are `$securitygroups`, baseline security group were `$securitygroups(Baseline)`.
Current platform details are `$platformdetails`, baseline platform details were `$platformdetails(Baseline)`.



Else: enter a message indicating no change:

On this VM \$this_device instance type, availability zone, private dns name, private ip address, public dns name, public ip address, security groups, and platform details have not changed.

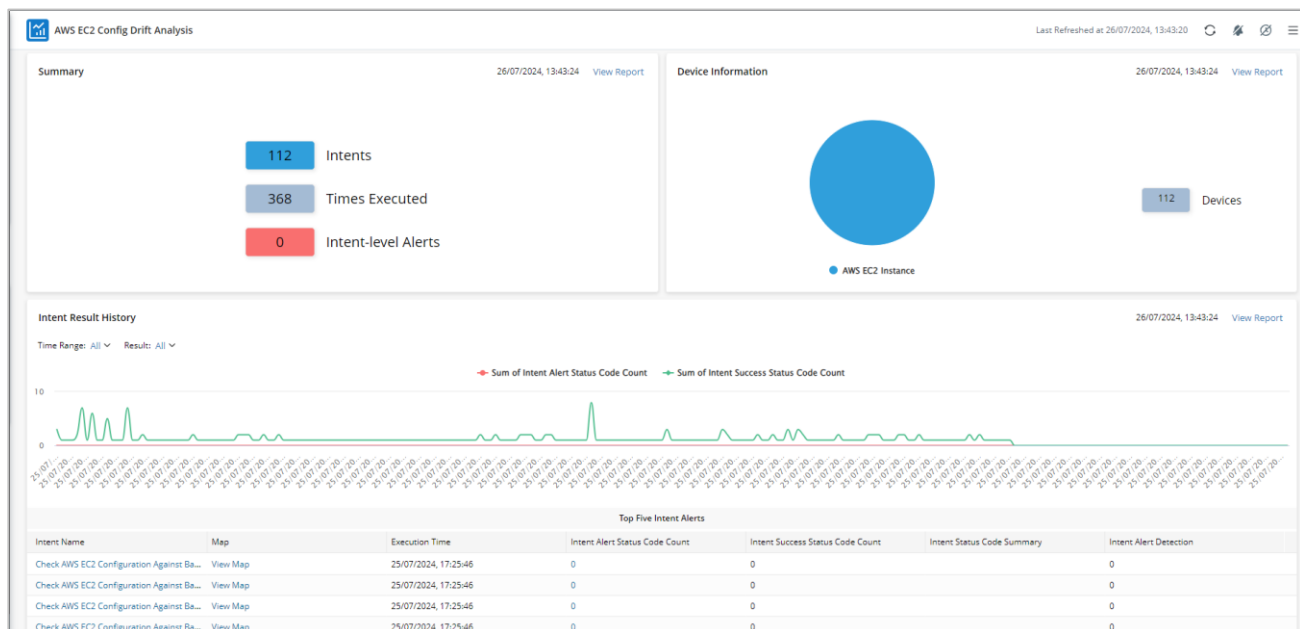
7.4.3 Replicate Intent to All AWS EC2

You can follow the same flow of Section 7.1 to replicate the intent to all AWS EC2 instances. Select the device group you created in the Section 7.4.1.

After running once all replicate intent, rebuild the table to see the intent status codes in the ADT table:

AWS EC2 Configuration Settings			
Table Builder		Last Updated at: 07/25/2024 05:22 PM	Rebuild Table
Description: Type description here...			
Items: 118 Rows 3 Columns			
No.	Device	Check EC2 Config	Intent Status Code
1	&2*` \n&(i-094cb7dd86455412c)	Check AWS EC2 Configuration Against Baseline &2*` \n&(i-094cb7...	On this VM &2*` \n&(i-094cb7dd86455412c) instance type, availa...
2	(i-00aefbb9ecd7b69de)	Check AWS EC2 Configuration Against Baseline (i-00aefbb9ecd7b69...	On this VM (i-00aefbb9ecd7b69de) instance type, availability zone...
3	(i-03a18e36322f36e25)	Check AWS EC2 Configuration Against Baseline (i-03a18e36322f36e...	On this VM (i-03a18e36322f36e25) instance type, availability zone...
4	(i-04b6d5d8acdd36c9d)	Check AWS EC2 Configuration Against Baseline (i-04b6d5d8acdd36c...	On this VM (i-04b6d5d8acdd36c9d) instance type, availability zon...
5	(i-05363037671558dd8)	Check AWS EC2 Configuration Against Baseline (i-05363037671558...	On this VM (i-05363037671558dd8) instance type, availability zon...
6	(i-0bdd3caa55517ba3c)	Check AWS EC2 Configuration Against Baseline (i-0bdd3caa55517b...	On this VM (i-0bdd3caa55517ba3c) instance type, availability zone...
7	(i-0bf20d0e607b910e9)	Check AWS EC2 Configuration Against Baseline (i-0bf20d0e607b91...	On this VM (i-0bf20d0e607b910e9) instance type, availability zone...
8	5325-to-7925-sharing-Subnet-ec2-1(i-02ce07443072f3320)	Check AWS EC2 Configuration Against Baseline 5325-to-7925-shari...	On this VM 5325-to-7925-sharing-Subnet-ec2-1(i-02ce07443072f33...
9	5325-to-7925-sharing-TGW-ec2-1(i-06c23d9c39f1b129a)	Check AWS EC2 Configuration Against Baseline 5325-to-7925-shari...	On this VM 5325-to-7925-sharing-TGW-ec2-1(i-06c23d9c39f1b129a...
10	5325To7925-VPC2-instance(i-0c985b246b1801c1e)	Check AWS EC2 Configuration Against Baseline 5325To7925-VPC2-I...	On this VM 5325To7925-VPC2-instance(i-0c985b246b1801c1e) ins...
11	7925-vpc1-ec2(i-06bbadaec14656c48)	Check AWS EC2 Configuration Against Baseline 7925-vpc1-ec2(i-06...	On this VM 7925-vpc1-ec2(i-06bbadaec14656c48) instance type, a...
12	ASAv10(i-0fa494c7d572e5478)	Check AWS EC2 Configuration Against Baseline ASAv10(i-0fa494c7d...	On this VM ASAv10(i-0fa494c7d572e5478) instance type, availabi...
13	ASAv10-NACL-appliance-ACL-Test-notcommonlyused(i-0eaa22b7bf3de...	Check AWS EC2 Configuration Against Baseline ASAv10-NACL-appli...	On this VM ASAv10-NACL-appliance-ACL-Test-notcommonlyused(i-...
14	ASAv10-duplicateip-ec2-103.6(i-0b264b6506edaae5b)	Check AWS EC2 Configuration Against Baseline ASAv10-duplicateip...	On this VM ASAv10-duplicateip-ec2-103.6(i-0b264b6506edaae5b) l...
15	AWS-Firewall-Lab-Instance-01(i-02bd82556acef3e05)	Check AWS EC2 Configuration Against Baseline AWS-Firewall-Lab-in...	On this VM AWS-Firewall-Lab-Instance-01(i-02bd82556acef3e05) l...
16	AWS-Firewall-Lab-Instance-02(i-009aeb5c93aface06)	Check AWS EC2 Configuration Against Baseline AWS-Firewall-Lab-in...	On this VM AWS-Firewall-Lab-Instance-02(i-009aeb5c93aface06) in...
17	AWS-Firewall-NAT-Lab-Instance(i-04527181641150e4b)	Check AWS EC2 Configuration Against Baseline AWS-Firewall-NAT-L...	On this VM AWS-Firewall-NAT-Lab-Instance(i-04527181641150e4b) ...
18	Benchmark-Added-EC2(i-0167fe60a98a5a95a)	Check AWS EC2 Configuration Against Baseline Benchmark-Added...	On this VM Benchmark-Added-EC2(i-0167fe60a98a5a95a) instanc...

You can also create a Dashboard to view the results:



8 Network Assessment Case Study: Failover

Failover and redundancy are one of the important aspects of your network. Network outages and performance degrading often happen when the failover occurs. In this chapter, you will create the intent to check the Failover status (whether the failover occurred) and assess the performance degradation, which usually occurs after the failover. Then, we are going to create automation to check the configuration consistency between the failover pair.

1. [Check Failover Status Change](#)
2. [Follow-up to Check the Performance Degrade After Failover](#)
3. [Check Failover Consistency of Pair Devices](#)

8.1 Check Failover Status Change

In this section, let us begin with creating a group of HSRP devices and then proceed with intent creation and dashboard as follows:

1. [Create a group for HSRP devices.](#)
2. [Create an intent to check the failover status change.](#)
3. [Replicate the intent to the HSRP device group using ADT.](#)
4. [Create dashboard.](#)

8.1.1 Create HSRP Device Group

First, you need to create a device group, **HSRP Devices**, to include all HSRP devices with the dynamic criteria, **Config File contains standby**, and **Vendor** contains **Cisco**.

The device group includes all Cisco HSRP devices:

The screenshot shows the 'Dynamic Search Device' dialog box. It has a 'Search Scope' dropdown set to 'All Devices'. Under 'Device Criteria', there are three rows: Row A has 'Config File' (1) with 'Contains' (2) and 'standby' (2); Row B has 'Vendor' (3) with 'Contains' (4) and 'cisco' (4); Row C has 'Select Criteria'. The 'Boolean Expression' field (5) contains 'A and B'. A yellow 'Search' button (6) is on the right. Below, the 'Search Result' section shows a table with 5 columns: Hostname, Vendor, Model, and Management IP. The table lists 5 devices. At the bottom right are 'Cancel' and 'OK' (7) buttons.

Dynamic Search Device

Search Scope: All Devices

Device Criteria:

A Config File (1) Contains (2) standby (2)

B Vendor (3) Contains (4) cisco (4)

C Select Criteria

Boolean Expression: A and B (5)

Search (6)

Search Result:

Hostname	Vendor	Model	Management IP
BJ-L2-Core-A	Cisco	catalyst356024TS	172.26.3.10
BJ-L2-coreB	Cisco	WS-C3560-24TS	172.24.101.3
BJ_L2_Core_3	Cisco	WS-C3750-24TS	172.26.3.20
BJ_L2_Core_4	Cisco	WS-C3750-24TS	172.24.101.5
BJ_core_3550	Cisco	WS-C3550-24	172.24.36.1

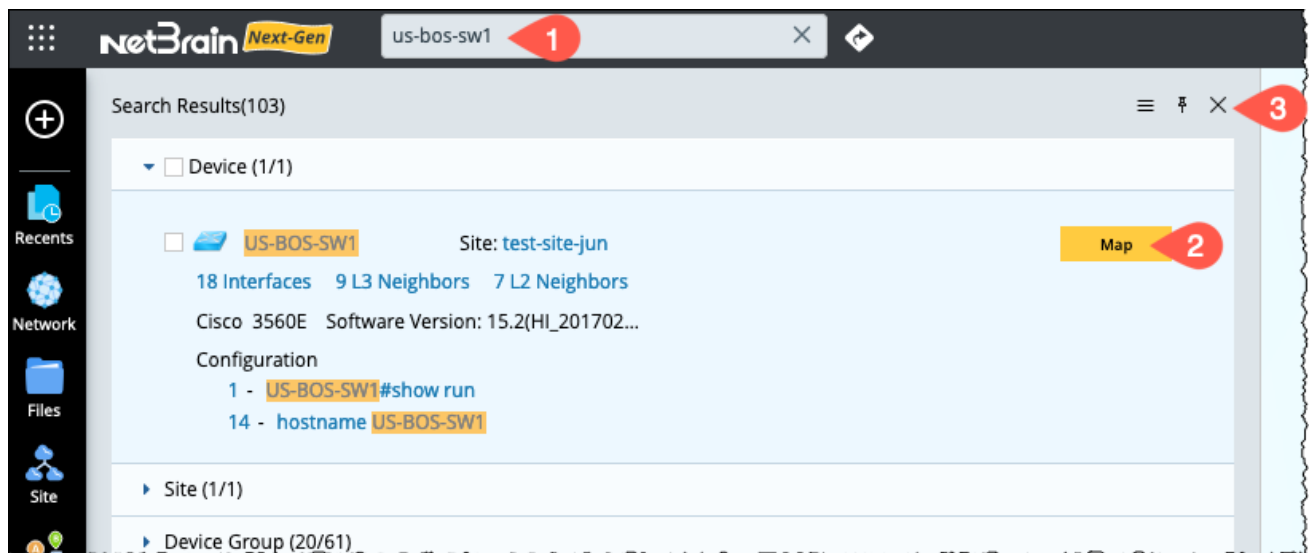
Cancel OK (7)

8.1.2 Create an intent to check the failover status change

1. [Select a device](#)
2. [Create a Paragraph Parser](#)
3. [Define the diagnosis to check the status](#)

8.1.2.1 Select a device

1. From the End User Desktop, click into the **Search Bar**, type **US-BOS-SW1**, and press **Enter**.
2. In the search results, click **Map** beside **US-BOS- SW1**.
3. Close the search results.



4. In the upper-left corner of the map, click **Intent** to expand the Intent pane. Then click the **Quick Intent** tab header.
5. In the **Input Command....** field, enter **show standby**.
6. Click **Retrieve**.
7. Click **New** located next to the **Select Parser** dropdown. A Visual parser window will appear on the screen to configure the variables.

Map12 (Master) ▾ * > Page 1 ▾ Summary Over

I Intent R Runbook D Data View

Auto Intent Quick Intent 4 Published Intents Map Intent

Collect Data US-BOS-SW1 ▾ <<

show standby 5 × ▾ Retrieve 6

05:11:19 PM: Successfully retrieved!

US-BOS-SW1 ▾ Parser: Select parser ▾ New 7

Add to Intent >>

07/23/2024 05:11:14 PM (Live) 🟢

```
JS-BOS-SW1>show standby
Vlan100 - Group 1 (version 2)
  State is Standby
    1 state change, last state change 31w0d
  Virtual IP address is 10.8.1.1
  Active virtual MAC address is 0000.0c9f.f001 (MAC Not In Use)
```

8.1.2.2 Create a Paragraph Parser

1. On **Line 2**, double-click **Vlan100** to parse the *interface variable* from the command output.

CLI Command Diagnosis

US-BOS-SW1 show standby Retrieve with Live Data

1. Define Variable 2. Define Diagnosis

Format1 +

Double-click a variable to parse. Select multiple lines to parse a table. Critical Variable (0)

Current Device 07/22/2024 06:40:32 PM Search...

1 US-BOS-SW1>show standby
2 Vlan100 1 (version 2)
3 State is Standby
4 1 state change, last state change 30w6d
5 Virtual IP address is 10.8.1.1
6 Active virtual MAC address is 0000.0c9f.f001 (MAC N
7 Local virtual MAC address is 0000.0c9f.f001 (v2 d
8 Hello time 5 sec, hold time 15 sec
9 Next hello sent in 3.696 secs
10 Preemption enabled
11 Active router is 10.8.1.3, priority 105 (expires in
12 MAC address is aabb.cc80.1500

P1-ID Line

Name: Pattern1 Type: Single

ID Line ^\$var1 - Group

2 Vlan100 - Group 1 (version 2)

2. In the right Variable Definition pane, update:
 - a) Variable name **\$var1** to **\$interface**.
 - b) Parser name **Pattern1** to **HSRP_failover**

CLI Command Diagnosis

US-BOS-SW1 show standby Retrieve with Live Data

1. Define Variable 2. Define Diagnosis

Format1 +

Double-click a variable to parse. Select multiple lines to parse a table. Critical Variable (0)

Current Device 07/22/2024 06:40:32 PM Search...

2 Vlan100 - Group 1 (version 2)
3 State is Standby
4 1 state change, last state change 30w6d
5 Virtual IP address is 10.8.1.1
6 Active virtual MAC address is 0000.0c9f.f001 (MAC N
7 Local virtual MAC address is 0000.0c9f.f001 (v2 d

P1-ID Line

Name: HSRP_failover Type: Single Multiple

ID Line ^\$interface up

2 Vlan100 - Group 1 (version 2)

3. Similar to the step 1, parse other following variables from the sample data:

- a) **\$State**
- b) **\$change_time**
- c) **\$virtual_ip**

The screenshot shows the NetBrain interface with two tabs: "1. Define Variable" and "2. Define Diagnosis". In the "Define Variable" tab, a list of variables is shown, with "Vlan100 - Group 1 (version 2)" selected. The variable's value is "State is Standby". In the "Define Diagnosis" tab, three variables are defined: "Var Line 1" (State is \$state), "Var Line 2" (last state change \$change_time), and "Var Line 3" (Virtual IP address is \$virtual_ip). Red dashed arrows point from the variable definitions in the "Define Diagnosis" tab to the corresponding values in the "Define Variable" tab. For example, "Var Line 1" points to "State is Standby", "Var Line 2" points to "last state change 30w6d", and "Var Line 3" points to "Virtual IP address is 10.8.1.1".

- d) **\$standby_router**
- e) **\$group_name**

The screenshot shows the NetBrain interface with two tabs: "1. Define Variable" and "2. Define Diagnosis". In the "Define Variable" tab, a list of variables is shown, with "Vlan100 - Group 1 (version 2)" selected. The variable's value is "State is Standby". In the "Define Diagnosis" tab, five variables are defined: "Var Line 1" (Virtual IP address is 10.8.1.1), "Var Line 2" (Standby router is \$Standby_router), "Var Line 3" (Group name is "\$group_name"), "Var Line 4" (Standby router is local), and "Var Line 5" (Group name is "hsrp-Vl100-1"). Red dashed arrows point from the variable definitions in the "Define Diagnosis" tab to the corresponding values in the "Define Variable" tab. For example, "Var Line 2" points to "Standby router is local", "Var Line 3" points to "Group name is 'hsrp-Vl100-1'", "Var Line 4" points to "Standby router is local", and "Var Line 5" points to "Group name is 'hsrp-Vl100-1'".

- Click **Apply** to exit Visual Parser Auto-Pattern mode
- Exit the Visual Parser window.

5

2. Define Diagnosis

Test on Devices: 0

variable (0) HSRP_failover Type: Paragraph + New Pattern

5 Virtual IP address is 10.8.1.1 > 3 Lines

Var Line 4 Standby router is \$Standby_router

13 Standby router is local > 3 Lines

Var Line 5 Group name is "\$group_name"

Output + Parse Lines

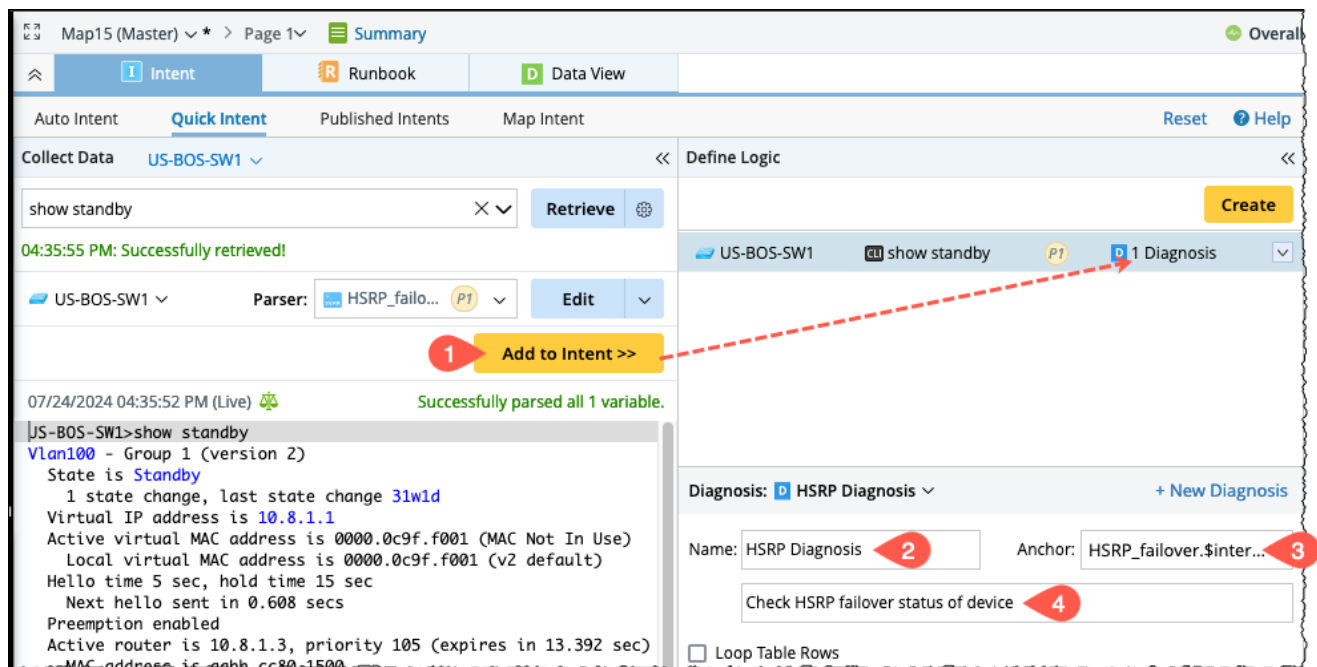
\$interface	\$state	\$change_time	\$virtual_ip	\$Standby_ro...
Vlan100	Standby	30w6d	10.8.1.1	local
Vlan100	Active	30w6d	10.8.1.8	10.8.1.3,
Vlan101	Standby	30w6d	10.8.1.17	local

Cancel 4 Apply

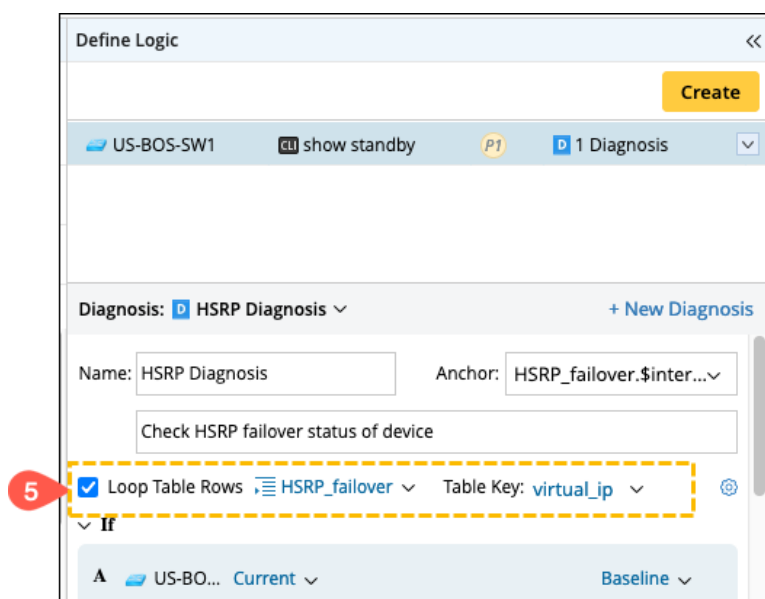
The output of the parsed variables

8.1.2.3 Define Diagnosis

1. Click **Add to Intent** >> back in the quick intent tab.
2. In the **Define Logic** pane, name the diagnosis **HSRP Diagnosis**.
3. Anchor: Select the variable **interface** from the drop-down.
4. Click in the **Type description of the diagnosis...** field and enter **Check HSRP failover status of device**.



5. Check-in **Loop Table Rows** and select the table **HSRP_failover** and **Virtual_ip** as table key from the corresponding dropdown menu.



6. **If:** Define the condition as follows:
- a) **A:** *Change_time* (**Current**) | **Does not equal** | *Change_time* (**baseline**)
 - b) **B:** *Change_time* (**Current**) | **Does not contain** | d
 - c) **C:** *Change_time* (**Current**) | **Does not contain** | w
7. Boolean Expression: A and (B or C)

Diagnosis: **D** HSRP Diagnosis + New Diagnosis

Name: HSRP Diagnosis Anchor: HSRP_failover.\$inter...
Check HSRP failover status of device

☒ Loop Table Rows HSRP_failover Table Key: virtual_ip

If 6

A US-BO... Current Baseline
change_time Does not e... change_time

B US-BO... Current
change_time Does not c... d

C US-BO... Current
change_time Does not c... w

D Select Variable

Boolean Expression: A and (B or C) 7

8. **Then:** With the diagnosis logic established, let's display an **Error** message if the result of the logic is TRUE.
 - a) Click the drop-down under **Diagnosis Message** and click the **RED** square.
 - b) Click into the Diagnosis Message text box (*\$intf is down...*), then type: **`$this_device HSRP $group_name has failover, last failover is at $change_time`**.
 - c) Check the boxes next to **Set Status Code for Device** and **Set Status Code for Intent**.
9. **Else:** Click "+ Else" to display a success message if the diagnosis logic is successful.
 - a) Click the drop-down under **Diagnosis Message** and click the **GREEN** square.
 - b) Click into the Diagnosis Message text box (*\$intf is down...*), then type: **`$this_device HSRP $group_name is stable`**.
 - c) Check the selection boxes: **Set Status Code for Device** and **Set Status Code for Intent**.

The screenshot displays the configuration interface for a diagnosis named "HSRP Diagnosis". The interface is divided into two main sections: "Then" (labeled with a red circle 8) and "Else" (labeled with a red circle 9).

Then Section:

- Diagnosis Message:** A red square is selected in the drop-down menu. The text box contains the message: `$this_device HSRP $group_name has failover, last failover is at $change_time`.
- Set Status Code for Device:** Checked. The status code is set to "Error". The message text is: `$this_device HSRP $group_name has failover, last failover is at $change_time`.
- Set Status Code for Intent:** Checked. The status code is set to "Error". The message text is: `$this_device HSRP $group_name has failover, last failover is at $change_time`.

Else Section:

- Diagnosis Message:** A green square is selected in the drop-down menu. The text box contains the message: `$this_device HSRP $group_name is stable`.
- Set Status Code for Device:** Checked. The status code is set to "Success". The message text is: `$this_device HSRP $group_name is stable`.
- Set Status Code for Intent:** Checked. The status code is set to "Success". The message text is: `$this_device HSRP $group_name is stable`.

Buttons for "+ New Diagnosis", "+ Else", and "Delete" are visible.

10. Click **Create**.

The image consists of two screenshots of the NetBrain interface. The top screenshot shows the 'Define Logic' pane with a red circle and the number '10' highlighting the 'create' button. A red dashed arrow points from this button to the 'Run Intent' pane in the bottom screenshot. The bottom screenshot shows the 'Run Intent' pane, which is highlighted with a yellow dashed border. A yellow speech bubble points to the 'Run Intent' pane with the text: 'Run Intent pane will appear with diagnosis decoded to seed device'. The 'Run Intent' pane displays the command 'show standby' for device 'US-BOS-SW1' and shows the output of the command, including 'Vlan100 - Group 1 (version 2)', 'State is Standby', '1 state change, last state change 3...', 'Virtual IP address is 10.8.1.1', 'Standby router is local', and 'Group name is "hsrp-Vl100-1" (default)'.

Overall Health View Network Stencils Map

Define Logic <<

Last Created: 05:12:56 PM

10 create

US-BOS-SW1 CLI show standby P1 1 Diagnosis

Diagnosis: HSRP Diagnosis + New Diagnosis

Overall Health View Network Stencils Map | 100%

+ New Incident Reset Help

Define Logic << Run Intent Save to <<

Last Created: 05:12:56 PM Recreate Not Executed Run with Live Data

US-BOS-SW1 CLI show standby P1 1 Diagnosis View Intent Diagnosis Tree Show in Map

Diagnosis: HSRP Diagnosis + New Diagnosis

US-BOS-SW1

show standby 1 Diagnosis

Vlan100 - Group 1 (version 2)

State is Standby

1 state change, last state change 3...

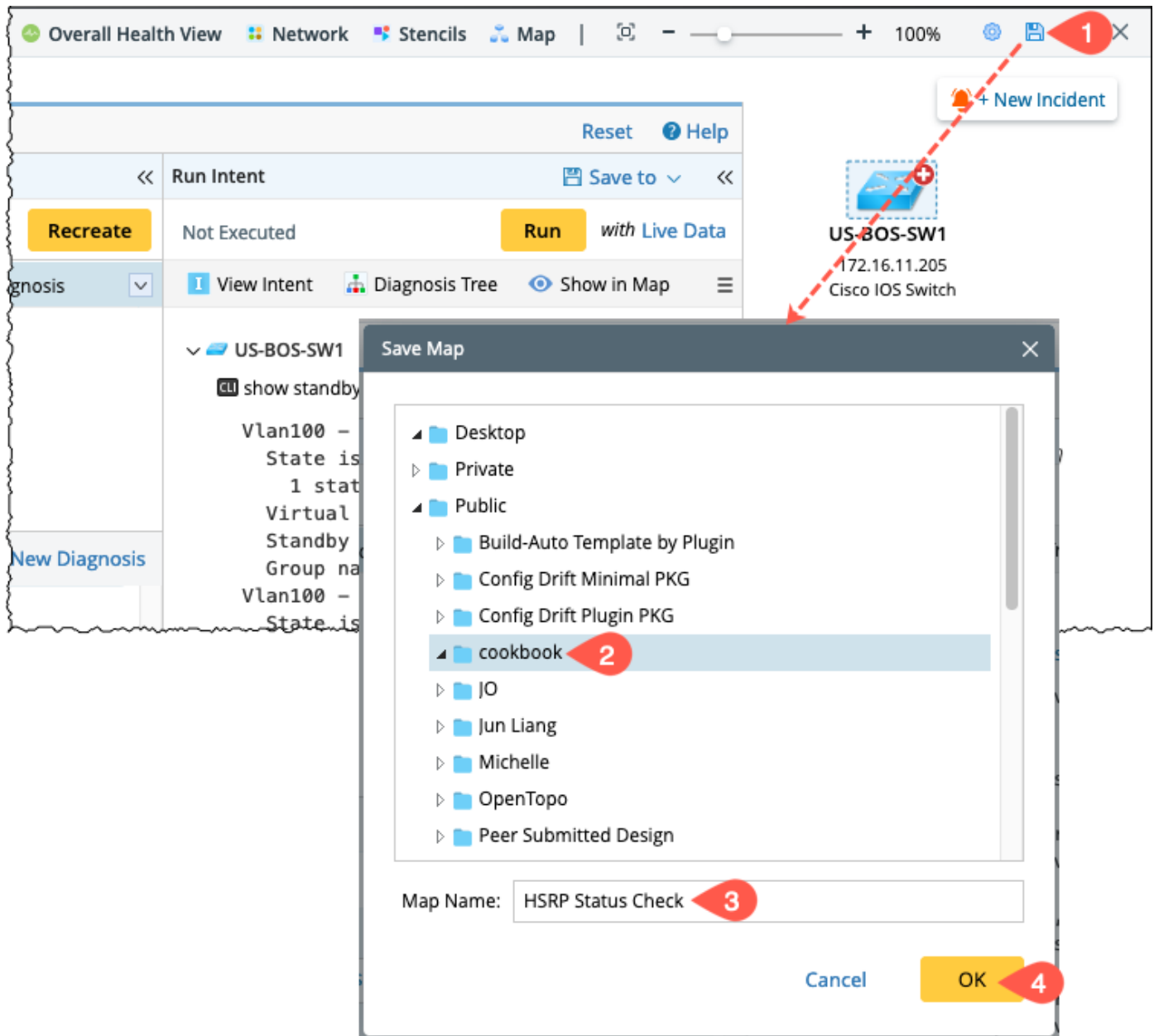
Virtual IP address is 10.8.1.1

Standby router is local

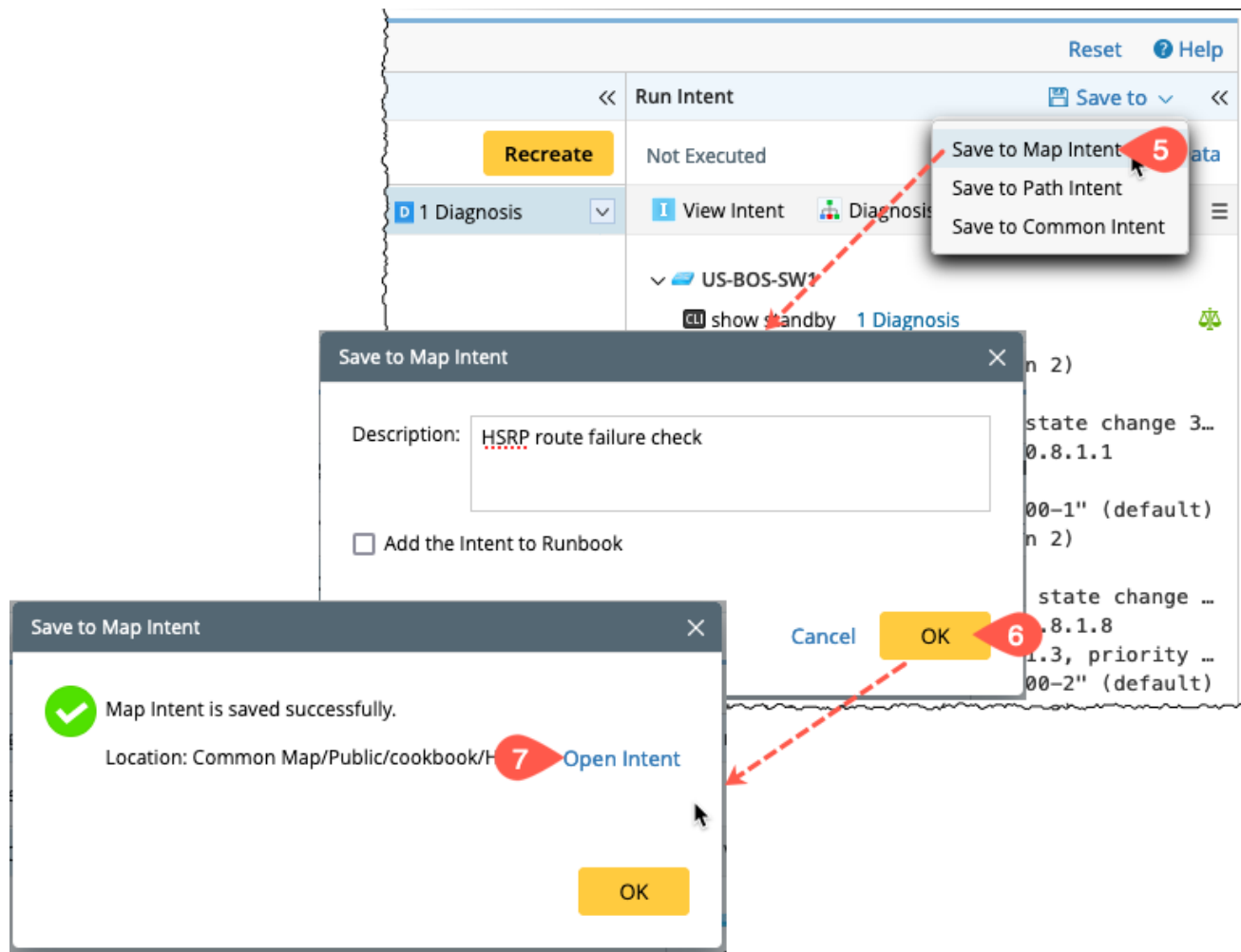
Group name is "hsrp-Vl100-1" (default)

8.1.2.4 Save and Execute Map Intent

1. Save the map by clicking on the **Save** button located in the top right corner.
2. In the **Save Map** dialog, navigate to **Public > cookbook**, then click on the directory.
3. Name the map **HSRP Status Check**.
4. Click **OK**.

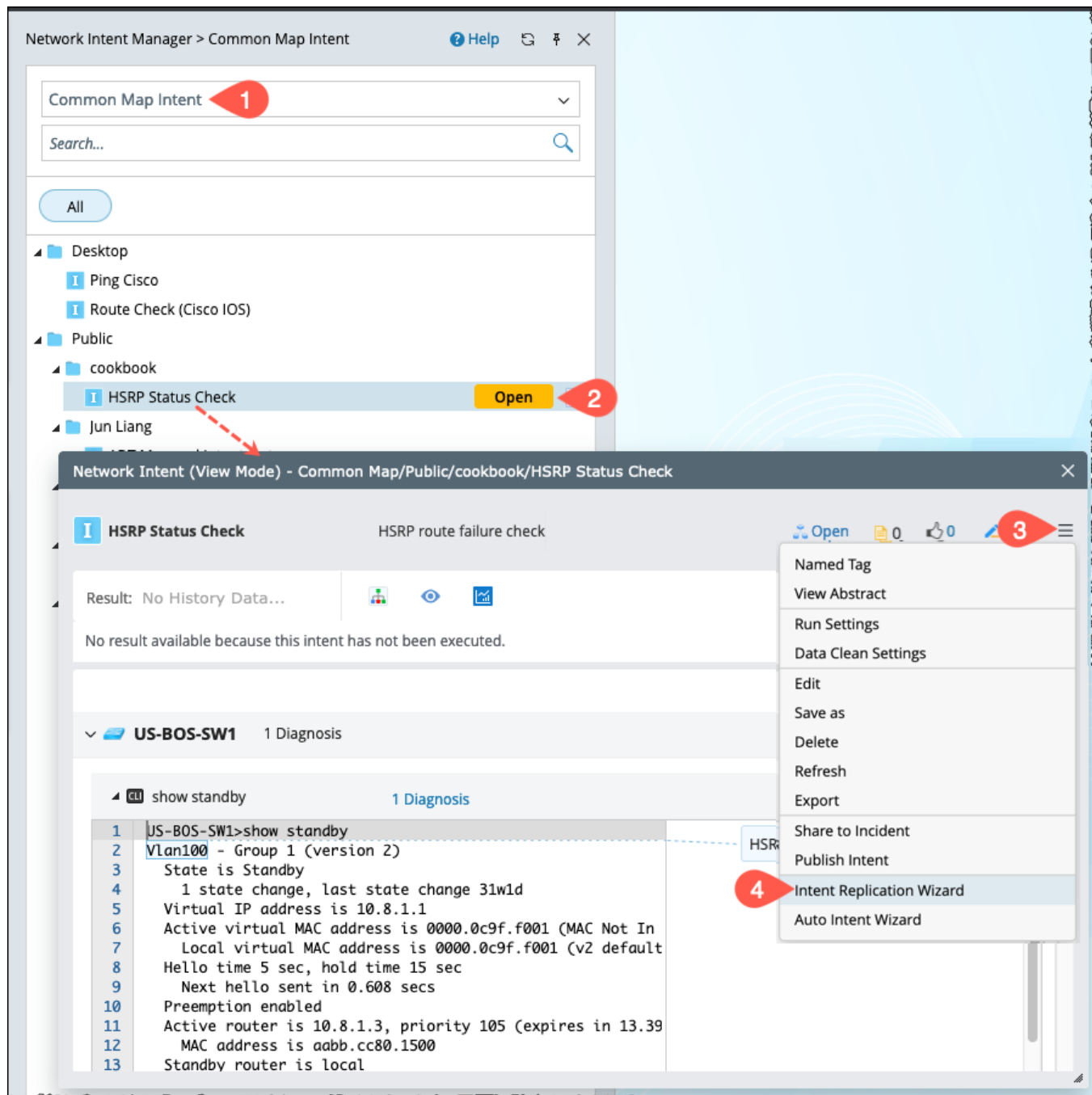


5. In the Run Intent pane, click **Save to** > **Save to Map Intent**.
6. Click **OK** in the dialogue.
7. Click **Open Intent**.

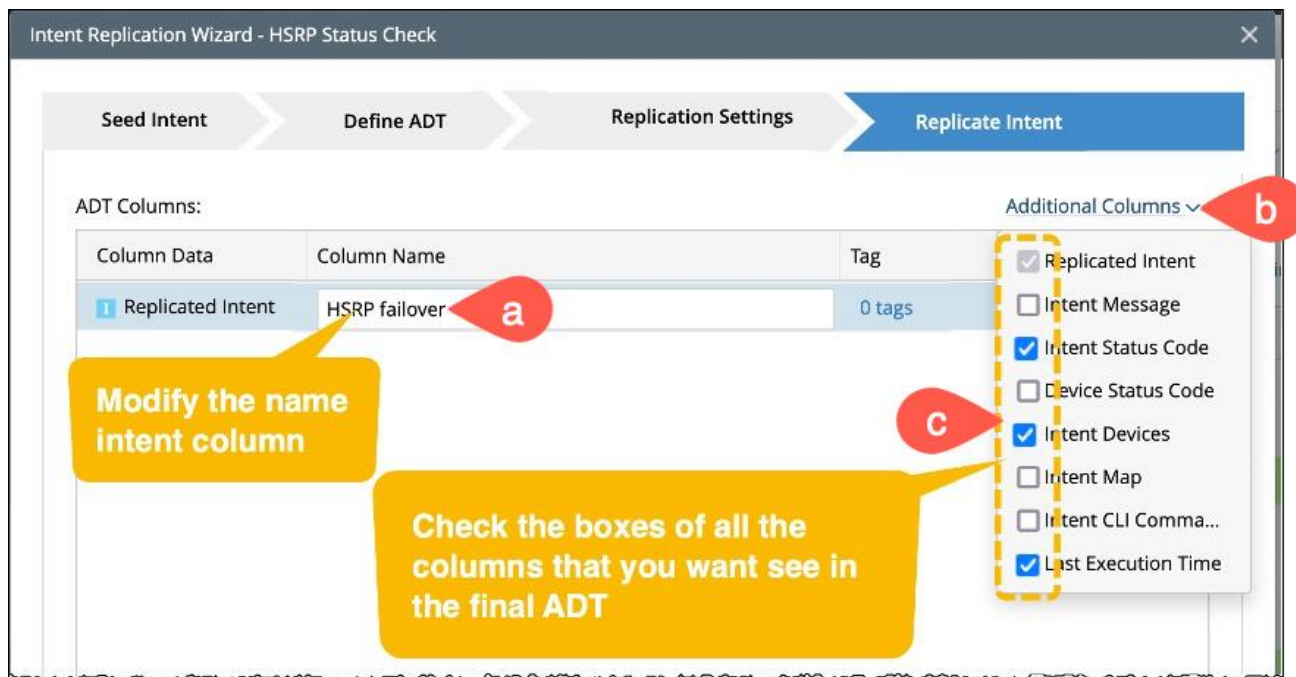


8.1.3 Replicate the intents for all HSRP devices using ADT

In this section, you will use the **Intent Replication Wizard** to replicate the intent to all devices in the HSRP device group. As a first step, open the intent from the intent manager and the Intent replication wizard as shown:



1. In the first step (**Seed Intent**), Validate the default seed intent and go to the next section.
2. In the second step (**Define ADT**), provide the basic input to create a new ADT, such as name, ADT location and target devices.
3. In the third step, define the **intent qualification** with device group **HSRP Devices** to include devices for the seed intent to replicate on.
4. In the fourth step, **Replicate Intent**, modify the name of the **Replicated Intent** In the **ADT Columns** section and add more columns that you want in the final ADT.



5. Click **Save and Replicate** to save all the settings and create ADT. An option **Open Output ADT** will appear. Click it to open the table in the **ADT Manager**.

Seed Intent





Define ADT

Replication Settings

Replicate Intent


ADT Columns:

Additional Columns ▾

Column Data	Column Name	Tag
 Replicated Intent	HSRP failover	0 tags
 Intent Message	Intent Message	
 Intent Devices	Intent Devices	
 Last Execution Time	Last Execution Time	

5

Save and Replicate

 Selection Mode: Device-based Replication, ADT: HSRP Route Failure, 0 Macro Variables.

Previous

Finish

Seed Intent





Define ADT

Replication Settings

Replicate Intent


ADT Columns:

Additional Columns ▾

Column Data	Column Name	Tag
 Replicated Intent	HSRP failover	0 tags
 Intent Message	Intent Message	
 Intent Devices	Intent Devices	
 Last Execution Time	Last Execution Time	

Click to open
the table in
ADT manager.

Save and Replicate

Replication Request submitted at: 07/26/2024 09:17 PM | [Open Output ADT](#) Selection Mode: Device-based Replication, ADT: HSRP Route Failure, 0 Macro Variables.

Previous

Finish

Review the new Intent columns that are added to the table and results.

Automation Data Table Manager

Search...

Shared Tables (1201)

My Tables (5)

Ping (1)

BGP Config Change Diagnosis

Day 1 Lab

HSRP Route Failure

NTP

HSRP Route Failure

Table Builder

Last Updated at: 07/30/2024 12:57 PM

Rebuild Table

Add Data Manually

Description: Type description here...


Items: 19 Rows 4 Columns

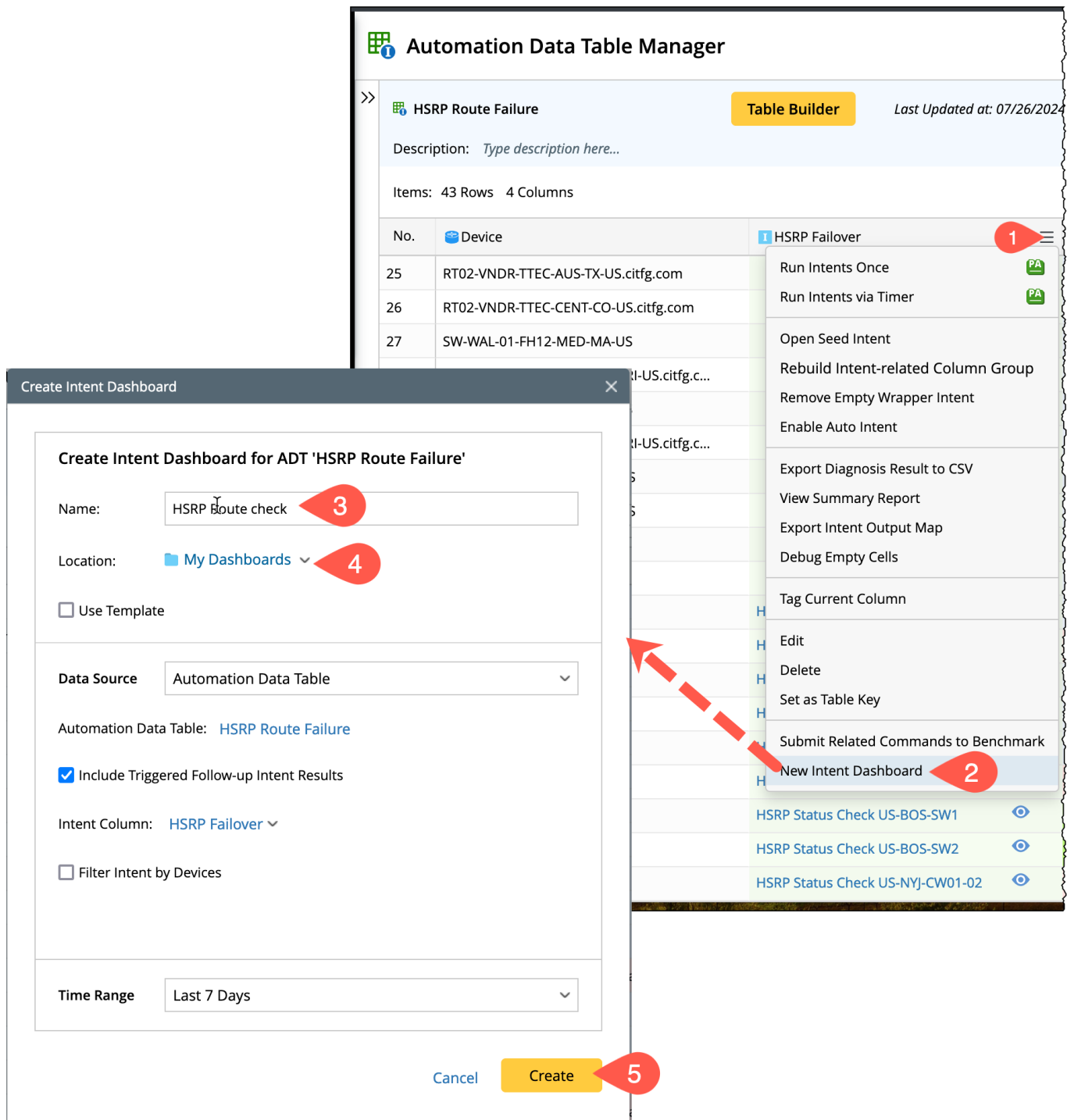
Search...

Advanced Filter: Undefined

No.	Device	HSRP failover	Intent Status Code	Last Execution Time
1	BJ-L2-coreB	HSRP Status Check BJ-L2-coreB	BJ-L2-coreB HSRP hsrp-VI10-100 is stable	07/29/2024 03:55:07 PM
2	BJ_L2_Core_3	HSRP Status Check BJ_L2_Core_3	BJ_L2_Core_3 HSRP hsrp-Fa1/0/9-10 is stable	07/29/2024 03:55:07 PM
3	BJ_L2_Core_4	HSRP Status Check BJ_L2_Core_4	BJ_L2_Core_4 HSRP hsrp-Fa2/0/3-0 is stable	07/29/2024 03:55:05 PM
4	BJ_core_3550	HSRP Status Check BJ_core_3550	BJ_core_3550 HSRP None is stable	07/29/2024 03:55:07 PM
5	Bur-isp-gw1	HSRP Status Check Bur-isp-gw1	Bur-isp-gw1 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
6	IPv6Lab-SW8	HSRP Status Check IPv6Lab-SW8	IPv6Lab-SW8 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:05 PM
7	IPv6Lab-SW9	HSRP Status Check IPv6Lab-SW9	IPv6Lab-SW9 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:07 PM
8	PE-3600X-01	HSRP Status Check PE-3600X-01	PE-3600X-01 HSRP hsrp-Gi0/24-2 is stable	07/29/2024 03:55:07 PM
9	PE-3600X-02	HSRP Status Check PE-3600X-02	PE-3600X-02 HSRP hsrp-Gi0/13-2 is stable	07/29/2024 03:55:07 PM
10	PE-ASR1K-01	HSRP Status Check PE-ASR1K-01	PE-ASR1K-01 HSRP hsrp-Te0/0/0-15 is stable	07/29/2024 03:55:05 PM
11	PE-ASR1K-02	HSRP Status Check PE-ASR1K-02	PE-ASR1K-02 HSRP hsrp-Gi0/0/5-10 is stable	07/29/2024 03:55:05 PM
12	Sjc-Dist-3750-02	HSRP Status Check Sjc-Dist-3750-02	Sjc-Dist-3750-02 HSRP hsrp-VI30-0 is stable	07/29/2024 03:55:07 PM
13	US-BOS-SW1	HSRP Status Check US-BOS-SW1	US-BOS-SW1 HSRP hsrp-VI100-1 is stable	07/29/2024 03:55:07 PM
14	US-BOS-SW2	HSRP Status Check US-BOS-SW2	US-BOS-SW2 HSRP hsrp-VI100-1 is stable	07/29/2024 03:55:05 PM
15	bjta002237-SW2	HSRP Status Check bjta002237-SW2	bjta002237-SW2 HSRP hsrp-VI481-1 is stable	07/29/2024 03:55:07 PM
16	bjta002238-SW3	HSRP Status Check bjta002238-SW3	bjta002238-SW3 HSRP hsrp-VI481-1 is stable	07/29/2024 03:55:05 PM
17	bjta002444-SW13	HSRP Status Check bjta002444-SW13	bjta002444-SW13 HSRP hsrp-Et1/1-1 is stable	07/29/2024 03:55:05 PM
18	bur-isp-gw2	HSRP Status Check bur-isp-gw2	bur-isp-gw2 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
19	qapp-c3560-2	HSRP Status Check qapp-c3560-2	qapp-c3560-2 HSRP hsrp-Gi0/15-0 is stable	07/29/2024 03:55:07 PM

8.1.4 Create the Dashboard

Let us create the intent dashboard from the ADT. Open the **New Intent Dashboard** from the intent HSRP Failover column  menu.



The screenshot shows the **Automation Data Table Manager** interface. At the top, it displays the table name **HSRP Route Failure**, a **Table Builder** button, and the last update time **Last Updated at: 07/26/2024**. Below this is a description field and a summary of **Items: 43 Rows 4 Columns**. A table with columns **No.**, **Device**, and **HSRP Failover** is shown. The **HSRP Failover** column has a menu icon (three horizontal lines) with a red circle '1' next to it. A red dashed arrow points from this menu icon to the **Create Intent Dashboard** dialog box.

The **Create Intent Dashboard** dialog box is titled **Create Intent Dashboard for ADT 'HSRP Route Failure'**. It contains the following fields and options:

- Name:** (Red circle '3' next to the text)
- Location:** My Dashboards (Red circle '4' next to the dropdown)
- ☐ Use Template
- Data Source:** Automation Data Table (dropdown)
- Automation Data Table:** HSRP Route Failure
- ☒ Include Triggered Follow-up Intent Results
- Intent Column:** HSRP Failover (dropdown)
- ☐ Filter Intent by Devices
- Time Range:** Last 7 Days (dropdown)

At the bottom of the dialog are **Cancel** and **Create** buttons. The **Create** button has a red circle '5' next to it.

The **Automation Data Table Manager** table shows the following data:

No.	Device	HSRP Failover
25	RT02-VNDR-TTEC-AUS-TX-US.citfg.com	HSRP Failover
26	RT02-VNDR-TTEC-CENT-CO-US.citfg.com	HSRP Failover
27	SW-WAL-01-FH12-MED-MA-US	HSRP Failover

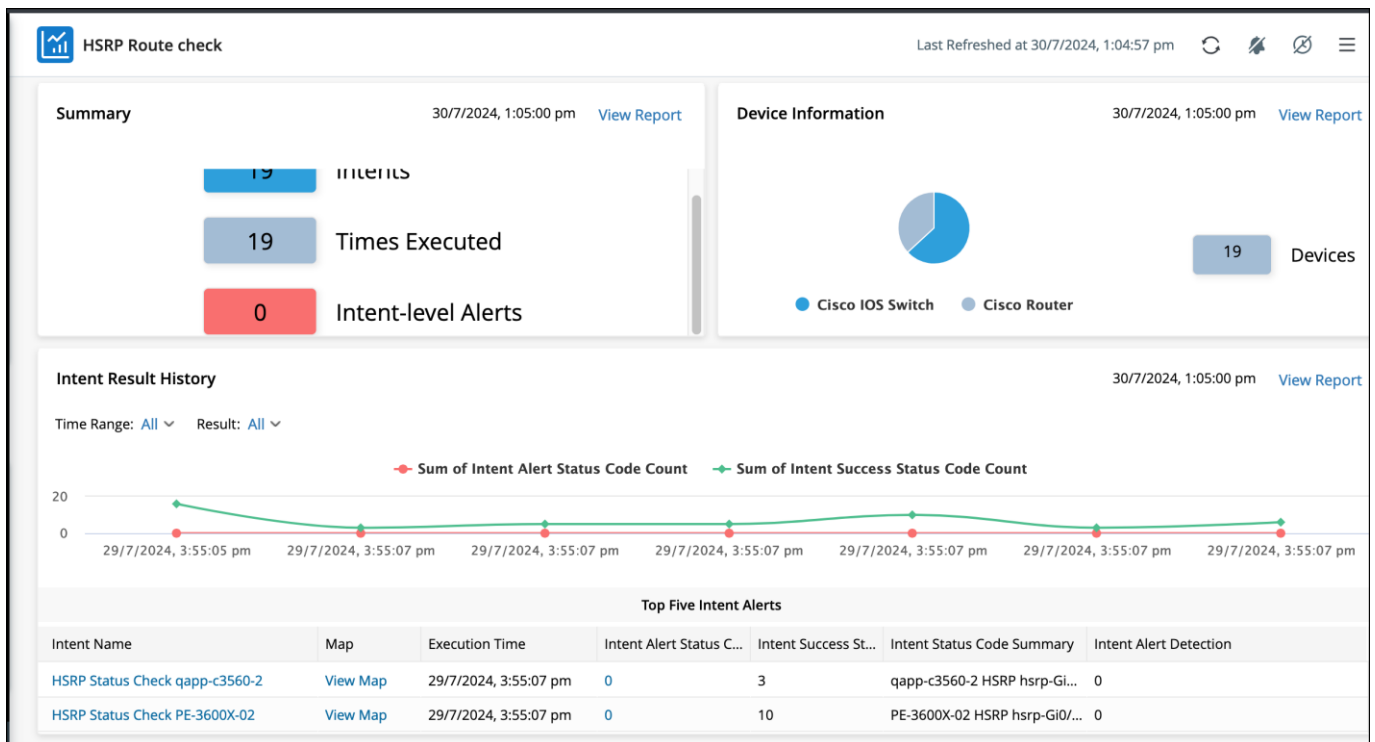
The **HSRP Failover** column menu (Red circle '1') includes the following options:

- Run Intents Once
- Run Intents via Timer
- Open Seed Intent
- Rebuild Intent-related Column Group
- Remove Empty Wrapper Intent
- Enable Auto Intent
- Export Diagnosis Result to CSV
- View Summary Report
- Export Intent Output Map
- Debug Empty Cells
- Tag Current Column
- Edit
- Delete
- Set as Table Key
- Submit Related Commands to Benchmark
- New Intent Dashboard** (Red circle '2' next to the option)

Below the menu, there are three rows of data in the **HSRP Failover** column:

- HSRP Status Check US-BOS-SW1
- HSRP Status Check US-BOS-SW2
- HSRP Status Check US-NYJ-CW01-02

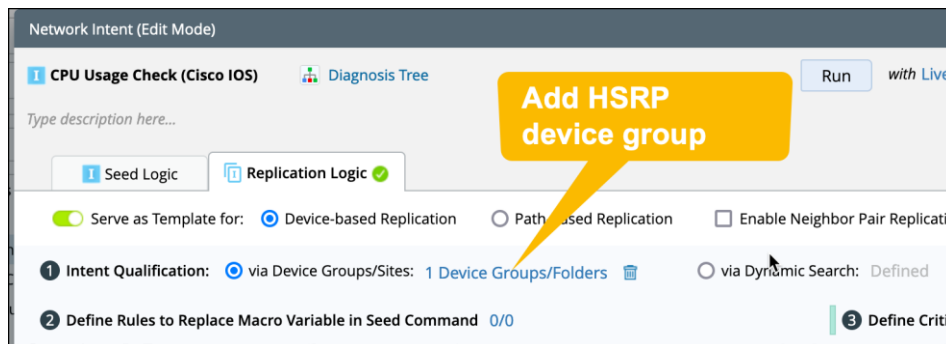
In the **Create Intent Dashboard** dialog, **Open Intent Dashboard** to view the results summary and result history.



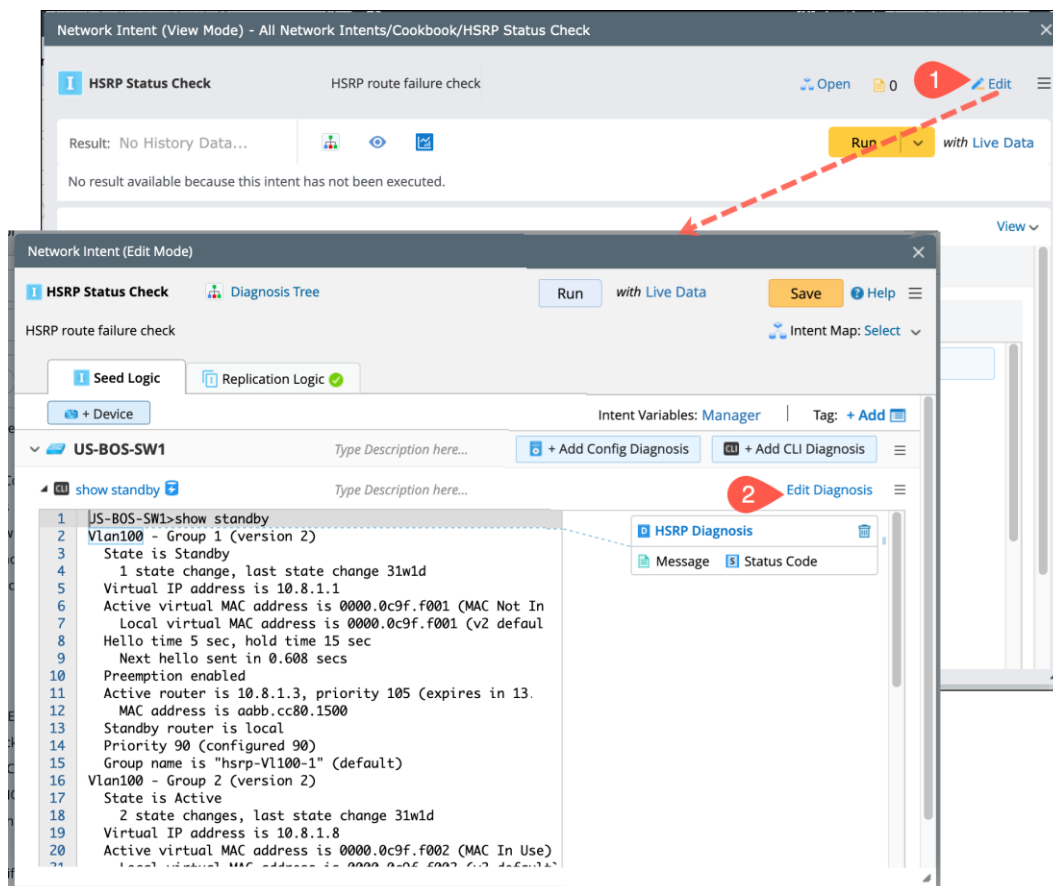
8.2 Follow-up to Check the Performance Degradate After Failover

In this section, we shall add two intents as follow-up for the HSRP device if it has failover in the last 24 hours. We shall use the intents created in [Section 3](#) (*CPU usage Check* and *Interface Status Check*) as a follow-up:

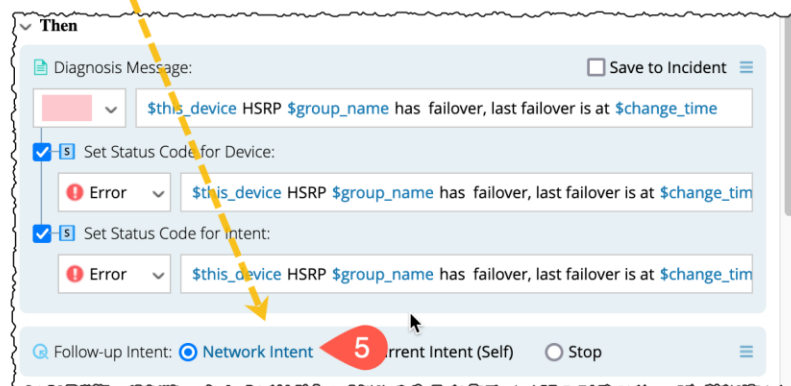
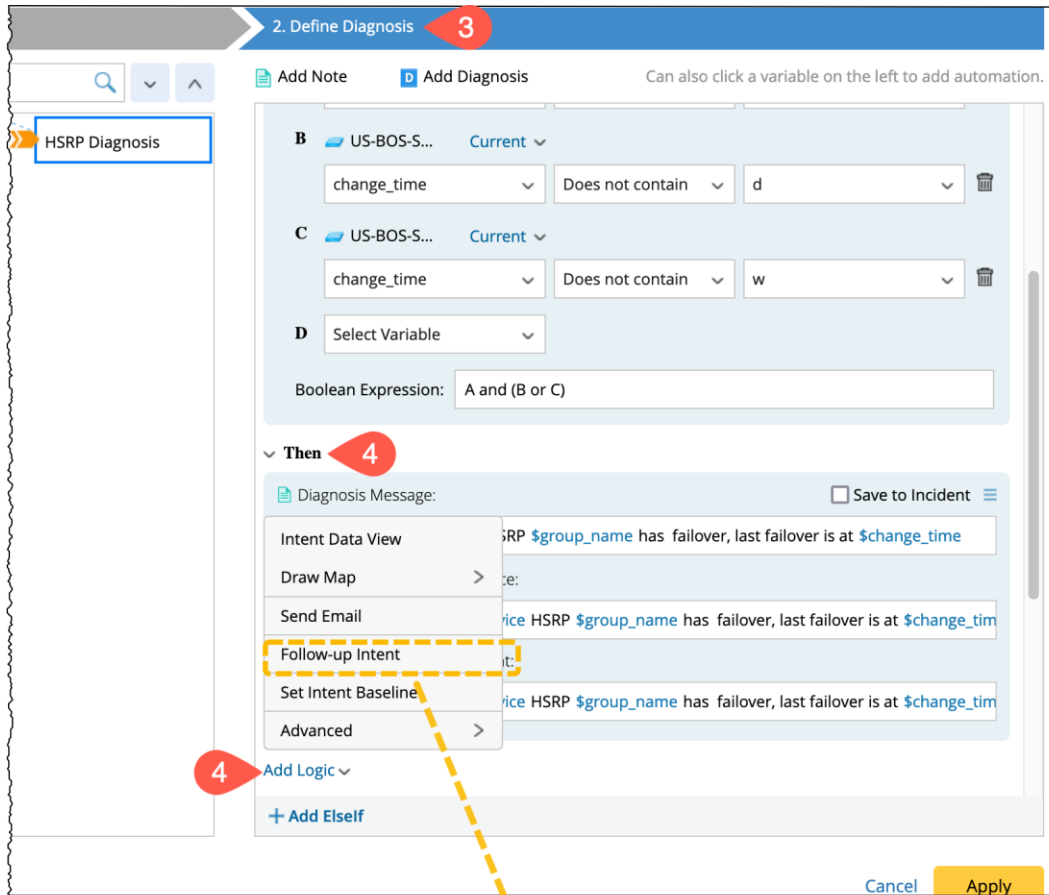
NOTE: Before proceeding further, go to the intents (*CPU usage Check* and *Interface Status Check*) **Edit Mode**>**Replication Logic** and define the **Intent Qualification** as HSRP device group.



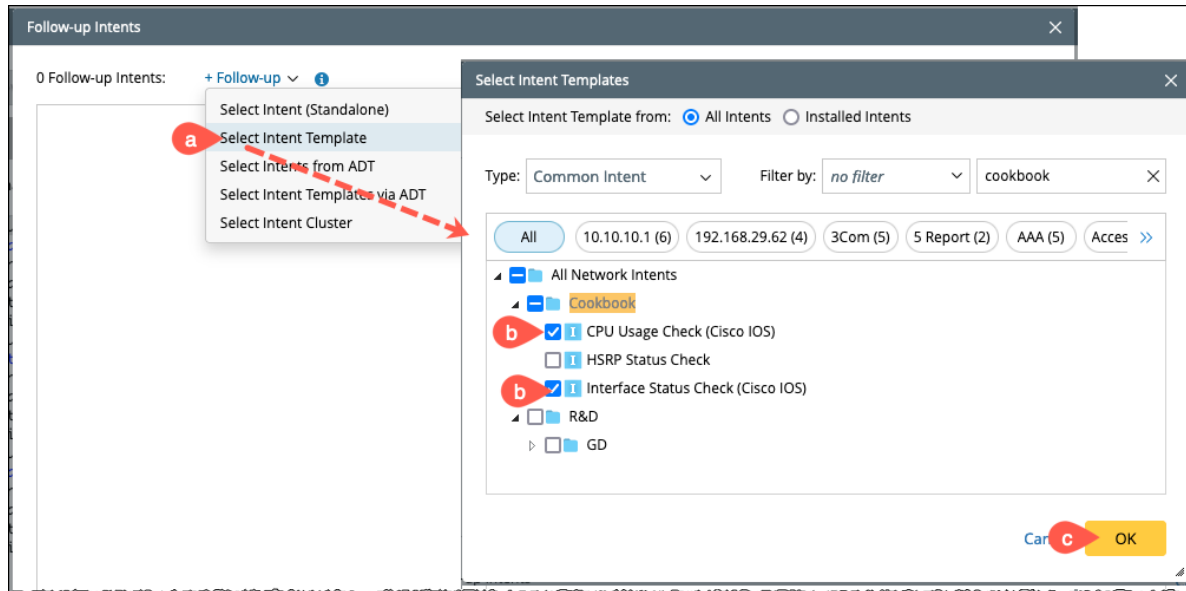
1. Open the seed intent **HSRP status check** in edit mode.
2. Click **Edit Diagnosis** to open the parser.



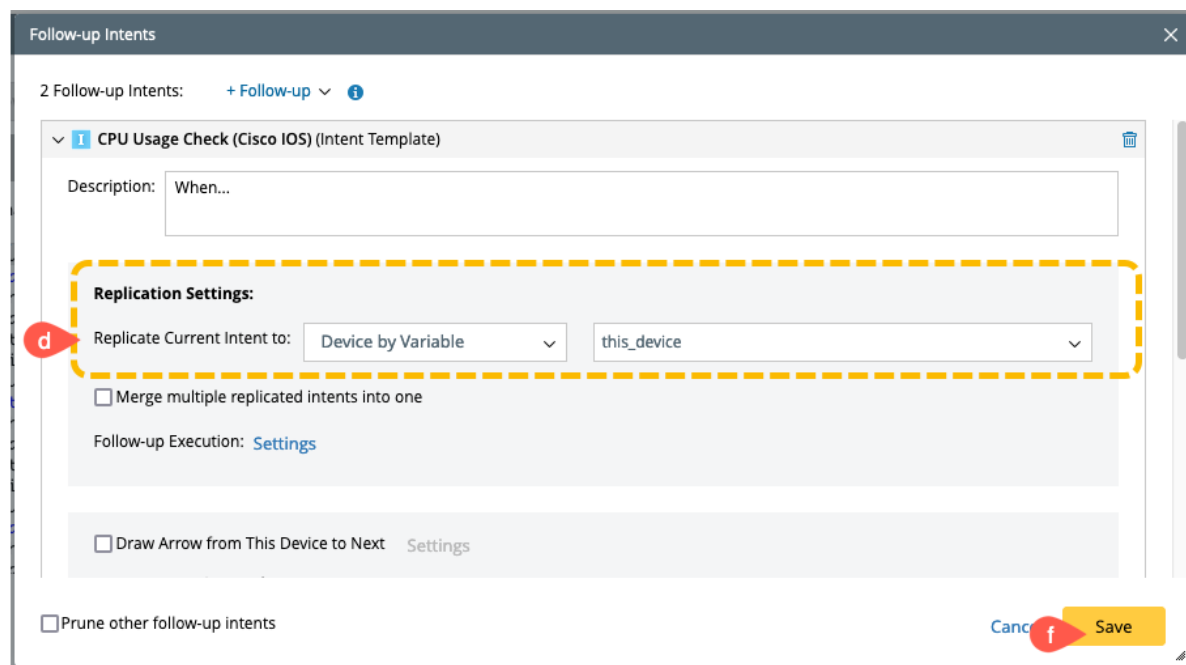
3. In the diagnosis window, go to the **Define Diagnosis** ribbon.
4. **Then:** Check the **CPU usage** and **Interface Status** by running the corresponding intents as the follow up intent. Remove the **Diagnosis Message** and add **Follow-up Intent** from the **Add logic** dropdown.



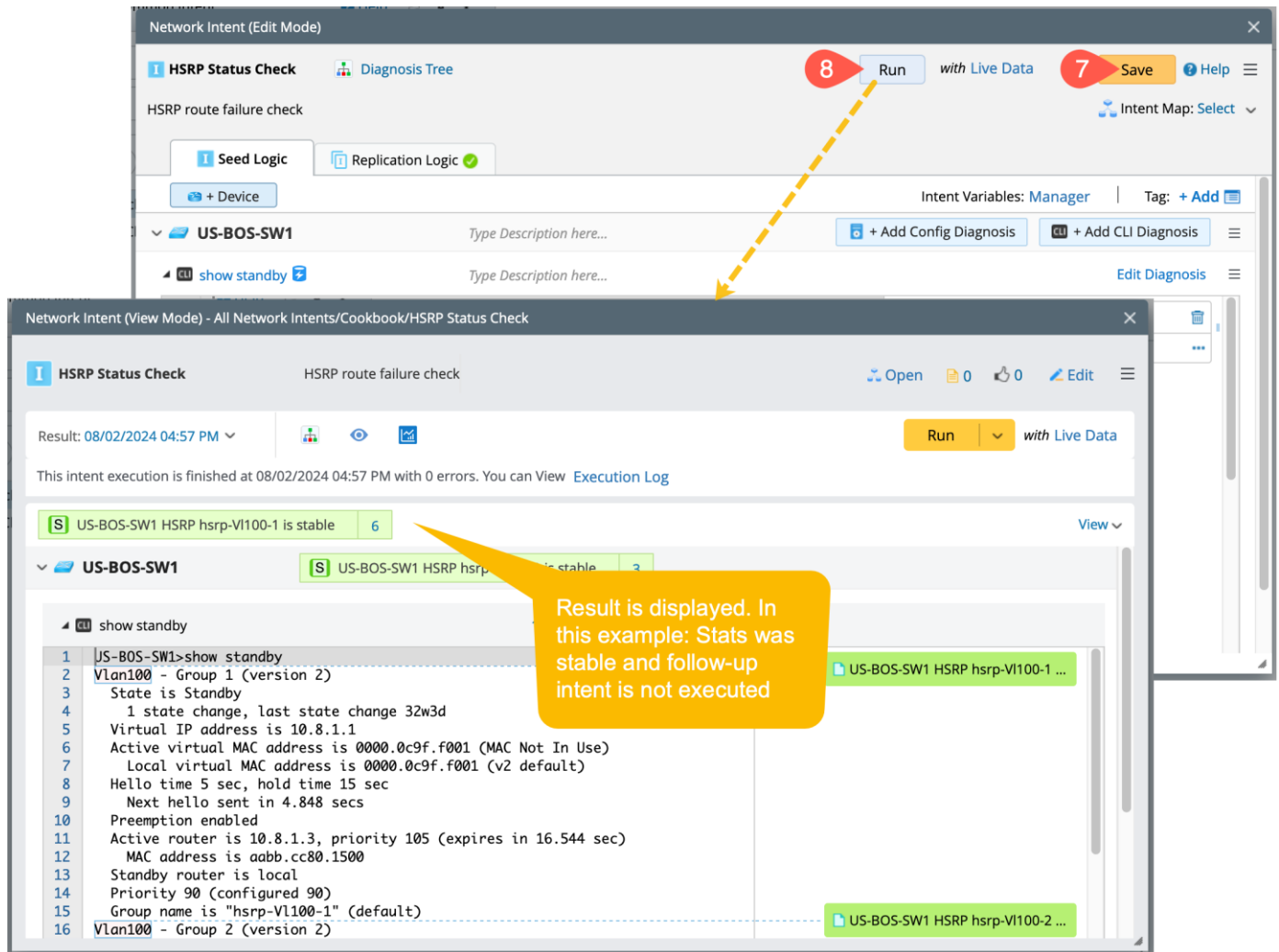
5. Click the **Network Intent** link to choose the follow-up intent in a new dialog **Follow-up Intents**.
 - a) In the Follow-up Intents window, choose **Select Intent Template**.
 - b) In the Select Intents dialog, navigate to the follow-up intents (**CPU usage** and **Interface Status**) and check the selection box.
 - c) Click **OK** to save the selection and close the window.



- d) Define Replicate Current Intent to: **Device by Variable** | **this_device**.
- e) Click **Save** to save and close the follow-up intents.




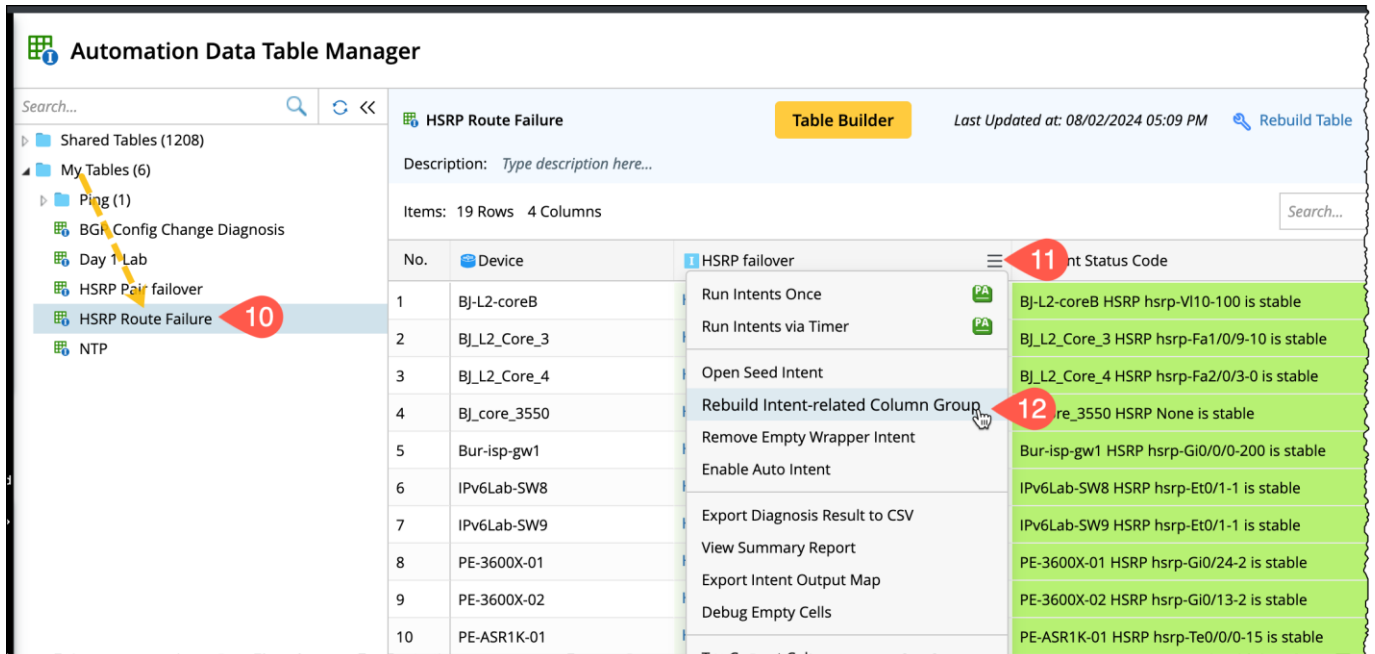
- Click **Apply** to save and close the diagnosis window.
- In the Network Intent (Edit Mode) dialog, click **Save**.
- Click **Run** to execute the intent.





- Close all the network intent dialogues and go to **ADT Manager** to update the replicated intents in ADT.

8.2.1 Update replicated intents in ADT

1. Go to Automation Data Table Manager > My Tables> HSRP Route Failure.
2. Open the  menu from the **Intent** Column.
3. Select **Rebuilt Intent-related Column Group**.



Automation Data Table Manager

Search...   <<

Shared Tables (1208)

My Tables (6)

Ping (1)


BGP Config Change Diagnosis

Day 1 Lab

HSRP Pair failover

HSRP Route Failure 10

NTP

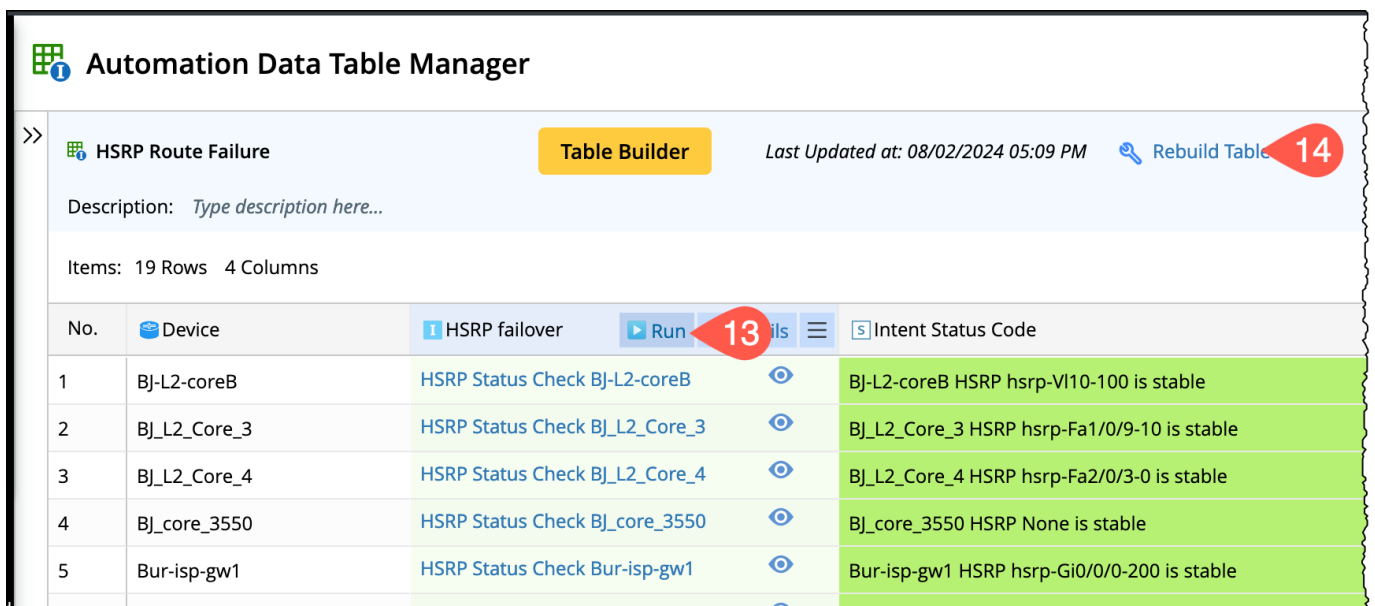
HSRP Route Failure Table Builder Last Updated at: 08/02/2024 05:09 PM  Rebuild Table

Description: Type description here...


Items: 19 Rows 4 Columns

No.	Device	HSRP failover	Intent Status Code
1	BJ-L2-coreB	Run Intents Once	BJ-L2-coreB HSRP hsrp-VI10-100 is stable
2	BJ_L2_Core_3	Run Intents via Timer	BJ_L2_Core_3 HSRP hsrp-Fa1/0/9-10 is stable
3	BJ_L2_Core_4	Open Seed Intent	BJ_L2_Core_4 HSRP hsrp-Fa2/0/3-0 is stable
4	BJ_core_3550	Rebuild Intent-related Column Group 12	BJ_core_3550 HSRP None is stable
5	Bur-isp-gw1	Remove Empty Wrapper Intent	Bur-isp-gw1 HSRP hsrp-Gi0/0/0-200 is stable
6	IPv6Lab-SW8	Enable Auto Intent	IPv6Lab-SW8 HSRP hsrp-Et0/1-1 is stable
7	IPv6Lab-SW9	Export Diagnosis Result to CSV	IPv6Lab-SW9 HSRP hsrp-Et0/1-1 is stable
8	PE-3600X-01	View Summary Report	PE-3600X-01 HSRP hsrp-Gi0/24-2 is stable
9	PE-3600X-02	Export Intent Output Map	PE-3600X-02 HSRP hsrp-Gi0/13-2 is stable
10	PE-ASR1K-01	Debug Empty Cells	PE-ASR1K-01 HSRP hsrp-Te0/0/0-15 is stable

4. In the Intents column, hover the mouse on the column header and click **Run**.
5. Click **Rebuild** to refresh the results in the **Intent Status code**.








Automation Data Table Manager

>> HSRP Route Failure Table Builder Last Updated at: 08/02/2024 05:09 PM  Rebuild Table 14

Description: Type description here...

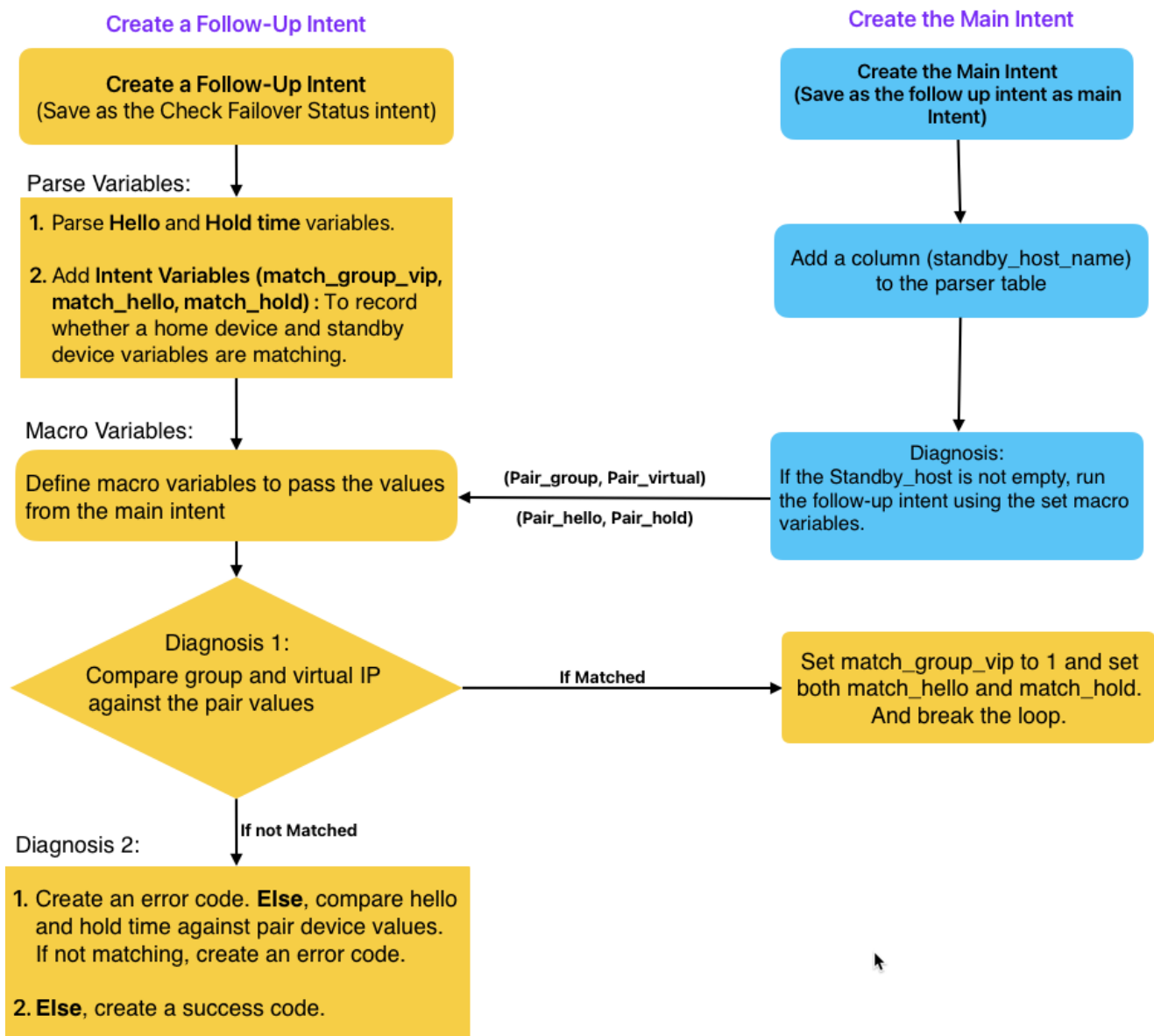
Items: 19 Rows 4 Columns

No.	Device	HSRP failover	Run 13	Intent Status Code
1	BJ-L2-coreB	HSRP Status Check BJ-L2-coreB		BJ-L2-coreB HSRP hsrp-VI10-100 is stable
2	BJ_L2_Core_3	HSRP Status Check BJ_L2_Core_3		BJ_L2_Core_3 HSRP hsrp-Fa1/0/9-10 is stable
3	BJ_L2_Core_4	HSRP Status Check BJ_L2_Core_4		BJ_L2_Core_4 HSRP hsrp-Fa2/0/3-0 is stable
4	BJ_core_3550	HSRP Status Check BJ_core_3550		BJ_core_3550 HSRP None is stable
5	Bur-isp-gw1	HSRP Status Check Bur-isp-gw1		Bur-isp-gw1 HSRP hsrp-Gi0/0/0-200 is stable

8.3 Check Failover Consistency of HSRP Pair Devices

Let us create an intent to check whether the hello and hold timers are equal for the failover pair devices. You are going to create a parent intent to parse all HSRP pair devices and call a follow-up intent for each of these pair devices, similar to Section 8.1. However, since you are going to compare the hello/hold timers between pair devices, you need to pass the values of these timers from the parent to the follow-up intents, which can be done by defining the variables as Macro Variables in the follow-up intents and setting the Macro Variables in the follow-up intent replicated settings. The following diagram shows the data flow from the parent and follow-up intents:

Check Failover Consistency of HSRP Pair Devices

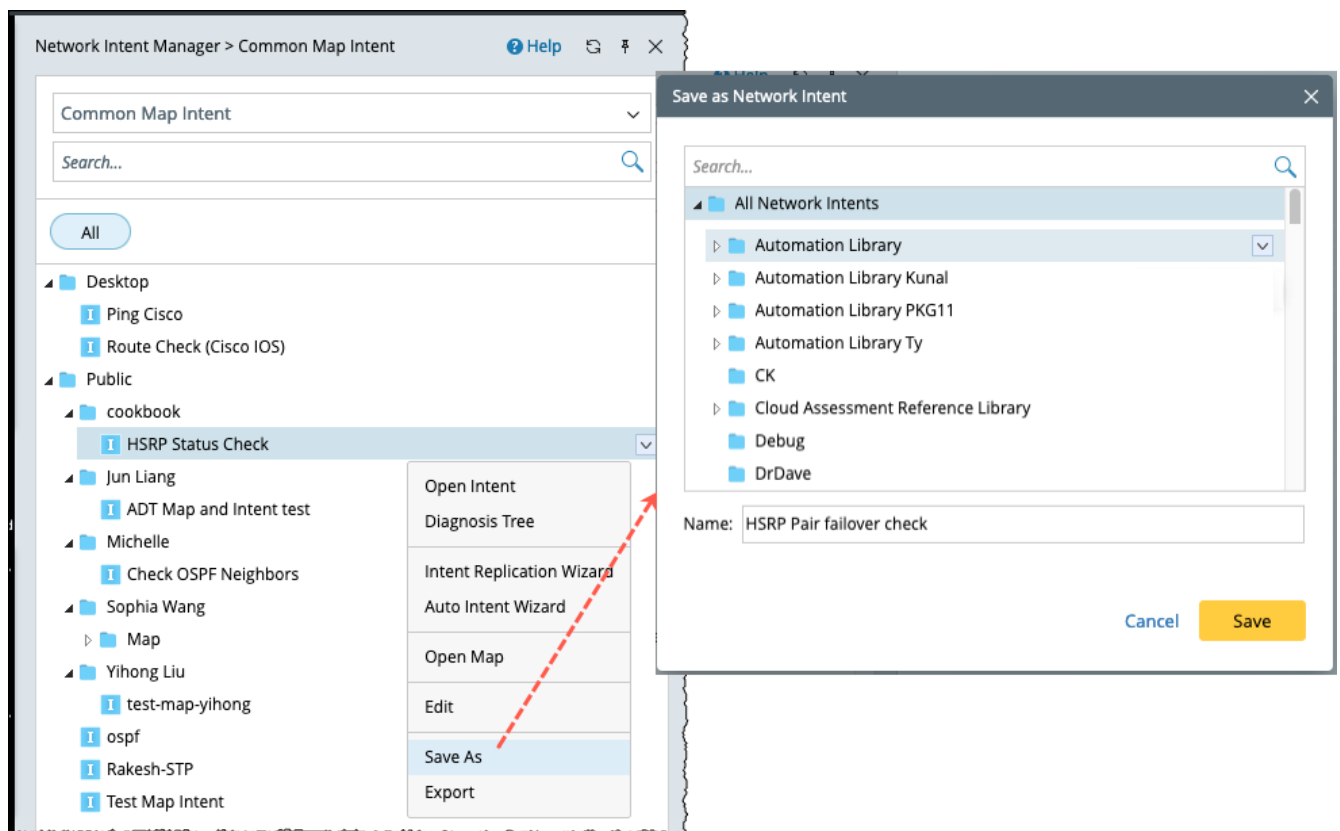


1. [Create a follow-up intent](#)
2. [Create a parent intent](#)
3. [Run the intent and view the diagnosis tree](#)
4. [Replicate the intent to all HSRP devices using ADT](#)

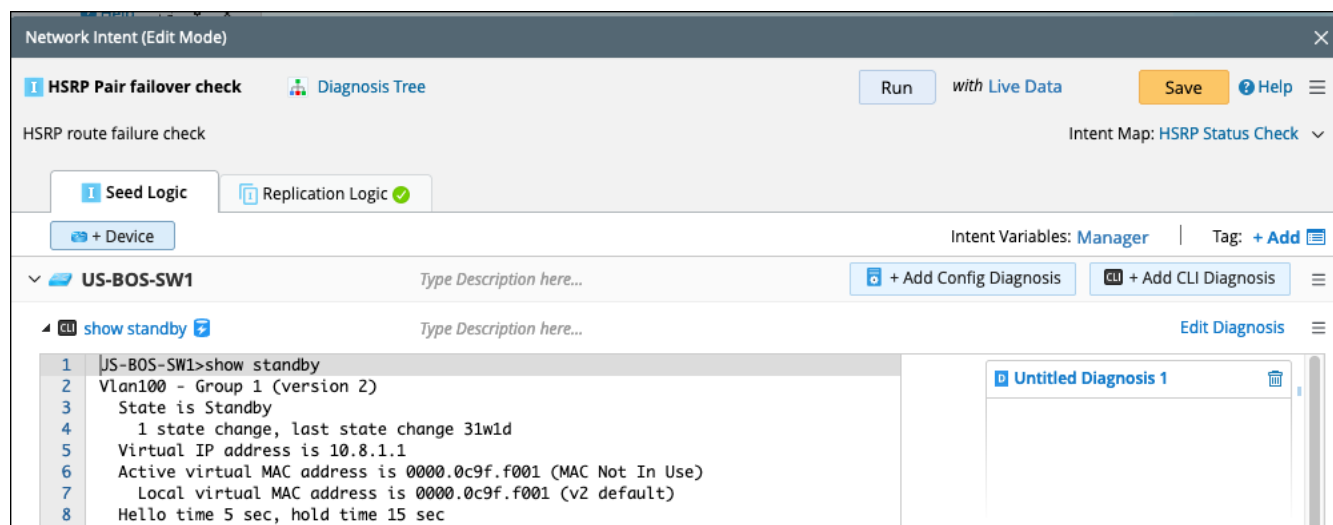
8.3.1 Create a follow-up intent

Create a follow-up intent using the intent created in Section 8.1 since we use the same CLI command and parser:

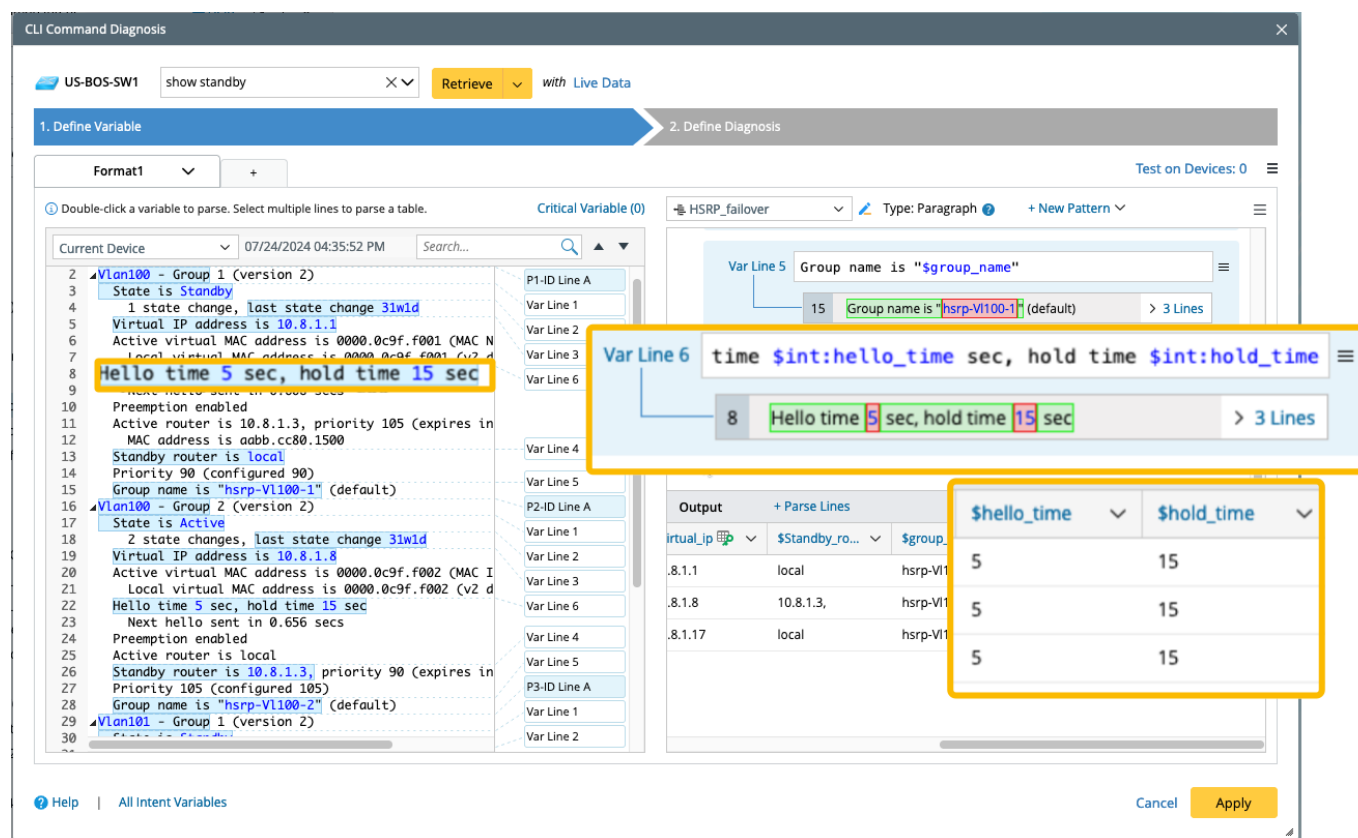
1. Save it as a new intent with the name **HSRP pair failover check_followup**.



- Open the new intent **HSRP pair failover check_followup** in edit mode and click **Edit Diagnosis** to open the parser.



- Parse the variables **hello** and **hold** time from retrieved CLI command data.



4. Add three intent variables:
 - a) **match_group_vip** (int, default value: 0),
 - b) **match_hello**, (int, -1)
 - c) **match_hold** (int, -1)


These variables will be used to record whether a group and virtual IP match the pair's group and virtual IP while looping through all groups, along with their corresponding **hello** and **hold** values.

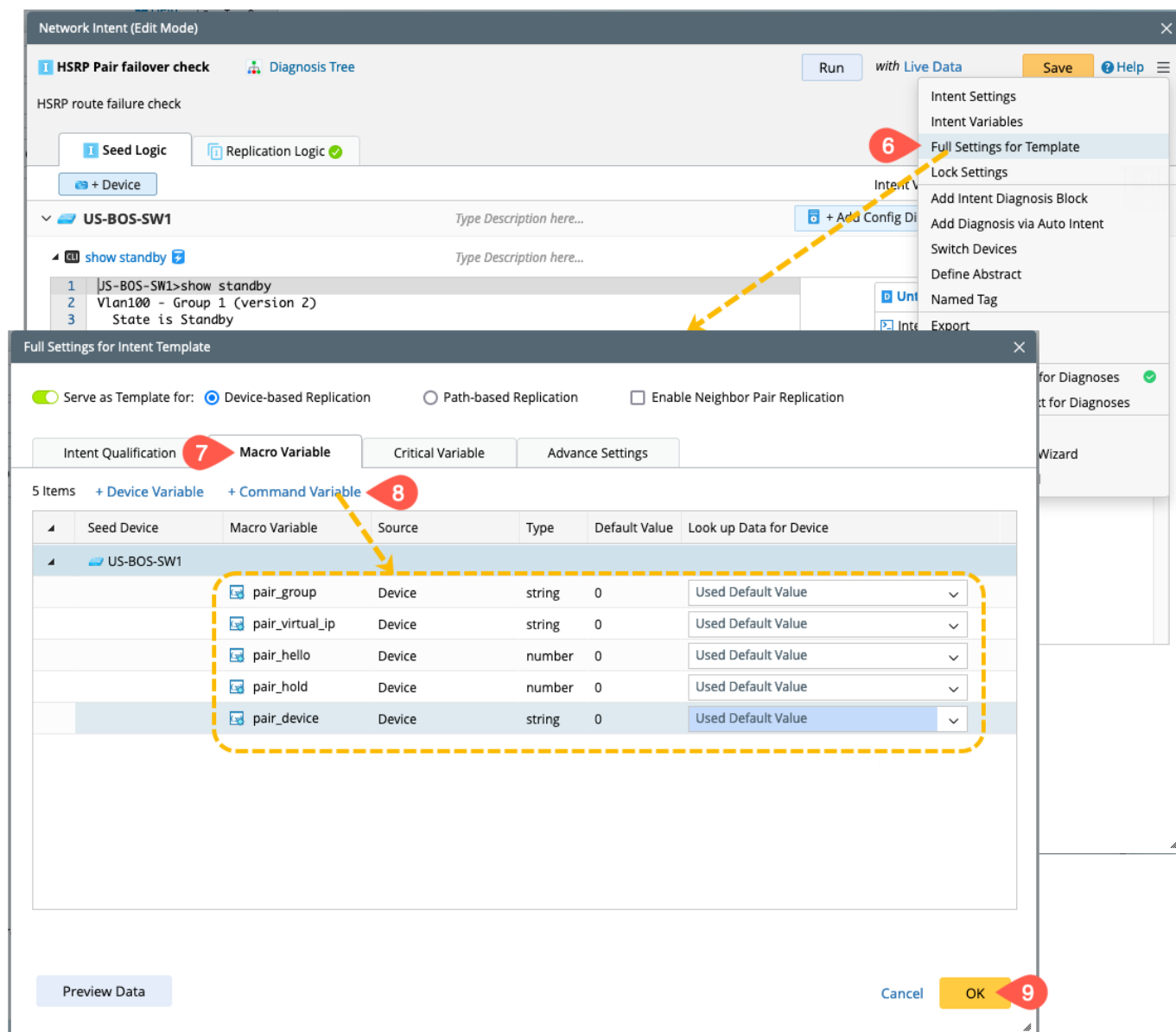
The screenshot shows the 'CLI Command Diagnosis' window with the command 'show standby' entered for device 'US-BOS-SW1'. The '1. Define Variable' tab is active, and the 'Intent Variables for Seed Logic' dialog is open. The dialog has three tabs: 'Intent Variable', 'Use Automation Data Table', and 'Task Variable'. The 'Intent Variable' tab is selected, showing a table of variables:

Intent Variable	Type	Initial Value
match_group_vip	number	0
match_hello	number	-1
match_hold	number	-1

Below the table, there are sections for 'US-BOS-SW1' showing 'show standby' output and 'Built-in Data'. A yellow arrow points from the 'Hello time 5 sec, hold time 15 sec' line in the background output to the 'match_hello' and 'match_hold' variables in the dialog.

5. Close the parser to go back to the **Network Intent (Edit Mode)** window.

6. From the menu , select **Full Settings for Template**.
7. Go to the **Macro Variable** tab, select the seed device and click **+Command Variable** to add macro variables.
8. Set macro variables: **pair_group**, **pair_virtual_ip**, **pair_hello**, **pair_hold**, and **pair_device**. We shall pass these variables from the parent intent to the follow-up intents for comparison.
9. Click **OK** and close the dialog.



10. Back in the **Network Intent (Edit Mode)** window, click **Edit Diagnosis** to open.
11. Go to **Define Diagnosis** ribbon and delete all the old diagnoses.

12. Add a diagnosis to loop through all groups:

- a) **If** condition: Compare `group_name` and `virtual_ip` against the `pair_group` and `pair_virtual_ip`.
- b) **Then**: If the condition is true, go to *Add Logic>Advanced>Set Intent Variable* and set the variables `match_group_vip` to 1 and `match_hello` and `match_hold` as in the image.
- c) Add the logic **Break Current Loop** from *Add Logic>Advanced> Break Current Loop*. This break is added here since we already found the paired device and so do not need to continue.
- d) Click **Apply**.

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: Untitled Diagnosis 1 Anchor: HSRP_failover.\$virtu...

Type description of the diagnosis...

☒ Loop Table Rows HSRP_failover Table Key: virtual_ip

If

A US-BOS-S... Current
group_name Equals pair_group

B US-BOS-S... Current
virtual_ip Equals pair_virtual_ip

C Select Variable

Boolean Expression: A and B

Then

Set Intent or Task Variable:
match_group_vip = 1

Set Intent or Task Variable:
match_hello = \$hello_time

Set Intent or Task Variable:
match_hold = \$hold_time

Break Current Loop

Add Logic

+ Add Elseif + Add Else

Apply

13. Add another diagnosis:

- a) **If** condition: Variable `match_group_vip` | Equals | 0.
- b) **Then:** If the condition is true, then choose **RED** and error message: `$pair_device` and `$this_device` are HSRP Pairs. Standby device Group and Virtual_IP are not the same.
- c) Check the selection boxes: **Set Status Code for Device** and **Set Status Code for Intent**.

d) Add an **Elseif** condition:

A: `match_hello` | Does not equal | `pair_hello`

B: `match_hold` | Does not equal | `pair_hold`

Boolean Expression: **A or B**

- e) **Then:** If Elseif condition is true, then choose **RED** and error message:
`$pair_device` and `$this_device` hello and hold time do not match:
`$pair_device` device Hello time `$pair_hello` and hold time `$pair_hold`
`$this_device` device Hello time `$match_hello` and hold time `$match_hold`

- f) **Else:** If Elself condition is not true, then choose **Green** and success message: `$pair_device` and `$this_device` hello and hold time is the same.
- g) Check the boxes next to **Set Status Code for Device** and **Set Status Code for Intent**.
- h) Click **Apply** to save and close the diagnosis window.

2. Define Diagnosis

Can also click a variable on the left to add automation.

Untitled Diagnosis 2

Untitled Diagnosis 1

Elself

A Current US-BOS-S...

match_hello Does not equal pair_hello

B Current US-BOS-S...

match_hold Does not equal pair_hold

C Select Variable

Boolean Expression: A or B

Then

Diagnosis Message: Save to Incident

\$pair_device and \$this_device hello and hold time do not match: \$pair_device dev

☒ Set Status Code for Device: Error \$pair_device and \$this_device hello and hold time do not match: \$pair_devi

☒ Set Status Code for Intent: Error \$pair_device and \$this_device hello and hold time do not match: \$pair_devi

Add Logic

Else

Diagnosis Message: Save to Incident

\$pair_device and \$this_device hello and hold time is same

☒ Set Status Code for Device: Success \$pair_device and \$this_device hello and hold time is same

☒ Set Status Code for Intent: Success \$pair_device and \$this_device hello and hold time is same

Add Logic

+ Add Elself

Apply

14. Back in the Network Intent (Edit Mode), Click **Save** and **Run** to execute the intent.

NOTE: You can adjust the default value of macro variables to cover all conditions.

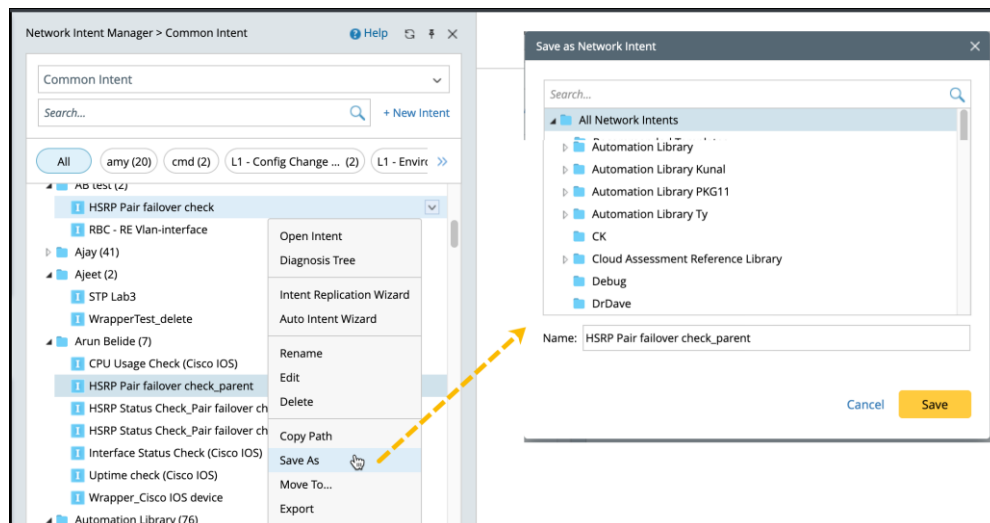
The screenshot displays the NetBrain Network Intent interface. The top window, 'Network Intent (Edit Mode)', shows the configuration for the 'HSRP Pair failover check_followup' intent. It includes a 'Run' button and a 'Save' button. The bottom window, 'Network Intent (View Mode)', shows the execution results. It displays a message: 'This intent execution is finished at 08/02/2024 01:49 PM with 0 errors. You can View Execution Log'. Below this, there are two error messages: 'Standby device and US-BOS-SW1 are HSRP Pairs and the standby device Group and Virtual_IP do not match' (2 errors) and 'Standby device and US-BOS-SW1 are HSRP Pairs' (1 error). A yellow callout box points to the first error message with the text: 'Error message appeared after successful execution'. The bottom window also shows the output of the 'show standby' command for 'US-BOS-SW1'.


```
1 JS-BOS-SW1>show standby
2 Vlan100 - Group 1 (version 2)
3 State is Standby
4 1 state change, last state change 32w3d
5 Virtual IP address is 10.8.1.1
6 Active virtual MAC address is 0000.0c9f.f001 (MAC Not In Use)
7 Local virtual MAC address is 0000.0c9f.f001 (v2 default)
8 Hello time 5 sec, hold time 15 sec
9 Next hello sent in 4.000 secs
10 Preemption enabled
11 Active router is 10.8.1.3, priority 105 (expires in 12.640 sec)
12 MAC address is aabb.cc80.1500
13 Standby router is local
14 Priority 90 (configured 90)
```

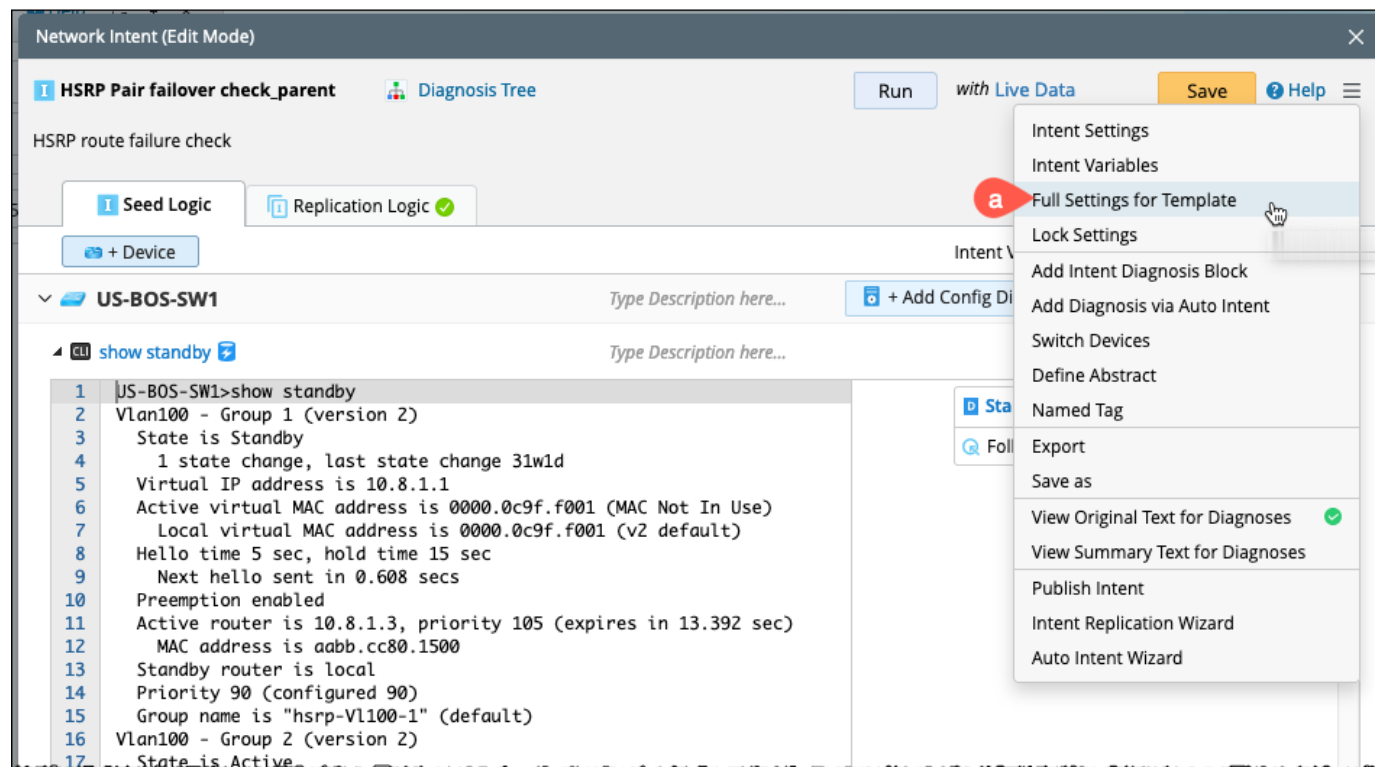

8.3.2 Create a parent intent

Let us create a parent intent based on the intent created in Section 8.3.1 as a parent intent.

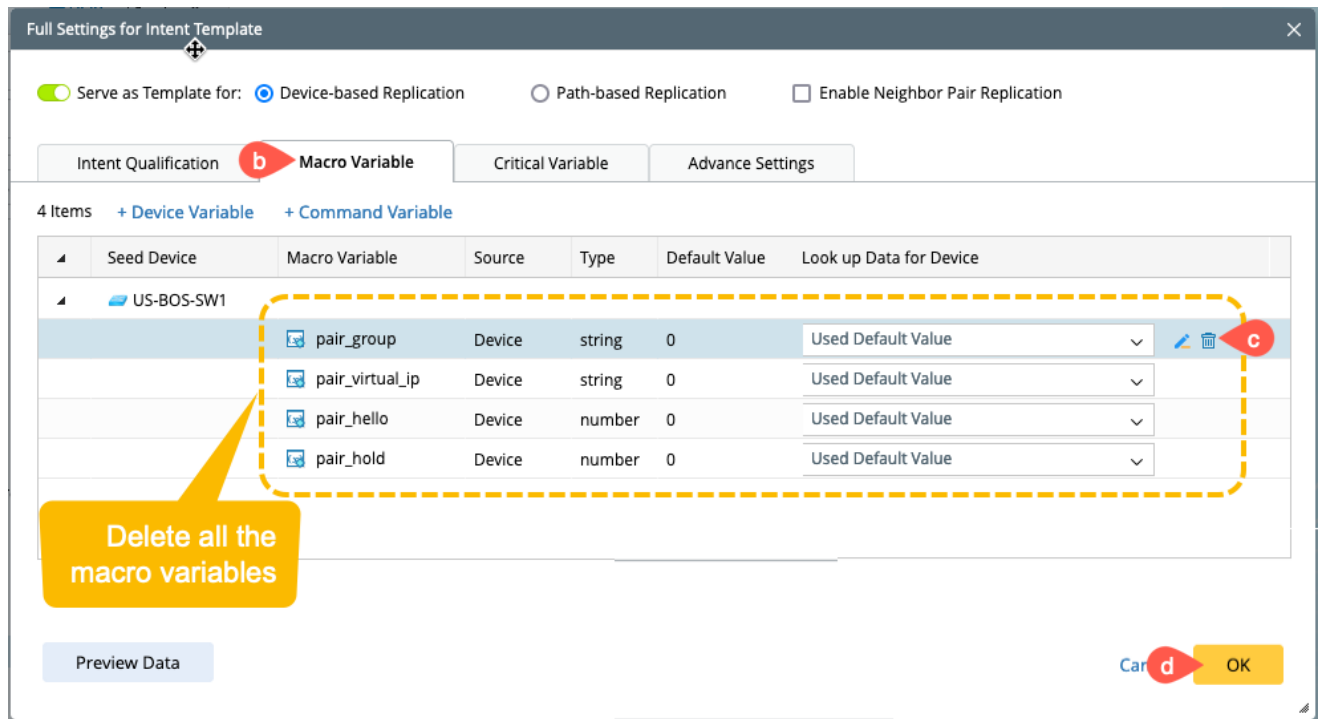
1. **Save As** the follow up intent as **HSRP Pair Failover Check_parent**.



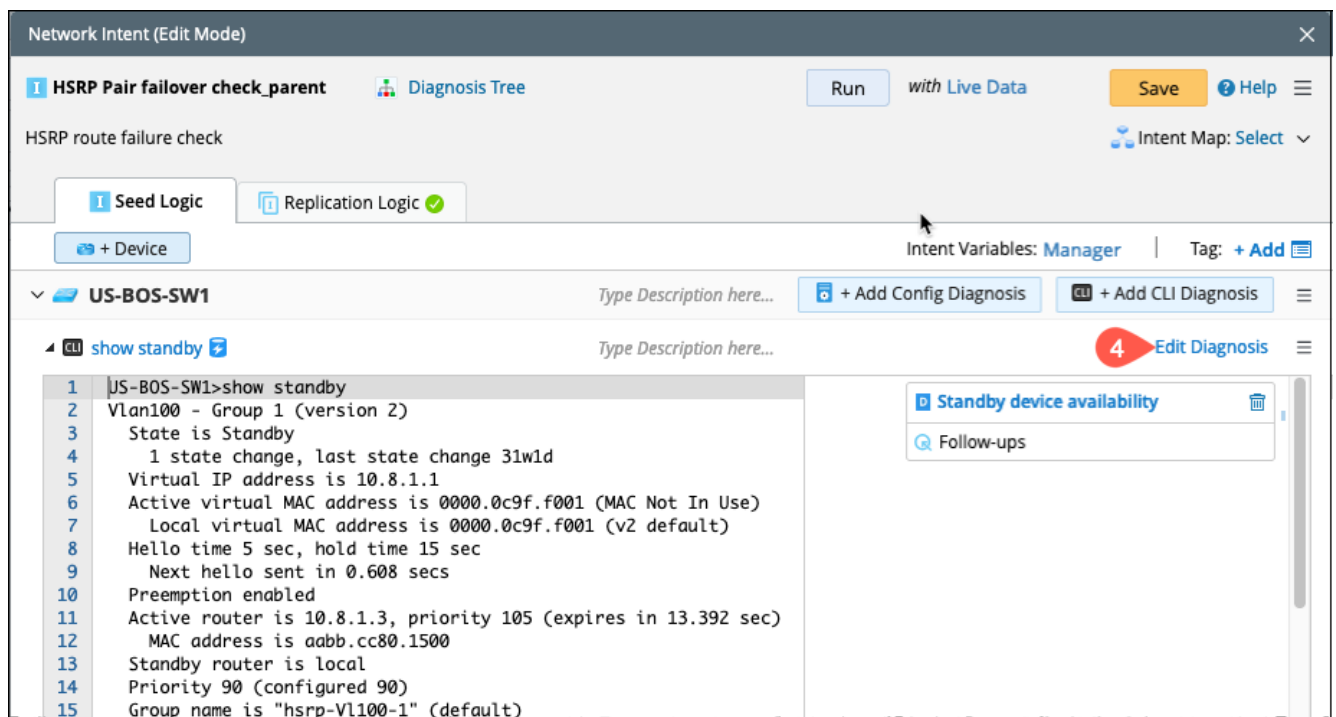
2. Open the new intent **HSRP pair failover check** in the edit mode.
3. Remove the macro variables that are set while configuring follow-up intent :
 - a) Open **Full settings for Template** from  menu.



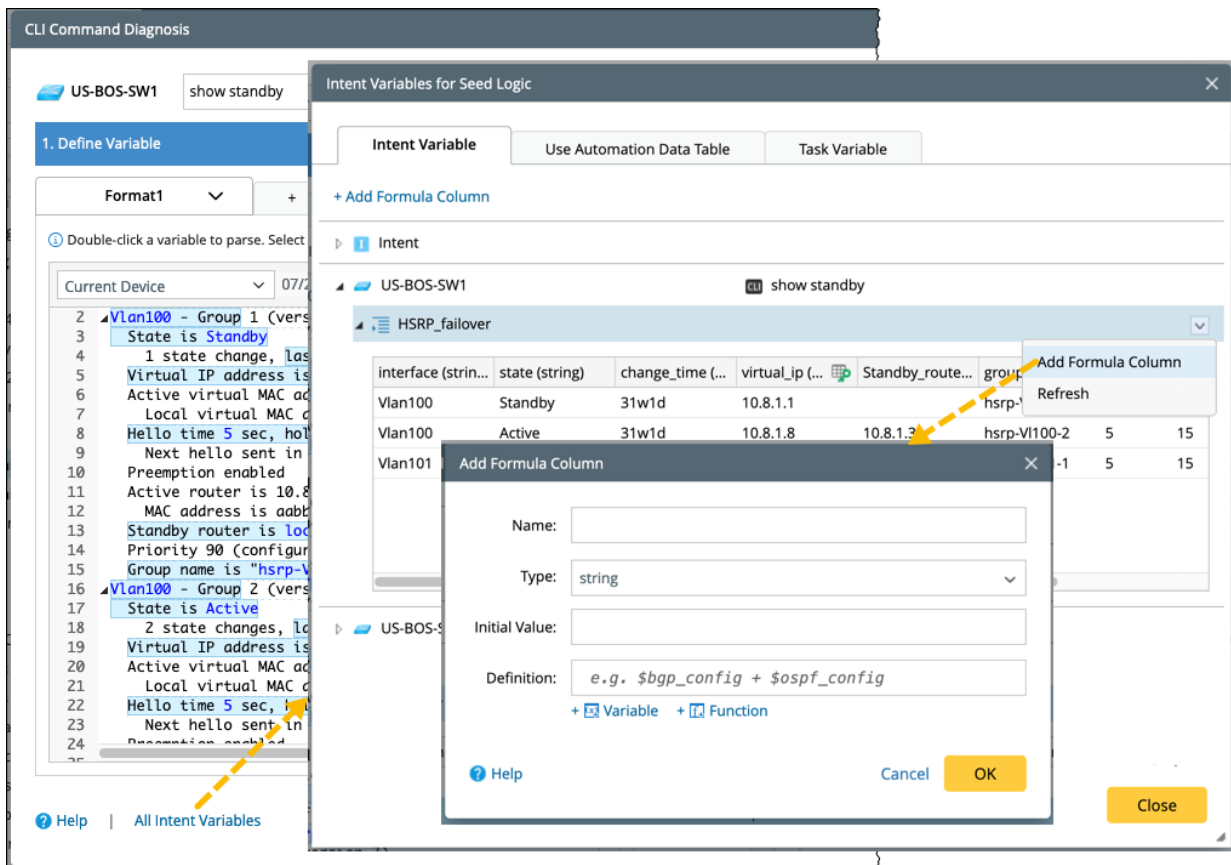
- b) Go to the **Macro Variable** tab section.
- c) Delete the variables (***pair_group***, ***pair_virtual_ip***, ***pair_hello***, and ***pair_hold***).
- d) Click **OK** to save the modifications and close the window.



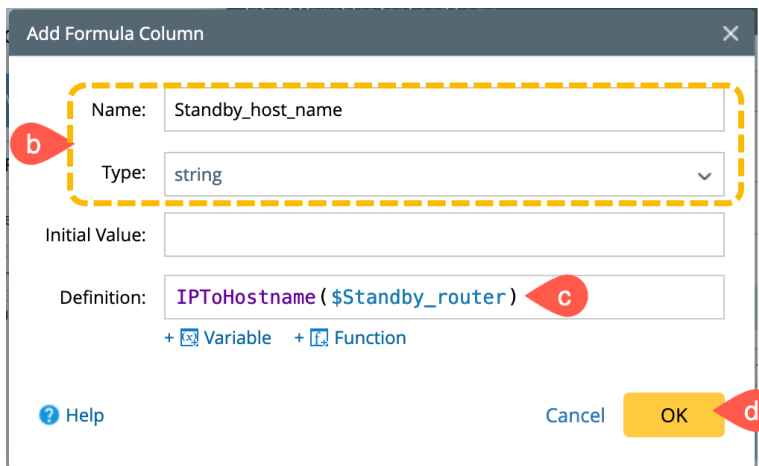
4. Click **Edit Diagnosis** to open the parser.



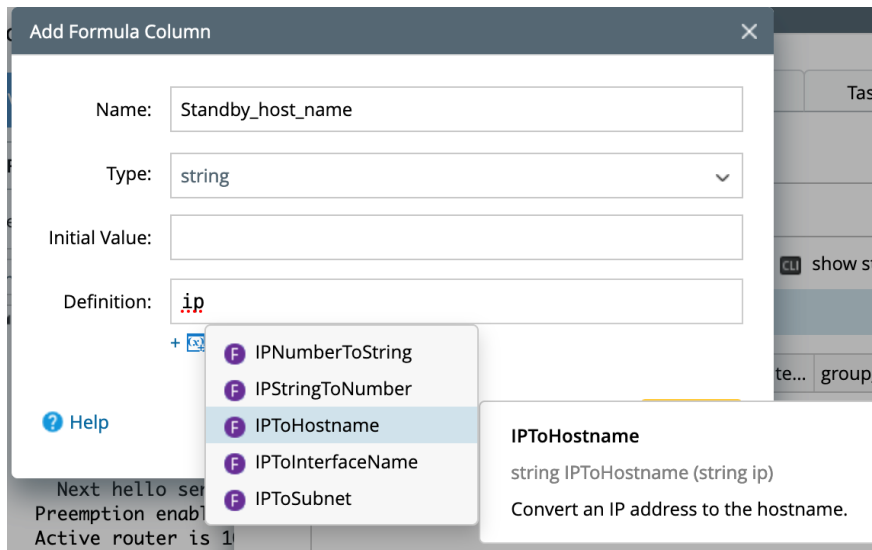
5. In the parser, add a column, **backup_ip device name**, to the parser output table by converting the **backup_ip** to the device name using the function call:
 - a) Go to **All Intent Variables > Intent Variable > HSRP_failover table**, click on the drop down menu, and choose **Add Formula Column**.



- b) In the **Add Formula Column** dialog, add the name **Standby_host_name** and Type is **string**.
- c) Definition: Choose the function **IpToHostname** and variable (**\$Standby_router**).



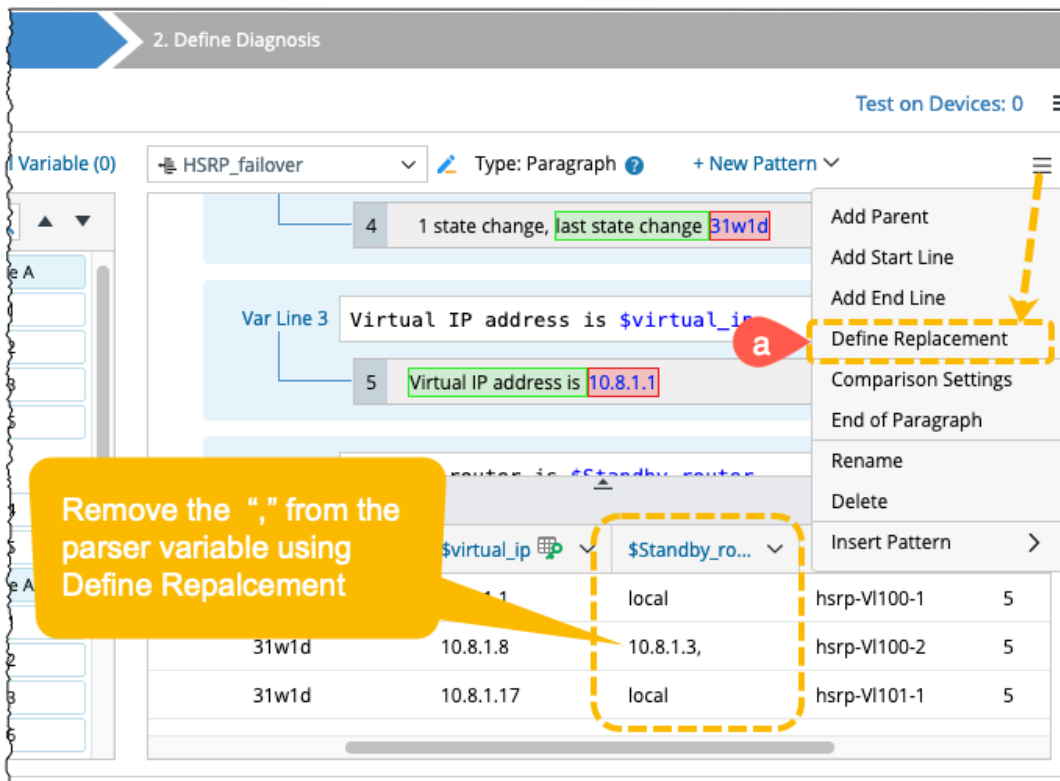
NOTE: As you type IP in the **Definition** field, the list of functions with IP will appear as a list. Choose the function **IpToHostname**.




d) Click **OK** to save and close the dialog.

e) Close the Intent Variables window.

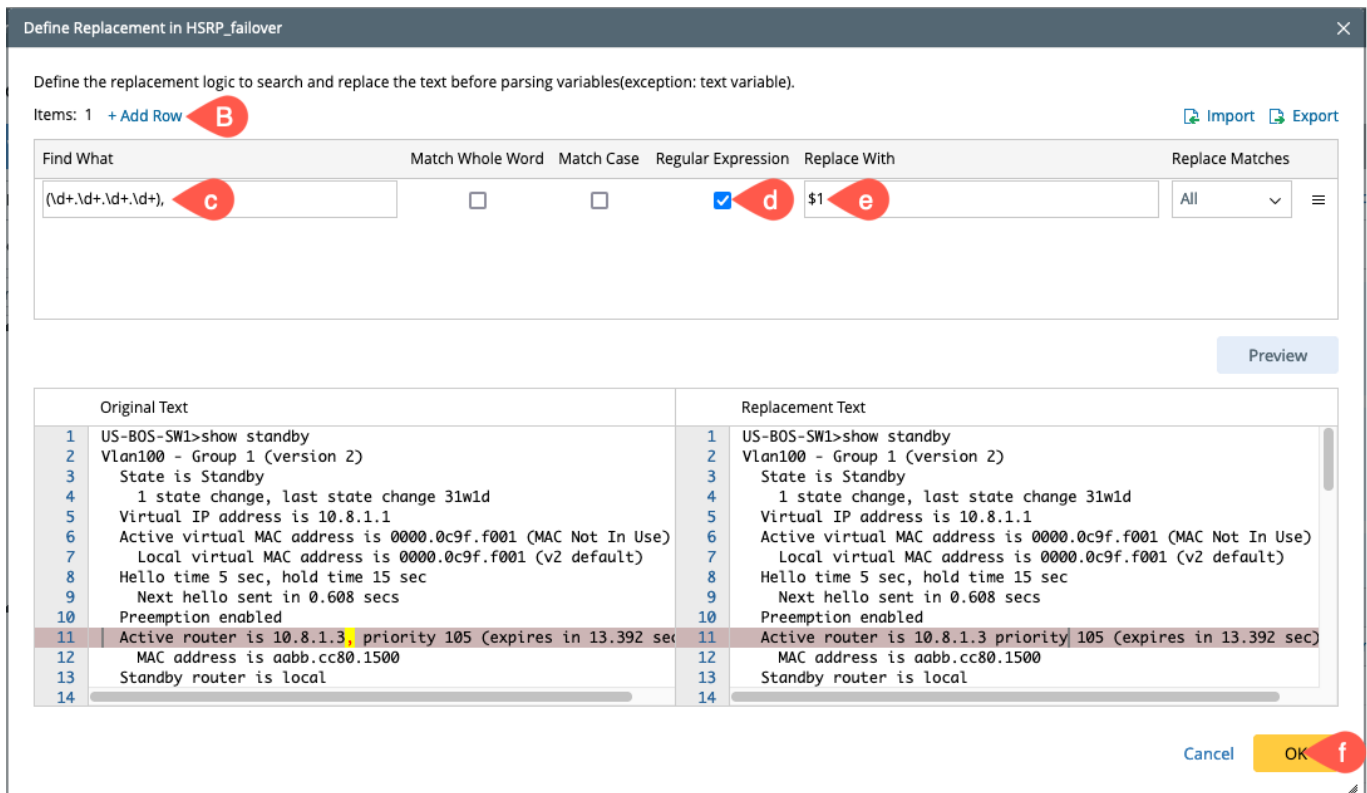
6. Refine the parser variable **standby_router** further to remove “,” from the ip to use the functionality **IpToHostname**:



- Select **Define Replacement** from  menu to the corresponding dialog.
- Click **+Add Row**.
- In the field **Find What**, enter the text “**(\d+.\d+.\d+.\d+),**”. We are constructing the variable for “**ip 10.8.1.3,**” inside the braces.
- Check the selection box under **Regular Expression**.
- In the field **Replace With**, enter the text “**\$1**”.

NOTE: **Replace With** is to replace the expression defined in the field **Find what**. Here, **\$1** stands for variable 1, and it will replace the expression defined inside the braces of the **Find What** field.

- Click **OK** to save the settings and close the window.



Define Replacement in HSRP_failover

Define the replacement logic to search and replace the text before parsing variables(exception: text variable).

Items: 1 **+ Add Row**

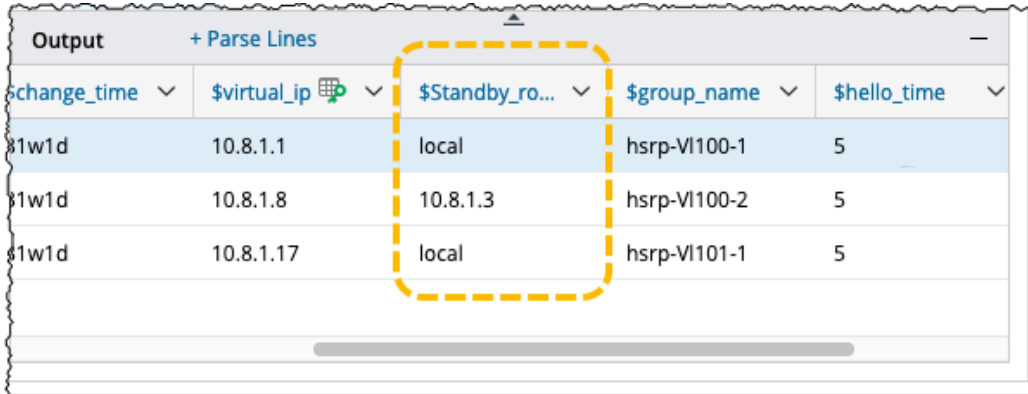
Find What	Match Whole Word	Match Case	Regular Expression	Replace With	Replace Matches
(\d+.\d+.\d+.\d+),	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	\$1	All

Preview

Original Text	Replacement Text
1 US-BOS-SW1>show standby	1 US-BOS-SW1>show standby
2 Vlan100 - Group 1 (version 2)	2 Vlan100 - Group 1 (version 2)
3 State is Standby	3 State is Standby
4 1 state change, last state change 31w1d	4 1 state change, last state change 31w1d
5 Virtual IP address is 10.8.1.1	5 Virtual IP address is 10.8.1.1
6 Active virtual MAC address is 0000.0c9f.f001 (MAC Not In Use)	6 Active virtual MAC address is 0000.0c9f.f001 (MAC Not In Use)
7 Local virtual MAC address is 0000.0c9f.f001 (v2 default)	7 Local virtual MAC address is 0000.0c9f.f001 (v2 default)
8 Hello time 5 sec, hold time 15 sec	8 Hello time 5 sec, hold time 15 sec
9 Next hello sent in 0.608 secs	9 Next hello sent in 0.608 secs
10 Preemption enabled	10 Preemption enabled
11 Active router is 10.8.1.3, priority 105 (expires in 13.392 sec)	11 Active router is 10.8.1.3 priority 105 (expires in 13.392 sec)
12 MAC address is aabb.cc80.1500	12 MAC address is aabb.cc80.1500
13 Standby router is local	13 Standby router is local
14	14

Cancel OK

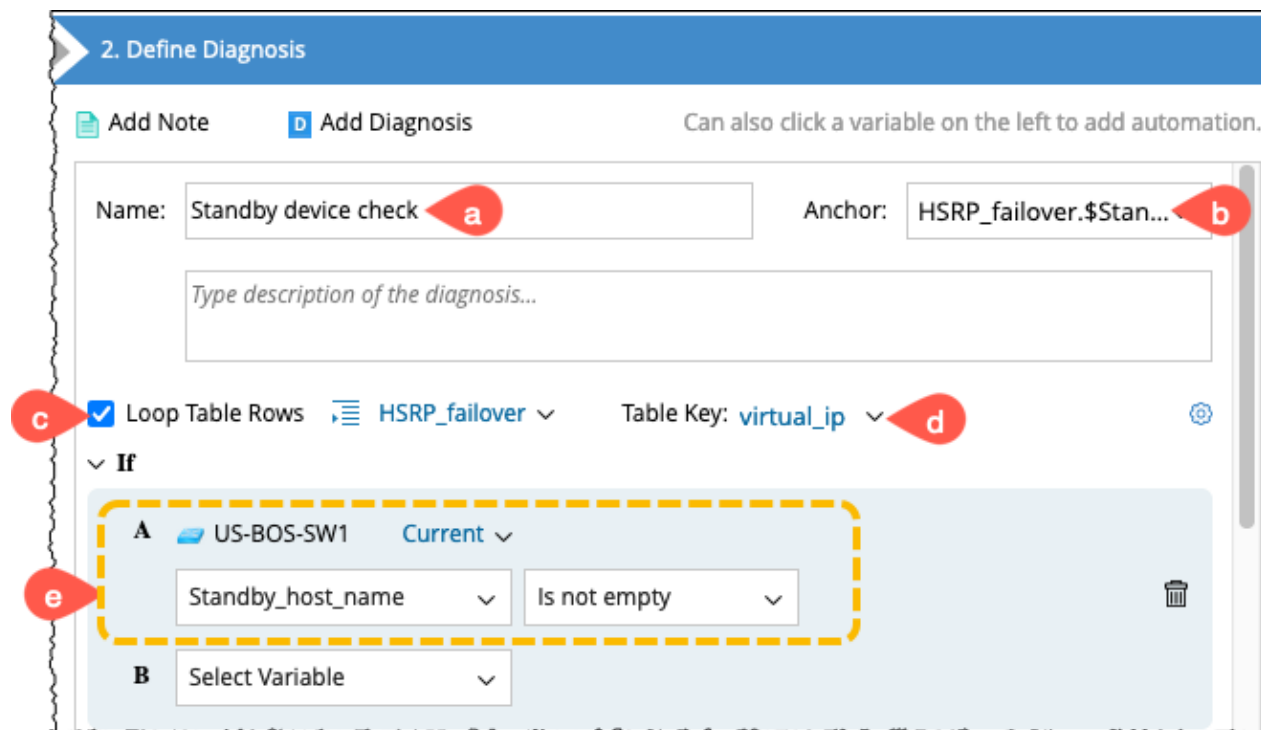
g) The final output of the variable `standby_router` will be:



change_time	\$virtual_ip	\$Standby_ro...	\$group_name	\$hello_time
1w1d	10.8.1.1	local	hsrp-VI100-1	5
1w1d	10.8.1.8	10.8.1.3	hsrp-VI100-2	5
1w1d	10.8.1.17	local	hsrp-VI101-1	5

7. Back in the diagnosis window, go to the **Define Diagnosis** ribbon, delete the diagnosis created for follow-up intent, and create a new diagnosis as follows:

- a) Name: Standby device check.
- b) Anchor: `Standby_router`.
- c) Check the selection box of **Loop Table Rows** and select the table **HSRP_failover**.
- d) Table Key: Select the variable `virtual_ip`.
- e) **If** condition: `Standby_router` | Is not empty.



2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: Standby device check **a** Anchor: HSRP_failover.\$Stan... **b**

Type description of the diagnosis...

c ☒ Loop Table Rows **HSRP_failover** Table Key: virtual_ip **d**

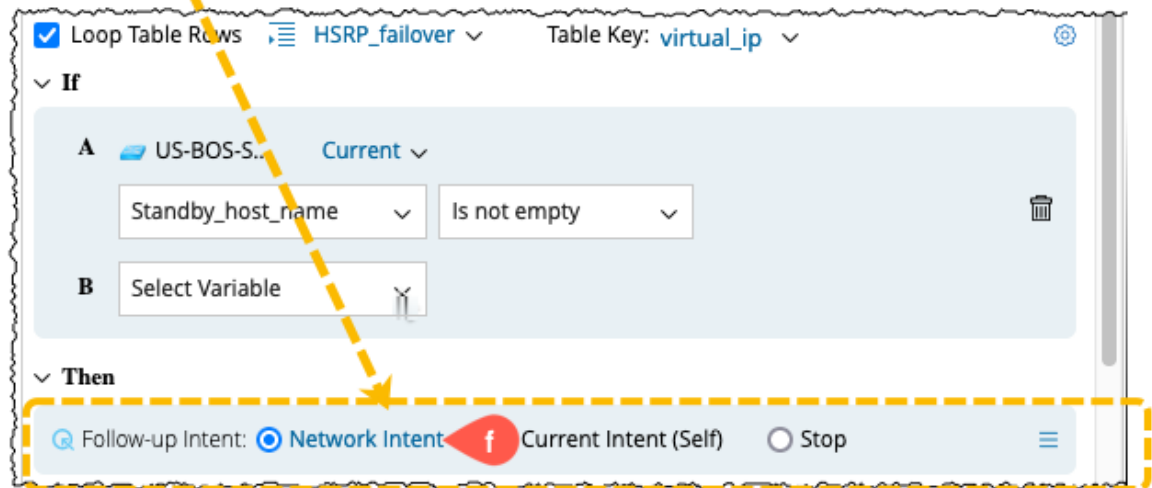
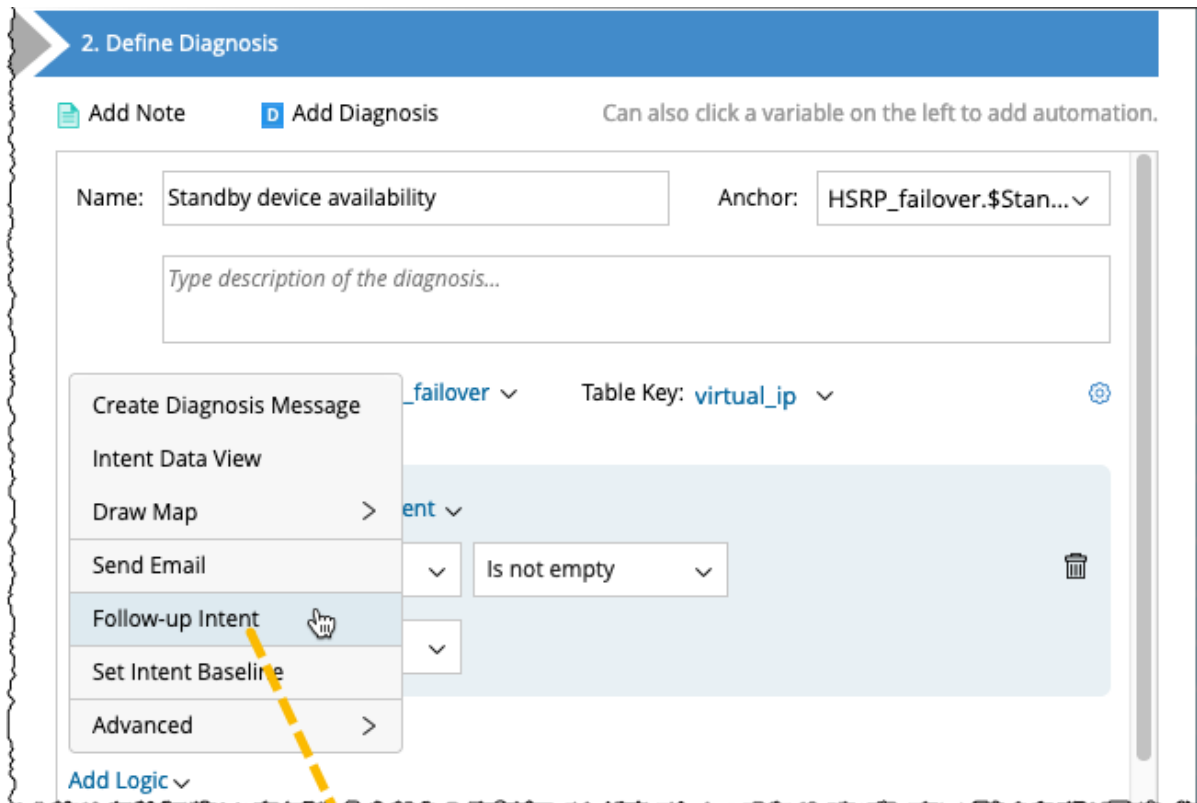
e **If**

A US-BOS-SW1 Current

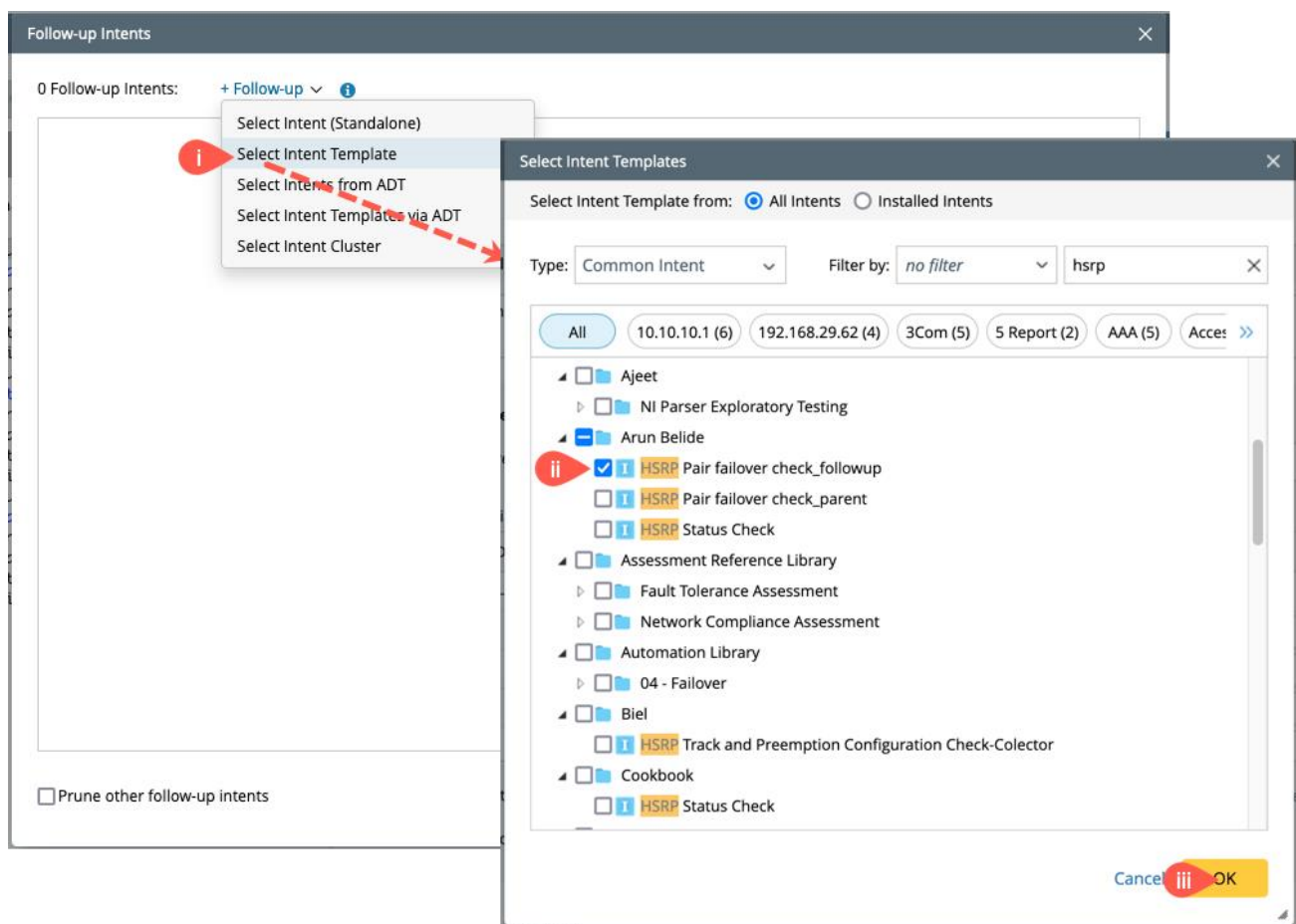
Standby_host_name Is not empty

B Select Variable

- f) **Then:** Remove the **Diagnosis Message** and add a follow-up action from the **Add logic** dropdown > **Follow-up Intent**.



- g) Click the **Network Intent** link to choose the follow-up intent in a new dialog **Follow-up Intents**.
- In the Follow-up Intents window, choose **Select Intent Template**.
 - In the Select Intents dialog, navigate to the follow-up intent (**HSRP pair failover check_followup**) and check the selection box.
 - Click **OK** to save the selection and close the window.



iv. In the Replication Settings:

Set the **Replicate Current Intent to: Device by Variable** and choose the variable Standby_host_name.

v. In the **Set Macro Variables of Seed Intent Template**: set the variables for macro variables as follows:

vi. Click **Save** to save and close the follow-up intents.

Follow-up Intents

1 Follow-up Intent: [+ Follow-up](#) [i](#)

HSRP Pair failover check_followup (Intent Template) [🗑️](#)

Description: When...

Replication Settings:

Replicate Current Intent to: **Device by Variable** **Standby_host_name (HSRP_failover)**

Set Macro Variables of Seed Intent Template:

Seed Device	Macro Variable	Type	Set Variable
US-BOS-SW1	pair_group	string	group_name (HSRP_failover)
	pair_virtual_ip	string	virtual_ip (HSRP_failover)
	pair_hello	number	hello_time (HSRP_failover)
	pair_hold	number	hold_time (HSRP_failover)

☐ Merge multiple replicated intents into one [1/1 Device Key Set for Selected Table](#)

Follow-up Execution: [Settings](#)

☐ Prune other follow-up intents

[Can](#) **vi** **Save**

h) Back in the **Define Diagnosis** ribbon, add an **ElseIf** condition for the unknown standby devices:

A: Standby_router | Contains | Unknown

i) **Then:** Define the diagnosis message: \$this device does not have standby HSRP.

j) Check the boxes next to **Set Status Code for Device** and **Set Status Code for Intent**.

k) Click **Apply** to save and close the diagnosis window.

The screenshot shows the 'Define Diagnosis' dialog box. At the top, there's a 'Add Logic' dropdown. Below it, the 'ElseIf' section is expanded. It contains two conditions: 'A' and 'B'. Condition 'A' is highlighted with a red circle and letter 'h'. It shows 'US-BOS-S...' as the device, 'Current' as the type, and the logic 'Standby_router' 'Contains' 'unknown'. Condition 'B' is 'Select Variable'. Below the conditions is the 'Then' section. It has a 'Diagnosis Message' field with the text '\$this_device do not have standby HSRP', which is also highlighted with a red circle and letter 'i'. Below the message field, there are two actions: 'Set Status Code for Device' and 'Set Status Code for Intent'. Both are checked with a blue box and have a status of 'Error' (highlighted with a red circle and letter 'j'). Both actions have the same message: '\$this_device do not have standby HSRP'. At the bottom right, there are 'Cancel' and 'Apply' buttons. The 'Apply' button is highlighted with a red circle and letter 'k'.

8. In the Network Intent (Edit Mode) dialog, click **Save**.

9. Click **Run** to execute the intent.

Network Intent (Edit Mode)

HSRP Pair failover check_parent Diagnosis Tree 9 Run with Live Data 8 Save Help

HSRP route failure check Intent Map: Select

Seed Logic Replication Logic

+ Device Intent Variables: Manager Tag: + Add

US-BOS-SW1 Type Description here... + Add Config Diagnosis + Add CLI Diagnosis

show standby Type Description here... Edit Diagnosis

```

1 US-BOS-SW1>show standby
2 Vlan100 - Group 1 (version 2)
3 State is Standby
4 1 state change, last state change 31w1d
5 Virtual IP address is 10.8.1.1
6 Active virtual MAC address is 0000.0c9f.f001 (MAC Not In Use)
7 Local virtual MAC address is 0000.0c9f.f001 (v2 default)
8 Hello time 5 sec, hold time 15 sec

```

Standby device availability Follow-ups

10. Open the diagnosis tree to see the results.

Network Intent (View Mode) - All Network Intents/0_Power User 2 Sandbox/HSRP Pair failover check_parent

HSRP Pair failover check_parent HSRP route failure check Open 0 0 Edit

Result: 08/02/2024 01:59 PM 10 Run with Live Data

This intent execution is finished at 08/02/2024 01:59 PM with 0 errors. You can View [Execution Log](#)

S US-BOS-SW1 and US-BOS-SW2 hello and hold time is same 2 View

US-BOS-SW1 1 Diagnosis

Diagnosis Tree of HSRP Pair failover check_parent

Source: ArunKumar.Belide@netbraintech... Current NI: **HSRP Pair failover check_pa...** Execution Time: 08/02/2024 01:59:30 PM View Settings Auto Layout

Pre-Execution | Post-Execution

```

graph LR
    Start([Start]) --> I1[I]
    I1 --> US-BOS-SW1[US-BOS-SW1]
    US-BOS-SW1 --> D1[D]
    D1 -- Because... --> I2[I]
    I2 --> US-BOS-SW2[US-BOS-SW2]
    US-BOS-SW2 --> D2[D]
    US-BOS-SW2 --> D3[D]
    D2 --> D3
    D3 --> D4[D]
    D4 --> D5[D]

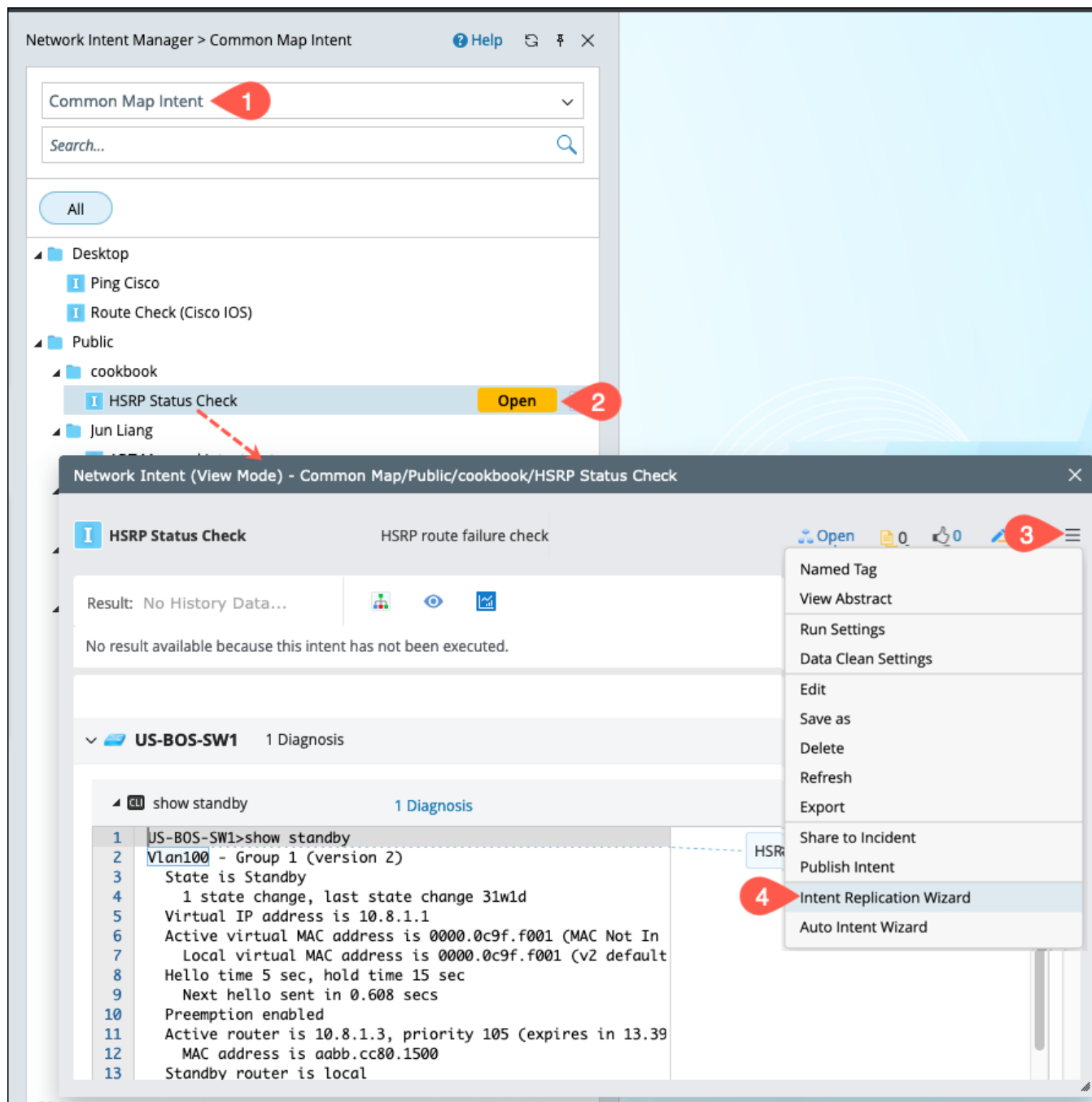
```

Legend

Network Intent Details - HSRP Pair failover check_parent

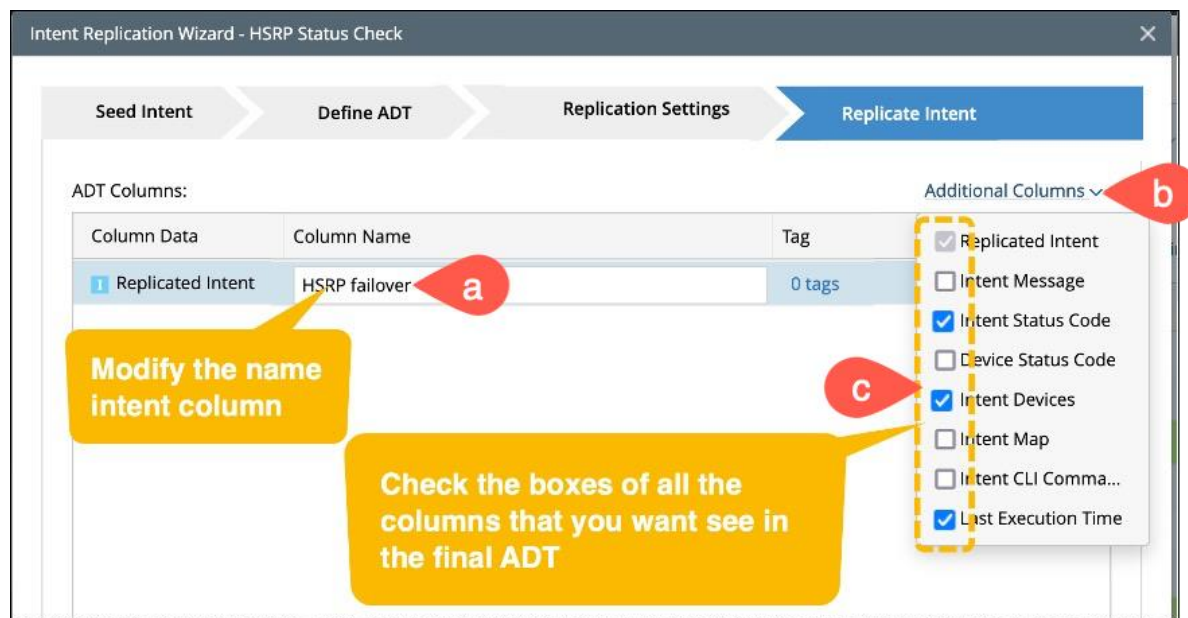
8.3.3 Replicate the intents for all HSRP devices using ADT

In this section, you will use the **Intent Replication Wizard** to replicate the intent to all devices in the HSRP device group. As a first step, open the intent from the intent manager and the Intent replication wizard as shown:

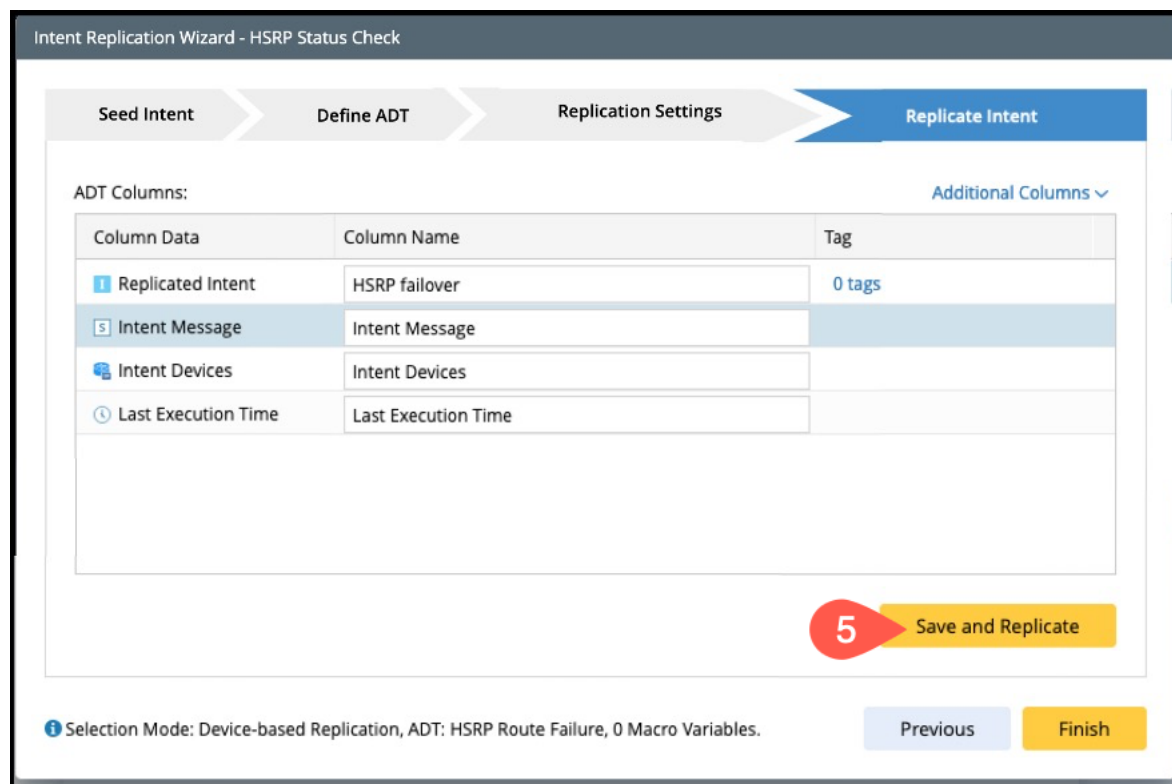


1. In the first step (**Seed Intent**), Validate the default seed intent and go to the next section.
2. In the second step (**Define ADT**), provide the basic input to create a new ADT, such as name, ADT location and target devices.

3. In the third step, define the **intent qualification** with device group **HSRP Devices** to include devices for the seed intent to replicate on.
4. In the fourth step, **Replicate Intent**, modify the name of the **Replicated Intent** In the **ADT Columns** section and add more columns that you want in the final ADT.



5. Click **Save and Replicate** to save all the settings and create ADT. An option **Open Output ADT** will appear. Click it to open the table in the **ADT Manager**.



Intent Replication Wizard - HSRP Status Check

Seed IntentDefine ADTReplication SettingsReplicate Intent

ADT Columns:Additional Columns

Column Data	Column Name	Tag
1 Replicated Intent	HSRP failover	0 tags
5 Intent Message	Intent Message	
Intent Devices	Intent Devices	
Last Execution Time	Last Execution Time	

Click to open the table in ADT manager.

Save and Replicate

Replication Request submitted at: 07/26/2024 09:17 PMOpen Output ADT

Selection Mode: Device-based Replication, ADT: HSRP Route Failure, 0 Macro Variables.

PreviousFinish

Review the new Intent columns that are added to the table and results.

Automation Data Table Manager

Search...

Shared Tables (1201)

My Tables (5)

Ping (1)

BGP Config Change Diagnosis

Day 1 Lab

HSRP Route Failure

NTP

HSRP Route Failure

Table Builder

Last Updated at: 07/30/2024 12:57 PM

Rebuild Table

Add Data Manually

Description: Type description here...

Items: 19 Rows 4 Columns

Search...

Advanced Filter: Undefined

No.	Device	HSRP failover	Intent Status Code	Last Execution Time
1	BJ-L2-coreB	HSRP Status Check BJ-L2-coreB	BJ-L2-coreB HSRP hsrp-VI10-100 is stable	07/29/2024 03:55:07 PM
2	BJ_L2_Core_3	HSRP Status Check BJ_L2_Core_3	BJ_L2_Core_3 HSRP hsrp-Fa1/0/9-10 is stable	07/29/2024 03:55:07 PM
3	BJ_L2_Core_4	HSRP Status Check BJ_L2_Core_4	BJ_L2_Core_4 HSRP hsrp-Fa2/0/3-0 is stable	07/29/2024 03:55:05 PM
4	BJ_core_3550	HSRP Status Check BJ_core_3550	BJ_core_3550 HSRP None is stable	07/29/2024 03:55:07 PM
5	Bur-isp-gw1	HSRP Status Check Bur-isp-gw1	Bur-isp-gw1 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
6	IPv6Lab-SW8	HSRP Status Check IPv6Lab-SW8	IPv6Lab-SW8 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:05 PM
7	IPv6Lab-SW9	HSRP Status Check IPv6Lab-SW9	IPv6Lab-SW9 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:07 PM
8	PE-3600X-01	HSRP Status Check PE-3600X-01	PE-3600X-01 HSRP hsrp-Gi0/24-2 is stable	07/29/2024 03:55:07 PM
9	PE-3600X-02	HSRP Status Check PE-3600X-02	PE-3600X-02 HSRP hsrp-Gi0/13-2 is stable	07/29/2024 03:55:07 PM
10	PE-ASR1K-01	HSRP Status Check PE-ASR1K-01	PE-ASR1K-01 HSRP hsrp-Te0/0/0-15 is stable	07/29/2024 03:55:05 PM
11	PE-ASR1K-02	HSRP Status Check PE-ASR1K-02	PE-ASR1K-02 HSRP hsrp-Gi0/0/5-10 is stable	07/29/2024 03:55:05 PM
12	Sjc-Dist-3750-02	HSRP Status Check Sjc-Dist-3750-02	Sjc-Dist-3750-02 HSRP hsrp-VI30-0 is stable	07/29/2024 03:55:07 PM
13	US-BOS-SW1	HSRP Status Check US-BOS-SW1	US-BOS-SW1 HSRP hsrp-VI100-1 is stable	07/29/2024 03:55:07 PM
14	US-BOS-SW2	HSRP Status Check US-BOS-SW2	US-BOS-SW2 HSRP hsrp-VI100-1 is stable	07/29/2024 03:55:05 PM
15	bjta002237-SW2	HSRP Status Check bjta002237-SW2	bjta002237-SW2 HSRP hsrp-VI481-1 is stable	07/29/2024 03:55:07 PM
16	bjta002238-SW3	HSRP Status Check bjta002238-SW3	bjta002238-SW3 HSRP hsrp-VI481-1 is stable	07/29/2024 03:55:05 PM
17	bjta002444-SW13	HSRP Status Check bjta002444-SW13	bjta002444-SW13 HSRP hsrp-Et1/1-1 is stable	07/29/2024 03:55:05 PM
18	bur-isp-gw2	HSRP Status Check bur-isp-gw2	bur-isp-gw2 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
19	qapp-c3560-2	HSRP Status Check qapp-c3560-2	qapp-c3560-2 HSRP hsrp-Gi0/15-0 is stable	07/29/2024 03:55:07 PM

8.3.4 Replicate the intent to all HSRP devices

Replicate the intent to all HSRP devices using the **Intent Replication Wizard** from the network intent window. In the intent replication wizard, input the seed intent and target devices as follows:

1. In the first step (**Seed Intent**), Validate the default seed intent and go to the next section.
2. In the second step (**Define ADT**), provide the basic input to create a new ADT, such as name (HSRP Pair failover), ADT location and target device group.
3. In the third step, define the **intent qualification** with device group **HSRP Devices** to include devices for the seed intent to replicate on.
4. In the fourth step, **Replicate Intent**, modify the name of the **Replicated Intent** In the **ADT Columns** section and add other columns like **Intent Status Code** and **Last Execution Time** as needed in the final ADT.

5. Select **Save and Replicate** to save all the settings and create ADT. An option **Open Output ADT** will appear. Click it to open the table in the **ADT Manager**.

Intent Replication Wizard - HSRP Pair failover check_parent

Seed Intent
Define ADT
Replication Settings
Replicate Intent

ADT Columns:
Additional Columns

Column Data	Column Name	Tag
Replicated Intent	HSRP Pair failover check	0 tags
Intent Status Code	Intent Status Code	
Last Execution Time	Last Execution Time	

Click the link to open ADT in new window

Save and Replicate

Replication Request submitted at: 08/02/2024 02:30 PM
Open Output ADT

Selection Mode: Device-based Replication, ADT: HSRP Pair failover, 0 Macro Variables.
Previous
Finish

Review the new Intent columns that are added to the table and results.

Automation Data Table Manager

Search...

Shared Tables (1201)
My Tables (5)
Ping (1)
BGP Config Change Diagnosis
Day 1 Lab
HSRP Route Failure
NTP

HSRP Route Failure
Table Builder
Last Updated at: 07/30/2024 12:57 PM
Rebuild Table
Add Data Manually

Description: Type description here...
Items: 19 Rows 4 Columns
Search...
Advanced Filter: Undefined

No.	Device	HSRP failover	Intent Status Code	Last Execution Time
1	BJ-L2-coreB	HSRP Status Check BJ-L2-coreB	BJ-L2-coreB HSRP hsrp-V10-100 is stable	07/29/2024 03:55:07 PM
2	BJ_L2_Core_3	HSRP Status Check BJ_L2_Core_3	BJ_L2_Core_3 HSRP hsrp-Fa1/0/9-10 is stable	07/29/2024 03:55:07 PM
3	BJ_L2_Core_4	HSRP Status Check BJ_L2_Core_4	BJ_L2_Core_4 HSRP hsrp-Fa2/0/3-0 is stable	07/29/2024 03:55:05 PM
4	BJ_core_3550	HSRP Status Check BJ_core_3550	BJ_core_3550 HSRP None is stable	07/29/2024 03:55:07 PM
5	Bur-isp-gw1	HSRP Status Check Bur-isp-gw1	Bur-isp-gw1 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
6	IPv6Lab-SW8	HSRP Status Check IPv6Lab-SW8	IPv6Lab-SW8 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:05 PM
7	IPv6Lab-SW9	HSRP Status Check IPv6Lab-SW9	IPv6Lab-SW9 HSRP hsrp-Et0/1-1 is stable	07/29/2024 03:55:07 PM
8	PE-3600X-01	HSRP Status Check PE-3600X-01	PE-3600X-01 HSRP hsrp-Gi0/24-2 is stable	07/29/2024 03:55:07 PM
9	PE-3600X-02	HSRP Status Check PE-3600X-02	PE-3600X-02 HSRP hsrp-Gi0/13-2 is stable	07/29/2024 03:55:07 PM
10	PE-ASR1K-01	HSRP Status Check PE-ASR1K-01	PE-ASR1K-01 HSRP hsrp-Te0/0/0-15 is stable	07/29/2024 03:55:05 PM
11	PE-ASR1K-02	HSRP Status Check PE-ASR1K-02	PE-ASR1K-02 HSRP hsrp-Gi0/0/5-10 is stable	07/29/2024 03:55:05 PM
12	Sjc-Dist-3750-02	HSRP Status Check Sjc-Dist-3750-02	Sjc-Dist-3750-02 HSRP hsrp-Vi30-0 is stable	07/29/2024 03:55:07 PM
13	US-BOS-SW1	HSRP Status Check US-BOS-SW1	US-BOS-SW1 HSRP hsrp-Vi100-1 is stable	07/29/2024 03:55:07 PM
14	US-BOS-SW2	HSRP Status Check US-BOS-SW2	US-BOS-SW2 HSRP hsrp-Vi100-1 is stable	07/29/2024 03:55:05 PM
15	bjta002237-SW2	HSRP Status Check bjta002237-SW2	bjta002237-SW2 HSRP hsrp-Vi481-1 is stable	07/29/2024 03:55:07 PM
16	bjta002238-SW3	HSRP Status Check bjta002238-SW3	bjta002238-SW3 HSRP hsrp-Vi481-1 is stable	07/29/2024 03:55:05 PM
17	bjta002444-SW13	HSRP Status Check bjta002444-SW13	bjta002444-SW13 HSRP hsrp-Et1/1-1 is stable	07/29/2024 03:55:05 PM
18	bur-isp-gw2	HSRP Status Check bur-isp-gw2	bur-isp-gw2 HSRP hsrp-Gi0/0/0-200 is stable	07/29/2024 03:55:07 PM
19	qapp-c3560-2	HSRP Status Check qapp-c3560-2	qapp-c3560-2 HSRP hsrp-Gi0/15-0 is stable	07/29/2024 03:55:07 PM

8.3.5 View and Run the intents in ADT

Navigate to the ADT you have just created, and we shall run the intent as follows:

1. Go to the Intents column, hover the mouse on the column header, and click **Run**.
2. Click **Rebuild** to refresh the results in the **Intent Status code**.

Automation Data Table Manager

Search... [Refresh] [Back]

Shared Tables (1208)
My Tables (6)
Ping (1)
BGP Config Change Diagnosis
Day 1 Lab
HSRP Pair failover
HSRP Route Failure
NTP


HSRP Pair failover [Table Builder] Last Updated at: 08/02/2024 03:36 PM [Rebuild Table]

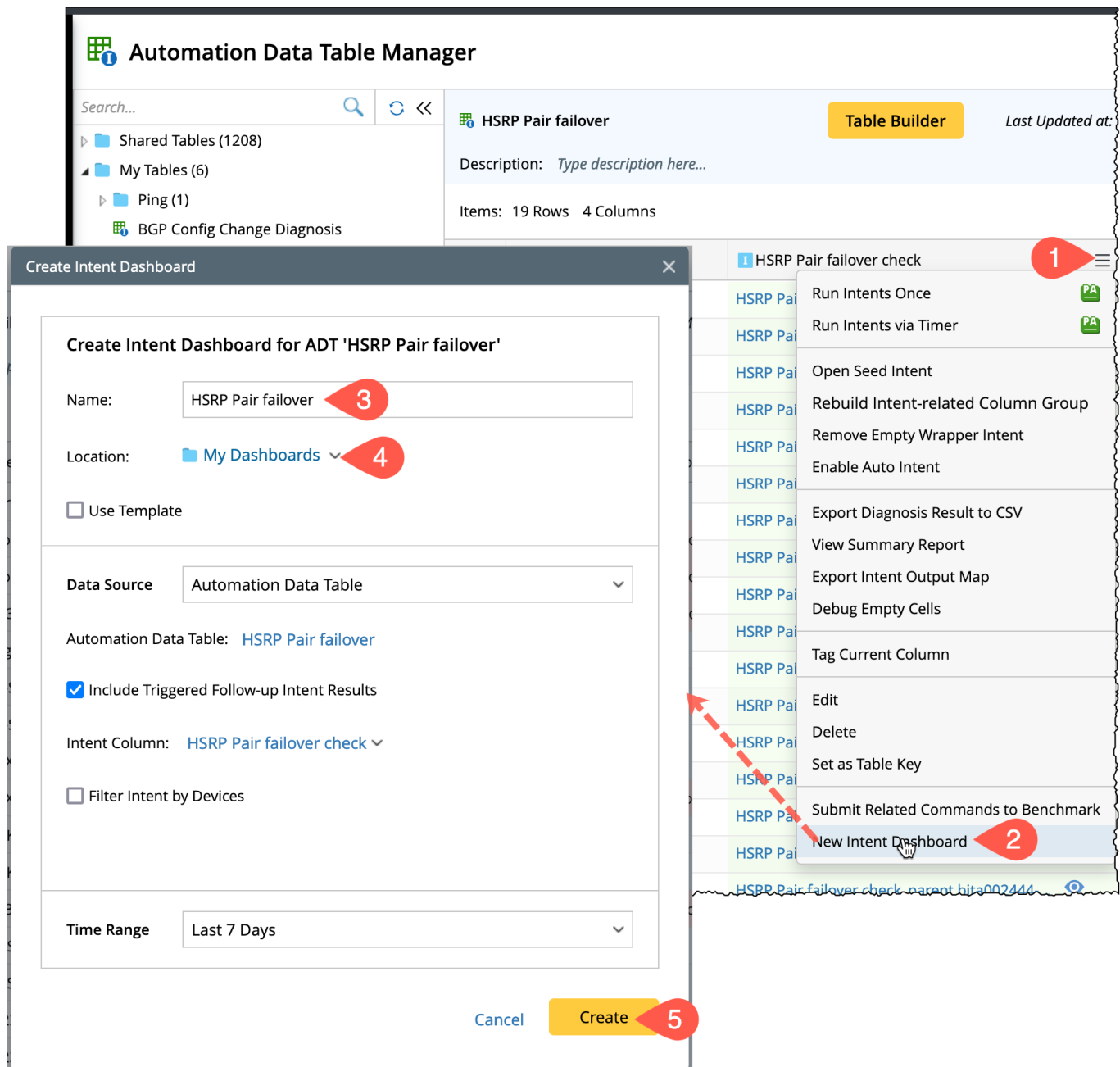
Description: Type description here...

Items: 19 Rows 4 Columns [Search...]

No.	Device	HSRP Pair failover c	Run	Details	Intent Status Code
1	BJ-L2-coreB	HSRP Pair failover check_parent BJ-L2-coreB			BJ-L2-coreB do not have standby HSRP
2	BJ_L2_Core_3	HSRP Pair failover check_parent BJ_L2_Core_3			BJ_L2_Core_3 do not have standby HSRP
3	BJ_L2_Core_4	HSRP Pair failover check_parent BJ_L2_Core_4			BJ_L2_Core_4 do not have standby HSRP
4	BJ_core_3550	HSRP Pair failover check_parent BJ_core_3550			BJ_core_3550 do not have standby HSRP
5	Bur-isp-gw1	HSRP Pair failover check_parent Bur-isp-gw1			Bur-isp-gw1 do not have standby HSRP
6	IPv6Lab-SW8	HSRP Pair failover check_parent IPv6Lab-SW8			IPv6Lab-SW8 do not have standby HSRP
7	IPv6Lab-SW9	HSRP Pair failover check_parent IPv6Lab-SW9			
8	PE-3600X-01	HSRP Pair failover check_parent PE-3600X-01			PE-3600X-01 do not have standby HSRP
9	PE-3600X-02	HSRP Pair failover check_parent PE-3600X-02			PE-3600X-02 do not have standby HSRP
10	PE-ASR1K-01	HSRP Pair failover check_parent PE-ASR1K-01			PE-ASR1K-01 do not have standby HSRP
11	PE-ASR1K-02	HSRP Pair failover check_parent PE-ASR1K-02			PE-ASR1K-02 do not have standby HSRP
12	Sjc-Dist-3750-02	HSRP Pair failover check_parent Sjc-Dist-375...			Sjc-Dist-3750-02 do not have standby HSRP
13	US-BOS-SW1	HSRP Pair failover check_parent US-BOS-SW1			US-BOS-SW1 do not have standby HSRP
14	US-BOS-SW2	HSRP Pair failover check_parent US-BOS-SW2			US-BOS-SW2 do not have standby HSRP
15	bjta002237-SW2	HSRP Pair failover check_parent bjta002237-...			bjta002237-SW2 do not have standby HSRP

8.3.6 Create the Dashboard

Let us create the intent dashboard from the ADT. Open the **New Intent Dashboard** from the intent HSRP Failover column  menu.



The screenshot shows the **Automation Data Table Manager** interface and the **Create Intent Dashboard** dialog. The dialog is open for the ADT 'HSRP Pair failover'.

Create Intent Dashboard for ADT 'HSRP Pair failover'

Name: (3)

Location: My Dashboards (4)

☐ Use Template

Data Source: Automation Data Table

Automation Data Table: HSRP Pair failover

☒ Include Triggered Follow-up Intent Results

Intent Column: HSRP Pair failover check

☐ Filter Intent by Devices

Time Range: Last 7 Days

Cancel Create (5)

The background shows the **Automation Data Table Manager** with the table **HSRP Pair failover** (19 Rows, 4 Columns). The **Table Builder** button is visible. The **HSRP Pair failover check** column (1) is selected, and the context menu is open, showing the **New Intent Dashboard** option (2).

In the **Create Intent Dashboard** dialog, **Open Intent Dashboard** to view the results summary and result history.

Create Intent Dashboard

✓

Intent dashboard has been created.

You may open to view the dashboard, or choose to add it to a summary dashboard.

[Add to Summary Dashboard](#)

[Open Intent Dashboard](#)

HSRP Pair failover

Last Refreshed at 2/8/2024, 2:58:00 pm

Summary

2/8/2024, 2:58:05 pm

[View Report](#)

33

Intents

118

Times Executed

61

Intent-level Alerts

Device Information

2/8/2024, 2:58:05 pm

[View Report](#)

19

Devices

Cisco IOS Switch

Cisco Router

Intent Result History

2/8/2024, 2:58:05 pm

[View Report](#)

Time Range: All

Result: All

Sum of Intent Alert Status Code Count

Sum of Intent Success Status Code Count

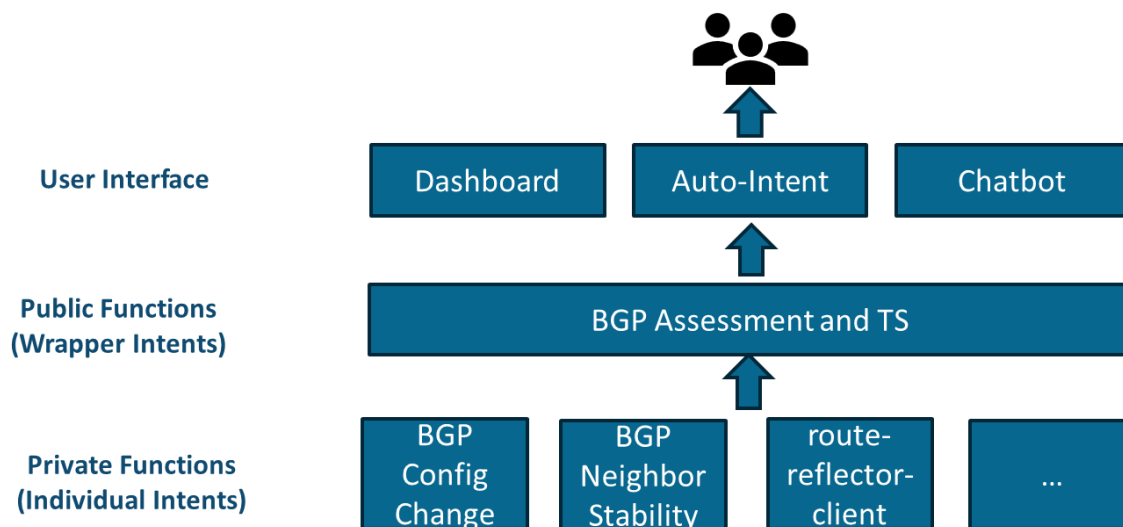
Top Five Intent Alerts

Intent Name	Map	Execution Time	Intent Alert Status Code Co...	Intent Success Status Code ...	Intent Status Code Summary	Intent Alert Detection
HSRP Pair failover check_p...	View Map	31/7/2024, 6:57:07 pm	8	0	PE-3600X-02 do not have st...	1
HSRP Pair failover check_p...	View Map	2/8/2024, 2:47:06 pm	8	0	PE-3600X-02 do not have st...	1

9 Collaborative Troubleshooting

In Chapter 4, you learned how an end user can troubleshoot a network problem with the auto intent and Chabot. You learned how to use the follow up intent to encapsulate a set of intents into a wrapper intent, which hides the detailed intent implementation from the end user and can function as a public interface to the end user, which will be accessed as an auto intent or as a chatbot.

In this chapter, we will dive deeper into the troubleshooting. We will use the BGP as an example to illustrate how to apply intent-based automation to collaborative troubleshooting. You will create and include all BGP-related intents, such as checking BGP config change, BGP neighbor stability, and route-reflector-client, into a wrapper intent. They will be exposed to the end user interface, such as Dashboard, Auto-intent, and Chatbot.



The following are covered in this chapter:

1. [Document BGP Devices and Configurations](#)
2. [Create a wrapper intent for TS BGP](#)
3. [Add more BGP troubleshooting intents](#)
4. [Create a chatbot for the BGP TS wrapper intent](#)

9.1 Document BGP Devices and Configurations

In this section, you will create an ADT with the base table to include network BGP devices, AS number, Router id, and BGP configurations. This ADT base table can be used as the base to replicate any BGP-related intent and the foundation of troubleshooting the BGP issues. The ADT will be created by the pre-replicated intent with the following steps:

1. [Create a dynamic device group to include all network devices.](#)
2. [Create a seed intent to parse BGP AS numbers and configurations.](#)
3. [Define the replication settings and install/decode it.](#)
4. [Create an ADT, building the base table with built-in device properties and signature variables \(As number and BGP configurations\).](#)

9.1.1 Create BGP Device Group

Create a device group (BGP Devices) with the dynamic criteria using **Device Property** (BGP Enabling) and **Vendor** contains **Cisco**.

1. Click **Select Criteria > Device Property** and select **BGP Enabling** from the dropdown.
2. Add the condition **True**.
3. Click **Search**.
4. Click **OK** to save and close the dialog.

Dynamic Search Device

Search Scope: All Devices

Device Criteria:

A Select Criteria is True

Boolean

Search


Hostname	Vendor	Model	Management IP
BUR-R206-Forti200F-1	Fortinet	200F	192.168.0.100
BUR-R206-Forti200F-1_(Cloud_...	Fortinet	200F	192.168.0.100
BUR-R206-Forti200F-1_(Demo-...	Fortinet	200F	192.168.0.100
Berlin-vEdge	Cisco	WS-C4500X-32	192.168.0.1
Bur-Netbond(Primary)(East-RG...	Microsoft	Azure MSEE	

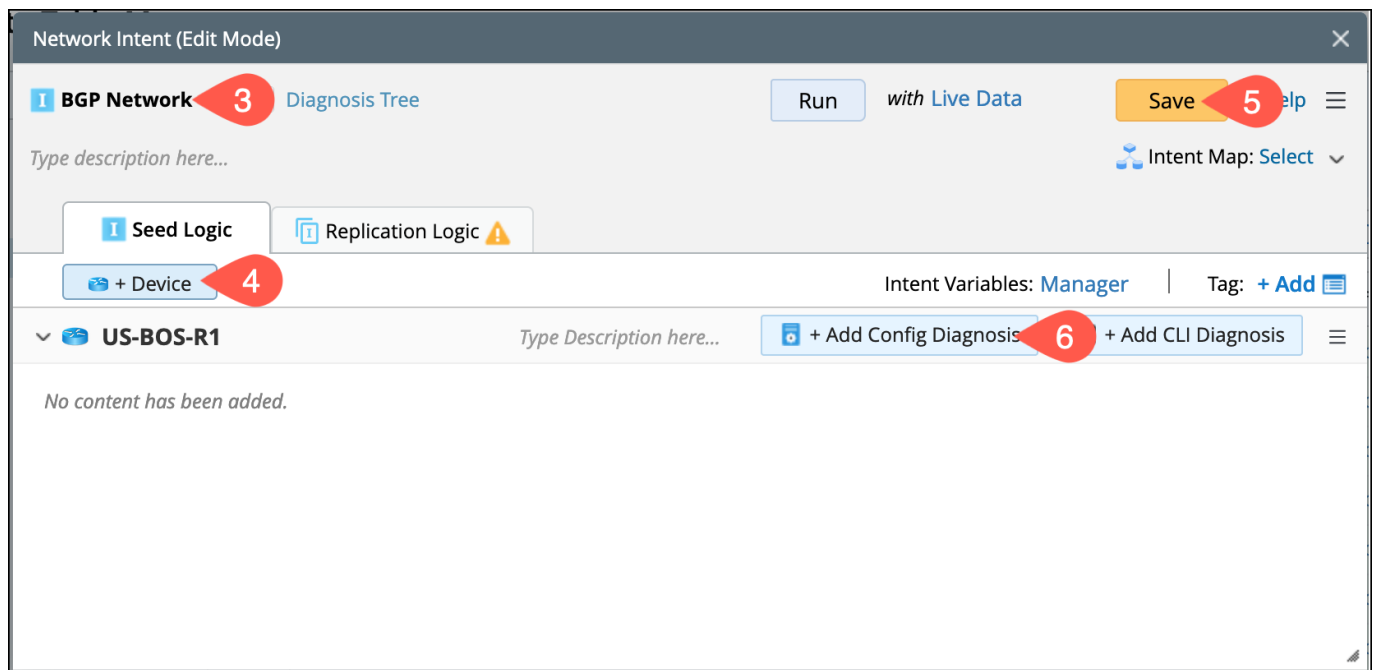
Cancel OK

9.1.2 Create an intent to parse BGP AS numbers and configuration

9.1.2.1 Select a device

Define the basic information of the intent, such as name, description, and device and save it to the intent manager as follows:

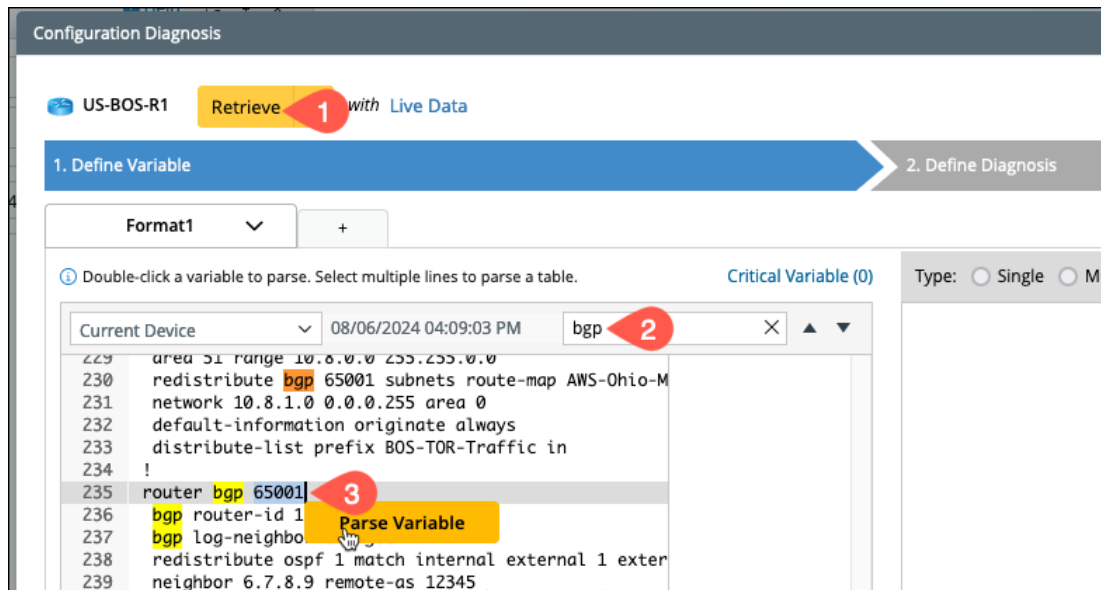
1. Click  icon from your desktop > **New Intent**.
2. A window **Network Intent** in edit mode will appear.
3. Enter the title and a brief description (optional) of the intent.
4. Add the seed devices by selecting the option **+Device**. You should select a **Cisco** device with BGP configured.
5. Click **Save** to save the intent to the Intent Manager.
6. Click **+Add CLI Diagnosis** to open the corresponding window and proceed to the next section to parse the variables and define the diagnosis.



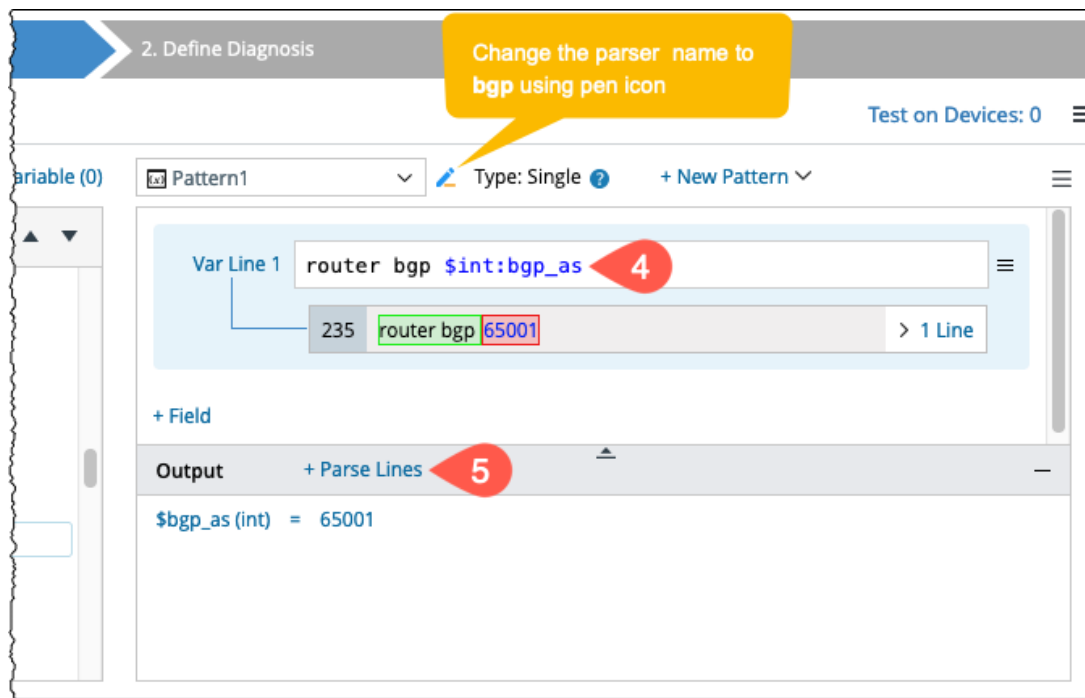
9.1.2.2 Retrieve Data and Parse Variables

1. Click **Retrieve** to collect the data from Live Data.
2. Search for bgp data using keywords in the search bar.
3. Select the bgp text **65001** and click **Parse Variable** in the tip window

Note: You can also double-click the text to get the same result.



4. In the right pane **Var Line 1** field, update the variable name to `router bgp $int:bgp_as`.



5. To parse the router id, select text **10.10.10.10** and click **Parse Variable** in the tip window.
6. Parsing BGP configuration: To parse the multiple lines into one variable, use the function **LineByVariables**:
 - a) Click **+ Parse Lines**.
 - b) Enter Variable name **bgp_config**.
 - c) Check **The line between** the radio button and parse the lines between **\$bgp_as** to **End**.
 - d) Review the **Pattern** `LinesByVariable[$bgp_config]:$bgp_as-` and click **Apply**.

Parse Lines

Name: **b**

☐ The line of variable:

☒ The line between: to **c**

☐ The line contains keyword:


Output:

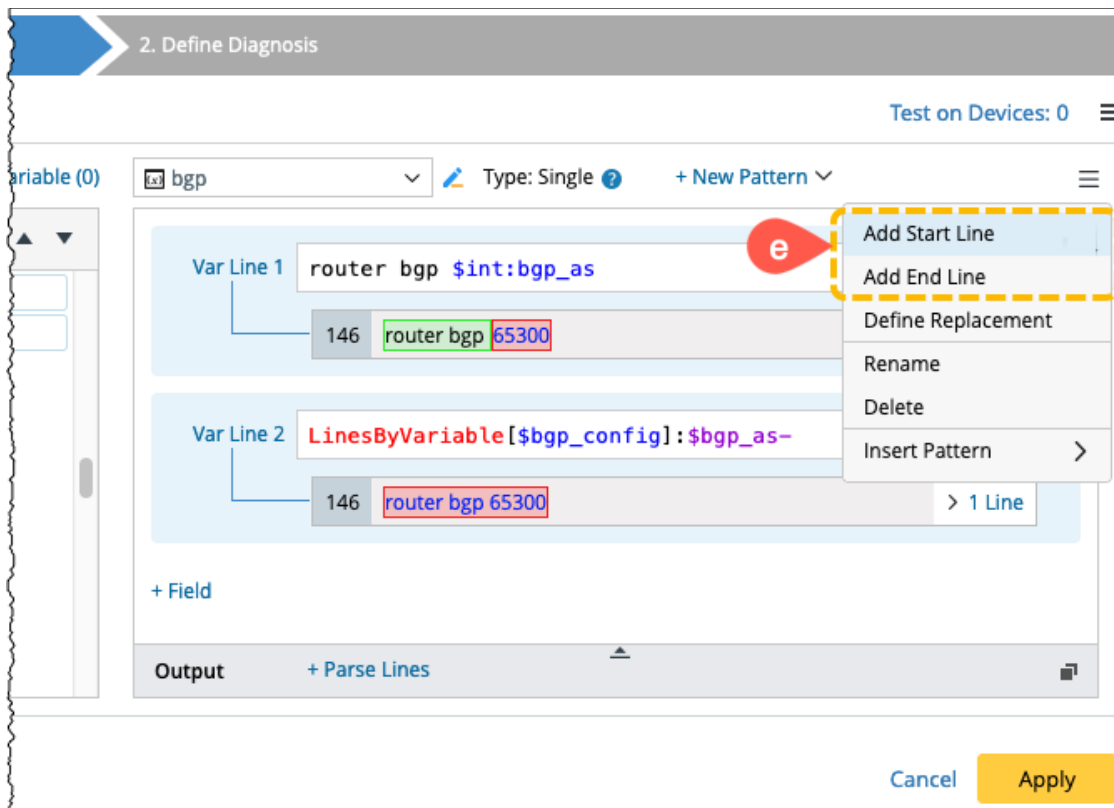
```
$bgp_config=
router bgp 65300
  bgp router-id 192.168.20.254
  bgp log-neighbor-changes
  redistribute eigrp 1 route-map E2B
  neighbor 100.30.0.1 remote-as 10000
  !
ip forward-protocol nd
```

Pattern: `LinesByVariable[$bgp_config]:$bgp_as-`

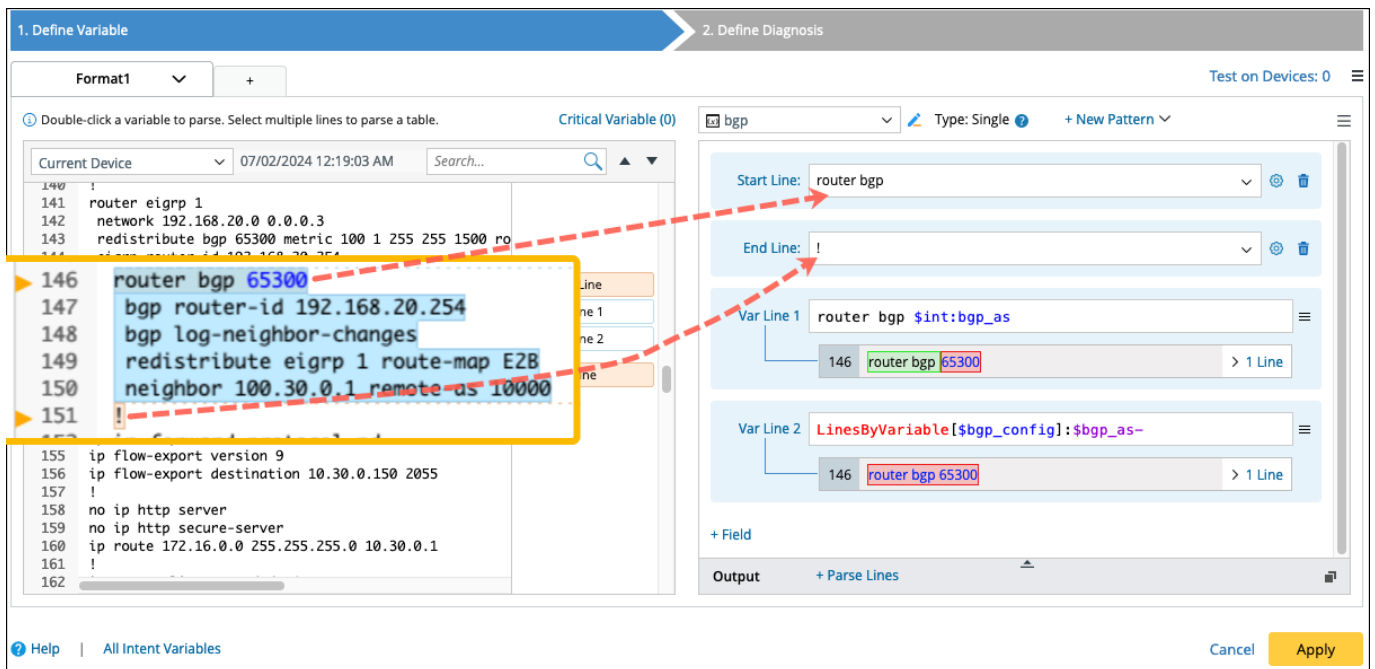
Pattern appears from step b and step c

d

- e) Back in the Configuration Diagnosis window, from the menu  add **Start Line** and **End Line** to limit the parsing lines to the BGP configuration data.



- f) Enter the starting line and end line text into these two fields: **Start Line** and **End Line**.



7. Parsed variables will be listed under the **Output** section in the right pane. Validate the variables.
8. Click **Apply** and proceed to the **Define Diagnosis** section. We will not define any diagnosis and status code for this intent since we only export the variables to the ADT base table.

2. Define Diagnosis

Test on Devices: 0

ble (0) Pattern1 Type: Single + New Pattern

Start Line: router bgp

End Line: !

Var Line 1: router bgp \$int:bgp_as
235 router bgp 65001 > 1 Line

Var Line 2: bgp router-id \$router_id
236 bgp router-id 10.10.10.10 > 1 Line

Var Line 3: LinesByVariable[\$bgp_config]:\$bgp_as-
235 router bgp 65001 > 1 Line

+ Field

Output + Parse Lines

\$bgp_as (int) = 65001

\$router_id (string) = 10.10.10.10


\$bgp_config (string) = router bgp 65001 bgp router-id 10.10.10.10 bgp log-neighbor-changes r...

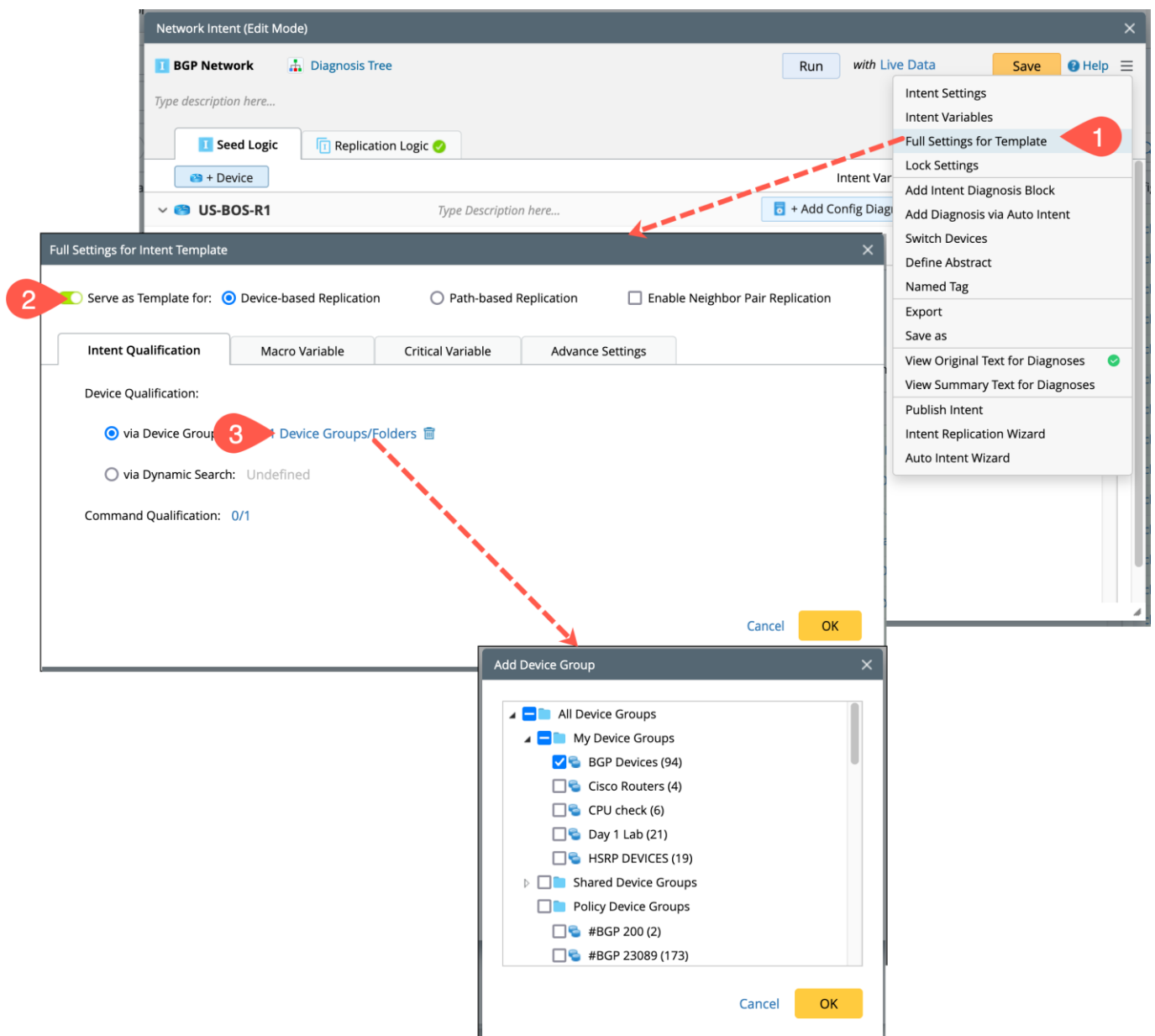
Cancel 8 Apply

9. Exit the **Visual Parser** window.

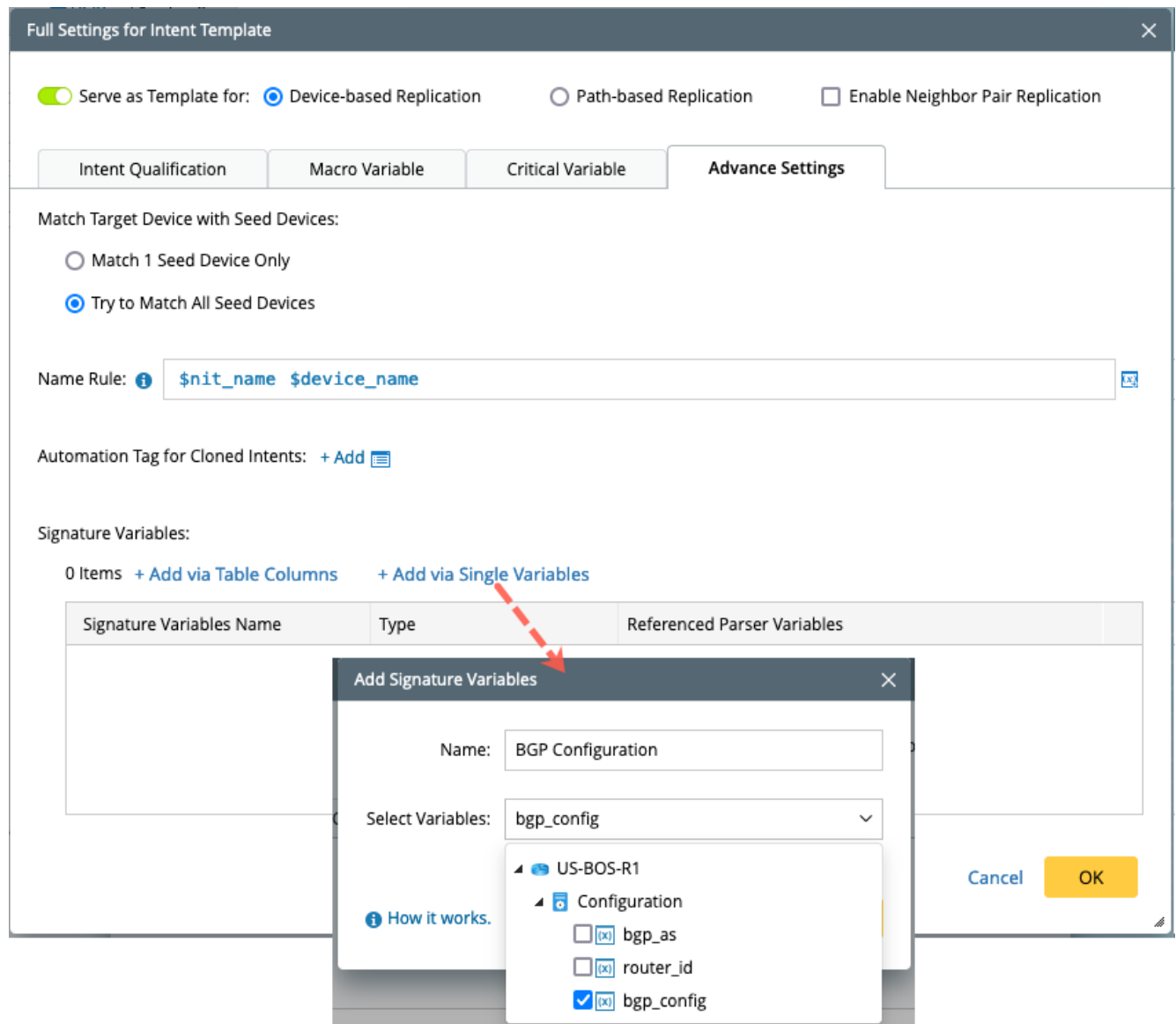
9.1.3 Define the replicate settings and decode

Replicate the settings to all the BGP devices using the device group created in Section 9.1.1. And define the signature variables to add them as columns in ADT:

1. From the menu , select **Full Settings for Template**.
2. Enable the field **Serve as Template** for **Deved-based Replication**.
3. In the **Intent Qualification** tab, select the device group created in earlier Section 9.1.1.



- Go to the **Advanced Settings** tab and click **+Add via Single Variables** to open the corresponding window.
- Enter a name for the variable and select the variable from the dropdown menu. Add three signature variables: **BGP Configuration**, **BGP ASN**, and **Router ID**.



6. Add all the variables and click **OK**.
7. Close the **Full settings for Intent Template** window.

Full Settings for Intent Template

☒ Serve as Template for: ☒ Device-based Replication ☐ Path-based Replication ☐ Enable Neighbor Pair Replication

Intent Qualification Macro Variable Critical Variable **Advance Settings**

Match Target Device with Seed Devices:

☐ Match 1 Seed Device Only
☒ Try to Match All Seed Devices

Name Rule:

Automation Tag for Cloned Intents: [+ Add](#)

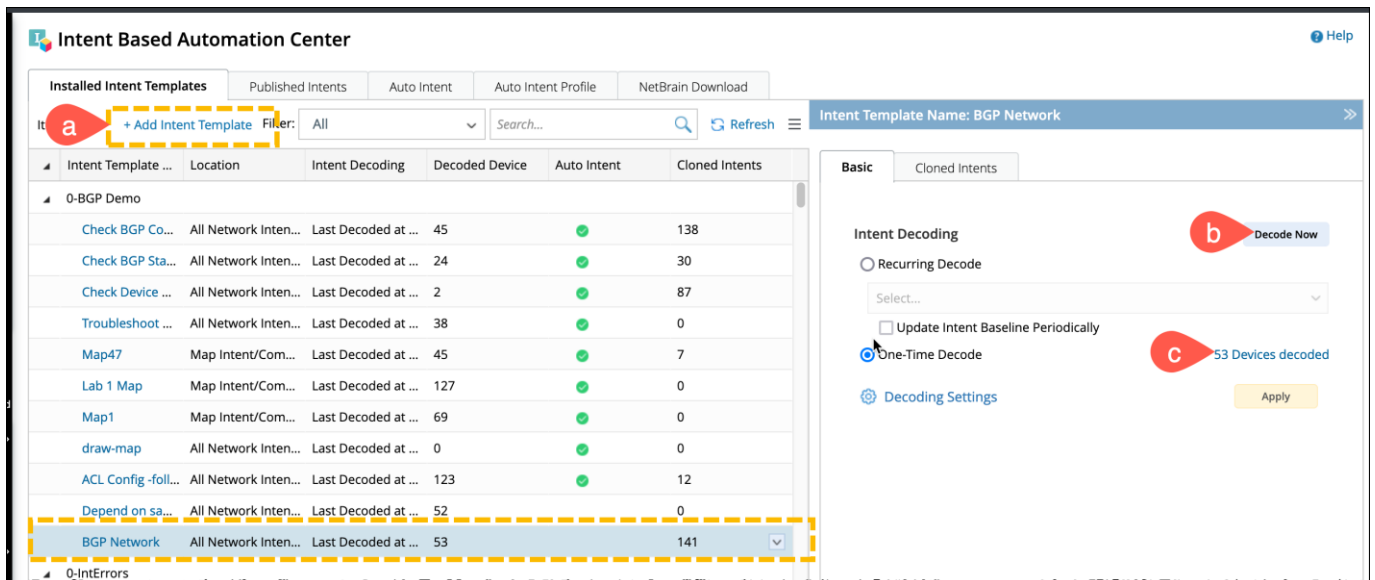
Signature Variables:

3 Items [+ Add via Table Columns](#) [+ Add via Single Variables](#)

Signature Variables Name	Type	Referenced Parser Variables
BGP ASN	Single Variables	1 Variable
Router ID	Single Variables	1 Variable
BGP Configuration	Single Variables	1 Variable

[Cancel](#) **OK**

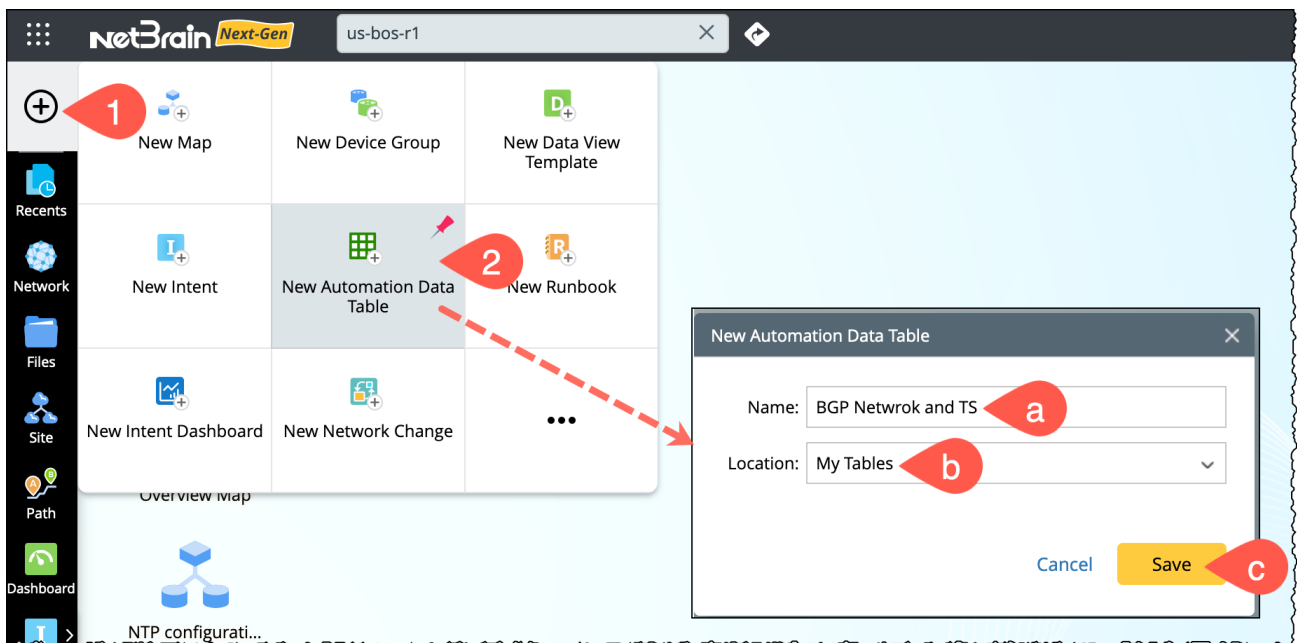
8. Install and decode the intent in the **Intent Based Automation Center**.
 - a) Open the **Intent Based Automation Center** and click **+Add an Intent Template**. Select the intent you just saved.
 - b) Click **Decode Now** to decode the intents.
 - c) Wait for the system to finish replicating and decoding intents. You can see the number of devices already decoded. Click it to view the execution log.



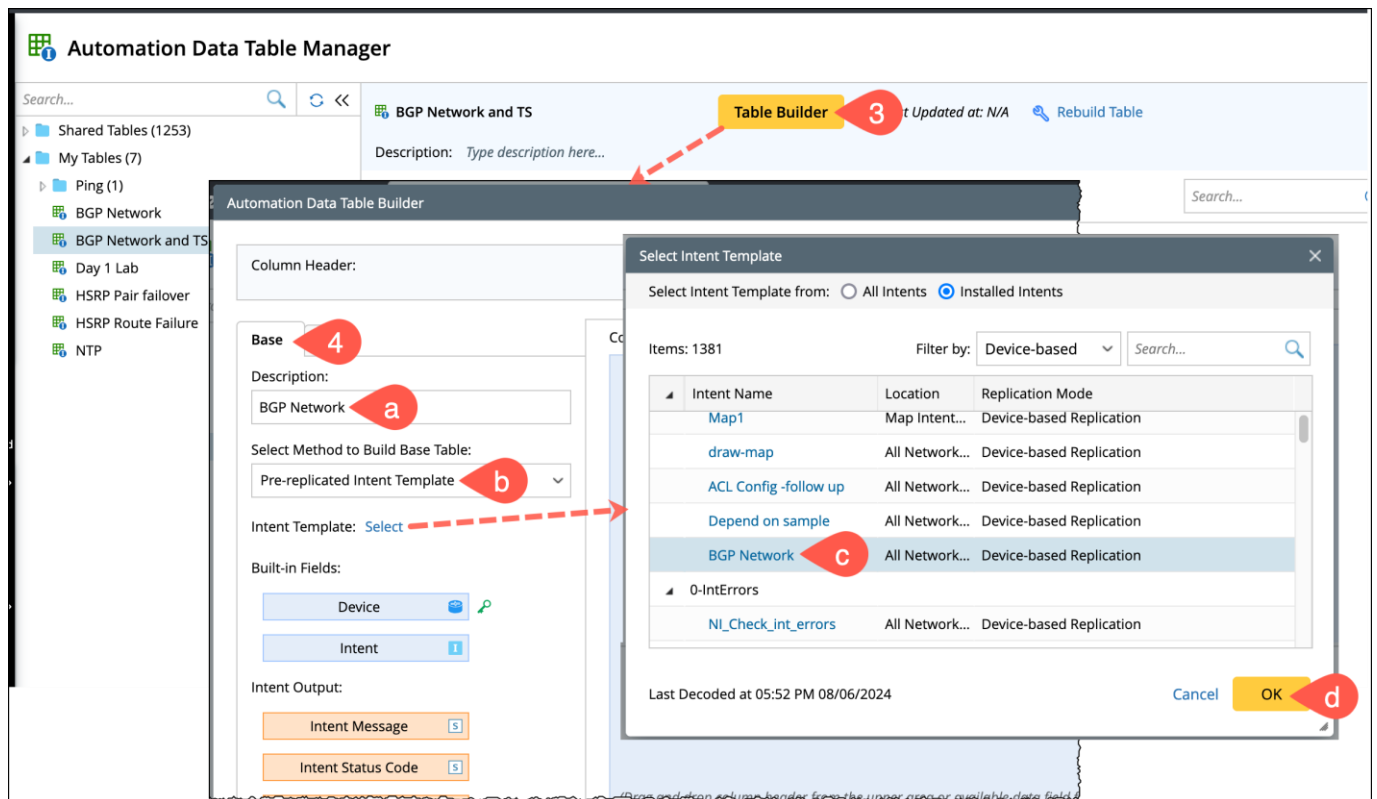
9.1.4 Build Base ADT

Build an ADT with the base table containing the built-in device properties and signature variables (*AS number* and *BGP configurations*) as columns:

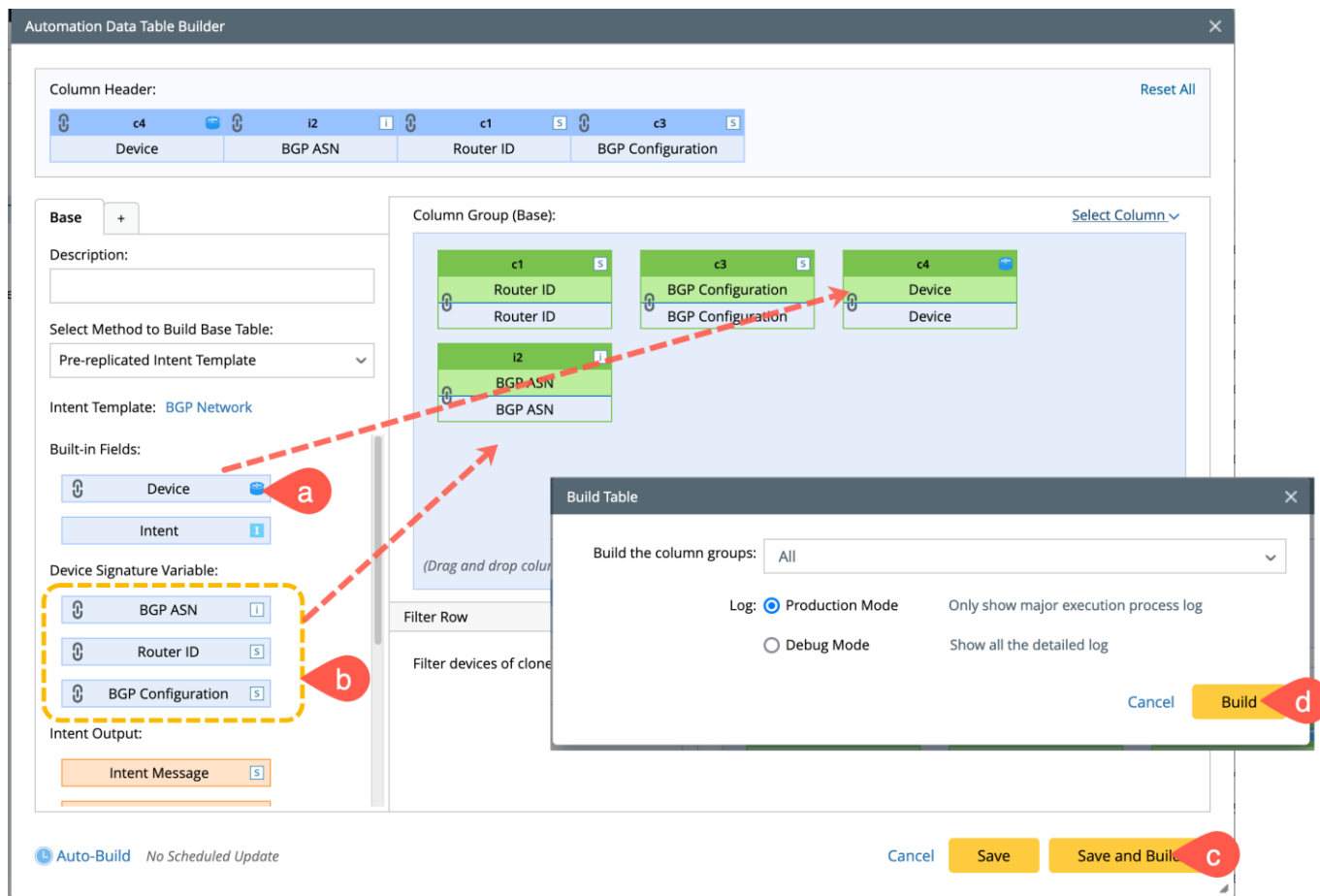
- From the desktop, click the icon > **New Automation Data Table**.
- In the **New Automation Data Table** popup, enter the name and select a folder to store the table.



3. Click **Table Builder** to open ADT Builder and define the Base Group.
4. Base Group tab settings: Under the **Base** tab, define the following settings:
 - a) Input **Description** for the base table to describe its use and function.
 - b) Select the method **Pre-replicated Intent Template** from the dropdown to build the base table.
 - c) Click the **Select** link to choose the intent **BGP Network** created in the previous section from **Intent Based Automation Center**.
 - d) Click **OK** to confirm the selection and close the Select Intent template window.



5. Build an ADT table with the columns **Device**, **BGP ASN**, **Router ID**, and **BGP Configuration**.
 - a) From the Built-in fields, drag and drop the **Device** field into the **Column Group (Base)** pane.
 - b) From the Device Signature Variables, drag and drop the **BGP ASN**, **Router ID**, and **BGP Configuration** into the **Column Group (Base)** pane.
 - c) Click **Save and Build** and the **Build Table** dialog appears.
 - d) Choose the settings as per your preferences and then click **Build** to save all the settings.



The final ADT output will appear with all the configured columns:

Automation Data Table Manager

Search...

Shared Tables (1253)

My Tables (7)

Ping (1)

BGP Network

BGP Network and TS

Day 1 Lab

HSRP Pair failover

HSRP Route Failure

NTP

BGP Network and TS

Description: Type description here...

Items: 53 Rows 4 Columns

Search...

Advanced Filter: Undefined

No.	Device	BGP ASN	Router ID	BGP Configuration
1	NIC-Lab-XR1	65501	10.8.76.200	router bgp 65501
2	NIC-Lab-PE2	65501	10.8.76.203	router bgp 65501
3	bjta002440-SW11	65145	10.88.192.6	router bgp 65145
4	US-SFO-R2	65001	10.8.2.250	router bgp 65001
5	US-SFO-R1	65001	10.9.9.9	router bgp 65001
6	US-BOS-R1	65001	10.10.10.10	router bgp 65001
7	bur-isp-gw2	64661		router bgp 64661
8	VRF-PE2	100		router bgp 100
9	Internet	80001	124.1.1.1	router bgp 80001

9.2 Create a wrapper intent for BGP Troubleshooting

In this section, you will create an intent to check the change of the BGP configuration and a wrapper intent with follow up intent execution and then replicate to all the devices in the BGP device group created in the previous section:

1. Create an intent to check changes in BGP configuration
2. Create a Wrapper Intent

9.2.1 Create an intent to Check BGP Configuration

9.2.1.1 Select a device and Parse variables

Create a new intent, **Check BGP Config Change**, select a BGP device, and **Add Config Diagnosis**.

The screenshot displays the 'Network Intent (Edit Mode)' window. At the top, the intent name 'Check BGP Config change' is highlighted with a red circle labeled '2'. To its right are buttons for 'Run', 'with Live Data', and 'Save' (labeled with a red circle '4'). Below the intent name is a text field for 'Type description here...'. The 'Seed Logic' tab is selected, showing a 'Replication Logic' status of '✓'. A '+ Device' button is highlighted with a red circle '3'. Below this, the device 'US-BOS-R1' is selected. To the right of the device name is a '+ Add Config Diagnosis' button (labeled with a red circle '5') and an 'Add CLI Diagnosis' button. The 'Configuration Diagnosis' section is expanded, showing a list of commands and their output for 'US-BOS-R1#show run'. The output includes configuration details such as 'Current configuration : 13636 bytes', 'Last configuration change at 21:48:46 EST Wed Jul 31 2024 by nb', 'version 15.4', and various service configurations.

Parse the bgp AS number and bgp configuration using the function **LinesByVariables**. Refer to Section 9.1.2 for details.

2. Define Diagnosis

Test on Devices: 0

(0) bgp Type: Single + New Pattern

Start Line: router bgp

End Line: !

Var Line 1 router bgp \$int:bgp_as
146 router bgp 65300 > 1 Line

Var Line 2 LinesByVariable[\$bgp_config]:\$bgp_as-
146 router bgp 65300 > 1 Line

+ Field

Output + Parse Lines

\$bgp_as (int) = 65300

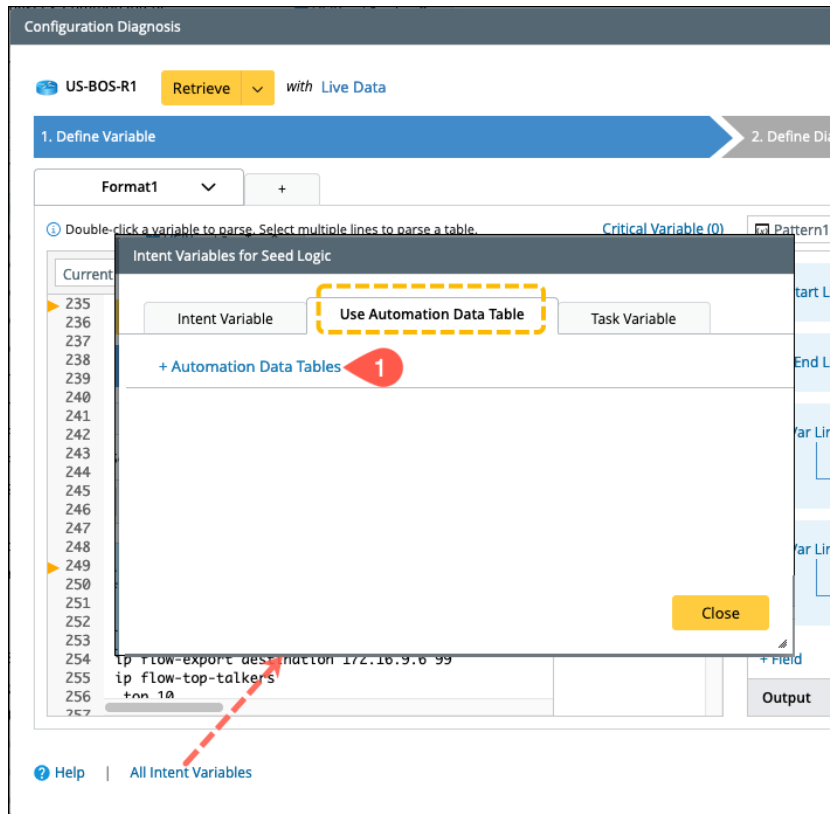
\$bgp_config (string) = router bgp 65300 bgp router-id 192.168.20.254 bgp log-neighbor-change...

Car 7 Apply

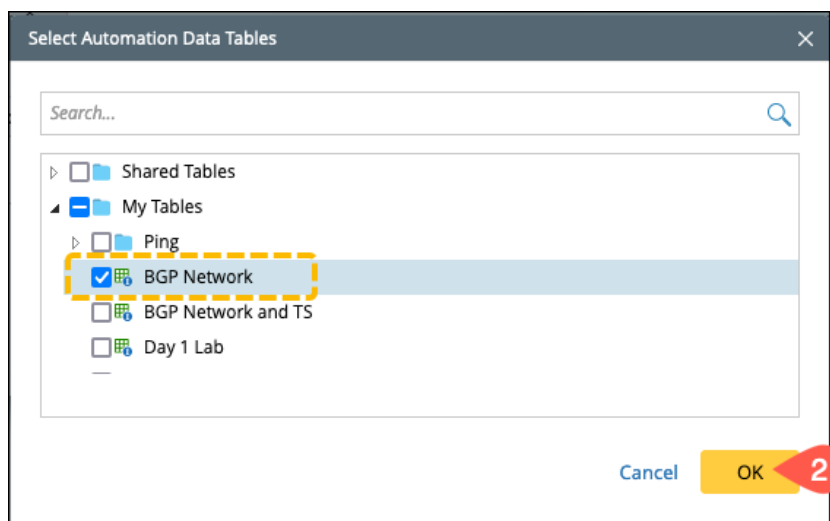
9.2.1.2 Add ADT Table as Intent Variable

In the diagnosis, you will compare the parsed **bgp configuration** against the configuration defined in the ADT. In order to refer to the ADT elements, you need to add the ADT table as an **Intent Variable**:

1. Go to **All Intent Variables > Use Automation Data Table > + Automation Data Tables** to select ADT.



2. In the **Select Automation Data Table** window, select the **BGP Network** ADT and then **OK**.



9.2.1.3 Define Diagnosis

In the diagnosis, compare the configurations against the configuration defined in the ADT (**BGP Network**) using two functions:

- **Match Pattern (MP)**: To compare two strings line by line and report the differences, including the changed, added and removed lines.
- **Get_Table_Cell**: To retrieve a cell value from an ADT table.

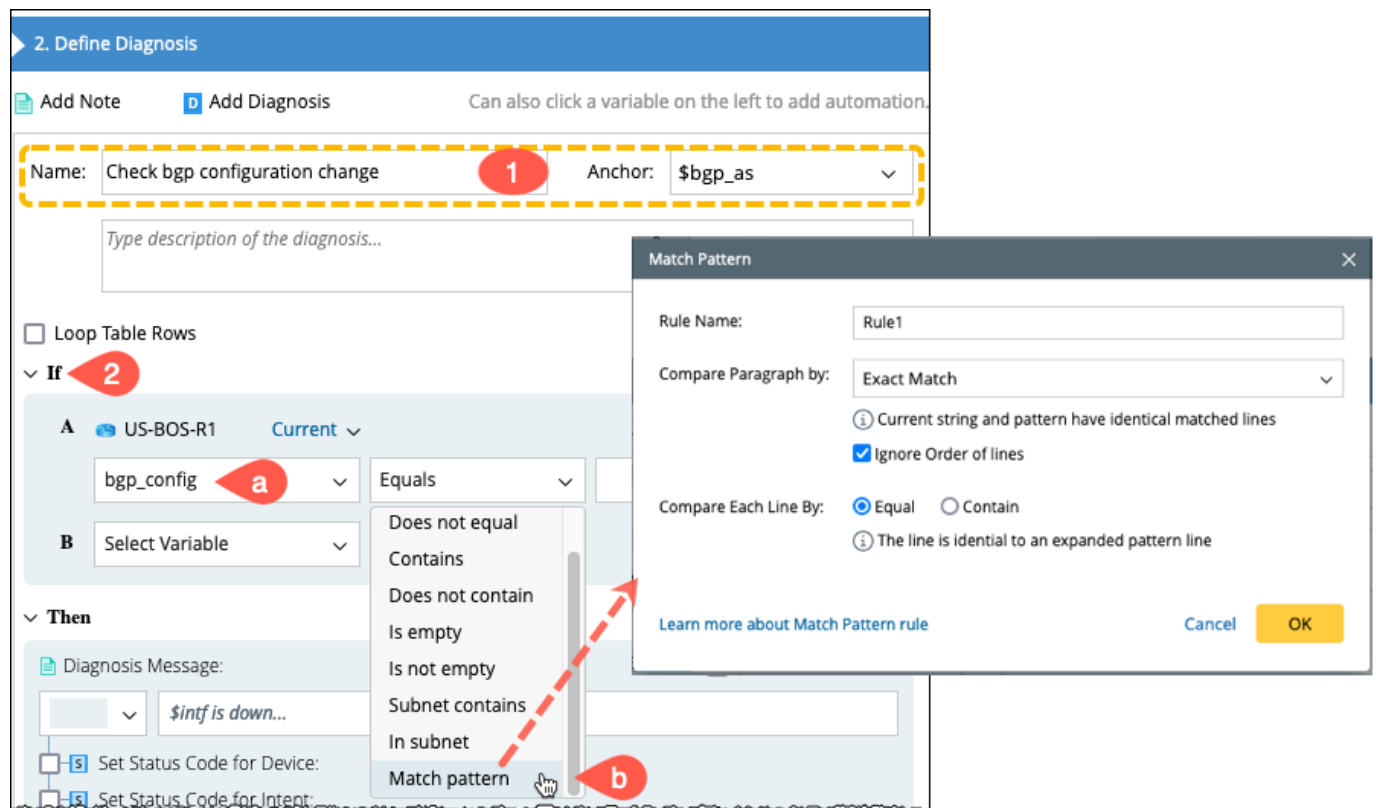
1. Define basic information such as name and anchor.

2. Define the **If** condition **A**:

a) Select Variable **bgp_config**.

b) Select **Match Pattern** from the dropdown, validate the default settings and then click **OK**.

NOTE: Change the default rule name if you have multiple match pattern rules in one diagnosis.



- c) To retrieve the configuration from the ADT table, select **Expression** > **+ Get_Table_Cell** function.

2. Define Diagnosis

Add Note **D Add Diagnosis** Can also click a variable on the left to add automation.

Name: Anchor:

Type description of the diagnosis...

☐ Loop Table Rows

▼ If

A US-BOS-R1 **Current** ▼

B

▼ Then

Diagnosis Message:

☐ Set Status Code for Device:

☐ Set Status Code for Intent:

Add Logic ▼

US-BOS-R1

- Configuration**
 - bgp_as
 - bgp_config
- Rule1**
 - Result
 - Matched_li

Expression

Define Variable

Expression

Expression:

+ Variable + Function + Get_Table_Cell **C**

Help **Cancel** **OK**

d) In the **Get Table Cell** window, select the ADT table, column, and condition to retrieve a table cell:

- i. Select the table **BGP_Network**.
- ii. Select the Column **BGP_Configuration**.
- iii. Define Row Matching Condition A: Device | Equals | this_device.

The Expression will be: *Get_Table_Cell (BGP_Network, BGP_Configuration, <Condition: A>)*

3. **Then:** In case **If** logic is true, define the color (**green**), status (**Success**), and:

Message: *BGP configuration did not change.*

4. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.

5. **Else:** In case **If** logic is not true, define the color (**red**), status (**Error**) and:
Message: *BGP configuration changed. Missing lines: `$Rule1.Unmatched_lines`. Extra lines: `$Rule1.Unused_pattern_lines`.*
6. Check the selection boxes of the **Set Status Code for Device** and **Set Status Code for Intent** to duplicate the message to the Device Status Code.
7. Click **Apply** to save the settings and then close the window.

2. Define Diagnosis

Add Note **D** Add Diagnosis Can also click a variable on the left to add automation.

A JP-TYO-CR... **Current** ▼

bgp_config ▼ MP(Rule1) ▼ Get_Table_Cell(BG ▼ 🗑️

B Select Variable ▼

Then

Diagnosis Message: ☐ Save to Incident ☰

🟢 ▼ BGP configuration did not change

☒ **S** Set Status Code for Device:

🟢 Success ▼ BGP configuration did not change

☒ **S** Set Status Code for Intent:

🟢 Success ▼ BGP configuration did not change

Else 🗑️ Delete

Diagnosis Message: ☐ Save to Incident ☰

🔴 ▼ BGP configuration changes. Missing lines: `$Rule1.Unmatched_lines`. Extra lines: `$R`

☒ **S** Set Status Code for Device:

🔴 Error ▼ BGP configuration changes. Missing lines: `$Rule1.Unmatched_lines`. Extra lin

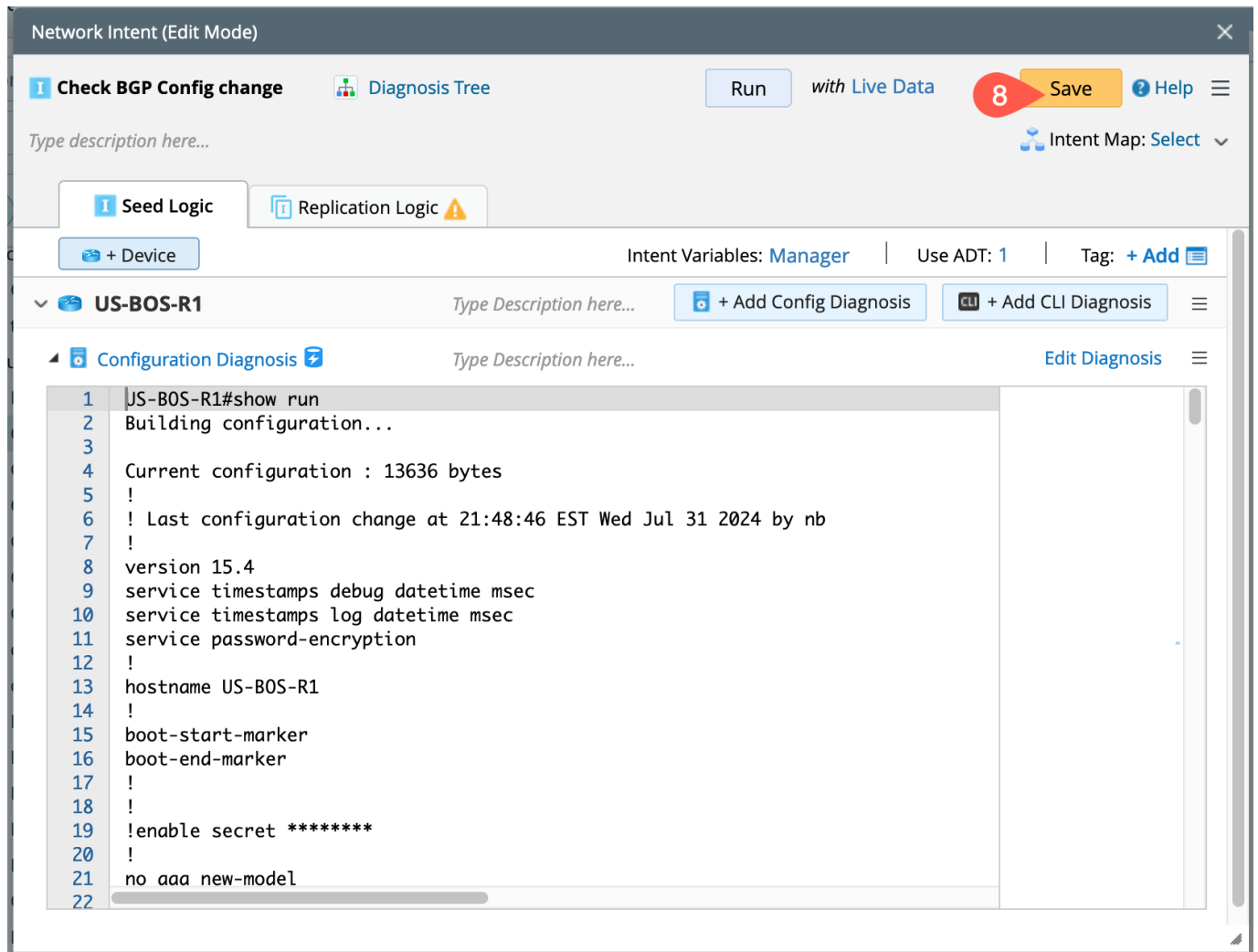
☒ **S** Set Status Code for Intent:

🔴 Error ▼ BGP configuration changes. Missing lines: `$Rule1.Unmatched_lines`. Extra lin

+ Add Elself

Can **7** Apply


8. In the Network Intent (Edit Mode) dialog, click **Save**.

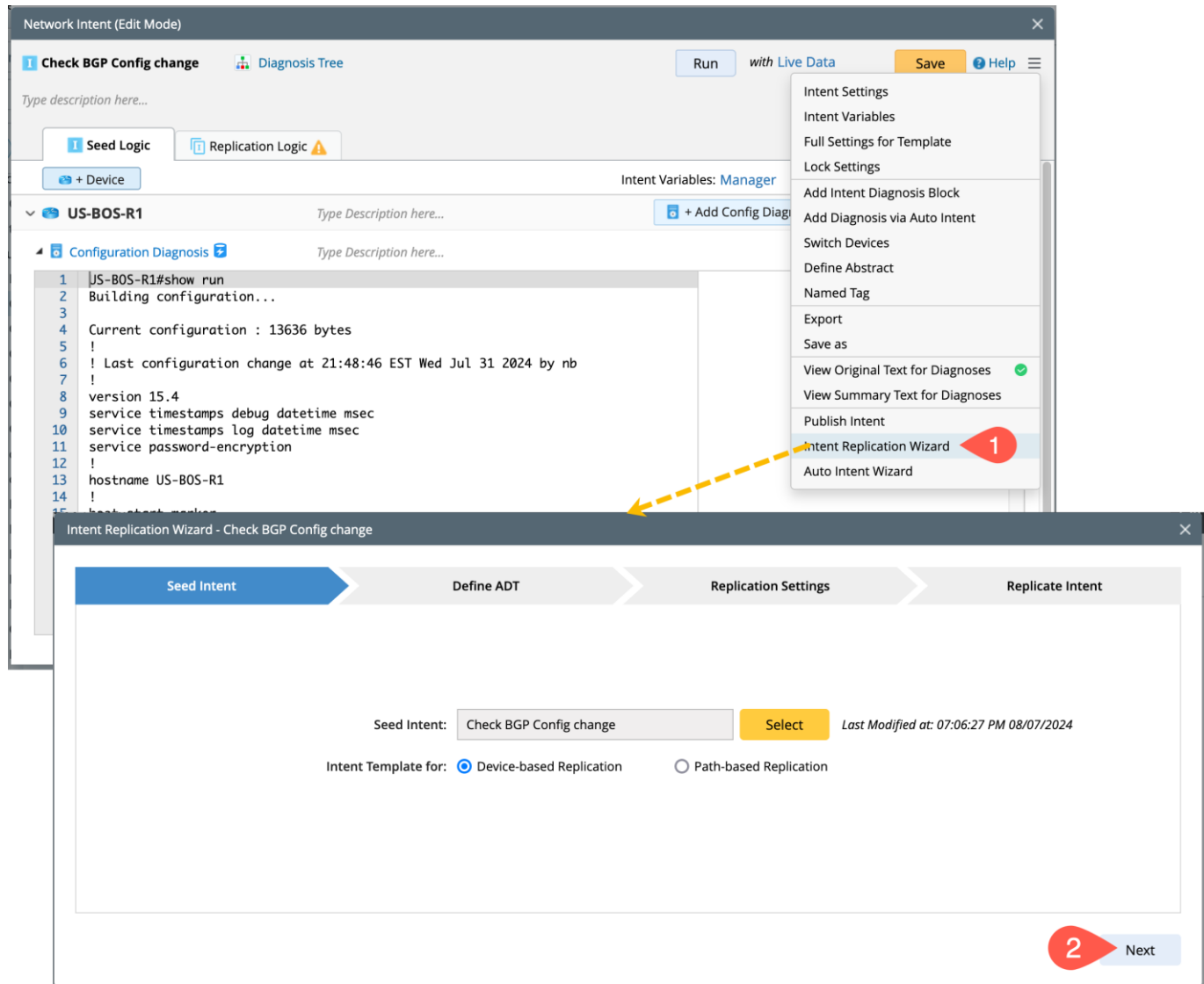


9. Without closing the window, replicate the intent to the ADT as detailed in the next section.
10. Run the intent to test its correctness. Often, your BGP configuration did not change. You can modify the ADT table to emulate a changed BGP to test your intent. The following is an example result.

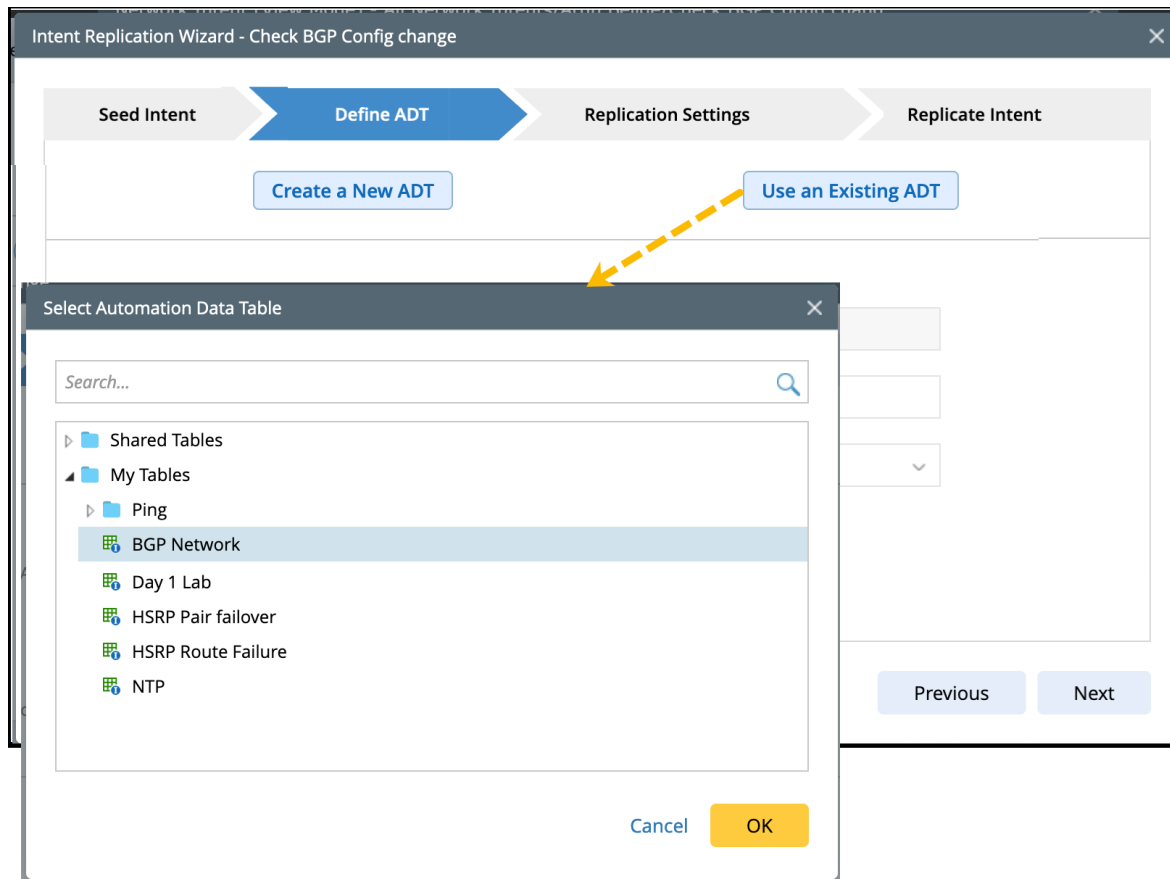
9.2.1.4 Replicate the Intent to the existing ADT

Use the **Intent Replication Wizard** to replicate the **Check BGP config change** intent to the ADT table BGP_Network created in Section 9.1.1.

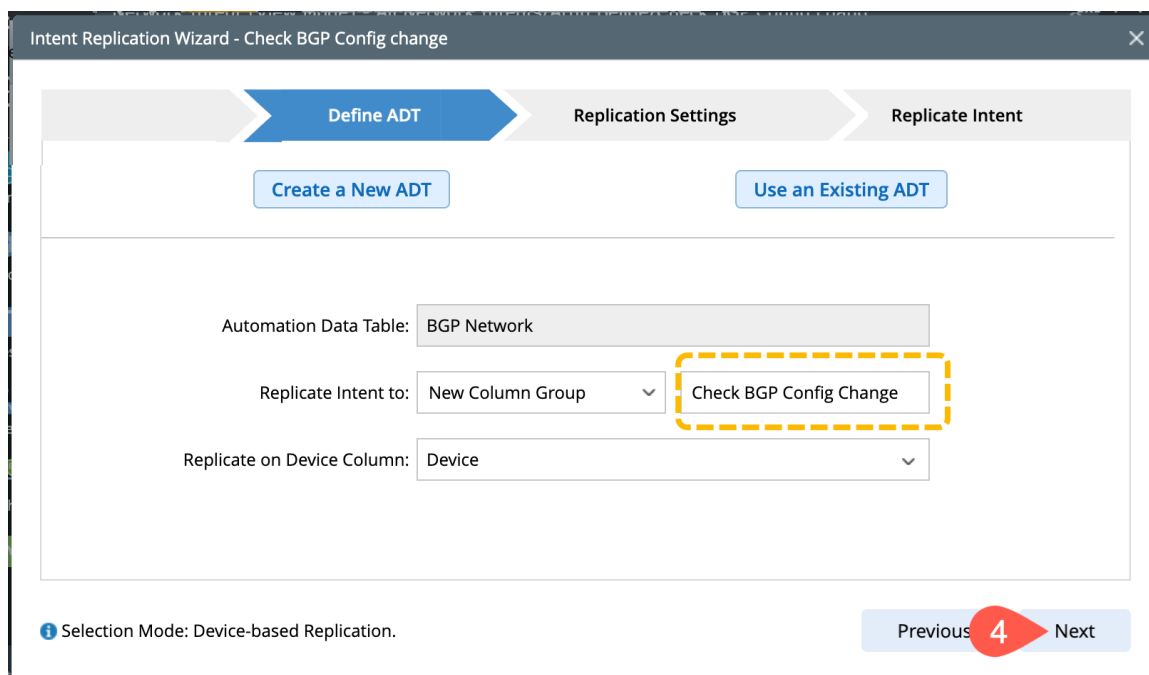
1. Launch the **Intent Replication Wizard** from the  menu.
2. The seed intent will appear by default as you have launched it from the **Network Intent** window. Click **Next**.



3. In the **Define ADT** ribbon, go to **Use an Existing ADT > BGP Network ADT >OK**.



4. Replicate Intent to: select **New Column Group** and enter the name **Check BGP Config Change** and click **Next**.



5. Add the BGP device group created in Section 9.1.4 and click **Next**.

Intent Replication Wizard - HSRP Pair failover check_parent

Seed Intent Define ADT **Replication Settings** Replicate Intent

Full Settings for Template

Intent Qualification: ☒ via Device Groups/Sites: 1 Device Groups/Folders ☐ via Dynamic Search: Undefined

Define Macro Variables and Rules for Their Substitution:

Item: 1

Seed Device	Seed Command	Macro Variables
US-BOS-SW1	Configuration	

Please click to select an entry from the above table.

More Replication Settings

Selection Mode: Device-based Replication, ADT: BGP Network.

Previous **5** Next

6. In the **Replication Settings**, modify the replicated intent **Column Name** and add additional columns such as **Intent Status Code** and **Device Status Code**.

Intent Replication Wizard - Check BGP Config change

Seed Intent Define ADT Replication Settings **Replicate Intent**

ADT Columns:

Column Data	Column Name	Tag
Replicated Intent	Check BGP config change	0 tags

Additional Columns

- ☒ Replicated Intent
- ☐ Intent Message
- ☐ Intent Status Code
- ☐ Device Status Code
- ☐ Intent Devices
- ☐ Intent Map
- ☐ Intent CLI Comma...
- ☐ Last Execution Time

Add columns to the table as needed

7. After selecting **Save and Replicate** to save all the settings and create ADT.

Intent Replication Wizard - Check BGP Config change

Seed Intent > Define ADT > Replication Settings > **Replicate Intent**

ADT Columns: Additional Columns v

Column Data	Column Name	Tag
I Replicated Intent	Check BGP config change	0 tags
S Intent Status Code	Intent Status Code	
S Device Status Code	Device Status Code	

7 **Save and Replicate**

Replication Request submitted at: 08/08/2024 01:06 PM 8 **Open Output ADT**

i Selection Mode: Device-based Replication, ADT: BGP Network, 0 Macro Variables. Previous Finish

The table will now be populated with devices and the replicated Intents (Check BGP config Change).

Automation Data Table Manager

BGP Network Table Builder Last Updated at: 08/08/2024 01:39 PM Rebuild Table Add Data Manually

Description: *Type description here...*

Items: 53 Rows 7 Columns

Replicated Intent

No.	Device	I BGP ASN	S Router ID	S BGP Configuration	I Check BGP config change
1	US-SFO-R1	65001	10.9.9.9	router bgp 65001	Check BGP Config change US-SF...
2	US-BOS-R1	65001	10.10.10.10	router bgp 65001	Check BGP Config change US-BO...
3	US-SFO-R2	65001	10.8.2.250	router bgp 65001	Check BGP Config change US-SF...
4	NIC-Lab-PE2	65501	10.8.76.203	router bgp 65501	Check BGP Config change NIC-La...
5	ip-172-26-0-114	64513		router bgp 64513	Check BGP Config change ip-172...
6	PE-3600X-01	64550	10.88.255.1	router bgp 64550	Check BGP Config change PE-36...
7	VRF-PE2	100		router bgp 100	Check BGP Config change VRF-PE2
8	bjta002440-SW11	65145	10.88.192.6	router bgp 65145	Check BGP Config change bjta00...

9.2.1.5 Execute and view Results

Run the replicated intent in the ADT once and rebuild the table. Review the intent result in the column **Intent Status Code**.

Automation Data Table Manager

BGP Network Table Builder Last Updated at: 08/08/2024 01:39 PM Rebuild Table

Description: Type description here...

Items: 53 Rows 7 Columns Search...

No.	Device	BGP ASN	Router ID	BGP Configuration	Check B	Run	Details	Intent Status Code
1	US-SFO-R1	65001	10.9.9.9	router bgp 65001	Check BGP Config change US-SFO...			
2	US-BOS-R1	65001	10.10.10.10	router bgp 65001	Check BGP Config change US-BO...			
3	US-SFO-R2	65001	10.8.2.250	router bgp 65001	Check BGP Config change US-SF...			
4	NIC-Lab-PE2	65501	10.8.76.203	router bgp 65501	Check BGP Config change NIC-La...			
5	ip-172-26-0-114	64513		router bgp 64513	Check BGP Config change ip-172...			
6	PE-3600X-01	64550	10.88.255.1	router bgp 64550	Check BGP Config change PE-36...			
7	VRF-PE2	100		router bgp 100	Check BGP Config change VRF-PE2			

Automation Data Table Manager

BGP Network Table Builder Last Updated at: 08/08/2024 02:12 PM Rebuild Add Data Manually

Description: Type description here...

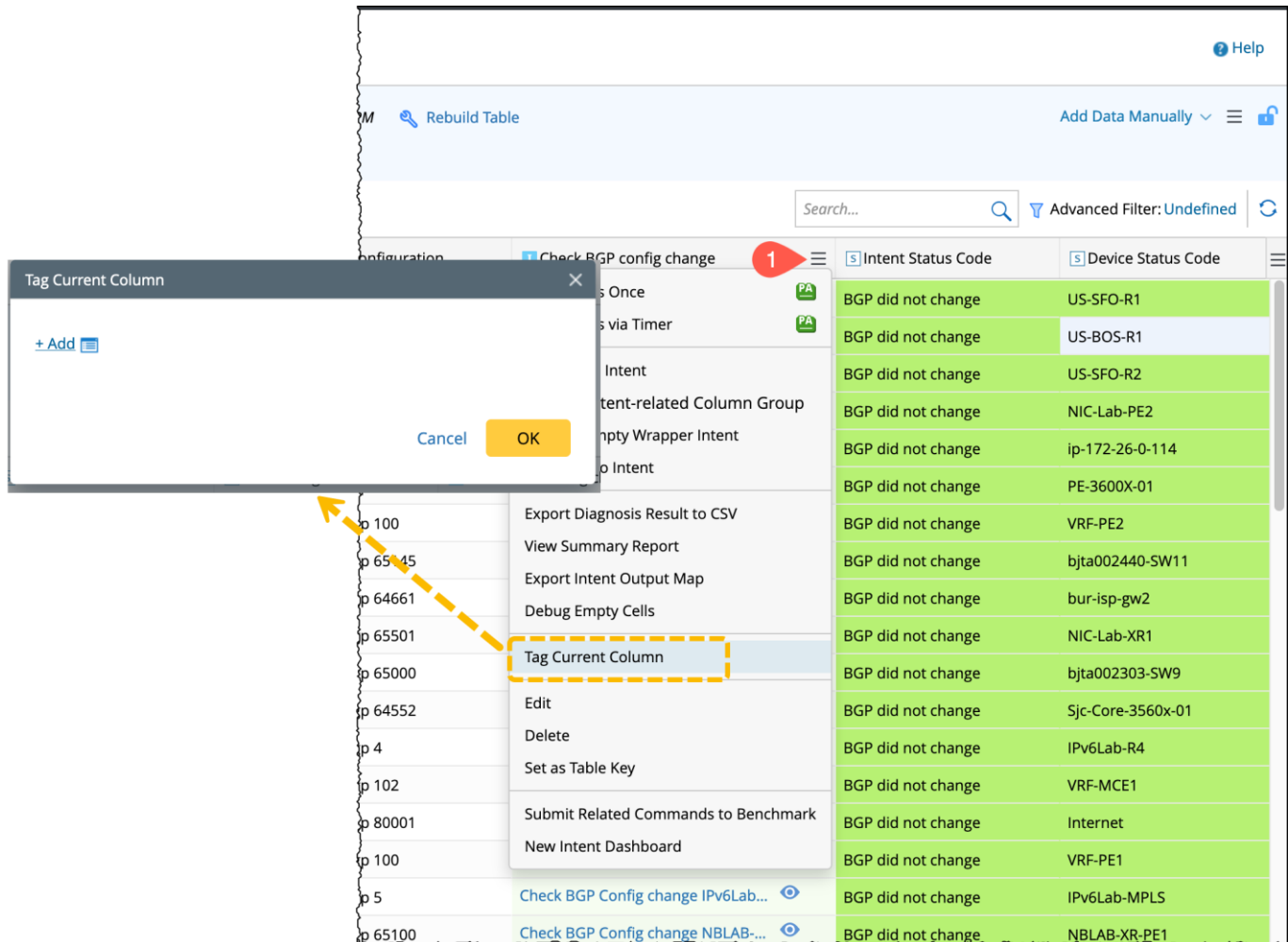
Items: 53 Rows 7 Columns Search... Advanced Filter: Undefined

No.	Device	BGP ASN	Router ID	BGP Configuration	Check BGP config change	Intent Status Code	Device Status Code
1	US-SFO-R1	65001	10.9.9.9	router bgp 65001	Check BGP Config change US-SFO...	BGP did not change	US-SFO-R1
2	US-BOS-R1	65001	10.10.10.10	router bgp 65001	Check BGP Config change US-BOS...	BGP did not change	US-BOS-R1
3	US-SFO-R2	65001	10.8.2.250	router bgp 65001	Check BGP Config change US-SFO...	BGP did not change	ip-172-26-0-114
4	NIC-Lab-PE2	65501	10.8.76.203	router bgp 65501	Check BGP Config change NIC-Lab...	BGP did not change	BGP did not change
5	ip-172-26-0-114	64513		router bgp 64513	Check BGP Config change ip-172-2...	BGP did not change	ip-172-26-0-114
6	PE-3600X-01	64550	10.88.255.1	router bgp 64550	Check BGP Config change PE-3600...	BGP did not change	PE-3600X-01
7	VRF-PE2	100		router bgp 100	Check BGP Config change VRF-PE2	BGP did not change	VRF-PE2
8	bjta002440-SW11	65145	10.88.192.6	router bgp 65145	Check BGP Config change bjta002...	BGP did not change	bjta002440-SW11
9	bur-isp-gw2	64661		router bgp 64661	Check BGP Config change bur-isp-...	BGP did not change	bur-isp-gw2
10	NIC-Lab-XR1	65501	10.8.76.200	router bgp 65501	Check BGP Config change NIC-Lab...	BGP did not change	NIC-Lab-XR1
11	bjta002303-SW9	65000	10.61.4.25	router bgp 65000	Check BGP Config change bjta002...	BGP did not change	bjta002303-SW9
12	Sjc-Core-3560x-01	64552	10.88.255.81	router bgp 64552	Check BGP Config change Sjc-Core...	BGP did not change	Sjc-Core-3560x-01
13	IPv6Lab-R4	4	4.4.4.4	router bgp 4	Check BGP Config change IPv6Lab...	BGP did not change	IPv6Lab-R4
14	VRF-MCE1	102		router bgp 102	Check BGP Config change VRF-MCE1	BGP did not change	VRF-MCE1
15	Internet	80001	124.1.1.1	router bgp 80001	Check BGP Config change Internet	BGP did not change	Internet
16	VRF-PE1	100		router bgp 100	Check BGP Config change VRF-PE1	BGP did not change	VRF-PE1
17	IPv6Lab-MPLS	5	5.5.5.5	router bgp 5	Check BGP Config change IPv6Lab...	BGP did not change	IPv6Lab-MPLS
18	NBLAB-XR-PE1	65100	10.8.19.1	router bgp 65100	Check BGP Config change NBLAB-...	BGP did not change	NBLAB-XR-PE1
19	West-CSR1000v	65523		router bgp 65523	Check BGP Config change West-CS...	BGP did not change	West-CSR1000v

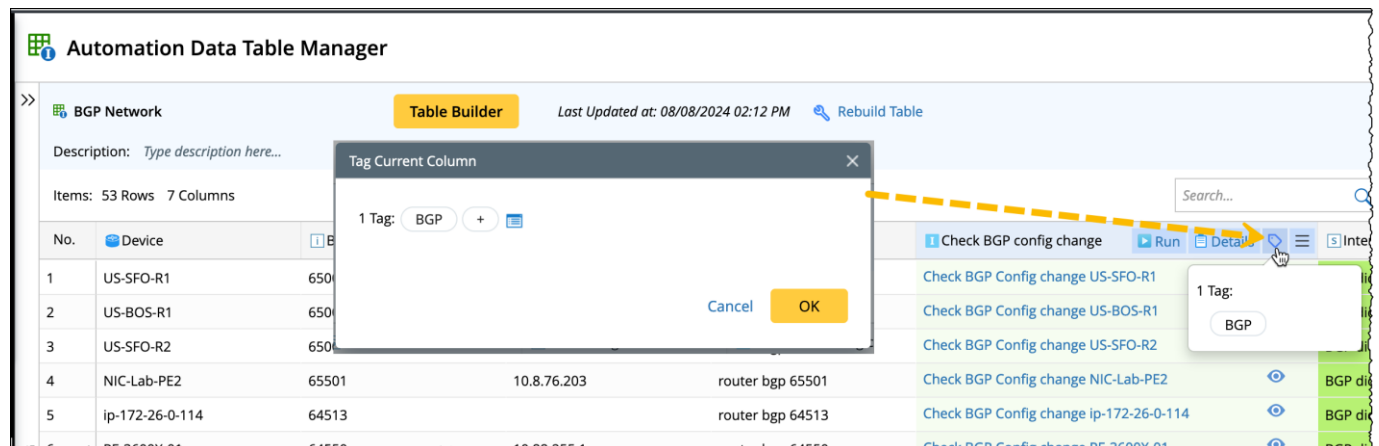
9.2.1.6 Set the intent tag as BGP.

Add a tag **BGP** to the intent column:

1. Go to the intent column  menu > Tag Current column:



2. The added tag can be seen in the intent column header.




9.2.2 Create a Wrapper Intent

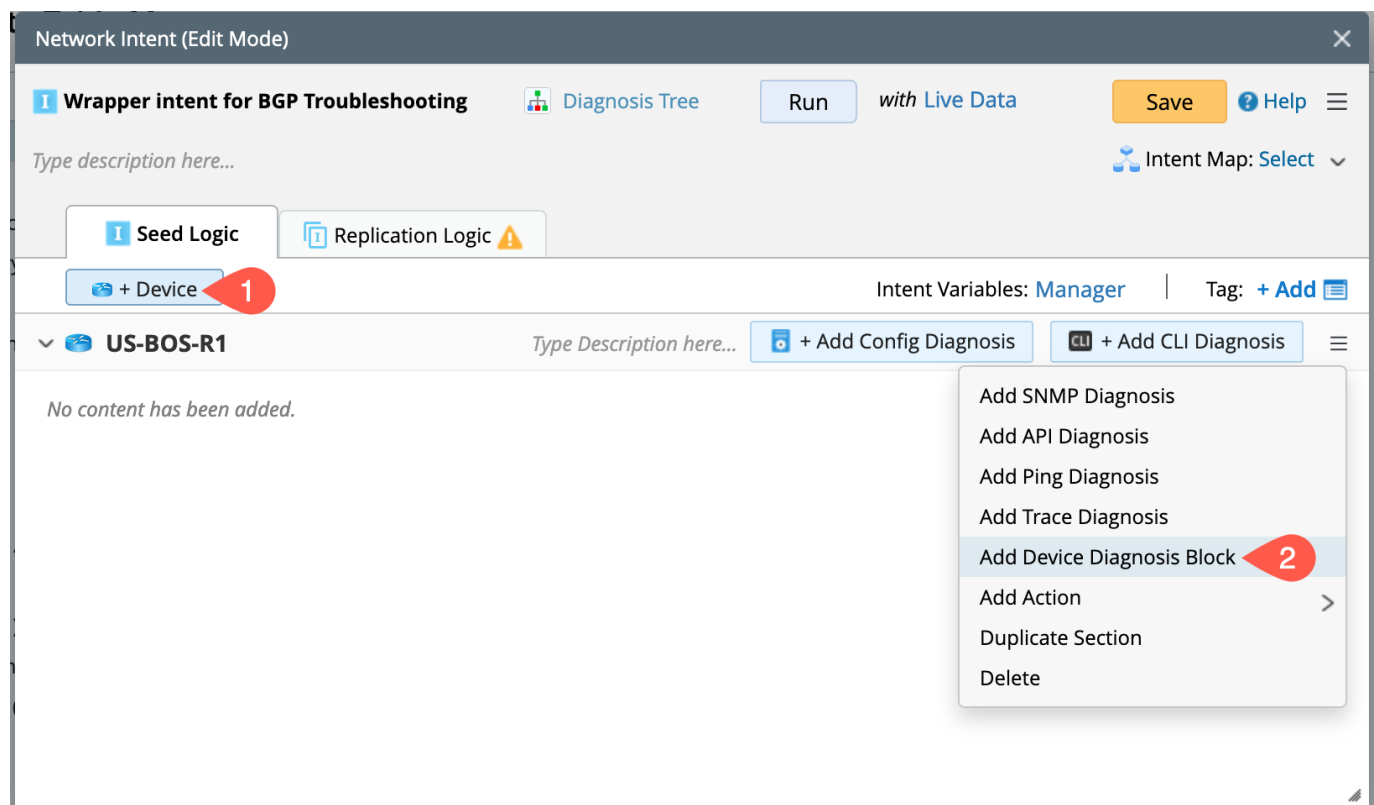
Create a wrapper intent as the public interface of intents to troubleshoot BGP-related problems. You will define the follow up intents logic as all the intent columns from the ADT with the tag BGP:

1. [Create a Wrapper Intent](#)
2. [Replicate and Execute the Wrapper Intent](#)

9.2.2.1 Create a Wrapper Intent

From the **Intent Manager**, create a new intent and name it as **Wrapper intent for BGP Troubleshooting**.

1. Click **+Device** and add a device from the BGP device group as the seed Intent, e.g., **US-BOS-R1**.
2. Click  located beside the add diagnosis buttons and then **Add Device Diagnosis Block**.



3. Add a description to the diagnosis about the follow up action.
4. In the **If** condition, select **True** from the Select Variable dropdown.
5. Remove the **Diagnosis Message** and add **Follow-up Intent** from the **Add logic** dropdown.

6. Click the **Network Intent** link to add the follow up intent from the ADT.

Edit Device Diagnosis Block


Diagnosis1 +

Name:

Description: Call all intents tagged with BGP in ADT table (BGP Devices) 3

☐ Loop Table Rows

▼ If

A  US-BOS-R1

True 4

B Select Variable

▼ Then

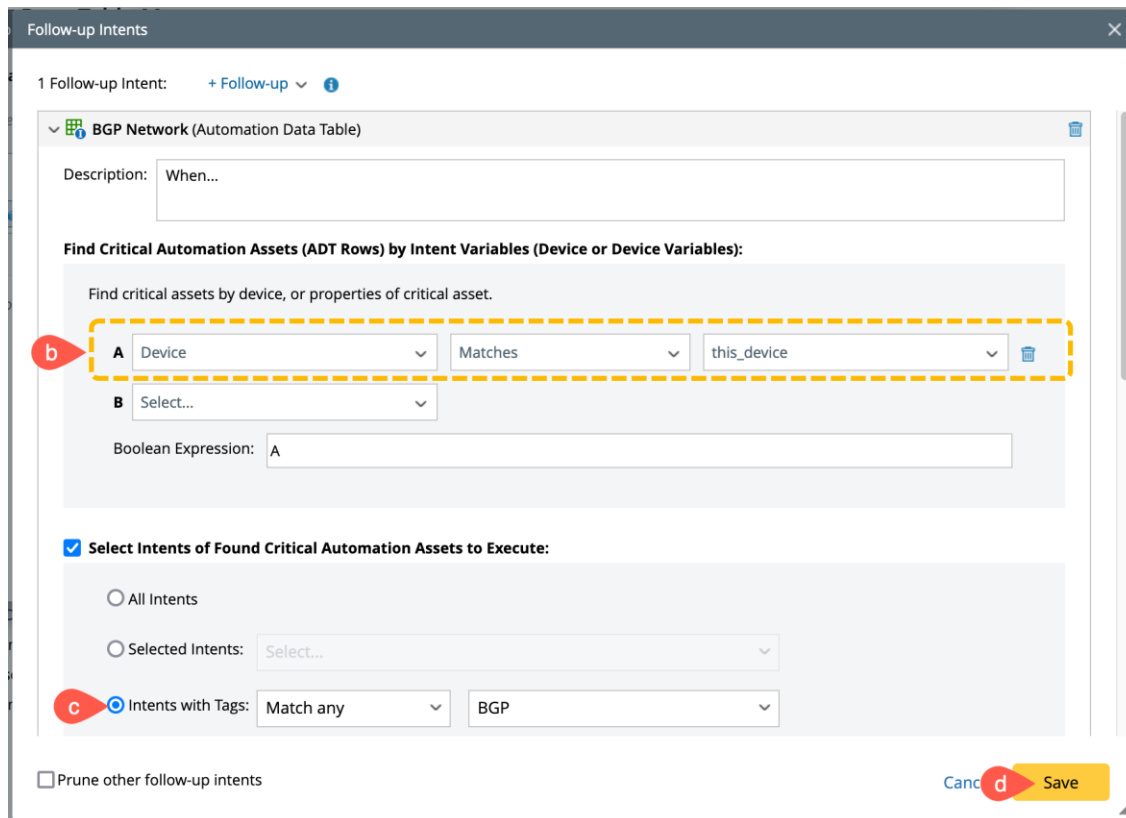
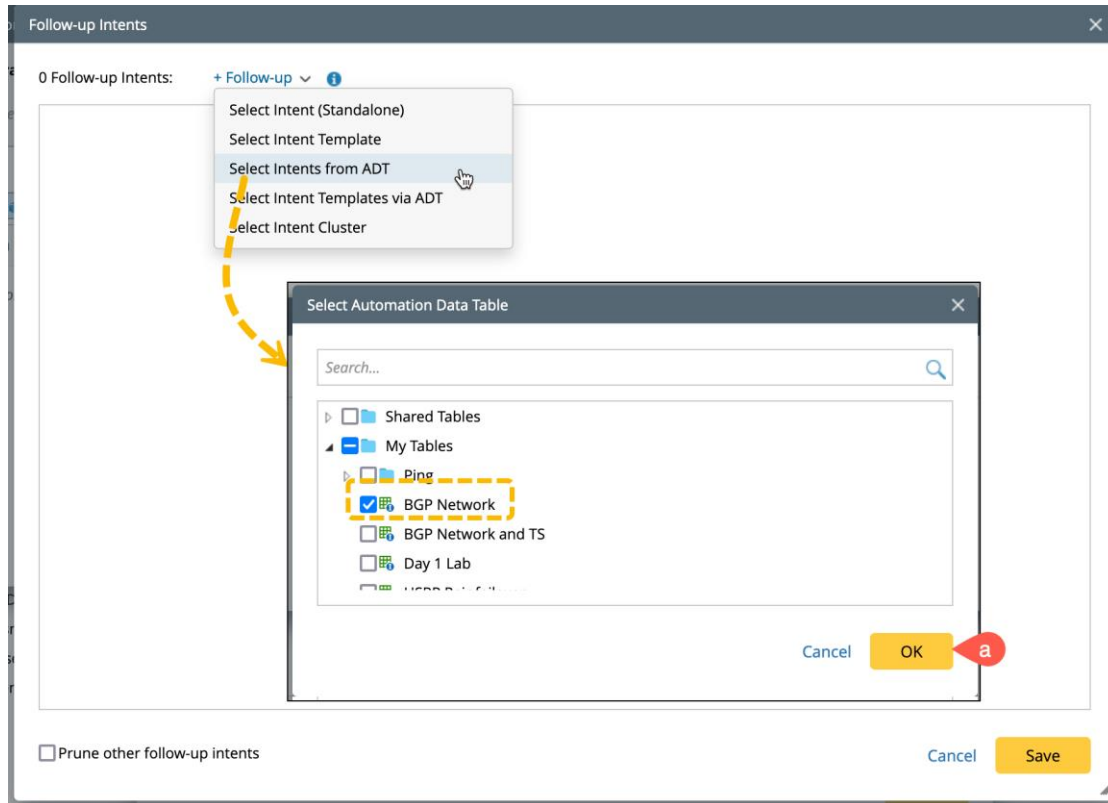
5 Follow-up Intent: 6 Network Intent Current Intent (Self) Stop

Add Logic ▼

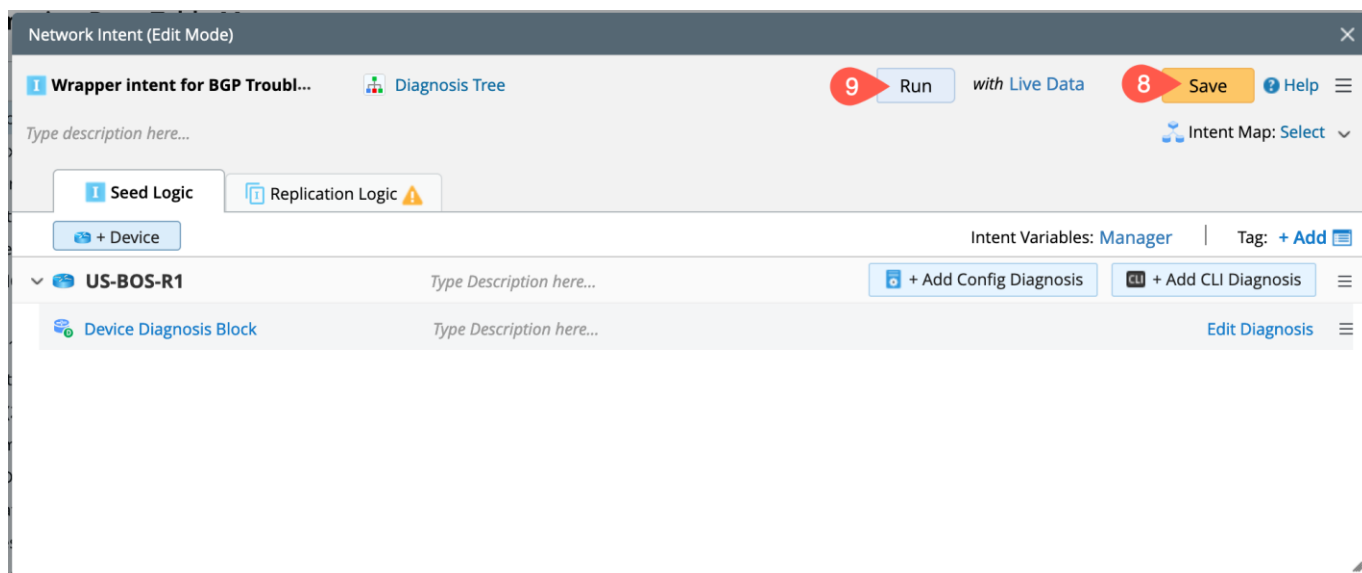
+ Add Elself + Add Else

Cancel OK

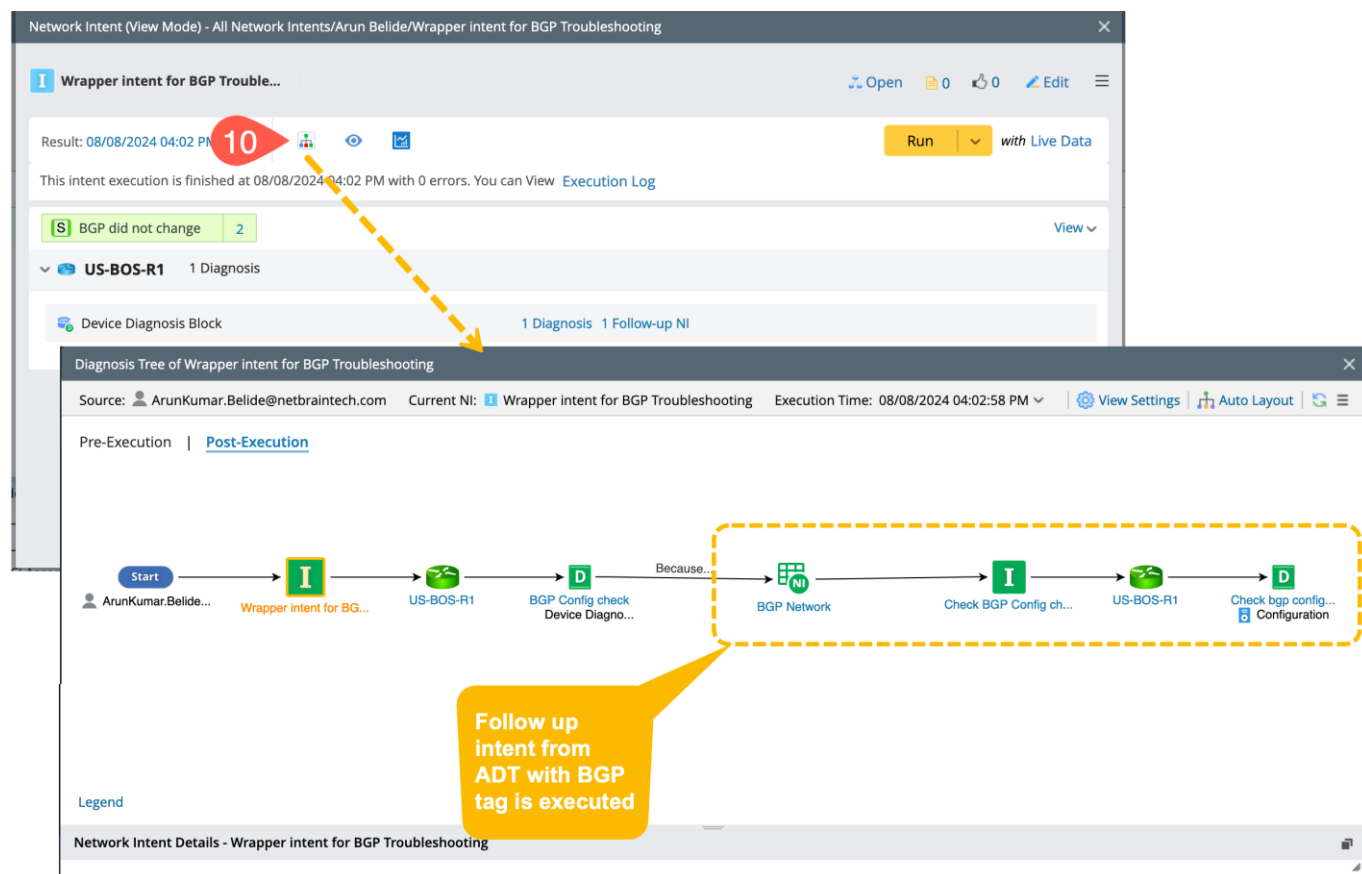
7. Click **Select Intents from ADT**.



8. Back in the Network Intent (Edit Mode) dialog, click **Save**.
9. Click **Run** to execute the intent.

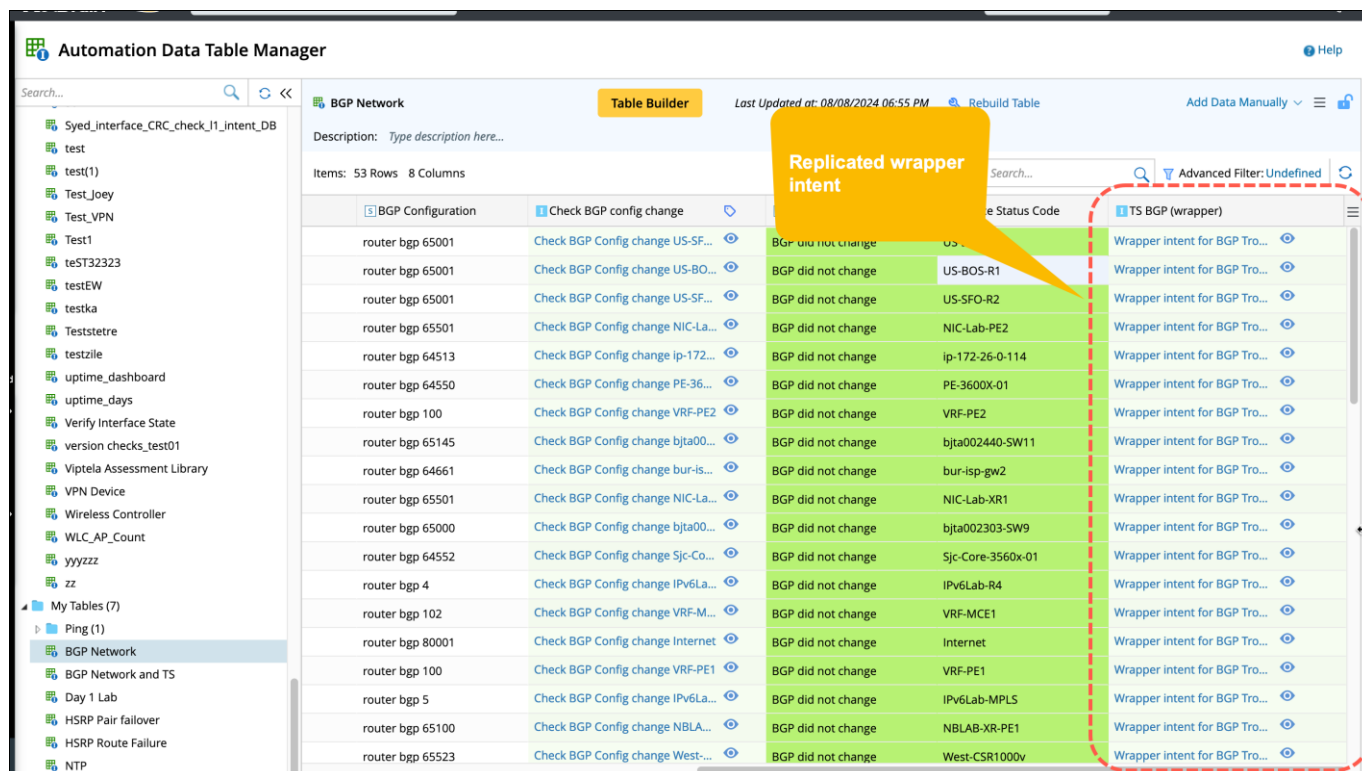


10. Open the diagnosis tree to see the results.



9.2.2.2 Replicate and Execute the Wrapper Intent

Now, you can use the **Intent Replication Wizard** to replicate the **wrapper** intent to all the devices in the ADT table **BGP Network**. Refer to Section 9.2.1.4 for details. In the second step (**Define ADT**), select the existing ADT, **BGP Network**. In the third step, define the **intent qualification** with device group **BGP Devices** to include devices for the wrapper intent to replicate on.




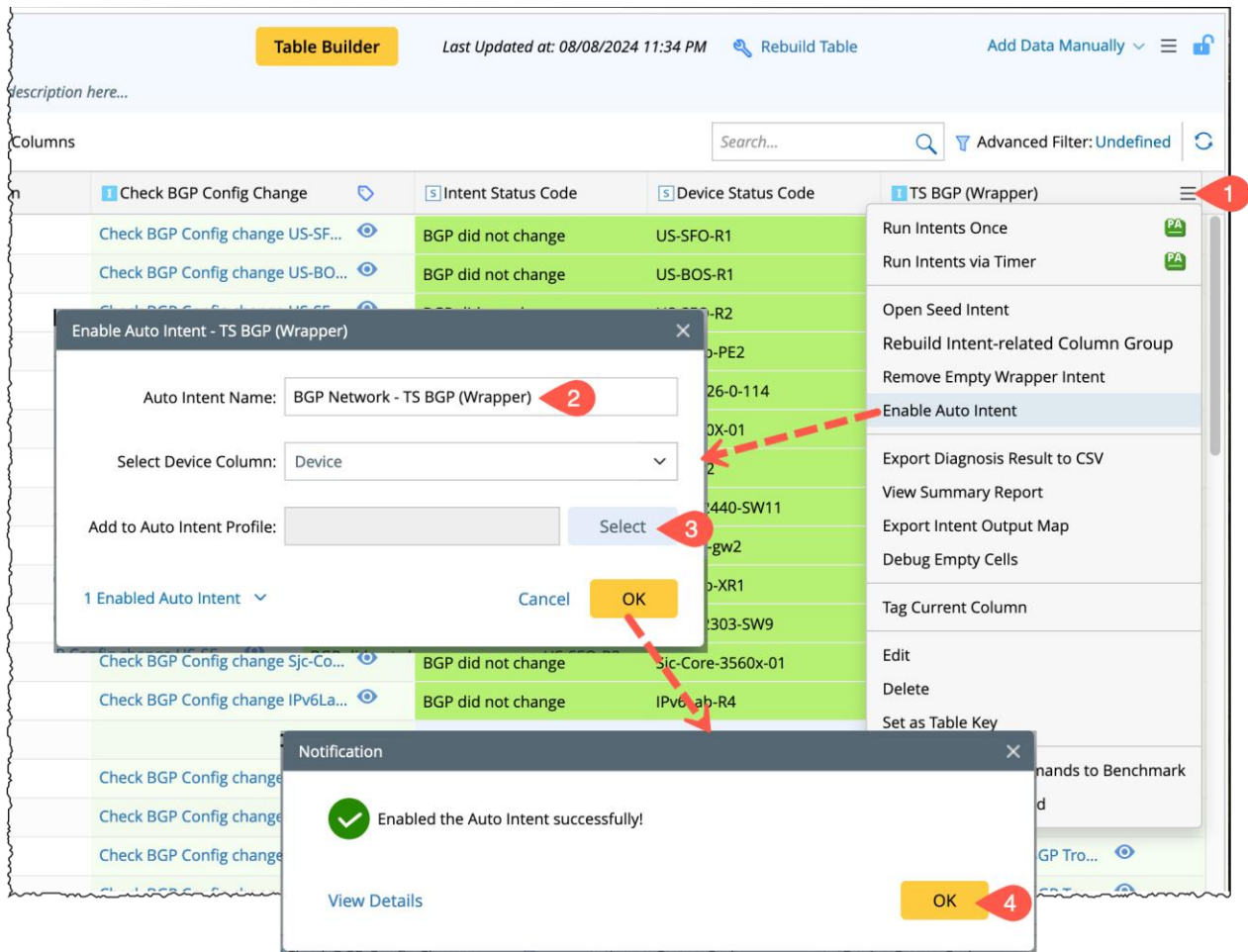
The screenshot shows the Automation Data Table Manager interface. The table is titled "BGP Network" and contains 53 rows and 8 columns. The columns are: BGP Configuration, Check BGP config change, Status Code, and TS BGP (wrapper). The table is filtered to show only rows where the status is "BGP did not change". A red dashed box highlights the "TS BGP (wrapper)" column, and a yellow callout bubble points to it with the text "Replicated wrapper intent".

BGP Configuration	Check BGP config change	Status Code	TS BGP (wrapper)
router bgp 65001	Check BGP Config change US-SF...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65001	Check BGP Config change US-BO...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65001	Check BGP Config change US-SF...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65501	Check BGP Config change NIC-La...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 64513	Check BGP Config change ip-172...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 64550	Check BGP Config change PE-36...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 100	Check BGP Config change VRF-PE2	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65145	Check BGP Config change bjta00...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 64661	Check BGP Config change bur-is...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65501	Check BGP Config change NIC-La...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65000	Check BGP Config change bjta00...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 64552	Check BGP Config change Sjc-Co...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 4	Check BGP Config change IPv6La...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 102	Check BGP Config change VRF-M...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 80001	Check BGP Config change Internet	BGP did not change	Wrapper intent for BGP Tro...
router bgp 100	Check BGP Config change VRF-PE1	BGP did not change	Wrapper intent for BGP Tro...
router bgp 5	Check BGP Config change IPv6La...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65100	Check BGP Config change NBLA...	BGP did not change	Wrapper intent for BGP Tro...
router bgp 65523	Check BGP Config change West...	BGP did not change	Wrapper intent for BGP Tro...

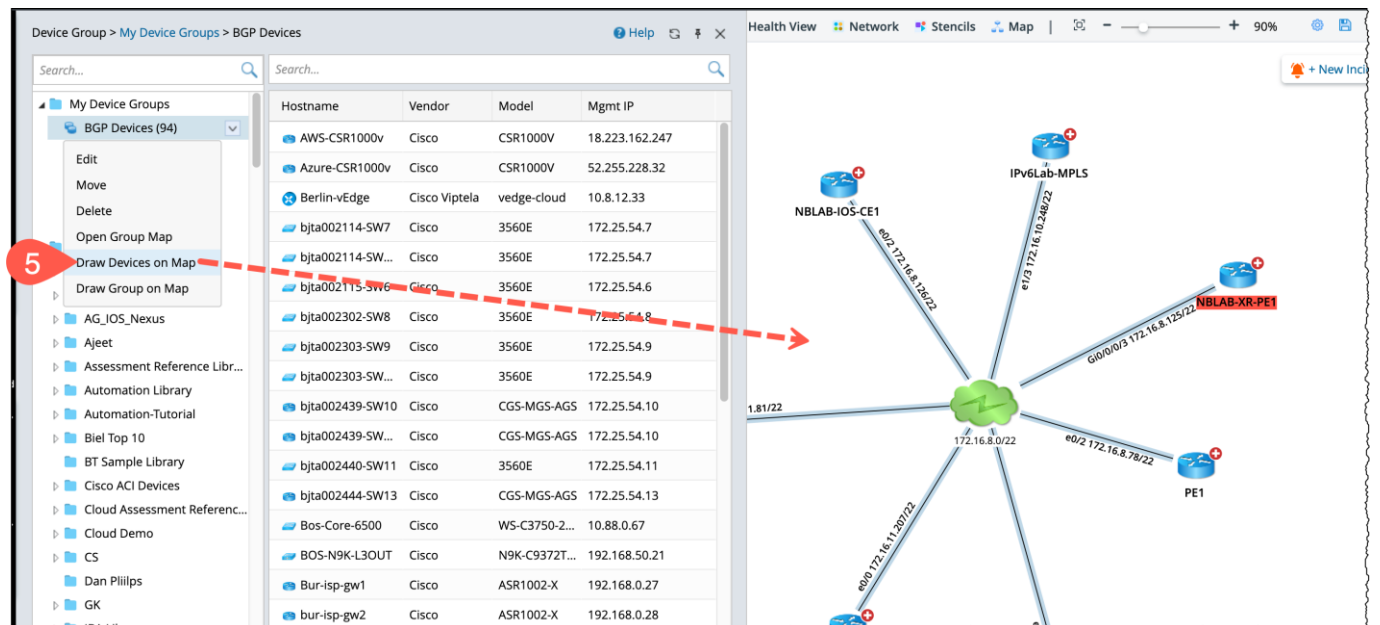
9.2.3 Execute the Auto Intent BGP Wrapper

Let us enable the auto intent feature for wrapper intent on a map and execute it on the devices in the map:

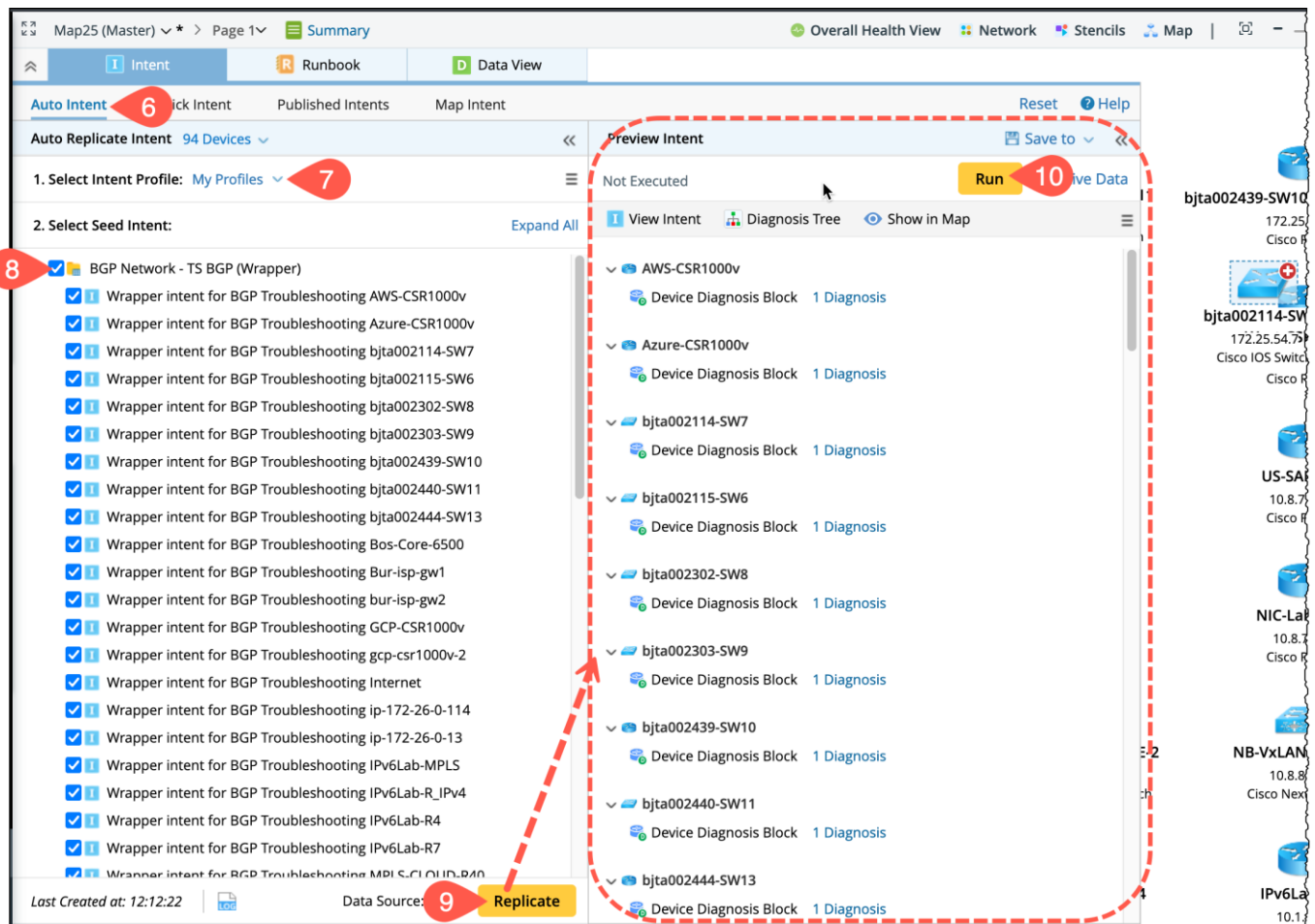
1. Click the  icon in the intent column **TS BGP (wrapper)**, then click **Enable Auto Intent**.
2. In the Enable Auto Intent dialog, validate the Auto Intent default name **BGP Network - TS BGP (wrapper)**.
3. Click **Select** > Select Profile > **Shared Profiles** > [**<Your profile>**] > click **OK**.
4. Click **OK** to complete the creation of the Auto Intent for the replicated **TS BGP (wrapper)**.



- Go to **Device Group** > Navigate to **BGP Device Group** > **Draw Devices on Map** and close the device group pane.



6. In the upper-left corner of the map, go to **Intent** > **Auto Intent**.
7. In **Select Intent Profile**, click the down arrow and navigate to the folder location with the auto intent **BGP Network - TS BGP (wrapper)** you saved in earlier steps 2 and 3.
8. Click the checkbox next to the **BGP Network - TS BGP (wrapper)**.
9. Click **Replicate** and review the list of created intent automation in the **Preview Intent** pane.
10. Click **Run**.



11. Validate the successful display of the follow up intent result.
12. In the Preview Intent page, click **Diagnosis Tree** to trace the execution flow.
13. Review the Diagnosis Tree and validate to yourself that you can properly trace the execution of the **Auto Intent** > **Wrapper Intent** > **Automation Data Tables** > **Replicated Intents** > **Diagnosis Logic**.
14. Close the Diagnosis Tree.

Map25 (Master) > Page 1 > Summary

Overall Health View Network Stencils Map | 110%

Intent Runbook Data View

Auto Intent Quick Intent Published Intents Map Intent Reset Help

Auto Replicate Intent 94 Devices

1. Select Intent Profile: My Profiles

2. Select Seed Intent:

BGP Network - TS BGP (Wrapper)

- Wrapper intent for BGP Troubleshooting AWS-CSR1000v
- Wrapper intent for BGP Troubleshooting Azure-CSR1000v
- Wrapper intent for BGP Troubleshooting bjta002114-SW7
- Wrapper intent for BGP Troubleshooting bjta002114-SW6
- Wrapper intent for BGP Troubleshooting bjta002114-SW5
- Wrapper intent for BGP Troubleshooting bjta002114-SW4
- Wrapper intent for BGP Troubleshooting bjta002114-SW3
- Wrapper intent for BGP Troubleshooting bjta002114-SW2
- Wrapper intent for BGP Troubleshooting bjta002114-SW1
- Wrapper intent for BGP Troubleshooting bjta002114-SW0
- Wrapper intent for BGP Troubleshooting bjta002114-SW9
- Wrapper intent for BGP Troubleshooting bjta002114-SW8
- Wrapper intent for BGP Troubleshooting bjta002114-SW7
- Wrapper intent for BGP Troubleshooting bjta002114-SW6
- Wrapper intent for BGP Troubleshooting bjta002114-SW5
- Wrapper intent for BGP Troubleshooting bjta002114-SW4
- Wrapper intent for BGP Troubleshooting bjta002114-SW3
- Wrapper intent for BGP Troubleshooting bjta002114-SW2
- Wrapper intent for BGP Troubleshooting bjta002114-SW1
- Wrapper intent for BGP Troubleshooting bjta002114-SW0

Diagnosis Tree of Auto NI

Source: ArunKumar.Belide@netbraintech.com Current NI: Auto NI Execution Time: 08/09/2024 12:21:31 AM

Pre-Execution | Post-Execution

13

11

12

14

Start

Auto NI

MPLS-CLOUD-R40 BGP Config check Device Diagno... BGP Network Check BGP Config ch... MPLS-CLOUD-R40 Check bgp config... Configuration

MPLS-P-ASR9001-01 BGP Config check Device Diagno... BGP Network Check BGP Config ch... MPLS-P-ASR9001-01 Check bgp config... Configuration

NBLAB-IOS-CE1 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NBLAB-IOS-CE1 Check bgp config... Configuration

NBLAB-IOS-CE3 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NBLAB-IOS-CE3 Check bgp config... Configuration

NBLAB-XR-PE1 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NBLAB-XR-PE1 Check bgp config... Configuration

NBLAB-XR-PE3 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NBLAB-XR-PE3 Check bgp config... Configuration

NIC-Lab-PE1 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NIC-Lab-PE1 Check bgp config... Configuration

NIC-Lab-PE2 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NIC-Lab-PE2 Check bgp config... Configuration

NIC-Lab-PE3 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NIC-Lab-PE3 Check bgp config... Configuration

NIC-Lab-PE4 BGP Config check Device Diagno... BGP Network Check BGP Config ch... NIC-Lab-PE4 Check bgp config... Configuration

Legend

Start Details

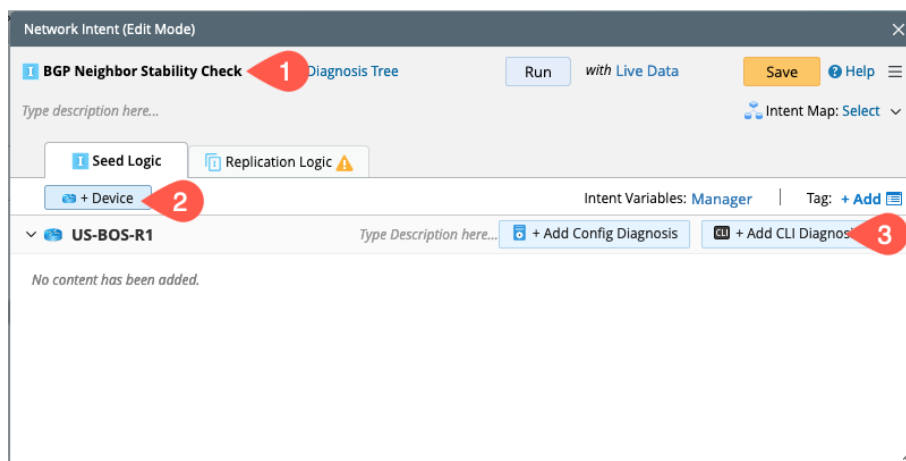
W10 bjta002303-SW9_TempName_T... 172.25.54.9 Cisco IOS Switch bjta002303-SW9 BGP did not change

9.3 Create frequently used BGP Troubleshooting Intents

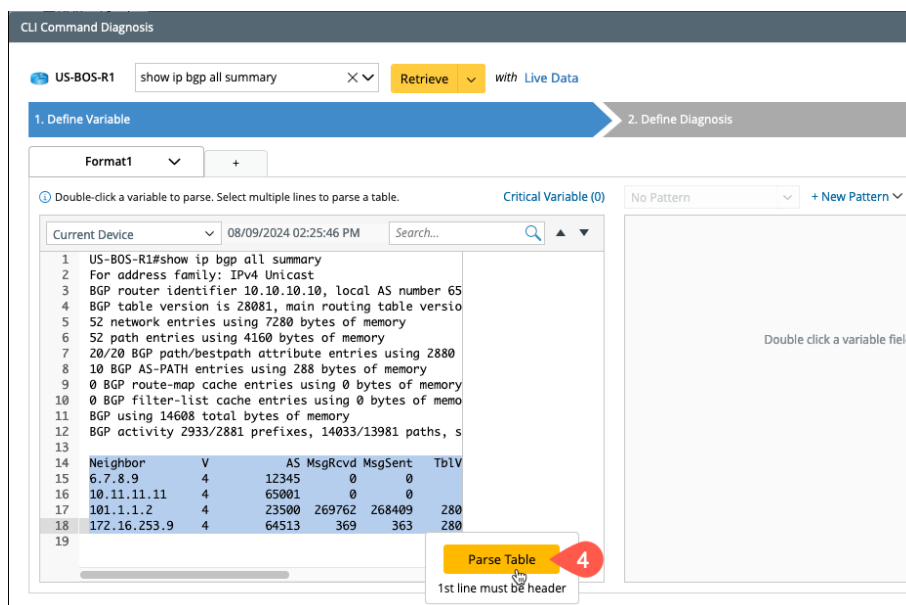
9.3.1 Create NI to Check BGP Neighbor Status Check

In this section, you will create an intent to check the change in BGP Neighbor status:

1. Create an Intent from the Intent Manager with Name, **BGP Neighbor Stability Check**.
2. Add a seed device from the BGP device group, e.g., **US-BOS-R1**.
3. Click **+Add CLI Diagnosis** to parse the variables and define the diagnosis.



4. The command will be: **show ip bgp all summary**. The required neighbor data is in tabular format, which can be parsed by the **Table Parser**. Select the table data and click **Parse Table** in the tip window.



5. In the parsed variables **output**, set the variable **\$neighbor** as Table key and click **Apply**. The table key will be used to compare two tables, such as the last and baseline tables.

2. Define Diagnosis

Test on Devices: 0

Critical Variable (0) bgp_nbrs Type: Table + New Pattern

Header Neighbor V AS Rcvd MsgSent TblVer InQ OutQ Up/Down
State/PfxRcd

Find 8 columns, enter semicolon to separate column.

15 10.10.10.10 4 65001 0 0 1 0 0 never ... > 2 Lines

Set Table Column

Column	Variable	Type
Neighbor	\$neighbor	string
V	\$v	int
AS	\$as	int
Rcvd	\$rcvd	int
MsgSent	\$msgsent	int
TblVer	\$tblver	int

Output

\$neighbor	\$v	\$as	\$rcvd	\$msgsent
4	65001	0	0	
4	30000	504848	504813	

Remove
Set as Interface Key
5 Set as Table Key

Cancel Apply

6. Define Diagnosis:

- a) **If** condition: A: **Current bgp_nbrs | Does not equal | Baseline bgp_nbrs**
- b) **Then**: In case **If** logic is true, define the color (**red**), status (**Error**), and **Message**: *\$this_device BGP changed from the baseline!*
- c) **Else**: In case **If** logic is not true, define the color (**green**), status (**Success**), and **Message**: *\$this_device BGP did not change.*

The screenshot shows the '2. Define Diagnosis' configuration window. At the top, there are tabs for 'Add Note' and 'Add Diagnosis', with a note that 'Can also click a variable on the left to add automation.' The 'Name' field is 'Check whether BGP Neighbor changed' and the 'Anchor' is 'bgp_nbrs.\$neighbor'. Below this is a text area for 'Type description of the diagnosis...'. The 'Loop Table Rows' checkbox is unchecked. The 'If' section (highlighted with a dashed yellow border and labeled 'a') contains a condition: 'A US-BOS-R2 Current Baseline' with a dropdown for 'bgp_nbrs', the operator 'Does not equal', and another dropdown for 'bgp_nbrs'. The 'Then' section (highlighted with a dashed yellow border and labeled 'b') contains a 'Diagnosis Message' field with a red color selector and the message '\$this_device BGP neighbor changed from the baseline!'. It also has checkboxes for 'Set Status Code for Device' and 'Set Status Code for Intent', both set to 'Error' with the same message. The 'Else' section (highlighted with a dashed yellow border and labeled 'c') contains a 'Diagnosis Message' field with a green color selector and the message '\$this_device BGP neighbor did not change.'. It also has checkboxes for 'Set Status Code for Device' and 'Set Status Code for Intent', both set to 'Success' with the same message. At the bottom, there are 'Cancel' and 'Apply' buttons.

9.3.2 Replicate the Intent to ADT

Launch the **Intent Replication Wizard** from the Network Intent window to replicate the intent:

1. In the Seed Intent tab, select the Path-based Replication option.
2. **Define ADT: Use an Existing ADT** and select **BGP Network**. Enter a relevant Intent group name in the **Replicate Intent to** field.

The screenshot shows the 'Define ADT' step of the 'Intent Replication Wizard - BGP Neighbor Stability Check'. The wizard has four steps: Seed Intent, Define ADT (current), Replication Settings, and Replicate Intent. Below the step indicators are two buttons: 'Create a New ADT' and 'Use an Existing ADT'. The 'Automation Data Table' is set to 'BGP Network'. The 'Replicate Intent to' dropdown is set to 'New Column Group', and the 'Neighbor Check' field is highlighted with a yellow callout bubble that says 'Add column name'. The 'Replicate on Device Column' dropdown is set to 'Device'. At the bottom, it says 'Selection Mode: Device-based Replication.' and has 'Previous' and 'Next' buttons.

Intent Replication Wizard - BGP Neighbor Stability Check

Seed Intent Define ADT Replication Settings Replicate Intent

Create a New ADT Use an Existing ADT

Automation Data Table: BGP Network

Replicate Intent to: New Column Group Neighbor Check

Replicate on Device Column: Device

Selection Mode: Device-based Replication.

Previous Next

3. Replication Settings:

The screenshot shows the 'Replication Settings' step of the 'Intent Replication Wizard - BGP Neighbor Stability Check'. The wizard has four steps: Seed Intent, Define ADT, Replication Settings (current), and Replicate Intent. Below the step indicators is a 'Full Settings for Template' link. The 'Intent Qualification' section has two options: 'via Device Groups/Sites: 1 Device Groups/Folders' (selected) and 'via Dynamic Search: Undefined'. Below this is a section titled 'Define Macro Variables and Rules for Their Substitution:' with 'Item: 1'. A table shows the 'Seed Device' as 'US-BOS-R1' and the 'Seed Command' as 'show ip bgp all summary'. A yellow callout bubble points to the '1 Device Groups/Folders' link with the text 'Add the BGP device group'. The 'Macro Variables' column is empty.

Intent Replication Wizard - BGP Neighbor Stability Check

Seed Intent Define ADT Replication Settings Replicate Intent

Full Settings for Template

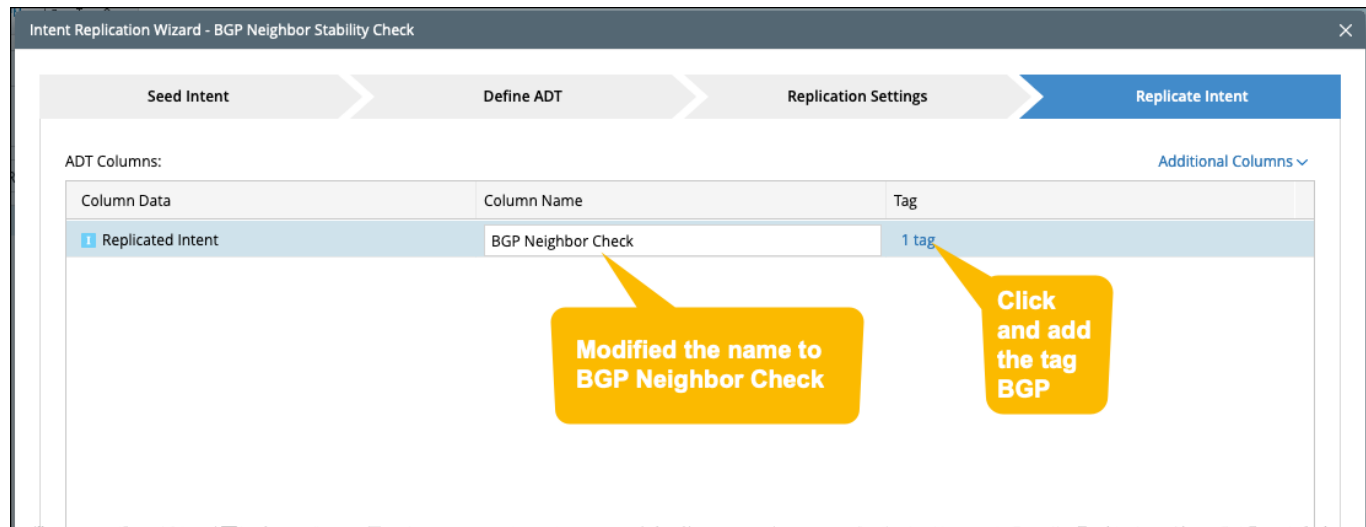
Intent Qualification: ☒ via Device Groups/Sites: 1 Device Groups/Folders ☐ via Dynamic Search: Undefined

Define Macro Variables and Rules for Their Substitution:

Item: 1

Seed Device	Seed Command	Macro Variables
US-BOS-R1	show ip bgp all summary	

4. Define the column name and tag for the Intent. The added automation column will be:



The table will now be populated with devices and the replicated Intents (**BGP Neighbor Check**).

Automation Data Table Manager

BGP Network

Description: Type description here...

Items: 53 Rows 9 Columns

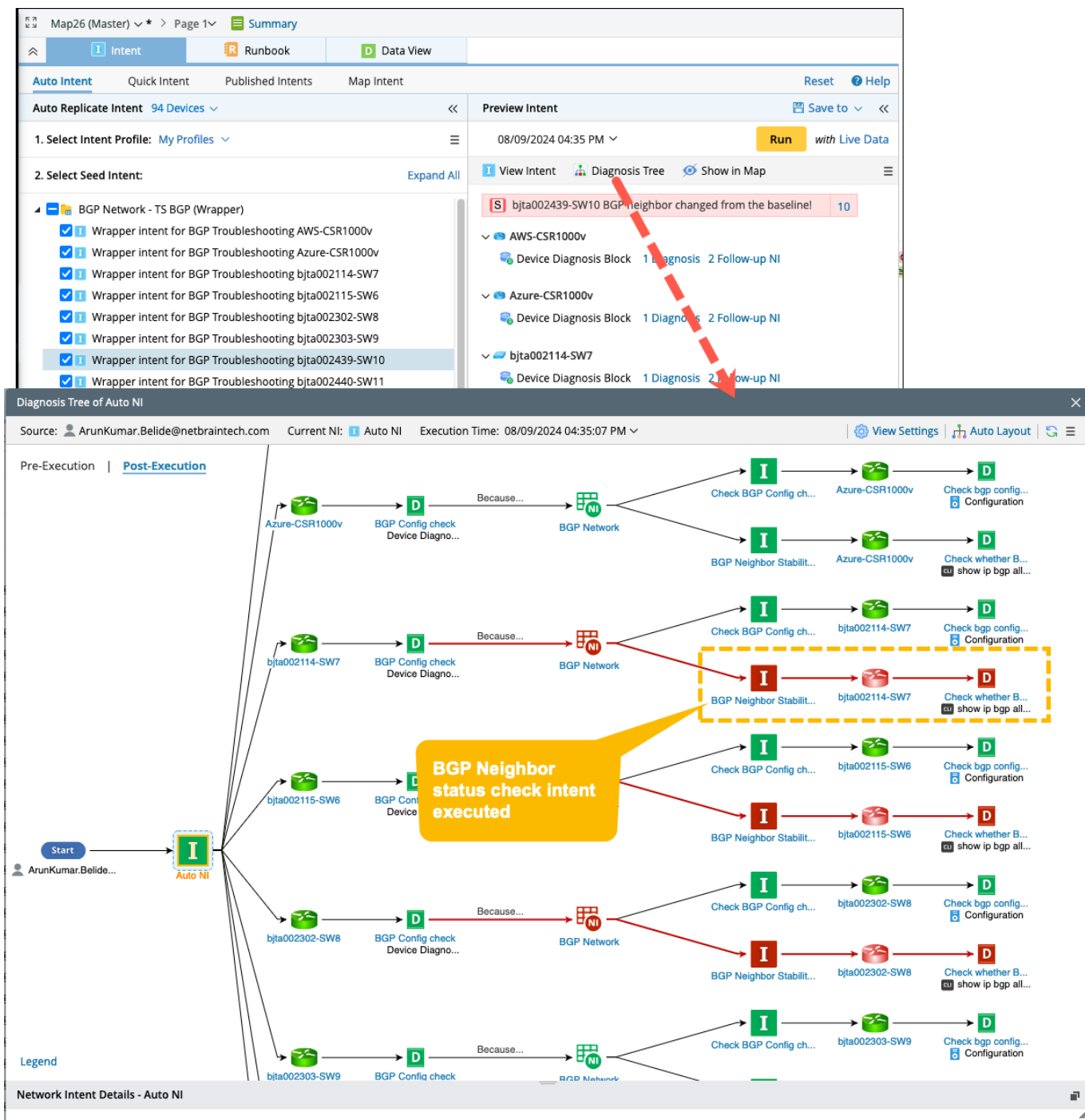
Table Builder Last Updated at: 08/09/2024 04:23 PM Rebuild Table

Configuration	1 Check BGP Config Change	5 Intent Status Code	5 Device Status Code	5 BGP Neighbor Check
bgp 65001	Check BGP Config change US-SF...	BGP did not change	US-SFO-R1	BGP Neighbor Stability Chec...
bgp 65001	Check BGP Config change US-BO...	BGP did not change	US-BOS-R1	BGP Neighbor Stability Chec...
bgp 65001	Check BGP Config change US-SF...	BGP did not change	US-SFO-R2	BGP Neighbor Stability Chec...
bgp 65501	Check BGP Config change NIC-La...	BGP did not change	NIC-Lab-PE2	BGP Neighbor Stability Chec...
bgp 64513	Check BGP Config change ip-172...	BGP did not change	ip-172-26-0-114	BGP Neighbor Stability Chec...
bgp 64550	Check BGP Config change PE-36...	BGP did not change	PE-3600X-01	BGP Neighbor Stability Chec...
bgp 100	Check BGP Config change VRF-PE2	BGP did not change	VRF-PE2	BGP Neighbor Stability Chec...
bgp 65145	Check BGP Config change bjta00...	BGP did not change	bjta002440-SW11	BGP Neighbor Stability Chec...
bgp 64661	Check BGP Config change bur-is...	BGP did not change	bur-isp-gw2	BGP Neighbor Stability Chec...
bgp 65501	Check BGP Config change NIC-La...	BGP did not change	NIC-Lab-XR1	BGP Neighbor Stability Chec...
bgp 65000	Check BGP Config change bjta00...	BGP did not change	bjta002303-SW9	BGP Neighbor Stability Chec...

Intent replicated to all the devices in ADT

BGP tag

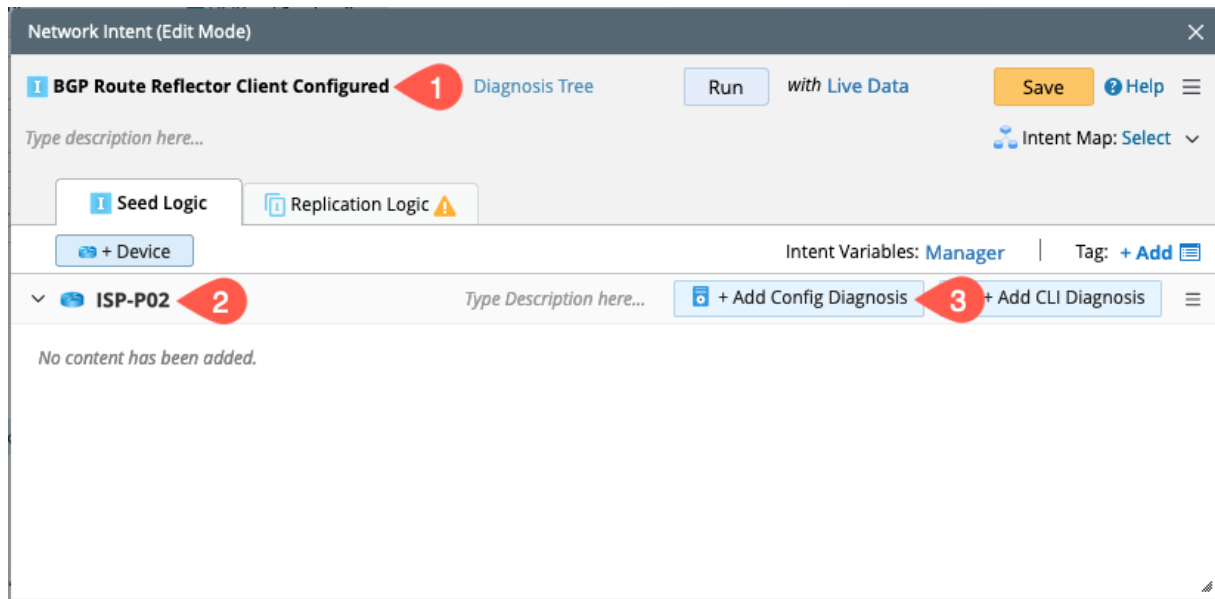
- Now, execute the wrapper intent on BGP devices with auto intent, repeating the same detailed in Section 9.2.3 to verify the **BGP Neighbor Check** (tagged BGP) is also executed.



9.3.3 Create NI to Check BGP Route Reflector Client

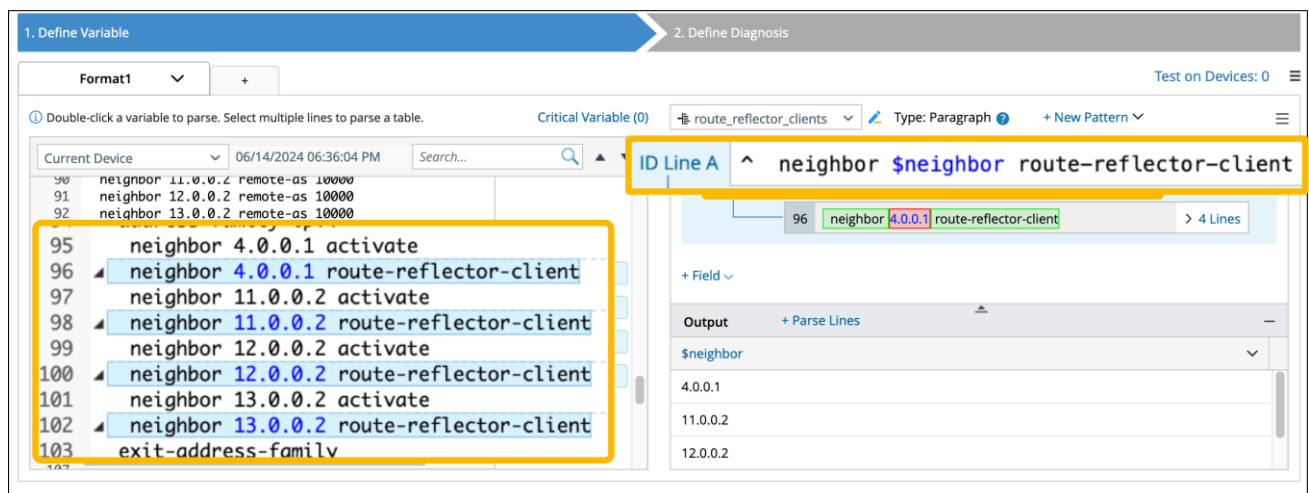
In this section, you will create an intent to check whether the status of the route reflector client is configured, and if so, ping each reflector client:

1. Create an Intent from the Intent Manager with Name, **BGP Route Reflector Client Configured**.
2. Add a seed device from the BGP device group, e.g., **ISP-P02**.
3. Click **+Add Config Diagnosis** to parse the variables and define the diagnosis.

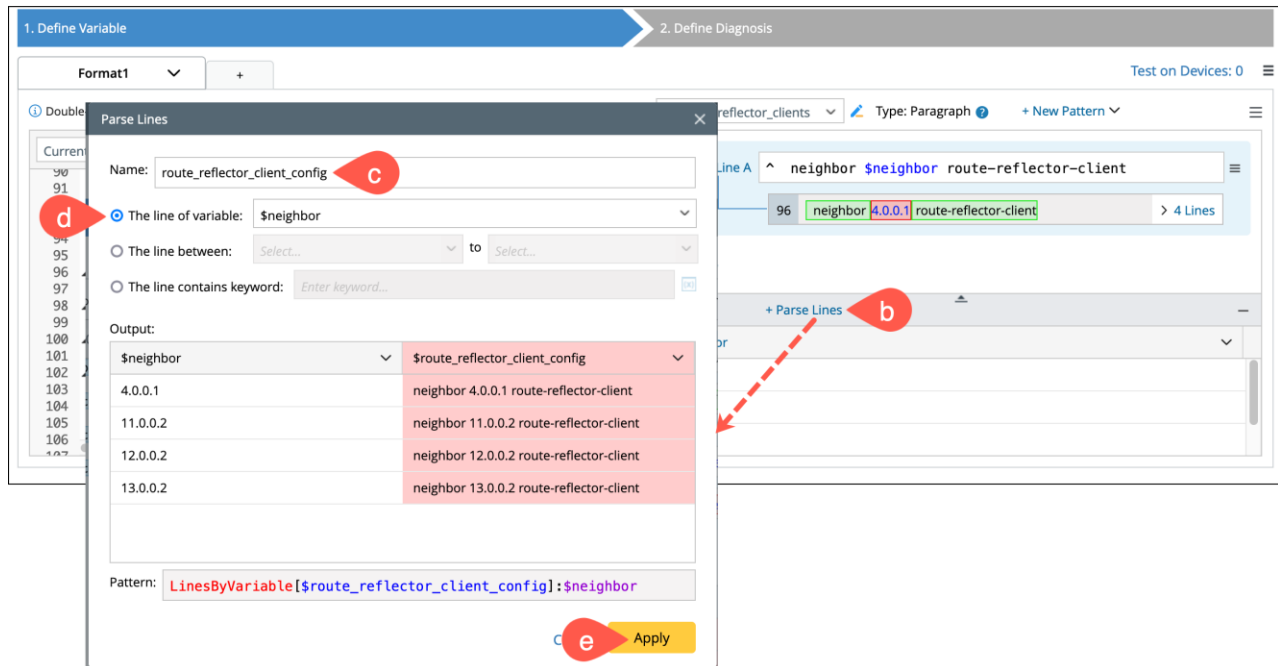


4. Click **Retrieve** to collect the data from Live Data. The required neighbor data is in paragraph format. Create the Paragraph Parser to parse the neighbor:

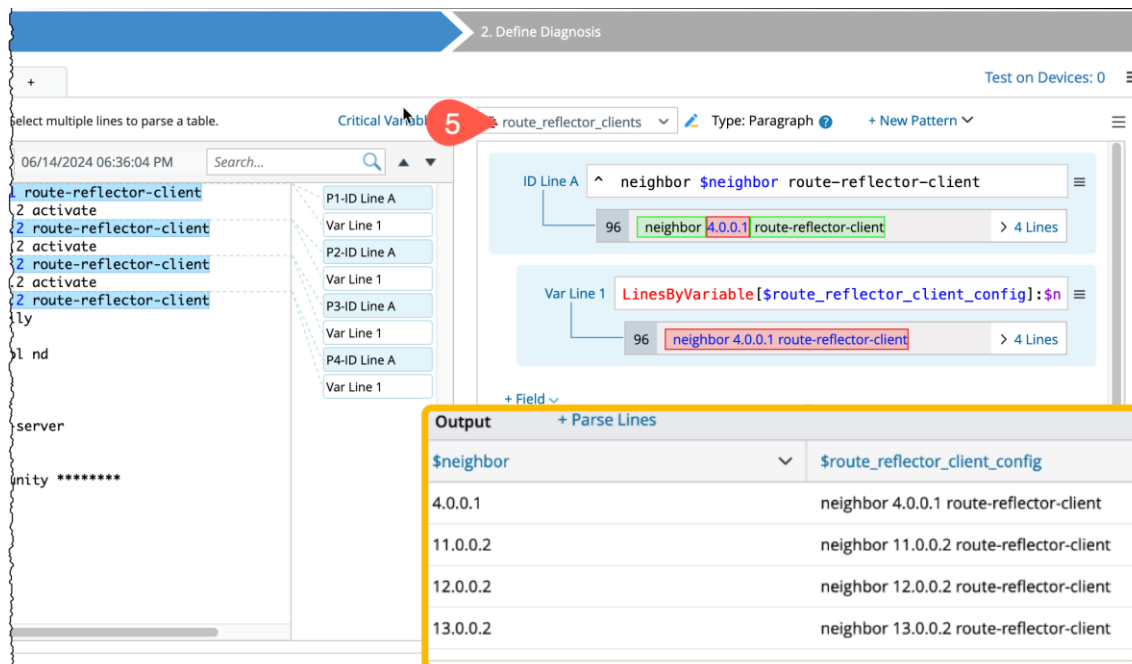
a) Define the ID Line as `^ neighbor $neighbor route-reflector-client`.



- b) Parse all the multiple lines with **route-reflector-client** using **+Parse Lines** to check whether the route reflector client is configured.
- c) Add the name: **route_reflector_client_config**.
- d) Parse the lines using the **line of variable**: select the variable **\$neighbor**.
- e) Click **Apply** to save and close the **Parse Lines** dialog.



5. Rename the parser as **route_reflector_clients**, and the final output would be:



6. Define Diagnosis 1: Check whether the route reflector client is configured as follows:
- a) **If** condition: A: **route_reflector_clients** | Is not empty.
 - b) **Then**: In case **If** logic is true, define the color (**green**), status (**Success**), and Message is **Route reflector client is configured**.
 - c) **Else**: In case **If** logic is not true, define the color (**grey**), status (**Error**), and Message is **Route reflector client is not configured**.

The screenshot shows the '2. Define Diagnosis' interface in NetBrain. The diagnosis is named 'Check whether route reflector client is configured' and is anchored to 'route_reflector_client'. The logic is defined as follows:

- If** condition: A: **ISP-P02** (Current) | **route_reflector_clients** | Is not empty.
- Then** block:
 - Diagnosis Message: **Route reflector client is configured** (Color: Green)
 - Set Status Code for Device: **Success** (Status: Success) | **Route reflector client is configured** (Message)
 - Set Status Code for Intent: **Success** (Status: Success) | **Route reflector client is configured** (Message)
- Else** block:
 - Diagnosis Message: **Route reflector client is not configured** (Color: Grey)
 - Set Status Code for Device: **Error** (Status: Error) | **Route reflector client is not configured** (Message)
 - Set Status Code for Intent: **Error** (Status: Error) | **Route reflector client is not configured** (Message)

Red callouts 'a', 'b', and 'c' highlight the 'If' condition, the 'Then' block, and the 'Else' block respectively.

7. Define Diagnosis 2: **Ping** the route reflector client if it is configured:
- a) **If** condition: A: **neighbor** | Is not empty.
 - b) **Then**: Add the **Ping <destination ip>** NIT created in Chapter 4 as a follow-up Intent.

2. Define Diagnosis

Can also click a variable on the left to add automation.

Name: Ping the route reflector client Anchor: route_reflector_clie... ▾

Type description of the diagnosis...

☒ Loop Table Rows ☐ route_reflector_clients ▾ Table Key: Please Select... ▾ ! ⚙

▼ If

A ☐ ISP-P02 Current ▾

a neighbor ▾ Is not empty ▾

B Select Variable ▾

▼ Then

Follow-up Intent: ☒ Network Intent ☐ Intent (Self) ☐ Stop

Intents Replicated from Template: Ping Check

Add Logic ▾

+ Add Elself + Add

Follow-up Intents

0 Follow-up Intents: + Follow-up ▾ i

- Select Intent (Standalone)
- Select Intent Template
- Select Intents from ADT
- Select Intent Templates via ADT
- Select Intent Cluster

8. In the Follow-up Intents window:
 - a) Define Replicate Current Intent to: **Device by Variable** | **this_device**.
 - b) Set macro Variable **dest_ip** to be the Variable **\$neighbor**.

Follow-up Intents

1 Follow-up Intent: + Follow-up ⓘ

▼ I Ping Check (Intent Template)

Description: When...

Replication Settings:

Replicate Current Intent to: Device by Variable this_device

Set Macro Variables of Seed Intent Template:

Seed Device	Macro Variable	Type	Set Variable
ISP-P02	dest_ip	string	neighbor (route_reflector_clients)

☐ Merge multiple replicated intents into one 0/1 Device Key Set for Selected Table

Follow-up Execution: [Settings](#)

☐ Prune other follow-up intents

Cancel Save

9. Close the diagnosis window and go back to the **Network Intent** window to replicate to **ADT BGP Network**.

9.3.4 Replicate the Intent to ADT

Launch the **Intent Replication Wizard** from the Network Intent window to replicate the intent:

1. In the Seed Intent tab, select the Path-based Replication option.
2. **Define ADT: Use an Existing ADT** and select **BGP Network**. Enter a relevant Intent group name in the **Replicate Intent to** field.

The screenshot shows the 'Define ADT' step of the 'Intent Replication Wizard - BGP Route Reflector Client Configured'. The wizard has four steps: Seed Intent, Define ADT (current), Replication Settings, and Replicate Intent. Below the steps are two buttons: 'Create a New ADT' and 'Use an Existing ADT'. The 'Automation Data Table' is set to 'BGP Network'. The 'Replicate Intent to' dropdown is set to 'New Column Group', and the 'Replicate on Device Column' dropdown is set to 'Device'. A yellow callout bubble points to the 'Replicate Intent to' dropdown with the text 'Add the column name'. At the bottom, there is a status bar indicating 'Selection Mode: Device-based Replication.' and 'Previous' and 'Next' buttons.

Intent Replication Wizard - BGP Route Reflector Client Configured

Seed Intent Define ADT Replication Settings Replicate Intent

Create a New ADT Use an Existing ADT

Automation Data Table: BGP Network

Replicate Intent to: New Column Group Reflector Client Status Check

Replicate on Device Column: Device

Selection Mode: Device-based Replication.

Previous Next

3. Replication Settings:

The screenshot shows the 'Replication Settings' step of the 'Intent Replication Wizard - BGP Route Reflector Client Configured'. The wizard has four steps: Seed Intent, Define ADT, Replication Settings (current), and Replicate Intent. Below the steps are two buttons: 'Create a New ADT' and 'Use an Existing ADT'. The 'Automation Data Table' is set to 'BGP Network'. The 'Replicate Intent to' dropdown is set to 'New Column Group', and the 'Replicate on Device Column' dropdown is set to 'Device'. A yellow callout bubble points to the 'Replicate Intent to' dropdown with the text 'Add the BGP device group'. At the bottom, there is a status bar indicating 'Selection Mode: Device-based Replication.' and 'Previous' and 'Next' buttons.

Intent Replication Wizard - BGP Route Reflector Client Configured

Seed Intent Define ADT Replication Settings Replicate Intent

Create a New ADT Use an Existing ADT

Automation Data Table: BGP Network

Replicate Intent to: New Column Group Reflector Client Status Check

Replicate on Device Column: Device

Selection Mode: Device-based Replication.

Previous Next

4. Define the column name and tag for the Intent. The added automation column will be:

Intent Replication Wizard - BGP Route Reflector Client Configured

Seed Intent > Define ADT > Replication Settings > Replicate Intent

ADT Columns: Additional Columns ▾

Column Data	Column Name	Tag
1 Replicated Intent	Check Reflector Client	1 tag
5 Intent Status Code	Route Reflector Client Configured or Not	

Default names of the columns are modified

Added the tag BGP

The table will now be populated with devices and the replicated Intent (**Check Reflector Client**).

Automation Data Table Manager

BGP Network Table Builder Last Updated at: 08/09/2024 06:33 PM Rebuild Table Add Data Manually ▾

Description: Type description here...

Items: 53 Rows 11 Columns

s Code	5 Device Status Code	1 TS BGP (Wrapper)	1 Check Reflector Client	5 Route Reflector Clie...
range	US-SFO-R1	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	US-BOS-R1	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	US-SFO-R2	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	NIC-Lab-PE2	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	ip-172-26-0-114	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	PE-3600X-01	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	VRF-PE2	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	bjta002440-SW11	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	bur-isp-gw2	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	NIC-Lab-XR1	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	bjta002303-SW9	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	Sjc-Core-3560x-01	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	
range	IPv6Lab-R4	Wrapper intent for BGP Troubles...	BGP Route Reflector Client Confi...	

BGP tag

Replicated Intent to the ADT

5. Now, execute the wrapper intent on BGP devices with auto intent, repeating the same detailed in Section 9.2.3 to verify the **BGP Route Reflector client** (tagged BGP) is also executed.

The screenshot displays the NetBrain interface with the 'Auto Intent' tab selected. The 'Auto Replicate Intent' section shows 94 devices. The 'Select Seed Intent' section lists various BGP troubleshooting intents. A red dashed arrow points from the 'Wrapper intent for BGP Troubleshooting bjt002302-SW8' to the 'Diagnosis Tree of Auto NI' window.

The 'Diagnosis Tree of Auto NI' window shows the execution flow of the intent. The flow starts with 'BGP Document and TS' leading to 'BGP Route Reflector ...'. This intent is then executed on multiple devices, including 'JP-TYO-CR01-01', 'SG-SIN-CR01-01', 'UK-LHR-CR01-01', and 'UK-LHR-CR01-01'. The flow continues through 'Check BGP configure ...', 'BGP Neighbor Stabilit...', and 'Check whether B... show ip bgp al...'. The flow ends with 'Ping the route refl...' and 'Configuration'.

Two yellow callout boxes highlight the execution of the BGP Route Reflector Client intent:


- BGP Route Reflector Client intent executed**
- All the intent with BGP tag in wrapper are executed**

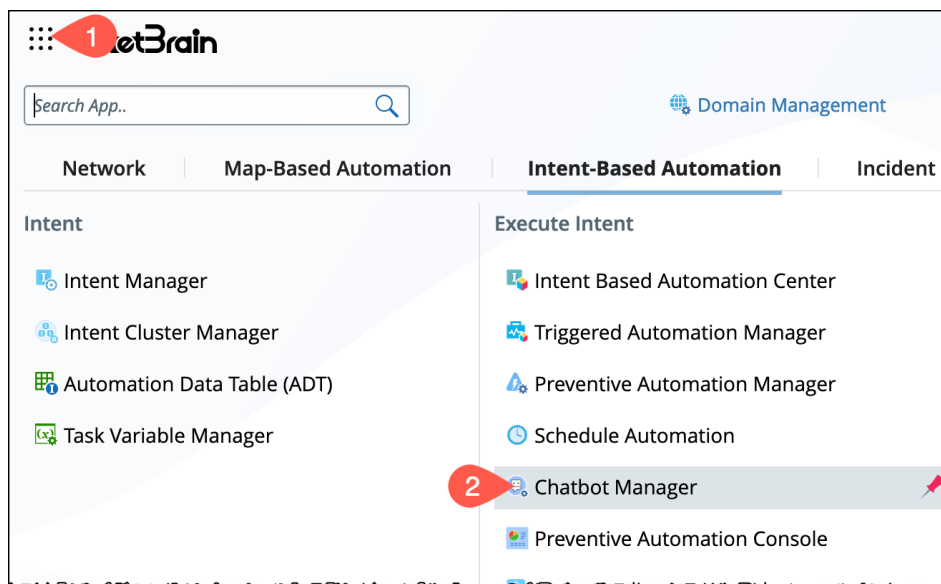
The right side of the interface shows a network map with various devices, including '2439-SW10_TempName...', 'West-CSR1000v', 'US-SAN-R1', 'bjta002439-SW10', 'VRF-PE2', and 'US-Portland-R1'.

9.4 Create a chatbot for the BGP TS wrapper intent

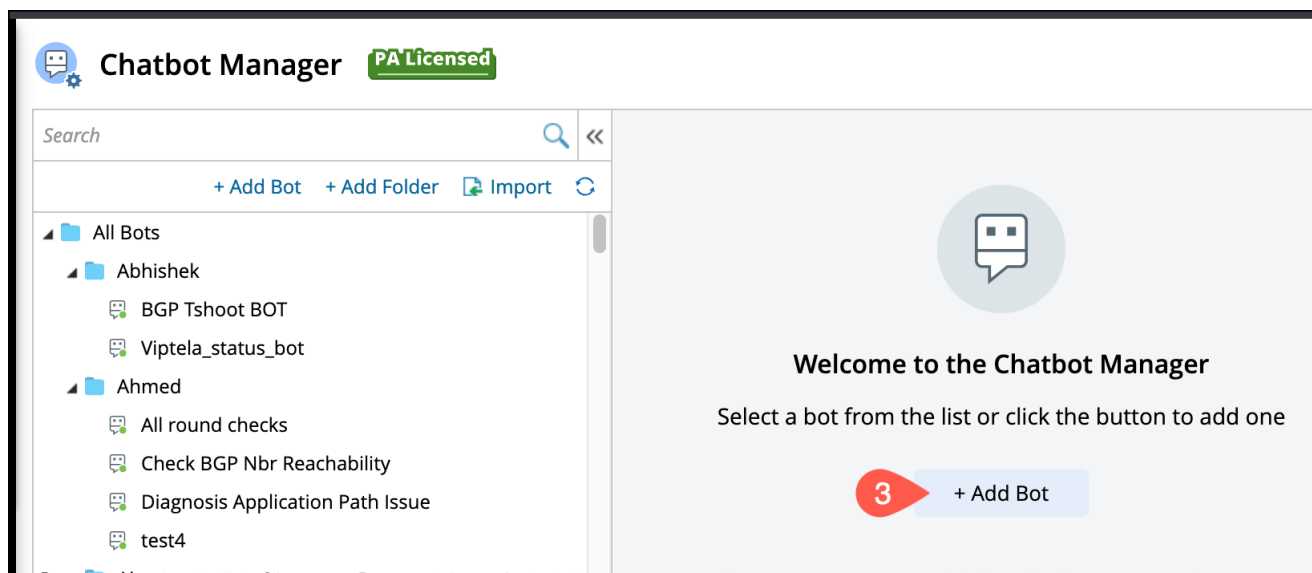
Chatbot allows power users to build interactive chatbots, so end users can execute intent-based automation to solve real-world challenges without accessing NetBrain system UI. Building a chatbot flow is straightforward.

9.4.1 Create the Netbrain Chatbot

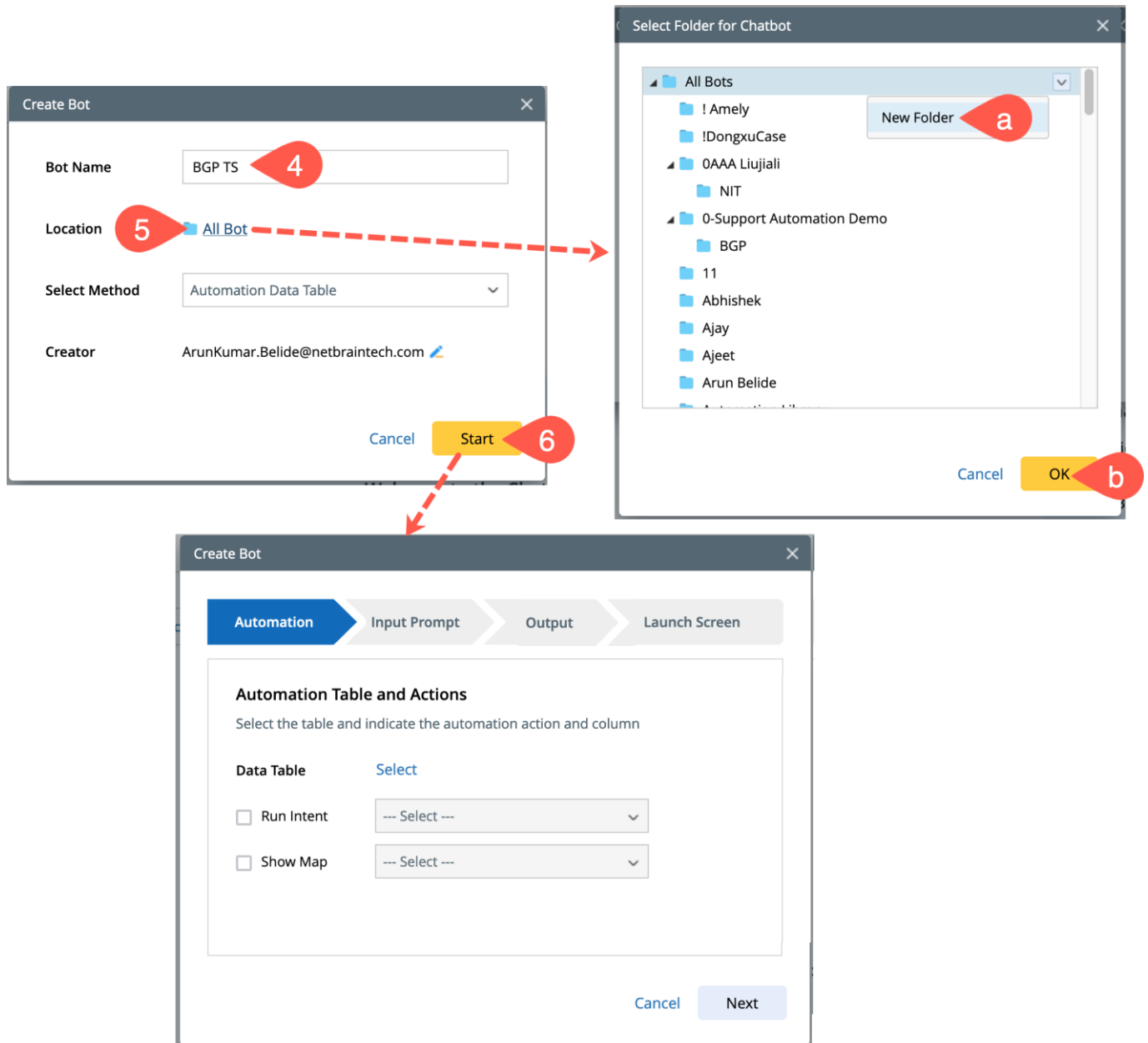
1. In the Upper-Left corner of the desktop, click .
2. Select **Chatbot Manager** under the **Intent-Based Automation** tab.



3. In the **Chatbot Manager**, click **+ Add Bot**.



4. Name the bot **BGP TS**.
5. Location: To choose a location for the bot, click **All Bot**.
 - a) In the dialog, **Select Folder for Chatbot**, select All Bots or create a folder under it using the option **New Folder** from the dropdown menu.
 - b) Click **OK**.
6. Click **Start** to begin the **Chatbot creation wizard**.



7. In the first step **Automation**, click **Select** located next to **Data Table** and choose the ADT **BGP Network**.
 - a) Check the selection box of the **Run Intent** field and select the intent **TS BGP (Wrapper)** from the dropdown.
 - b) Click **Next** to go to the next ribbon Output.

Create Bot

Automation Input Prompt Output Launch Screen

Automation Table and Actions
Select the table and indicate the automation action and column

Data Table My Tables/BGP Network 7

8 ☒ Run Intent ☐ Show Map

Check BGP Config Change

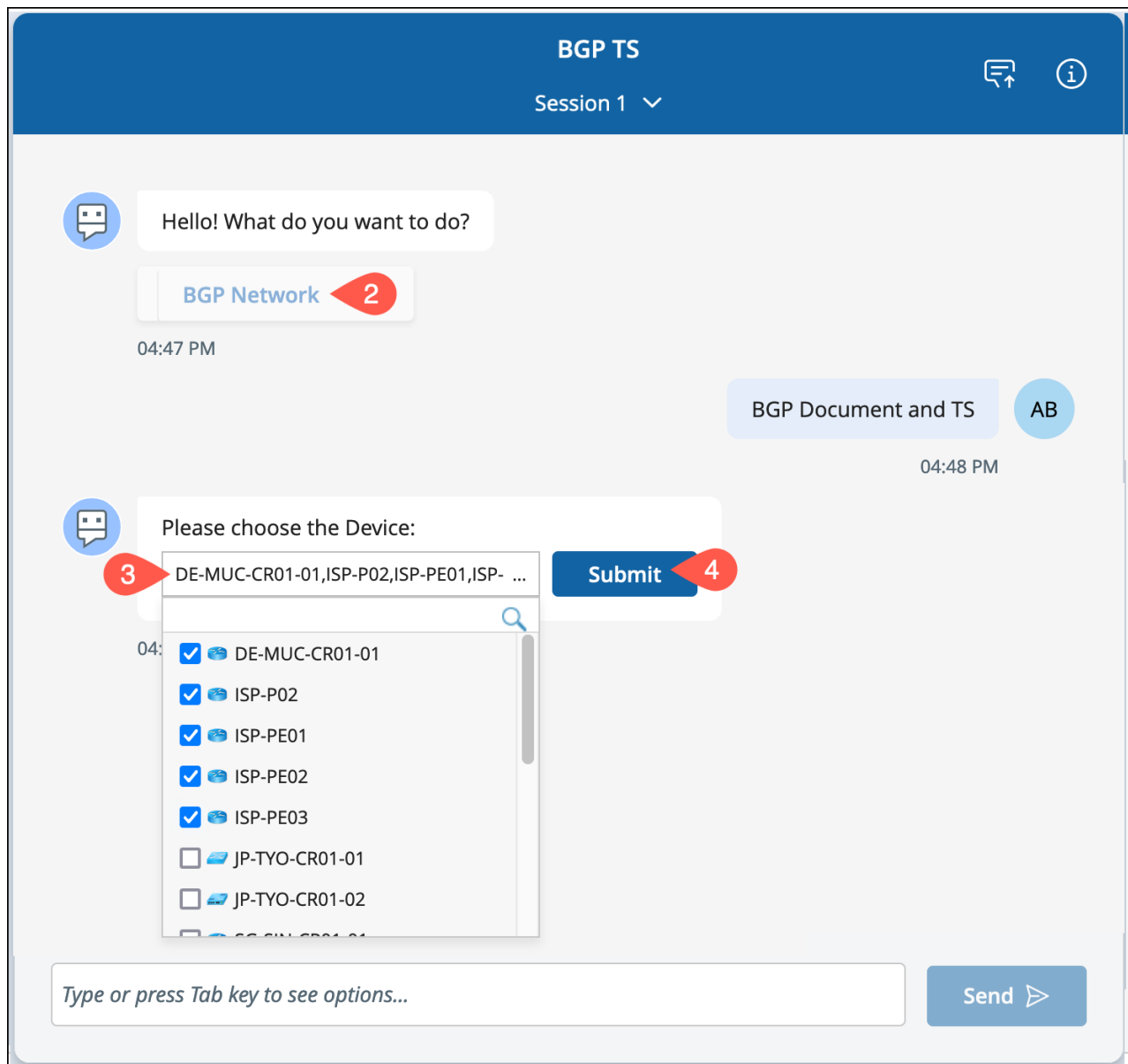
Check BGP Config Change
TS BGP (Wrapper)
BGP Neighbor Check
Check Reflector Client

Cancel Next 9

8. In the second step **Input prompt**, check the **Device** selection box for input and enable the multiple selection option.
9. In the third step **Input prompt**, review the automation output message and customize it if needed using the pencil icon.
10. The fourth step provides the option to personalize the logo and description on the launch screen. Do the modifications if needed or leave the default set up and click **OK**.

9.4.2 Use Chatbot

1. Click **Launch Bot > START**.
2. Click the Intent **BGP Network** (appears as the ADT table is BGP Network).
3. Under **Please choose the Device**, select multiple devices from the drop-down as required.
4. Click **Submit** to execute the intent.



5. The chatbot will execute the intent and provide the diagnosis output.

NOTE: Click the **Map** and **View Diagnosis tree** links to see the error details.

BGP TS

Devices selected for wrapper execution

DE-MUC-CR01-01, ISP-P02 AB

04:55 PM

"Wrapper intent for BGP Troubleshooting ISP-P02" executed.
1 errors found on device ISP-P02.

YES
BGP does not change
ISP-P02 BGP neighbor changed from the baseline
Ping succeeded

Map from Intent: [Intent Output Map](#)
View Diagnosis Tree: [Diagnosis Tree](#)

"Wrapper intent for BGP Troubleshooting DE-MUC-CR01-01" executed.
2 errors found on device DE-MUC-CR01-01.

DE-MUC-CR01-01 BGP neighbor changed from the baseline
NO
BGP configuration changes. Missing lines: None. Extra lines: ro
bgp router-id 192.168.11.254
bgp log-neighbor-changes
redistribute ospf 1 route-map O2B
neighbor 100.20.1.1 remote-as 10000
neighbor 100.20.1.1 description Connection_to_ISP-PE02
neighbor 100.20.1.1 weight 200

Map from Intent: [Intent Output Map](#)
View Diagnosis Tree: [Diagnosis Tree](#)

This is the last step. You can type "restart" to start over.

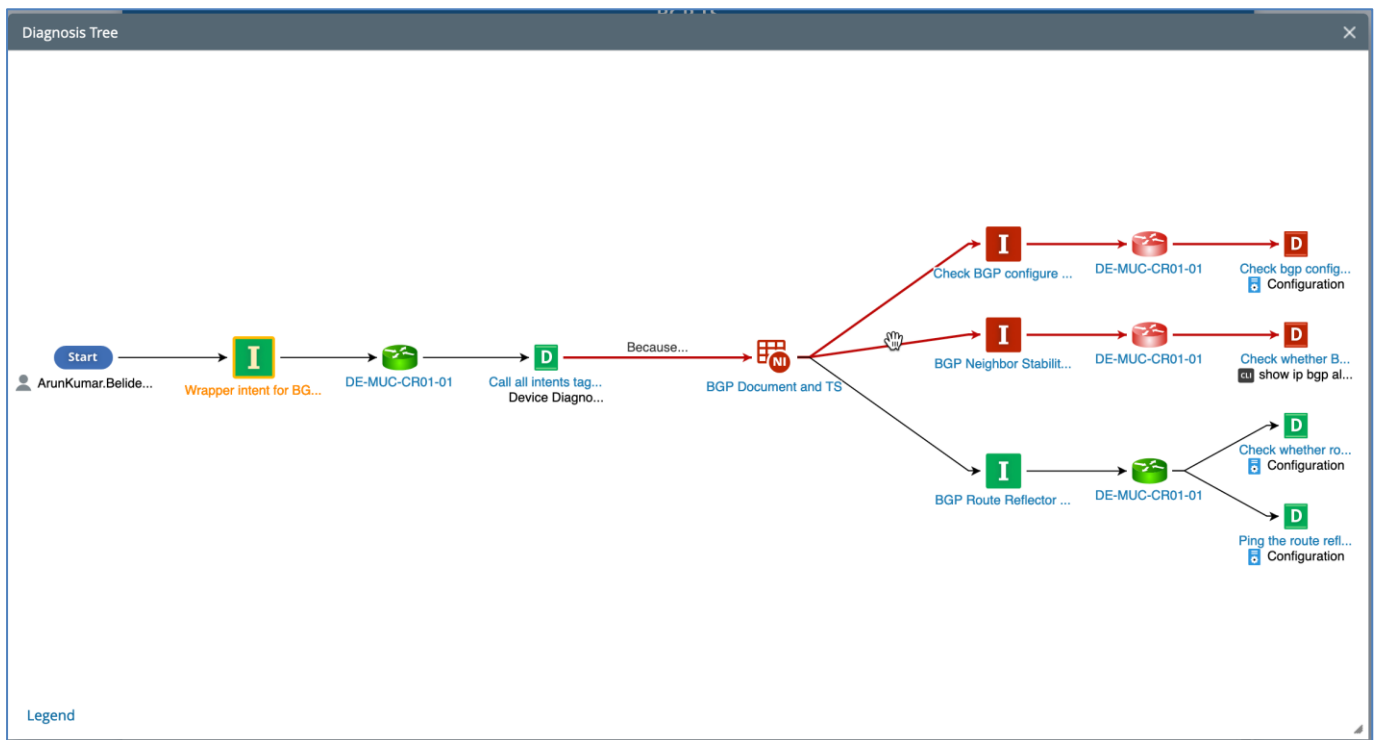
04:55 PM

Type "/" for shortcut commands

Send ➤

Automation output message

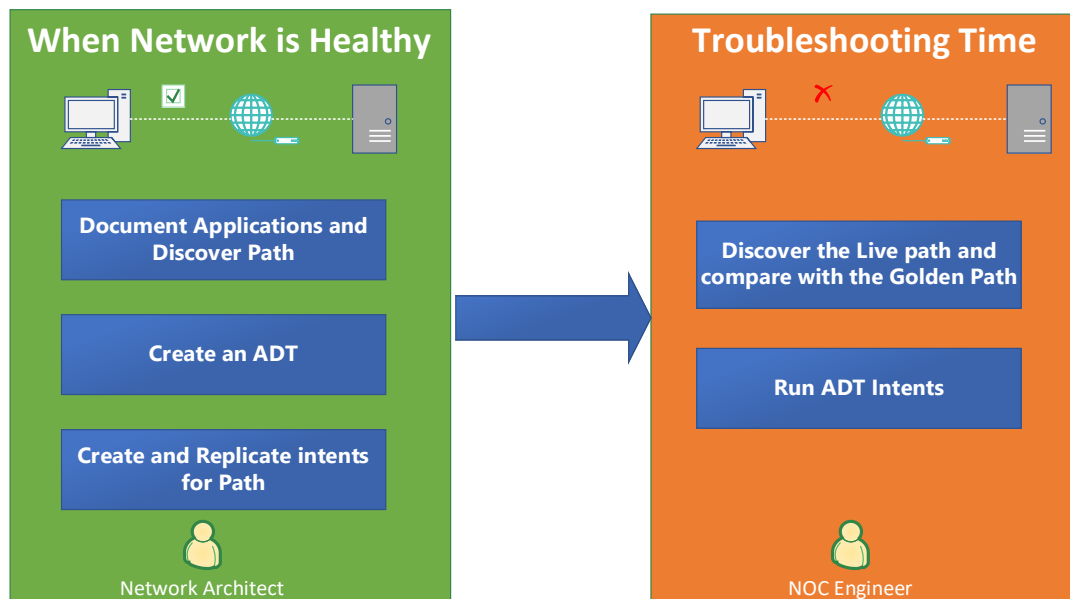
Click Diagnosis Tree to see intents and its follow up in tree view



6. Close all the browser tabs except for the **NetBrain desktop**.

10 Application Path Assessment and Troubleshooting

This chapter explains how to use Intents to troubleshoot path-related network issues. Firstly, you document the network path when it is functioning correctly to establish a Golden Baseline. Next, you will create an ADT for critical applications. Then, you replicate the intents that you think are useful to monitor and troubleshoot the applications to the ADT. And you can create a wrapper intent as the user interface. Finally, when you troubleshoot the path-related issues, you will discover the path and compare it with the Golden baseline for any difference and run the wrapper intent to find any error.



The intent-based path assessment can answer the following questions:

- Is the Path changed visually? Compare the cached path (when the network is healthy), golden baseline path, and live Path in the troubleshooting stage.
- Is the Path failing over programmatically? Use the intent to check for the routing next-hop change and the CAM table (for L2 failover) change.
- Is the Path healthy performance-wise? Use the intent to analyze utilization change, CPU/memory change, link error change, QoS buffer drop change, etc.
- Is Path configured properly? Use the intent to check the QoS configuration consistency across devices of the path and configuration consistency between the failover device pair for the Path.

In this chapter, you will learn how to replicate a seed intent with **path-based replication logic** (you use the device-based replication logic up to now).

10.1 Document the application path when the network is healthy

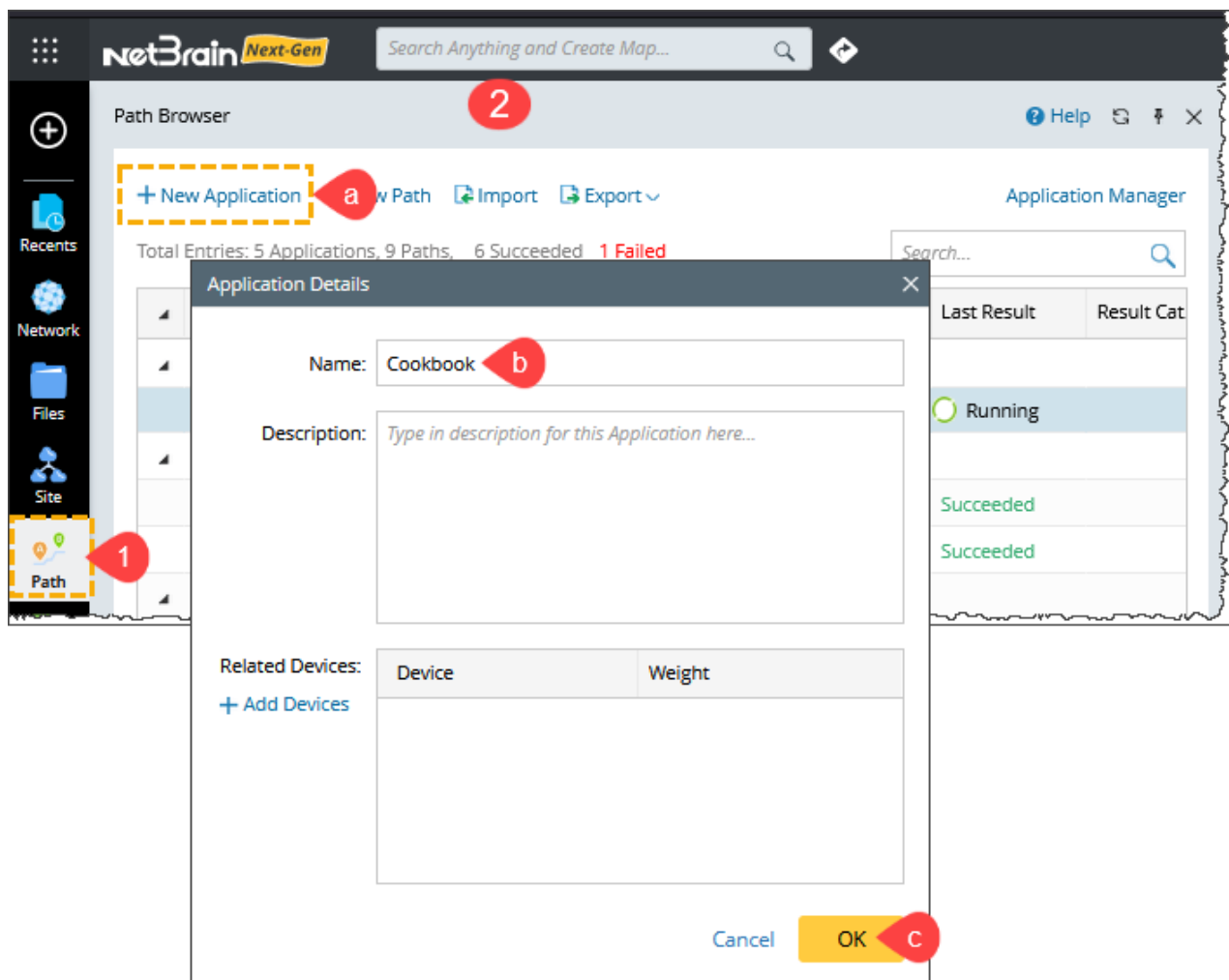
Note: To use the full functions of the **Path Browser**, you need to purchase an **AAM license**.

Otherwise, you cannot create a new application and only can save the path in "Untitled Application".

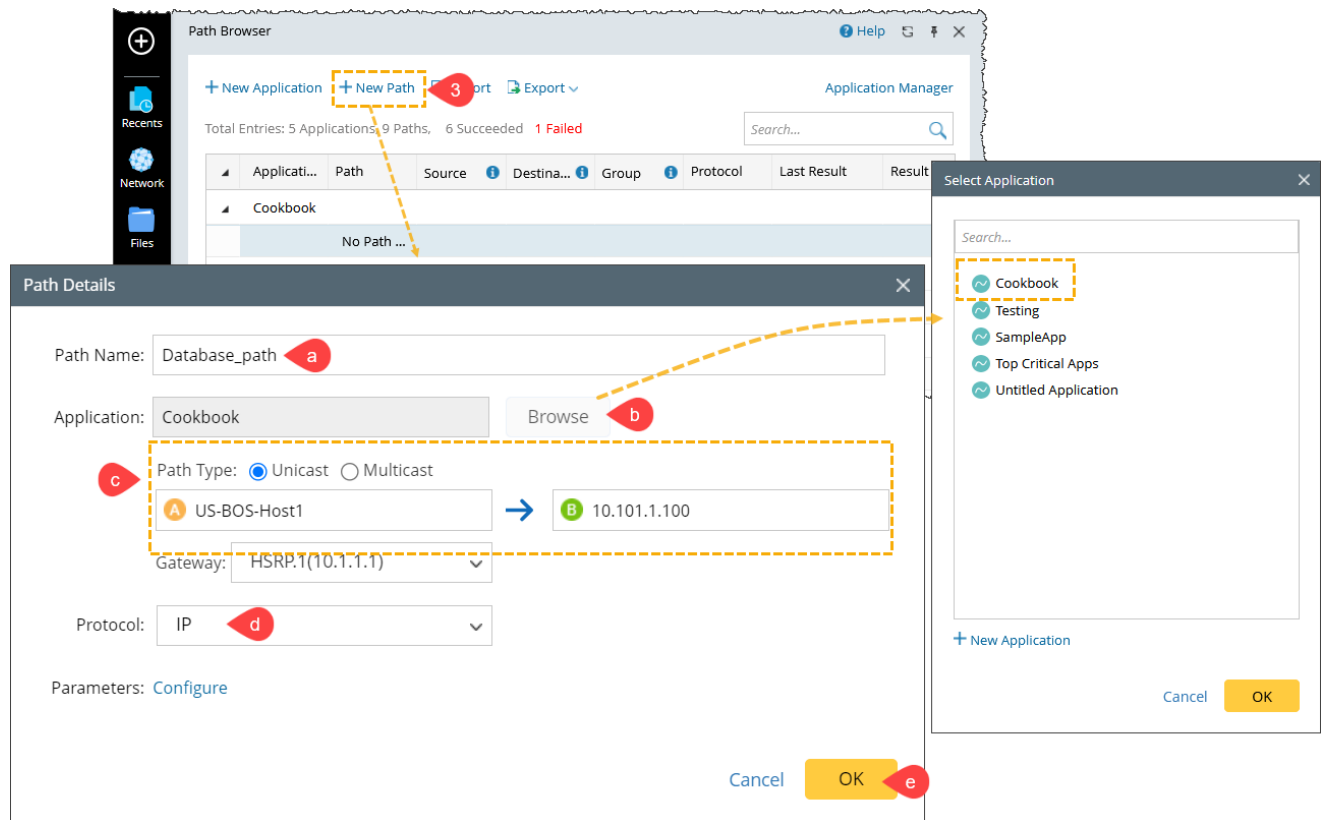
This section teaches how to define a map path for an application using **Path Browser**. You can manage this path based on the Application in the **Path Browser**.

Follow the step-by-step instructions to add paths in the **Path Browser**.

1. In the quick access toolbar, click **Path** to open the **Path Browser** window.
2. Create an Application to add paths in the application.

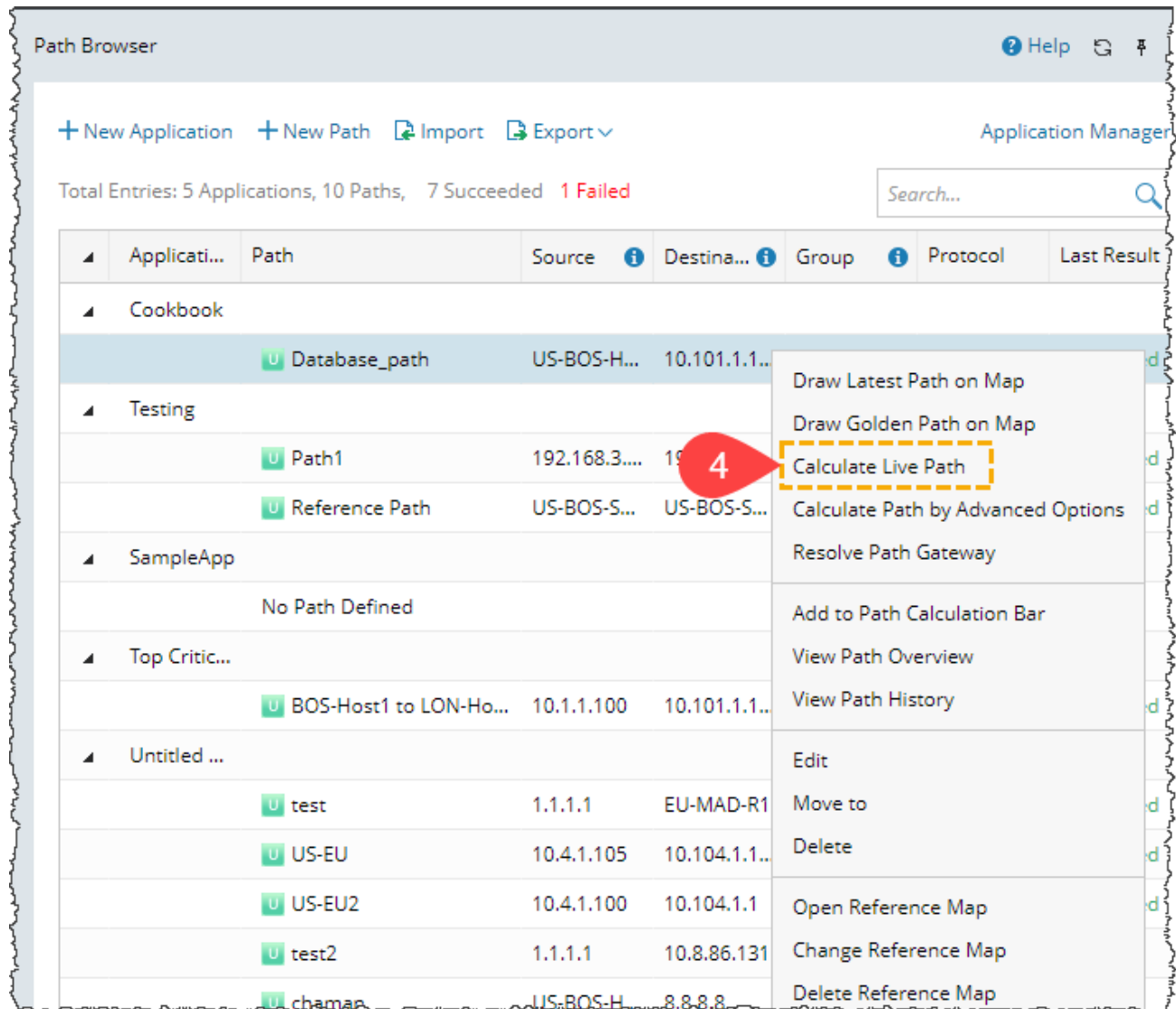


- a) Click **New Application** to open the **Application details** window.
 - b) Enter the application **Name**, i.e., **Cookbook**. The Description is optional.
 - c) Click **OK** to close the window. The **Cookbook application** will be created in the **Application Manager**. You can create a path under this Application.
3. Click **New Path** to define the path in the **Path Details** window.



- a) Enter **Path Name**, i.e., **Database_path**.
- b) Select an application (**Cookbook**) for the path using the **Browse** button.
- c) Enter **Source IP** and **Destination IP Addresses** (e.g., **Source IP: 10.1.1.100**, and **Dest IP: 10.101.1.100**). The related gateways will be auto-identified per your input. You can select the target one from the Gateway list if the device has multiple gateways.
- d) Select a **Protocol** from the **dropdown** list.
- e) Click **OK** to create a path. You can view your created path (**Database_path**) under the **Cookbook** application.

4. Right-click on the created Path and click **Calculate Live Path** from the menu.



You can wait to load the path until you see the **Succeeded** status in the **Last Result** column.

5. Double-click the path to check whether the path is healthy.

The screenshot shows the NetBrain interface. On the left is the 'Path Browser' window, and on the right is the 'Database_path' window showing path details.

Path Browser Table:

Applicati...	Path	Source	Destina...	Group	Protocol	Last Result
Cookbook						
	Database_path	US-BOS-H...	10.101.1.1...		IP	Succeeded
Testing						
	Path1	192.168.3....	192.168.3....		IP	Succeeded
	Reference Path	US-BOS-S...	US-BOS-S...		IP	Succeeded
SampleApp						
	No Path Defined					
Top Critic...						
	BOS-Host1 to LON-Ho...	10.1.1.100	10.101.1.1...		IP	Succeeded
Untitled ...						
	test	1.1.1.1	EU-MAD-R1		IP	Succeeded
	US-EU	10.4.1.105	10.104.1.1...		IP	Succeeded
	US-EU2	10.4.1.100	10.104.1.1		IP	Succeeded
	test2	1.1.1.1	10.8.86.131		IP	N/A
	chaman	US-BOS-H...	8.8.8.8		IP	Failed
	Se_test	10.1.1.100	10.101.1.1...		IP	N/A

Path Details Window (Database_path):

02/08/2024, 19:43:51 (Live) Path Details

Unicast This is Unicast path.

The diagram shows a unicast path from source A (10.1.1.100, US-BOS-Host1(IP)) to destination B (10.101.1.100, 10.101.1.100). The path includes the following hops:

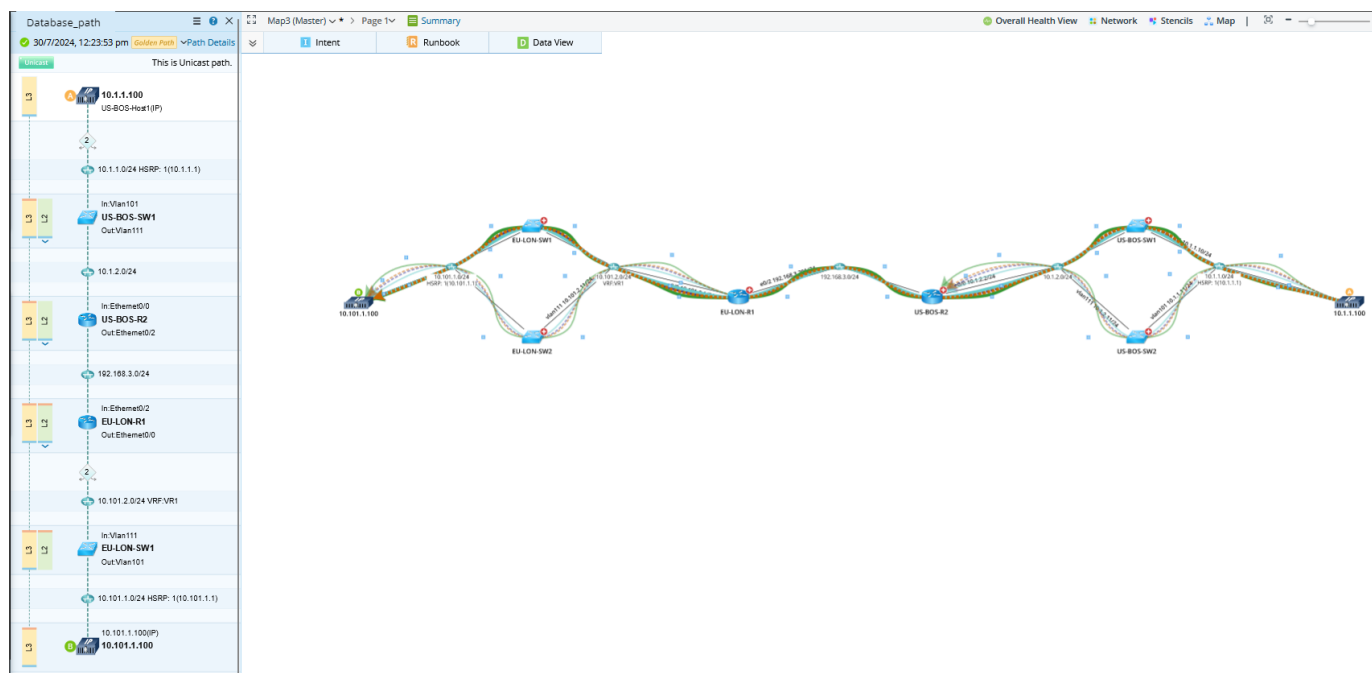
- 10.1.1.0/24 HSRP: 1(10.1.1.1)
- In:Vlan101, US-BOS-SW1, Out:Vlan111
- 10.1.2.0/24
- In:Ethernet0/0, US-BOS-R2, Out:Ethernet0/2
- 192.168.3.0/24
- In:Ethernet0/2, EU-LON-R1, Out:Ethernet0/0
- 10.101.2.0/24 VRF:VR1
- In:Vlan111, EU-LON-SW1, Out:Vlan101
- 10.101.1.0/24 HSRP: 1(10.101.1.1)

6. Click **Path Details** to view the path details for each hop and define the reference map.

- a) Select the **Reference Map** from the **Select Map** window.

Note: You can define a reference map for a saved path. The path reference map allows you to draw the golden baseline path on the map and use device notes to draw key configlets or troubleshooting recommendations on the map.

- b) Click **+ Add Note** in the bottom-right corner, enter a note and click **Save**.
- c) Set this path as a **Golden Path**.



10.2 Build and Manage Automations for Path

In this section, you will create a base ADT from the Application path. Furthermore, you will use the **Intent Replication Wizard** to add several Intents and a wrapper Intent to the ADT. Additionally, you will create a wrapper Intent for path-related automation and enable the auto intent functionality.

10.2.1 Create the ADT base table from the path browser

The system provides the method **Application Path** for you to import the application and path data into the ADT base table.

Follow the step-by-step instructions to build the base table with the data in devices:

1. Build the new base table by entering the Name, Path Stability Check and select the Location as per your preference.
2. Click **Table Builder** and define the base table using the method **Application Path**.
3. You can build an ADT table with the following columns: **Application Name, Path, Source, Destination, Path Devices, Path Status, and Path Hops**.

Automation Data Table Builder

3

Column Header:

Application Name

Path

Source

Destination

Path Status

Path Hops (Device Inter...

Reset All

Base

Description:

Path Stability Check

Select Method to Build Base Table:

Application Path

☐ All Applications except "Untitled Application"
 ☒ Specified Applications (3 Items)

Built-in Fields:

Path

Application Name

Path Name

Path Devices

Path Hops (Device Inte...

Inbound Interfaces

Outbound Interfaces

Path Map

Source

Column Group (Base):

Application Name

Path

Source

Destination

Path Status

Path Hops (Device I...

Select Column

Select Applications and Paths

Search...

☐ Top Critical Apps

☒ BOS-Host1 to LON-Host1(10.1.1.100 to 10.101.1.100)

☐ SampleApp

☐ Testing

☒ Path1(192.168.3.10 to 192.168.3.201)
 ☒ Reference Path(US-BOS-SW1 to US-BOS-SW2)
 ☒ Cookbook

☒ Path_Troubleshooting(US-BOS-Host1 to 10.101.1.100)

Select All Deselect All

Cancel OK

Auto-Build No Scheduled Update

Cancel Save Save and Build

The ADT will be:

Path Stability Check											
		Table Builder	Last Updated at: 08/09/2024 12:57 PM		Rebuild Table	Add Data Manual					
Description: Type description here...											
Items: 4 Rows 11 Columns											
No.	Application Name	Path	Path Devices	Source	Destination	Path Status	Path Hops (Device In...				
1	Cookbook	Database_path	10.1.1.100...	US-BOS-Host1	10.101.1.100	Succeeded	EU-LON-R1 - Ethernet0/...				
2	Testing	Path1	192.168.3....	192.168.3.10	192.168.3.201	Succeeded	192.168.3.10 - Ethernet...				
3	Testing	Reference Path	US-BOS-S...	US-BOS-SW1	US-BOS-SW2	Succeeded	US-BOS-SW1 - Loopback...				
4	Top Critical Apps	BOS-Host1 to LON-Host1	10.1.1.100...	10.1.1.100	10.101.1.100	Succeeded	EU-LON-R1 - Ethernet0/...				

388 | NetBrain R11.1b

10.2.2 Replicate the Intents of Device Health Check

In this section, you will replicate all the intents created in Chapter 3 for Basic Device Health Check. The intents you will replicate in the ADT will be:

- CPU usage check (Cisco IOS).
- Interface status check (Cisco IOS).
- Uptime check (Cisco IOS).

NOTE: Ensure when you use Intent Replication Wizard to replicate intent, define Path-based Replication instead of Device based replication.

10.2.2.1 Replicate CPU Usage Check (Cisco IOS) Intent to the ADT

1. From the Intent, go to the menu and click **Intent Replication Wizard**.
2. Seed Intent. Select the **Path-based Replication** option.

Intent Replication Wizard - CPU Usage Check (Cisco IOS)

Seed Intent 2 Define ADT Replication Settings

Seed Intent: CPU Usage Check (Cisco IOS) Select Last Modified

Intent Template for: ☐ Device-based Replication ☒ Path-based Replication a

3. Define ADT.

Intent Replication Wizard - CPU Usage Check (Cisco IOS)

Seed Intent Define ADT **3** Replication Settings

Create a New ADT Use an Existing ADT **a**

Automation Data Table: Path Stability Check

Replicate Intent to: New Column Group check cpu **b**

Replicate on Path Column: Path Path **c**

Selection Mode: Path-based Replication.

Select Automation Data Table

Search...

- Shared Tables
 - ADT_Micro_Learning
 - Assessment Reference ADT
 - Cloud Assessment Reference ADT
- Cookbook
 - Path Stability Check**
- Instructor Materials
- NBU_Materials

Cancel OK

4. Replication Settings.

Intent Replication Wizard - CPU Usage Check (Cisco IOS)

Seed Intent Define ADT Replication Settings **4** Replicate Intent

Full Settings for Template

Define Macro Variables and Rules for Their Substitution:

Item: 1

Seed Device	Seed Command	Macro Variables	Command Qualification
US-BOS-SW1	show process cpu		Defined

More Replication Settings

Selection Mode: Path-based Replication, ADT: Path Stability Check.

Previous Next

5. Replicate Intent and add the tag application or path.

Intent Replication Wizard - CPU Usage Check (Cisco IOS)

Seed Intent Define ADT Replication Settings Replicate Intent 5

ADT Columns:

Column Data	Column Name	Tag
1 Path Intent	CPU Usage	1 tag

Tag Current Column

1 Tag: path

Save and Replicate

submitted at: 07/30/2024 04:31 PM

Open Output ADT

Selection Mode: Path-based Replication, ADT: Path Stability Check, 0 Macro Variables.

Previous Finish

The ADT after adding **CPU Usage** Intent:

Path Stability Check Table Builder Last Updated at: 08/09/2024 12:57 PM Rebuild Table Add Data Manually

Description: Type description here...

Items: 4 Rows 8 Columns

Path	Path Devices	Source	Destination	Path Status	Path Hops (Device In...)	CPU Usage
Database_path	10.1.1.100...	US-BOS-Host1	10.101.1.100	Succeeded	EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS...)
Path1	192.168.3.10...	192.168.3.10	192.168.3.201	Succeeded	192.168.3.10 - Ethernet...	CPU Usage Check (Cisco IOS...)
Reference Path	US-BOS-S...	US-BOS-SW1	US-BOS-SW2	Succeeded	US-BOS-SW1 - Loopback...	CPU Usage Check (Cisco IOS...)
BOS-Host1 to LON-Host1	10.1.1.100...	10.1.1.100	10.101.1.100	Succeeded	EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS...)

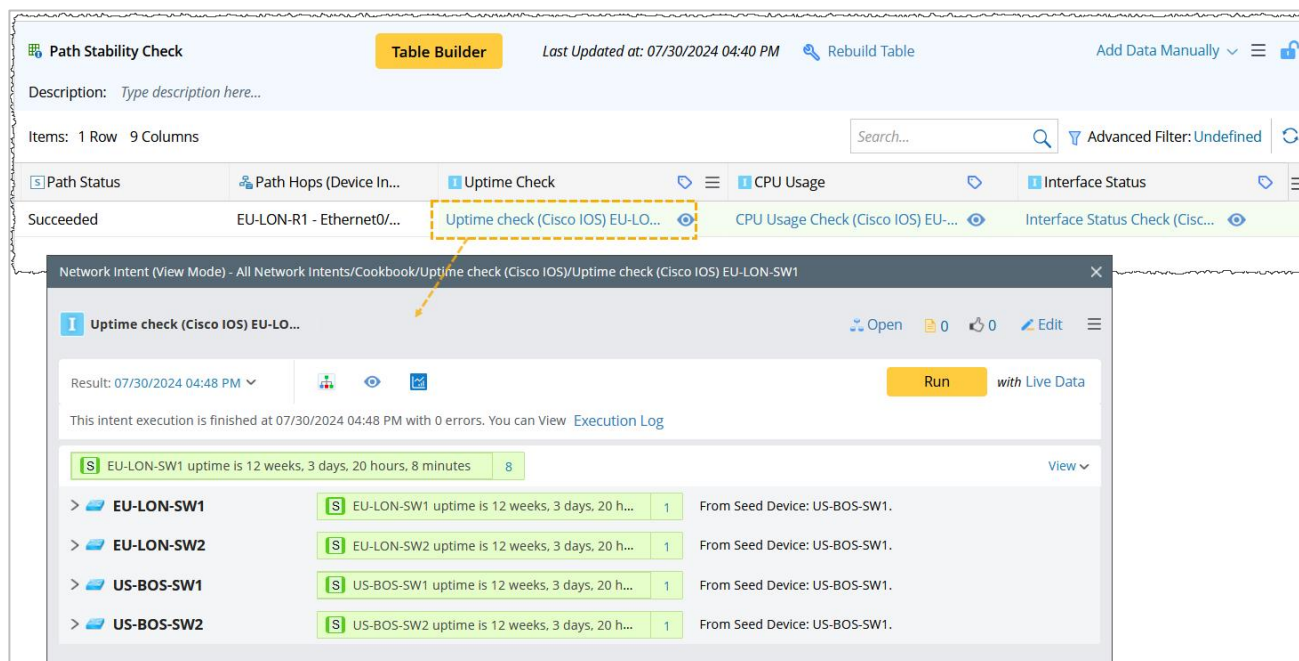
Similarly, add another two Intents (**Interface Status Check** and **Uptime check**) via Intent Replication Wizard.

Table Builder Last Updated at: 08/09/2024 12:57 PM Rebuild Table Add Data Manually

Search... Advanced Filter: Undefined

Status	Path Hops (Device In...)	CPU Usage	Uptime Check	Interface Status
ded	EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) EU-LO...	Interface Status Check (Cisc...
ded	192.168.3.10 - Ethernet...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) US-BO...	Interface Status Check (Cisc...
ded	US-BOS-SW1 - Loopback...	CPU Usage Check (Cisco IOS) US-...	Uptime check (Cisco IOS) US-BO...	Interface Status Check (Cisc...
ded	EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) EU-LO...	Interface Status Check (Cisc...

- Open a replicated intent to check whether all devices in the path are replicated.



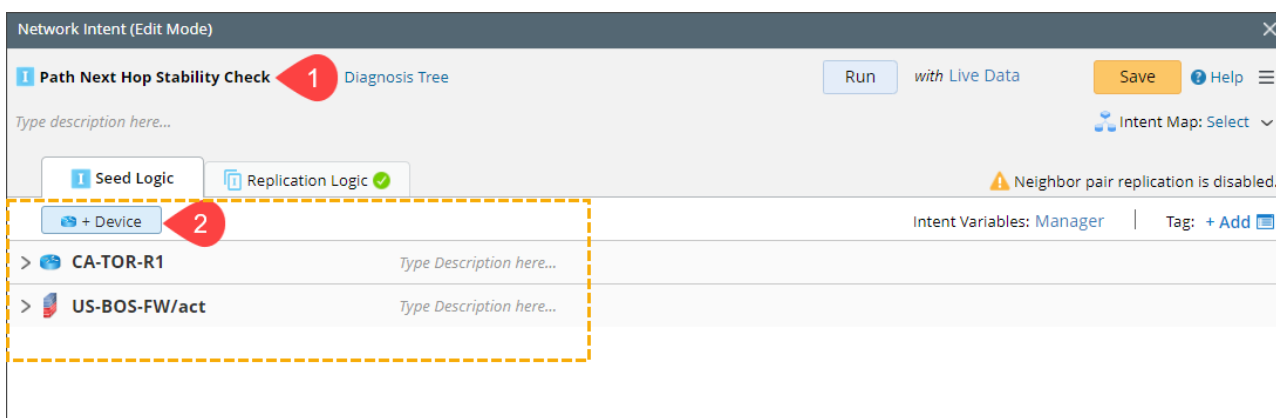
- Run Intent for all the Intent Columns and Rebuild Table.

10.2.3 Create Path Intent for Path Next Hop Stability check

In this section, you will create an intent to check whether the next hop changes for the path. You will use two seed devices of the different types to support the multi-vendor.

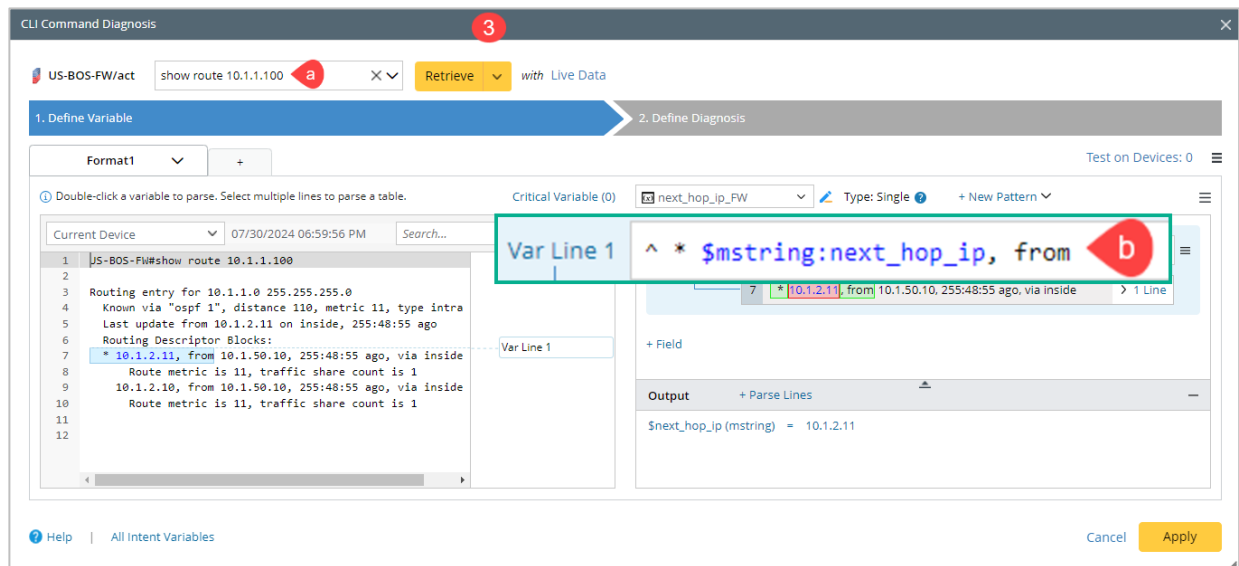
- Create an Intent from the Intent Manager with Name, **Path Next Hop Stability Check**.
- Add Seed devices.

Let us add two types of network devices, **Cisco Firewall** and **Cisco IOS Device**, to learn how to support the multi-vendor. Different CLI commands are used to retrieve the next hop for these devices.



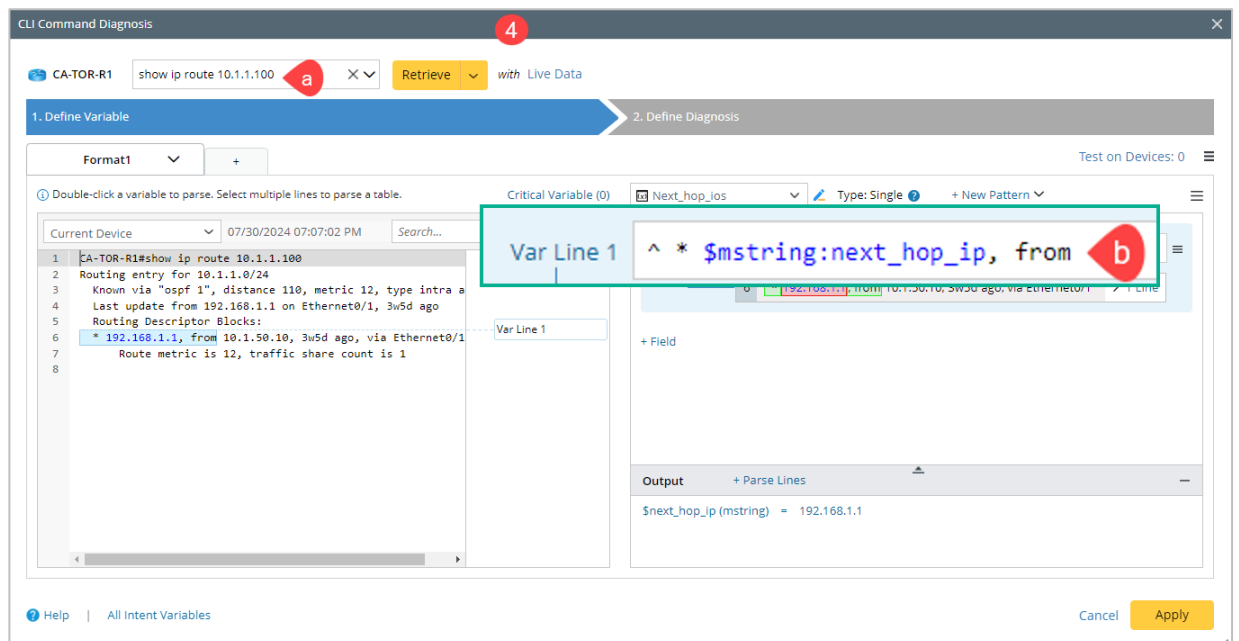
3. Define the Variable for the **Cisco Firewall** device.

- Define parser using **CLI Diagnosis**. The command will be: ***show route 10.1.1.100***.
- The Var Line 1 will be: ***^ * \$mstring:next_hop_ip, from***.



4. Define Variable for the **Cisco IOS Device**.

- Define parser using **CLI Diagnosis**. The command will be: ***show ip route 10.1.1.100***.
- The Var Line 1 will be: ***^ * \$mstring:next_hop_ip, from***.



5. Define Diagnosis for **Cisco Firewall**.

The condition **A** will be: The **Current next_hop_ip** does not equal the last next_hop_ip. And Define the Diagnosis message.

2. Define Diagnosis

5

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: check next hop ip for FW Anchor:

Type description of the diagnosis...

☐ Loop Table Rows

▼ If

A US-BOS-F... Current Last

next_hop_ip Does not equal next_hop_ip

B Select Variable

Current
Baseline
Last

6. Define Diagnosis for **Cisco IOS Device**.

The **If** condition **A** will be: The **Current next_hop_ip** dose not equal the last next_hop_ip. And Define the Diagnosis message.

2. Define Diagnosis

6

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: Next hope Ip check for ios device Anchor:

Type description of the diagnosis...

☐ Loop Table Rows

▼ If

A CA-TOR-R1 Current Last

next_hop_ip Does not equal next_hop_ip

B Select Variable

Current
Baseline
Last

10.2.3.1 Replicate the Intent to ADT

In this section, you will use the **Intent Replication Wizard** to replicate the intent, **Path Stability Check**. In the replication settings tabs, you will define the **destination IP** as **Macro Variable** and set it as the path destination.

1. In the Seed Intent tab, select the **Path-based Replication** option.
2. Define ADT: **Use an Existing ADT** and select **Path Stability Check**. Enter a relevant Intent group name here.
3. Replication Settings:

Replication Settings 3 Replicate Intent

Full Settings for Intent Template

☒ Serve as Template for: ☐ Device-based Replication ☒ Path-based Replication ☐ Enable Neighbor Pair Replication

Intent Qualification Macro Variable Critical Variable Advance Settings

0 Items + Device Variable + Command Variable b

Seed Device	Macro Variable	Source	Type	Default Value	Look up Data for Device
CA-TOR-R1					

Add Command Variable

cu show ip route 10.1.1.100

Cancel OK

Define Command Variable

Command: show ip route 10.1.1.100

Variable Pattern: show ip route \$dest_ip c

Tip: You can define a macro variable by replacing the string with \$var_name.
Example: show ip ospf 1 -> show ip ospf \$process_id

\$dest_ip:

Description:

Type: string

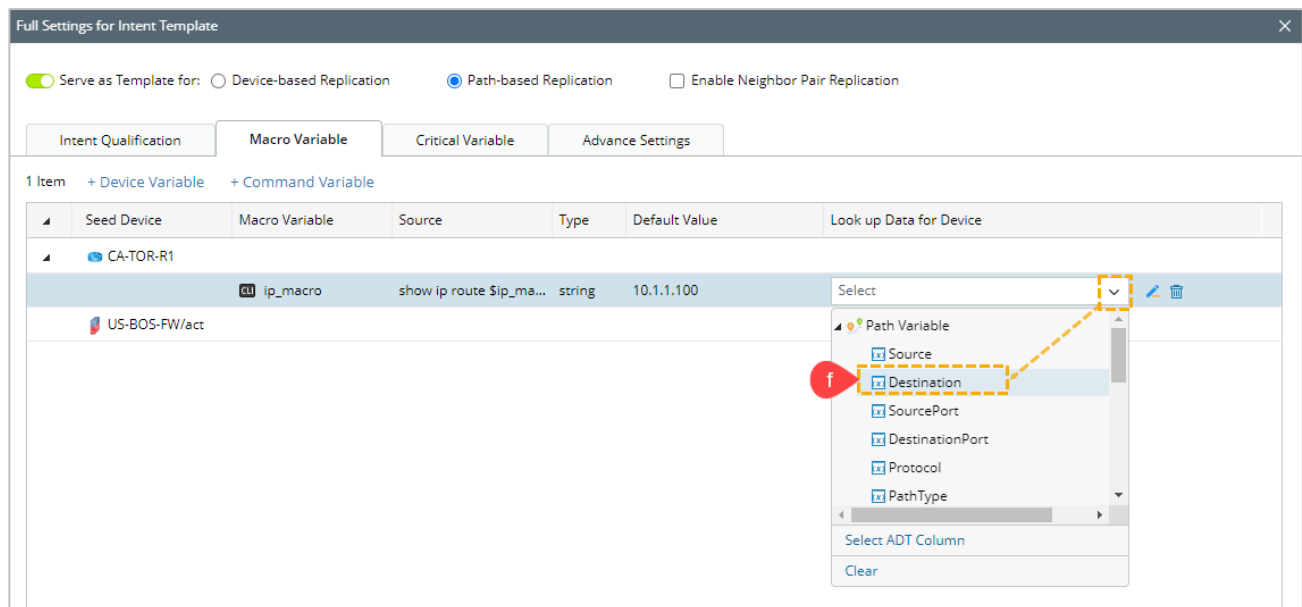
* Default Value: 10.101.1.100 d

Prompt for Input: dest_ip

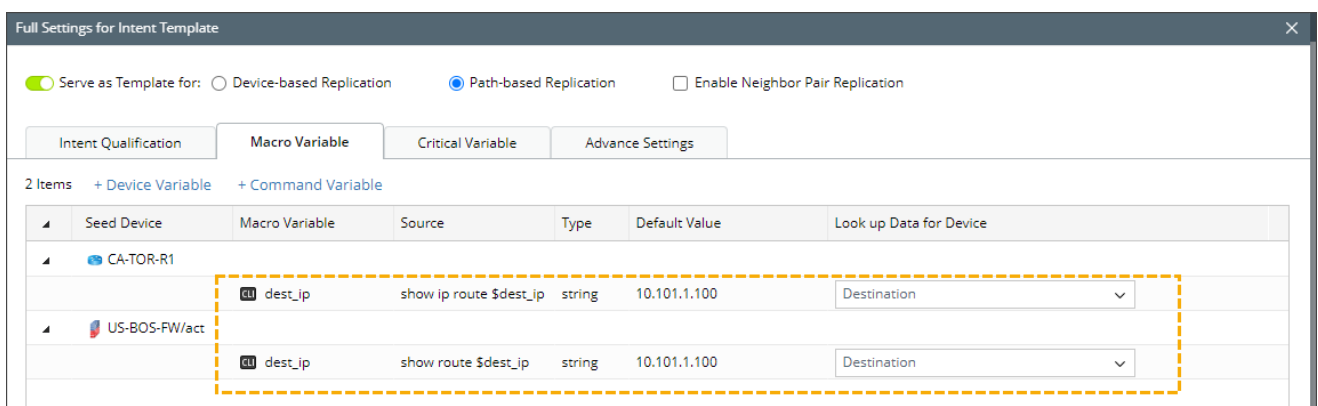
Hint: Set value...

Cancel OK e

- Click **Full Settings for Template** to define **Macro Variable** for the Cisco Device.
- Click **+ Command Variable** to open the **Add Command Variable** window, and double click the CLI command, i.e., **show ip route 10.1.1.100**.
- Define the command variable by replacing the command, **show ip route 10.1.1.100** with **show ip route \$dest_ip**.
- Enter the Default Value, i.e., destination IP, **10.101.1.100**.
- Click **OK** to add Macro Variable.
- Select **Destination** from the dropdown to define look-up data for the device.

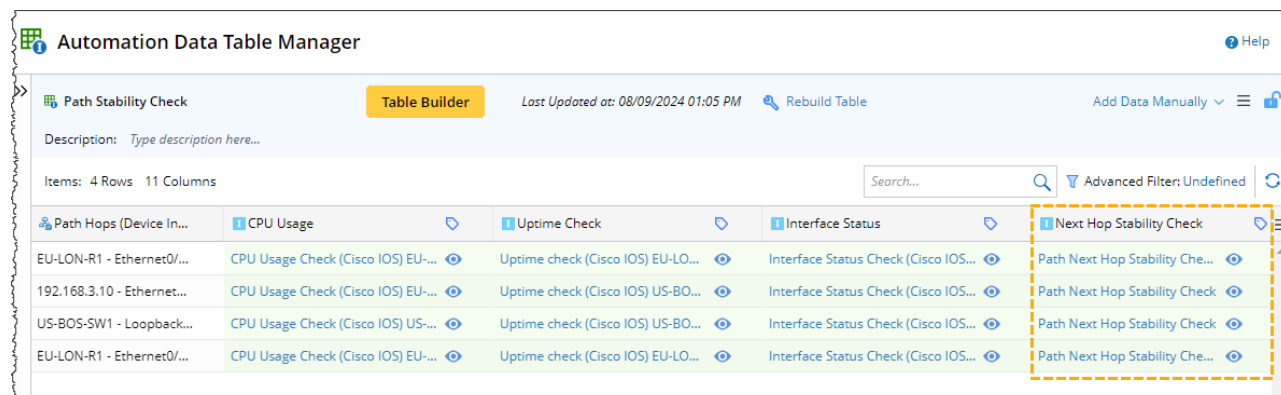


- Similarly, define Macro Variable for Cisco Firewall. The added macro variable will be:



4. Replicate Intent:

Define the column name and tag for the Intent. The added automation column will be:



Automation Data Table Manager

Path Stability Check Table Builder Last Updated at: 08/09/2024 01:05 PM Rebuild Table Add Data Manually

Description: Type description here...

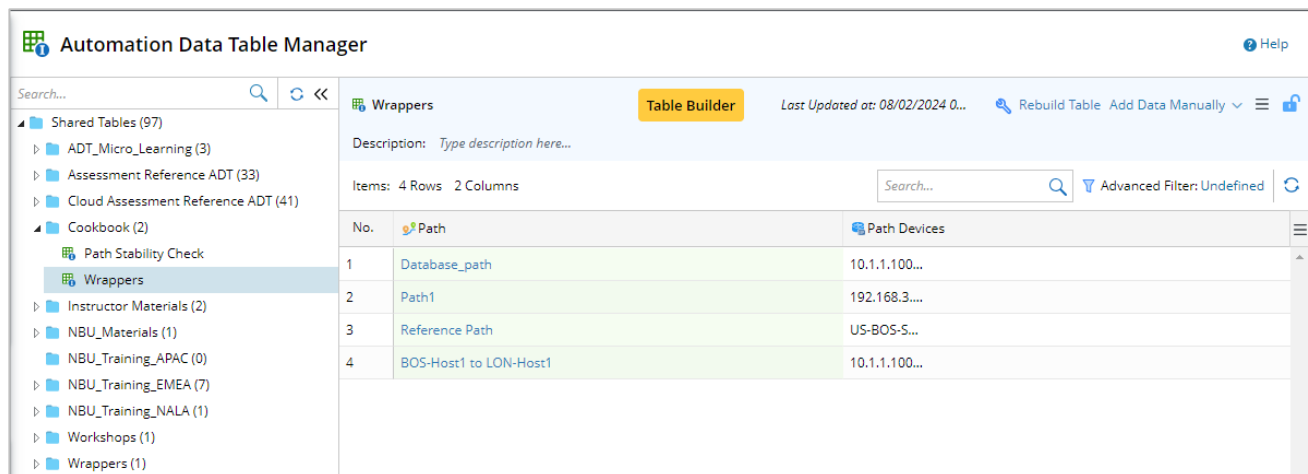
Items: 4 Rows 11 Columns Search... Advanced Filter: Undefined

Path Hops (Device In...	CPU Usage	Uptime Check	Interface Status	Next Hop Stability Check
EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) EU-LO...	Interface Status Check (Cisco IOS...	Path Next Hop Stability Che...
192.168.3.10 - Ethernet...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) US-BO...	Interface Status Check (Cisco IOS...	Path Next Hop Stability Check
US-BOS-SW1 - Loopback...	CPU Usage Check (Cisco IOS) US-...	Uptime check (Cisco IOS) US-BO...	Interface Status Check (Cisco IOS...	Path Next Hop Stability Check
EU-LON-R1 - Ethernet0/...	CPU Usage Check (Cisco IOS) EU-...	Uptime check (Cisco IOS) EU-LO...	Interface Status Check (Cisco IOS...	Path Next Hop Stability Che...

10.2.4 Create Wrapper Intent and Enable Auto Intent

In this section, you will create a Wrapper Intent as a public interface for the path-related intents. For detailed steps on how to create a wrapper intent, please refer to Section 4.2. A key difference here is that you will use the **Intent Diagnosis** method instead of the Device Diagnosis method since the Device Diagnosis method will not replicate all path devices. Also, you need to create a separate ADT for the **Wrapper Intent**.

1. Create an ADT for Wrapper Intent using the **Application Path** Method. Add the **Path** and **Path Devices** built-in fields to the ADT.



Automation Data Table Manager

Wrappers Table Builder Last Updated at: 08/02/2024 0... Rebuild Table Add Data Manually

Description: Type description here...

Items: 4 Rows 2 Columns Search... Advanced Filter: Undefined

No.	Path	Path Devices
1	Database_path	10.1.1.100...
2	Path1	192.168.3....
3	Reference Path	US-BOS-S...
4	BOS-Host1 to LON-Host1	10.1.1.100...

2. Create Wrapper Intent from **Intent Manager** with name, **Wrapper for Path**.
3. Add **Follow-up intent** by selecting **Select Intents from ADT** option.

Network Intent (Edit Mode)

3

Wrapper for path | Diagnosis Tree

Run | with Live Data | Save | Help

Type description here...

Seed Logic | + Device

Intent Diagnosis Block Definition

Diagnosis1

Name: Path Troubleshooting **b**

Description:

☐ Loop Table Rows

If

A True **c**

B Select Variable

Then

Follow-up Intent: Network Intent

Add Logic

- Intent Status Code
- Draw Map
- Send Email
- Follow-up Intent **d**
- Advanced

Follow-up Intents

0 Follow-up Intents: + Follow-up

- Select Intent (Standalone)
- Select Intent Template
- Select Intents from ADT
- Select Intent Cluster

Select Automation Data Table

Search...

- Shared Tables
 - ADT_Micro_Learning
 - Assessment Reference ADT
 - Cloud Assessment Reference ADT
 - ☒ Cookbook
 - ☒ Path Stability Check
 - Instructor Materials
 - NBU_Materials
 - NBU_Training_EMEA

Intent Settings

Intent Variables

Full Settings for Template

Lock Settings

Add Intent Diagnosis Block **a**

Add Diagnosis via Auto Intent

Switch Devices

Define Abstract

Named Tag

Export

Save as

View Original Text for Diagnoses

View Summary Text for Diagnoses

Publish Intent

Intent Replication Wizard

Auto Intent Wizard

4. In the Follow-up Intents window:

Follow-up Intents

1 Follow-up Intent: + Follow-up ⓘ

▼ Path Stability Check (Automation Data Table)

Description: When...

Find Critical Automation Assets (ADT Rows) by Intent Variables (Device or Device Variables):

Find critical assets by device, or properties of critical asset.

a A Application Name Is not none

B Select...

Boolean Expression: A

☒ **Select Intents of Found Critical Automation Assets to Execute:**

☐ All Intents

☐ Selected Intents: Select...

b ☒ Intents with Tags: Match all path

☐ Prune other follow-up intents

Cancel **c** Save

- Define condition **A: Application Name is not none**.
- Select the **Intents with Tags** option to Match all **path** tags.
- Click **Save** to save the wrapper intent.

5. Replicate this Wrapper Intent to the ADT using **Path-based Replication**.

Automation Data Table Manager

Search...

Shared Tables (97)

- ADT_Micro_Learning (3)
- Assessment Reference ADT (33)
- Cloud Assessment Reference ADT (41)
- Cookbook (2)
 - Path Stability Check
 - Wrappers**
- Instructor Materials (2)
- NBU_Materials (1)
- NBU_Training_APAC (0)
- NBU_Training_EMEA (7)
- NBU_Training_NALA (1)
- Workshops (1)

Wrappers Table Builder Last Updated at: 08/02/2024 0... Rebuild Table Add Data Manually

Description: Type description here...

Items: 4 Rows 3 Columns Search... Advanced Filter: Undefined

No.	Path	Path Devices	Wrapper Intent
1	Database_path	10.1.1.100...	Wrapper for path Path_Troubles...
2	Path1	192.168.3....	Wrapper for path Path1
3	Reference Path	US-BOS-S...	Wrapper for path Reference Path
4	BOS-Host1 to LON-Host1	10.1.1.100...	Wrapper for path BOS-Host1 to ...

6. **Run** the Wrapper Intent and check the diagnosis tree for all the replicated devices.

Network Intent (View Mode) - All Network Intents/Cookbook/New Wrapper

Wrapper for path Open 0 0 Edit

Result: 08/02/2024 12:26 PM Run with Live Data

This intent execution is finished at 08/02/2024 12:26 PM with 0 errors. You can View Execution Log

S on EU-LON-R1, Interface Ethernet0/3 is down, Current L1 status is administratively down and L2 status is down 66 View

Diagnosis Message (08/02/2024 12:26 PM)

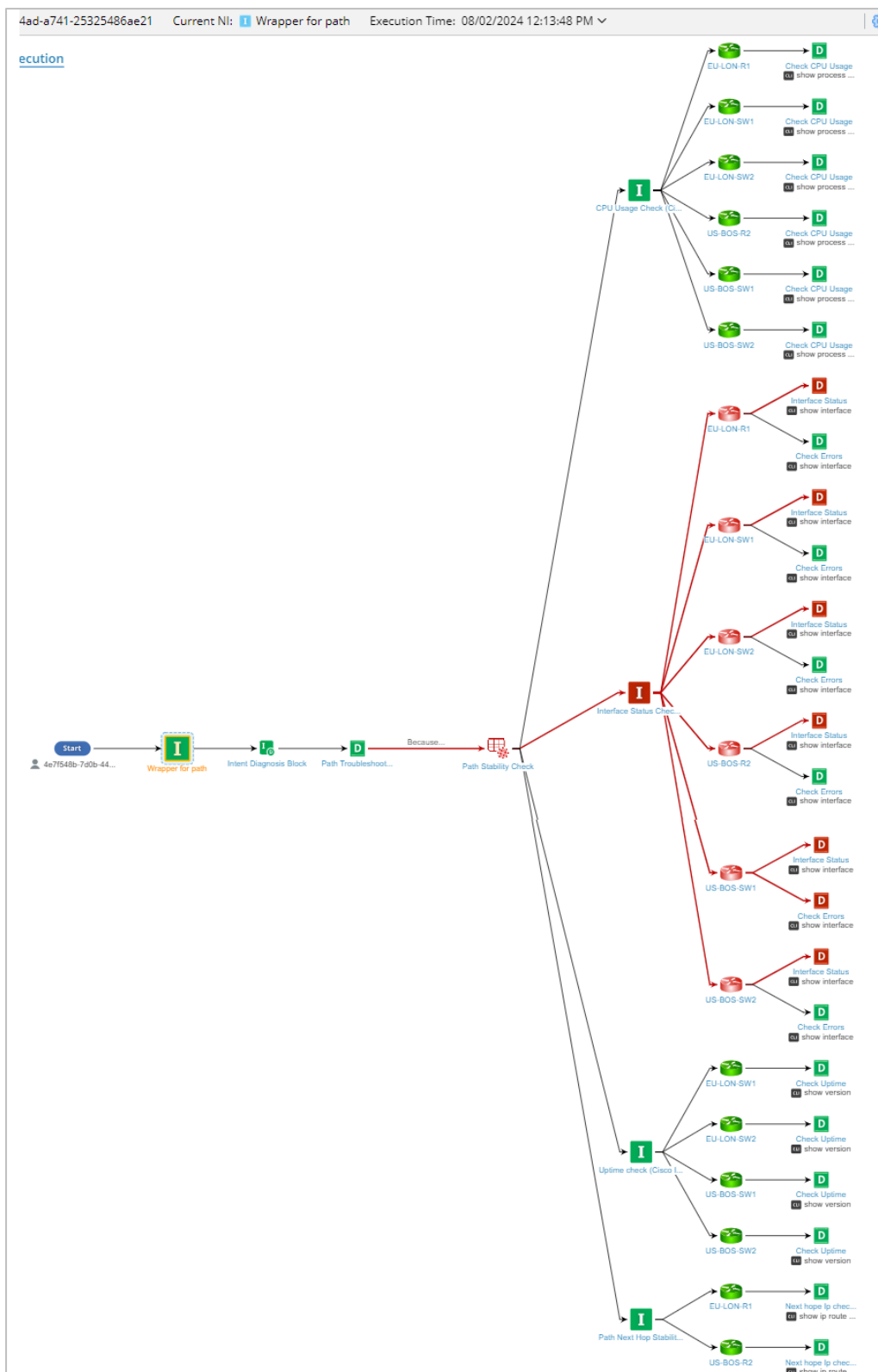
Intent: **I** Wrapper for path

With 4 Follow-up Intents 66 Alerts 326 Successes 196 Messages

View by Intent View by Device

- Wrapper for path
 - Path Stability Check / CPU Usage Check (Cisco IOS) EU-LO... 12 Successes 6 Messages
 - Path Stability Check / Interface Status Check... 66 Alerts 302 Successes 184 Messages
 - Path Stability Check / Uptime check (Cisco IOS) EU-LON-SW1 8 Successes 4 Messages
 - Path Stability Check / Path Next Hop Stability Check EU-LON... 4 Successes 2 Messages

7. Click the Diagnosis Tree  icon to view the results in a tree layout.



8. Enable Auto Intent for Wrapper Intent.

NOTE: Before Enabling **Auto Intent** for the Wrapper Intent column, you need to define **Auto Intent Profile** in the **IBA Center**.

a) Enable Auto Intent via ADT.

Intent Based Automation Center

Installed Intent Templates | Published Intents | **Auto Intent** | Auto Intent Profile | NetBrain Download

+Add Folder

Search...

Enable Auto Intent via ADT

Automation Data Table: Path Stability Check Description:

Auto Intent Name: Path Stability Check - Wrapper Intnet

Find critical assets by device properties or visible interfaces:

A Path Devices Matches Hostname

B Select...

Boolean Expression: A

Select intents/maps/paths to be listed in Auto Intent and set the display name:

List in Auto Intent	Intent Column	Display Name
<input checked="" type="checkbox"/>	Path	\$Path
<input type="checkbox"/>	Uptime Check	\$Path_Intent_2
<input type="checkbox"/>	CPU Usage	\$Path_Intent
<input type="checkbox"/>	Interface Status	\$Path_Intent_1
<input type="checkbox"/>	Next Hop Stability Check	\$Path_Intent_4

Automation Data Table Preview:

No.	Application Name	Path	Path Devices	Source	Destination
1	Cookbook	Path_Troubleshooting	10.1.1.100...	US-BOS-Host1	10.101.1.100

b) Define Auto Intent Profile.

Intent Based Automation Center

Installed Intent Templates | Published Intents | Auto Intent | **Auto Intent Profile** | NetBrain Download

+Add Profile

Search...

Shared Profiles

- NBU_EMEA_Training
- Network Essential
- Wireless_Troubleshooting

My Profiles

- path_troubleshooting

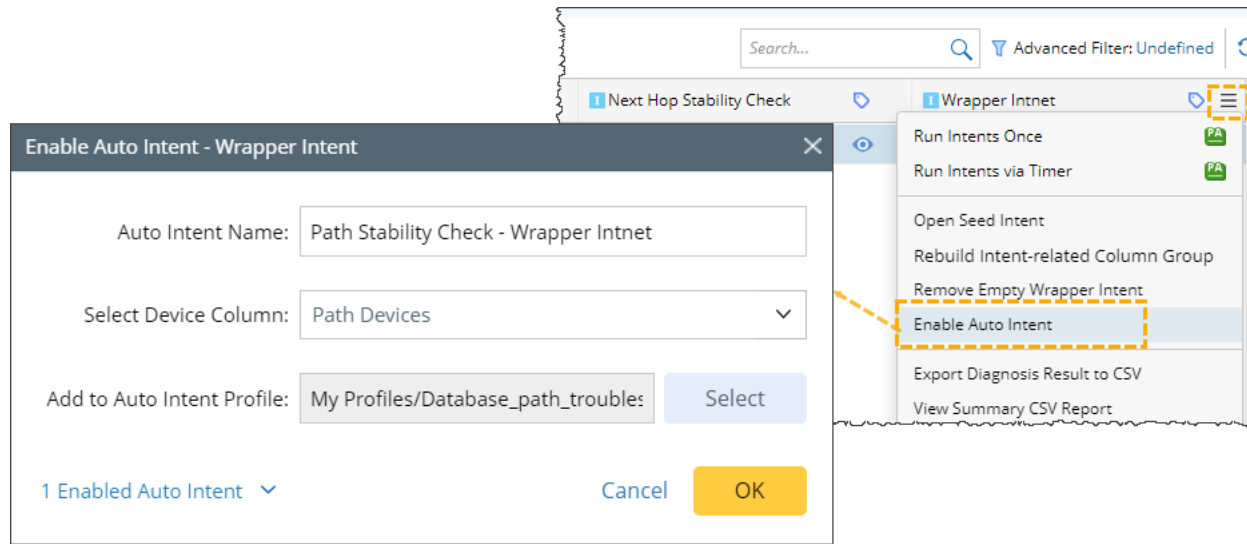
Name: Database_path_troubleshooting

Description: Input...

Included NITs/ADTs: +Add

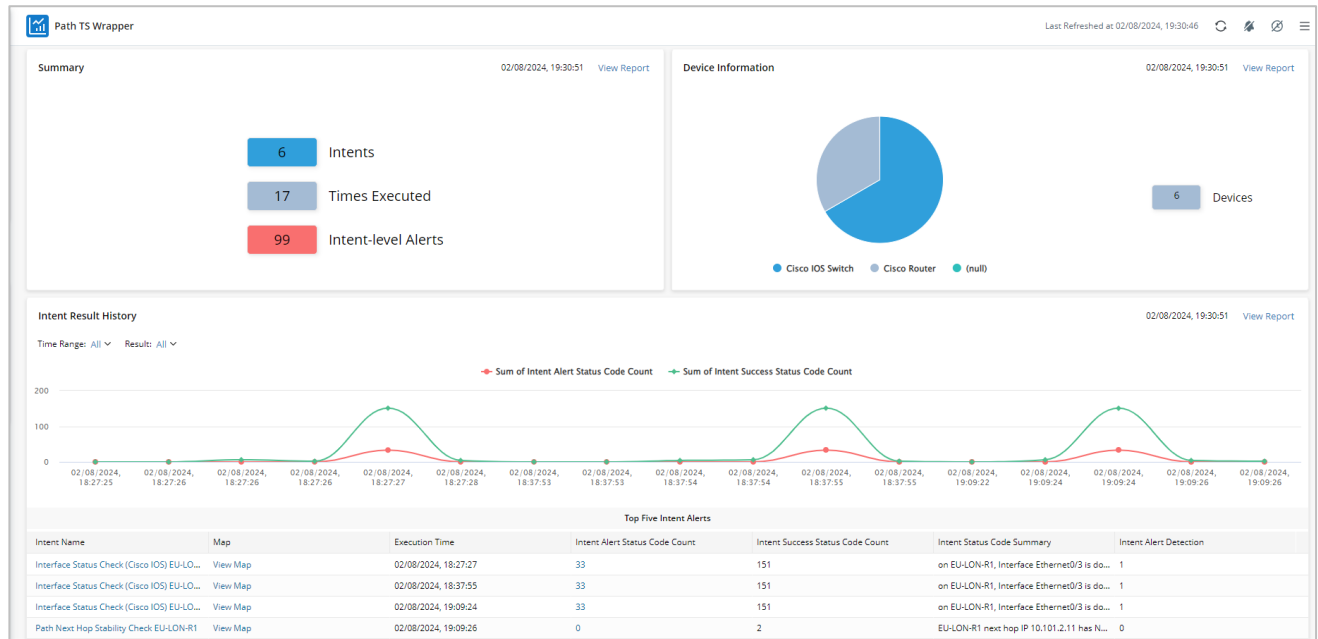
NIT/ADT	Location
Path Stability Check - Wrapper Intnet	/Enable Auto Intent via ADT

c) **Enable Auto Intent** from the ADT Wrapper Intent column.

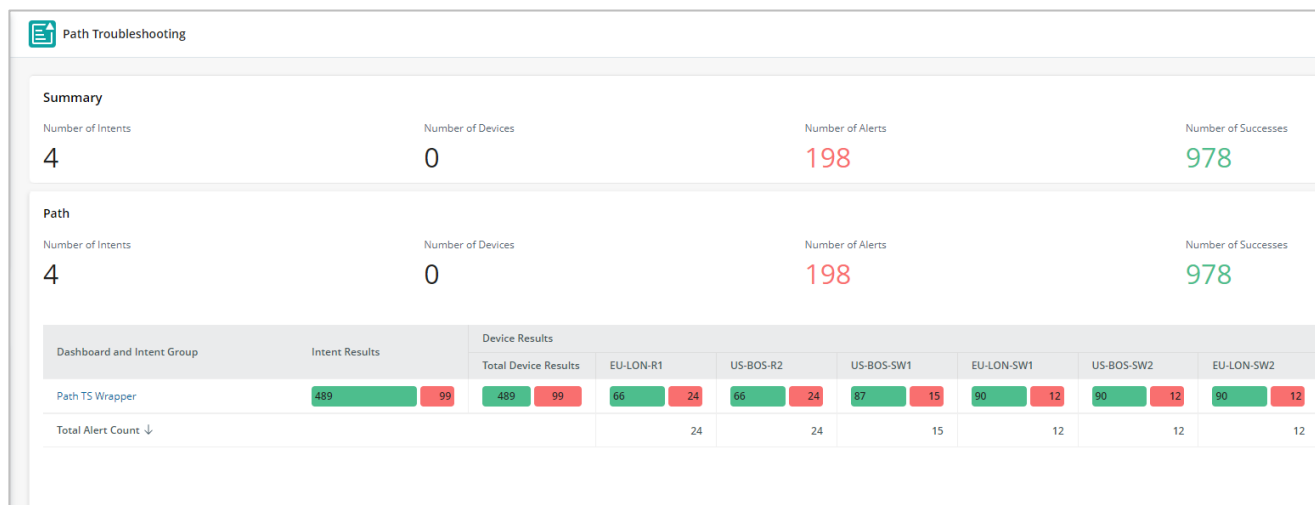


9. Run the Wrapper Intent and Create a Dashboard.

Intent Dashboard:



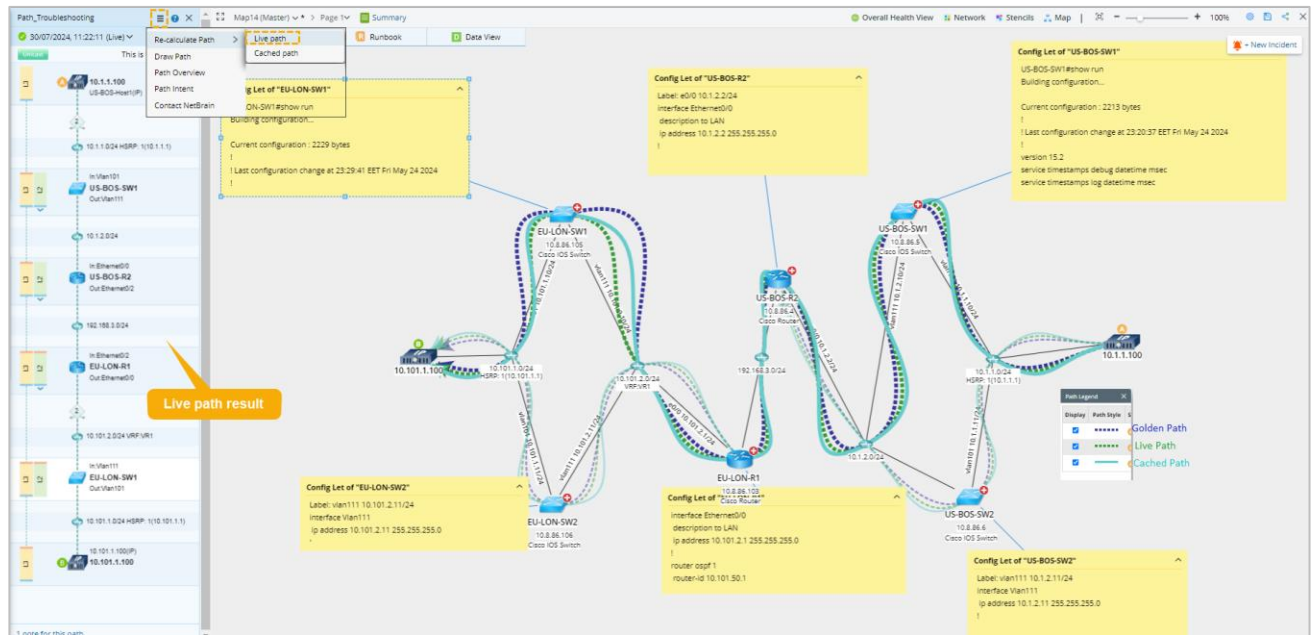
Summary Dashboard:



10.3 Troubleshoot the path issue

Calculate the live path and compare it with the golden or cached path to view the path differences.

1. Open your path from the path browser, draw the live path, and then draw the Golden or Cached path for comparison.



2. Run the Wrapper Intent and view the intent results in the dashboard.

Path Troubleshooting

Summary

Number of Intents

4

Number of Devices

0

Number of Alerts

462

▲ 264

Number of Successes

2,282

▲ 1,304

Path

Number of Intents

4

Number of Devices

0

Number of Alerts

462

▲ 264

Number of Successes

2,282

▲ 1,304

Dashboard and Intent Group	Intent Results	Device Results						
		Total Device Results	EU-LON-R1	US-BOS-R2	US-BOS-SW1	EU-LON-SW1	US-BOS-SW2	EU-LON-SW2
Path TS Wrapper	<div>1,141</div> <div>231</div>	<div>1,141</div> <div>231</div>	<div>154</div> <div>56</div>	<div>154</div> <div>56</div>	<div>203</div> <div>35</div>	<div>210</div> <div>28</div>	<div>210</div> <div>28</div>	<div>210</div> <div>28</div>
Total Alert Count ↓			56	56	35	28	28	28

11 Intent-based Change Verification and Assessment

NetBrain **Change Management** module is a comprehensive solution for controlling a network change process. Within a single Runbook, network engineers can define a change, deploy the configuration updates, and verify the impact of the changes in a fully auditable way and adherent to existing approval processes.

With intent-based automation, you can upgrade the change process with more accurate verifications of the change. One recommended change flow can be:

The screenshot displays the NetBrain Next-Gen interface. On the left, a sidebar shows navigation options like Recents, Network, Files, Site, Path, Dashboard, Intents, Chatbot, Data, Map, and Desktop. The main area shows a Runbook flow for 'Auto Test Intent (Before Change)'. The flow consists of several steps: 1. Define Change, 2. Benchmark Before, 3. Auto Test Intent (Before Change), 4. Execute, 5. What is Changed Intent, 6. Auto Test Intent (After Change), 7. Benchmark After, and 8. Compare. A red arrow points from the 'Auto Test Intent (Before Change)' step to a table of intents. The table has columns: Intent Name, Target Dev..., Status Code, CSV Repor..., and Actions. It lists four intents: Ping Check, Assess OSPF neighbor st, Check CPU, and Cookbook Server Farm F. To the right of the table, there is a network diagram showing a central cloud connected to three devices: US-BOS-R2 (Cisco Router), US-BOS-SW3 (Cisco IOS Switch), and US-BOS-SW2 (Cisco IOS Switch).

1. Map the devices you want to change and those that can be affected by the change, for example, the L3/L2/routing neighbors.
2. Define the change.
3. Prepare the intents you want to run before and after the change. You can add as many as intents to verify that the change does not violate the designs and that important operation status is normal. At least you may want to run the batch ping for the key applications to ensure that they are still accessible. Run these intents before the change.
4. Execute the change.
5. After the change, run an intent to highlight what has changed and verify that your change is indeed successfully pushed to the devices.
6. Repeat the same intents as step 3 to verify that the change does not violate the designs and that important operation status is normal.

You may still use the “old” verification methods: benchmark the data before and after the change and compare these data to check the difference. The comparison is based on the text only, while the intents provide a more accurate way.

In this chapter, you will learn how to create the intent for the change flow. We will focus on two types of intents: the intent to show what has changed and the other basic auto-test. Certainly, all intents you have created up to now can be added to the CM workflow. For example, the general device health check you created in Chapter 3 can be useful to verify that the device is healthy before and after the change.

As an example, we will modify the OSPF configurations of a router to add a new subnet.

11.1 Build Intents for Change Assessment

11.1.1 What has Changed

To find out what has changed between two times (for example, the time when your network is good and when your network is troubled) is not just useful for the change process but also essential for the troubleshooting process and network assessment. In Chapters 5 and 6, you have learned how to parse a subset of configurations, such as AAA, ACL, etc. and compare the configuration to the golden template or the industry standard and best practices. You may run these intents before and after the change.

For a change process, you can create an intent to find out the change of the whole configurations before and after the change.

Create an intent, **Configuration Change**, select a seed device (for CM, you can select a device you are going to change configurations), and add a **Configuration Diagnosis**. A Cisco IOS device configuration will be like:

```
US-BOS-R2#show run
```

```
Building configuration...
```

```
Current configuration : 12524 bytes
```

```
!
```

```
! Last configuration change at 10:11:43 EST Wed Jul 31 2024 by nb
```

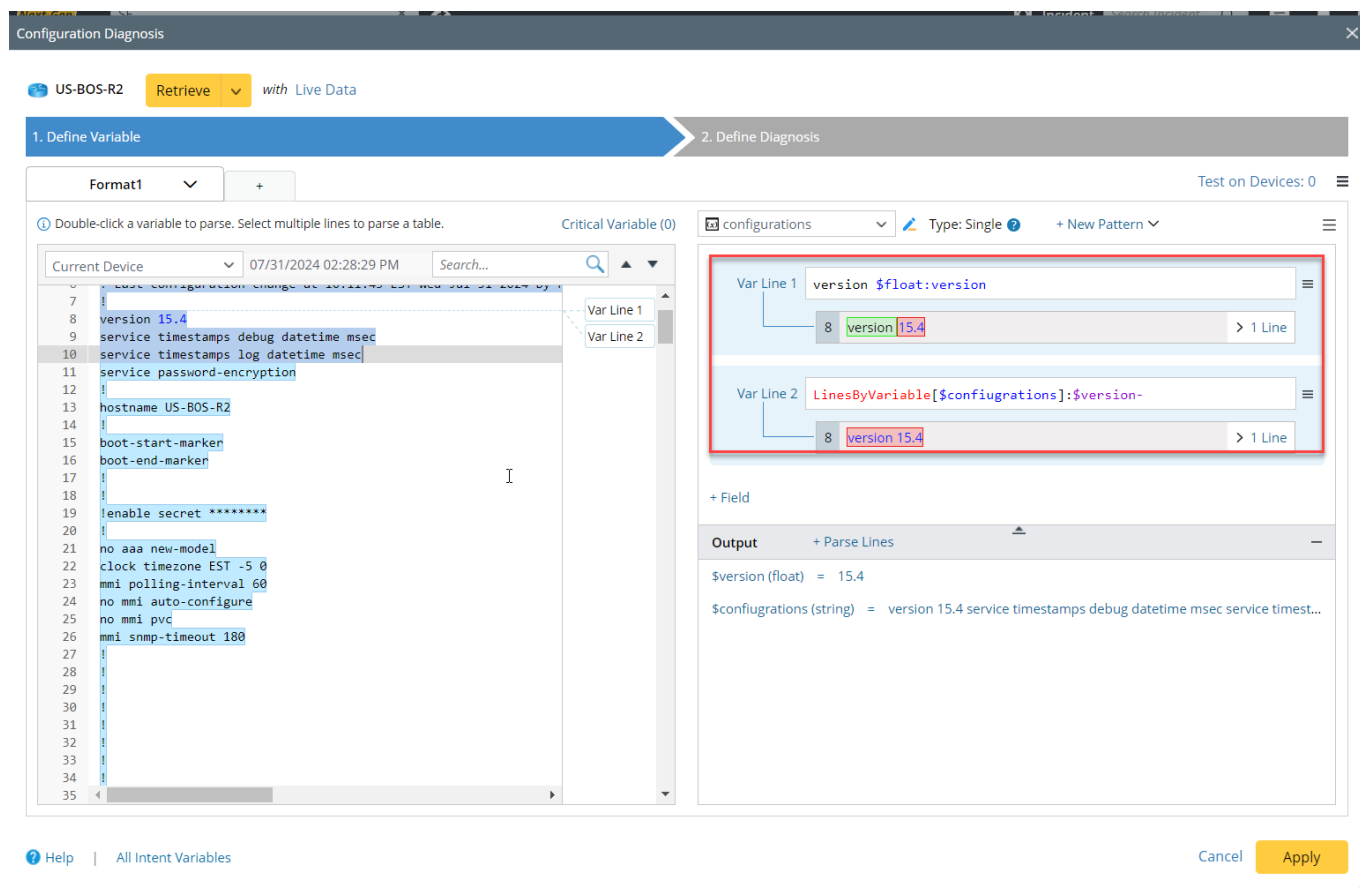
```
!
```

```
version 15.4
```

```
service timestamps debug datetime msec
```

You may notice that the bold lines can often change, and so you want to ignore them while comparing the configurations. So, you define a variable **\$configurations**, by the function

LinesByVariable, starting from the line, **version \$float:version**, to the end. You can click the link **Parse Lines** to define the function **LinesByVariable**. Check the variable value to confirm its correctness.



In the diagnosis, you compare the current and baseline values of **\$configurations** using the operator **Pattern Match**. If they do not match, create the status code to include the added and missing lines:

Configurations changed. Added lines:

\$Rule1.Unmatched_lines

Missing Lines:

\$Rule1.Unused_pattern_lines

Note:

- Select the **Pop up** option from the menu at the right of the message to pop up the status code editing window to enter the note with the multiple lines.
- You need to run the intent twice to have the last and current data.

1. Define Variable

2. Define Diagnosis

Summary Text

Original Text

Search...

What has Changed

8

version 15.4

9

service timestamps debug datetime msec

10

service timestamps log datetime msec

11

service password-encryption

12

!

13

hostname US-BOS-R2

14

!

15

boot-start-marker

16

boot-end-marker

17

!

18

!

19

enable secret *****

20

!

21

no aaa new-model

Diagnosis Message of Intent Diagnosis

Diagnosis Message:

Configurations changed. Added lines:

\$Rule1.Unmatched_lines

Missing Lines:

\$Rule1.Unused_pattern_lines

Status Code for Device:

Configurations changed. Added lines:

\$Rule1.Unmatched_lines

Missing Lines:

\$Rule1.Unused_pattern_lines

Cancel

OK

Add Note

Add Diagnosis

Can also click a variable on the left to add automation.

Loop Table Rows

▼ If

A

US-BOS-R2

Current

Last

configurations

MP(Rule1)

configurations

B

Select Variable

Then

Diagnosis Message:

Save to Incident

Configurations did not change

Set Status Code for Device:

Success

Configurations did not change

Set Status Code for Intent:

Add Logic

Else

Diagnosis Message:

Save to Incident

Configurations changed. Added lines:

\$Rule1.Unmatched_lines

Pop up

Set Status Code for Device:

Delete

Add Elself

Cancel

Apply

Help

All Intent Variables

Run the intent and check the results:

Diagnosis Details and Compare - Configuration Change

US-BOS-R2

Configuration

Execution Time: 08/01/2024 11:49:32 AM

Diagnosis Details

Compare

Summary Text

Original Text

08/01/2024 09:30:51 AM

Search...

1

US-BOS-R2#show run

2

Building configuration...

3

Current configuration : 12566 bytes

4

!

5

! Last configuration change at 21:49:12 EST Wed Jul 3

6

!

7

!

8

version 15.4

9

service timestamps debug datetime msec

10

service timestamps log datetime msec

11

service password-encryption

12

!

13

hostname US-BOS-R2

14

!

15

boot-start-marker

16

boot-end-marker

17

!

18

!

19

enable secret *****

20

!

21

no aaa new-model

22

clock timezone EST -5 0

23

mmi polling-interval 60

24

no mmi auto-configure

25

no mmi pvd

26

mmi snmp-timeout 180

27

!

28

!

29

!

30

!

31

!

Configurations ...

Diagnosis Logic (What has Changed)

Anchor: version 15.4 service timestamps debug datetime msec servic...

if

A: US-BO... Current

US-BOS-R2 Last

\$configurations

MP:Rule1

\$configurations

False

Boolean Expression

Value: vers

Configurations changed. Added lines:

!username *****

Missing Lines:

None

Else

Diagnosis Message:

Configurations changed. Added lines:

Status Code for Device:

Configurations changed. Added li...

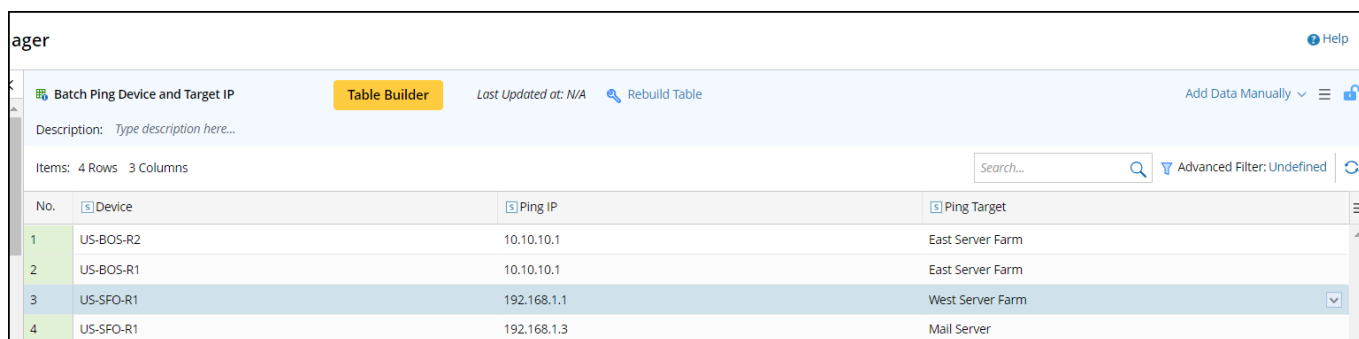
NetBrain R11.1b | 409

11.1.2 Auto Test

While making the network change, you want to ensure no accidental impact. For example, the server farm is still accessible, and the traceroute to the critical server does not change, etc. You can develop a set of automations for auto tests.

Let us use the ping as an example. To ensure that the critical server still functions, you may ping the server IP addresses from a set of devices that should have routes to the server. Ping for many servers from hundreds and even thousands of devices manually is not feasible. However, you can have the intents to do this for you automatically.

First, create an ADT including all pairs of the device and server IP. You can create a CSV file and import the CVS file into the ADT table (make sure that you set the device name column as the data type **Device**). The following is a small set of sample data:

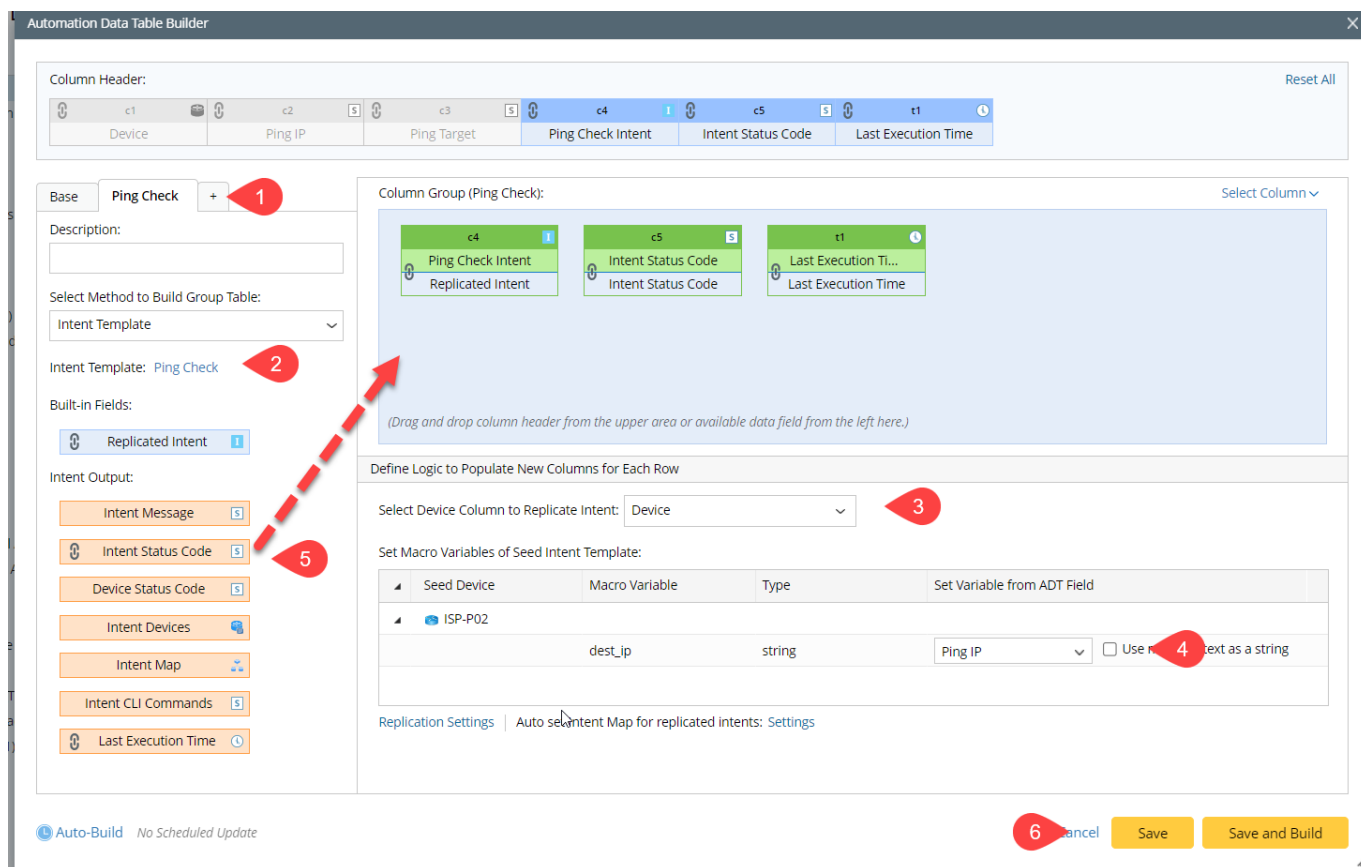


The screenshot shows the NetBrain Table Builder interface. At the top, there's a header with 'Batch Ping Device and Target IP', a 'Table Builder' button, 'Last Updated at: N/A', and a 'Rebuild Table' button. Below this is a description field 'Type description here...'. The table itself has 4 rows and 3 columns: 'No.', 'Device', 'Ping IP', and 'Ping Target'. The data is as follows:

No.	Device	Ping IP	Ping Target
1	US-BOS-R2	10.10.10.1	East Server Farm
2	US-BOS-R1	10.10.10.1	East Server Farm
3	US-SFO-R1	192.168.1.1	West Server Farm
4	US-SFO-R1	192.168.1.3	Mail Server

Now, you can build a new Column Group, **Ping Check**:

1. Add a new group, **Ping Check**.
2. Select the **Ping Check** intent created in Chapter 2.
3. Select the Device Column.
4. Set the macro variable, **dest_ip**, as the **Ping IP** column.
5. Add the intent output to the ADT.
6. Save and build.



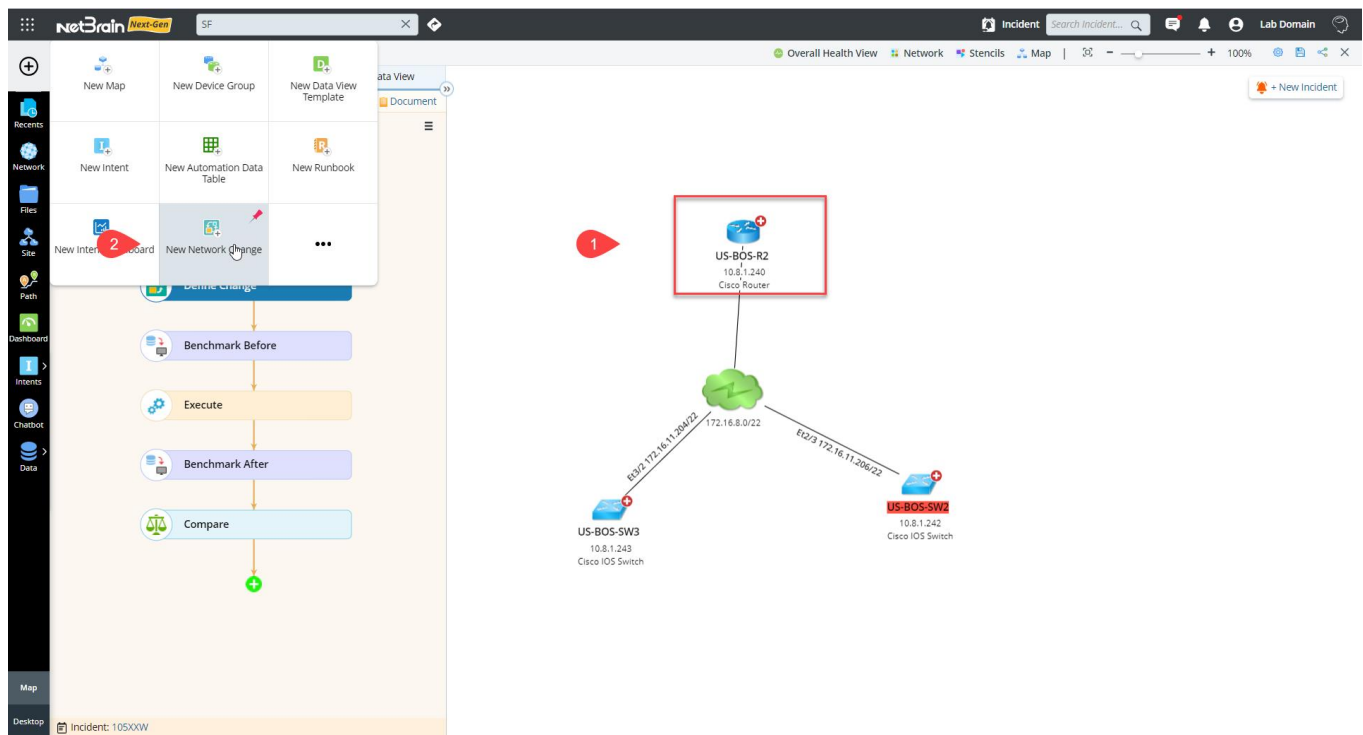
Run the replicated intents once and check the results. You can also create a dashboard.

Batch Ping							
Description: Type description here...		Table Builder		Last Updated at: 08/01/2024 02:49 PM		Rebuild Table	
Items: 4 Rows 6 Columns		Search...		Advanced Filter: Undefined			
No.	Device	Ping IP	Ping Target	Ping Check	Intent Status Code	Last Execution Time	
1	US-BOS-R2	10.8.1.1	East Server Farm	Ping Check US-BOS-R2 1	Ping succeeded	08/01/2024 02:49:09 PM	
2	US-BOS-R1	10.8.1.1	East Server Farm	Ping Check US-BOS-R1 1	Ping succeeded	08/01/2024 02:49:09 PM	
3	US-SFO-R1	192.168.1.1	West Server Farm	Ping Check US-SFO-R1 1	Ping failed with the success r...	08/01/2024 02:49:09 PM	
4	US-SFO-R1	192.168.1.3	Mail Server	Ping Check US-SFO-R1	Ping failed with the success r...	08/01/2024 02:49:09 PM	

11.2 Intent-Driven Change Management and Assessment

In this section, we will walk you through the intent-driven change management and assessment flow with a simple change: we will add a new subnet at one router, **US-BOS-R2**, and add this subnet into the OSPF configuration.

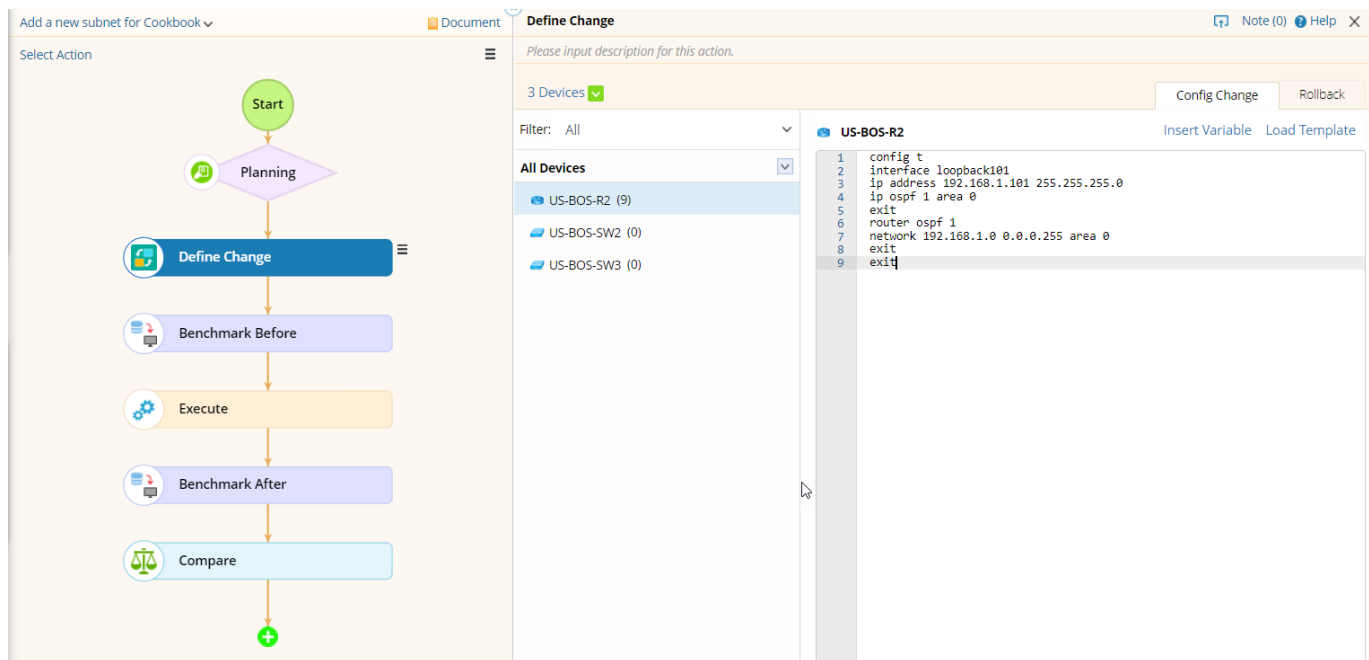
Map the devices you are going to make changes to and add the possibly affected devices, such as OSPF neighbors, in our example. Then, add a **New Network Change** task.



11.2.1 Define Change

Click the **Define Change** node, select the device, **US-BOS-R2**, and enter the **config change**, which adds a new loopback interface and the subnet into the **OSPF 1**.

```
config t
interface loopback101
ip address 192.168.1.101 255.255.255.0
ip ospf 1 area 0
exit
router ospf 1
network 192.168.1.0 0.0.0.255 area 0
exit
exit
```



You may also enter the configurations to roll back your change in case the change causes unexpected results.

After you review the configurations to be changed, you can click the **Planning** node to request approval. The change can only be executed after the change is approved. You can assign yourself as the approver for the exercise.

11.2.2 Create Intent to Analyze Route Changes

Since we are going to make changes to the OSPF configuration, it will affect the routing. So, we should add an intent to analyze the route changes.

Create a new intent, **Compare Dynamic Routes**, and select the device to which you will change configurations. Enter the CLI command **show ip route** and retrieve the data. At first glance, it is hard to parse the result. However, if you study the data closely, the routes have two types of formats: one for the dynamic routes and one for the directly connected routes. So, you can create two Paragraph Parsers to parse the dynamic and connected routes.

For the dynamic routes, copy a line to the **ID Line A** and replace the values with the variables (be cautious to keep the number of spaces intact):

\$mstring:protocol \$subnet [\$distance] via \$next_hop, \$_dummy

2. Define Diagnosis

Add Note

Add Diagnosis

Can also click a variable on the left to add automation.

Name: Compare dynamic routes

Anchor:

Type description of the diagnosis...

☐ Loop Table Rows

If

A

US-BOS-R2

Current

dynamic_routes

Does not equal

dynamic_routes

B

Select Variable

Then

Diagnosis Message:

Dynamic route changes

☐ Save to Incident

☒ Set Status Code for Device:

Error

Dynamic route changes

☐ Set Status Code for Intent:

Table Compare Settings

Pop up

Delete

Table Compare Settings

☒ Output table-compare summary message

Compare Table

dynamic_routes

of

US-BOS-R2

Between

Current

and

Last

Sample Summary Message

e.g. 'table' of \$device has changed (Current vs Baseline): 30 changed, 20 added, 10 removed.

☒ Include Top

20

Changed Entries

☒ Include Top

20

Added Entries

☒ Include Top

20

Removed Entries

Cancel

Apply

Cancel

OK

If you want to get more details, you can create a diagnosis to find out all the changed, added and missing entries. For this purpose, you can add a **Compound Table** to merge the current and last tables. The system supports different rules for merging tables, and the most common one is the **full join**:

Full Join

Table1

Table2

Returns all rows when there is a match in left (table1) or right (table2) table rows.

Left Table (Table1)

Column1	Column2
1	a
2	b
3	c

Right Table (Table2)

Column1	Column2
b	x
d	z

Merged Table

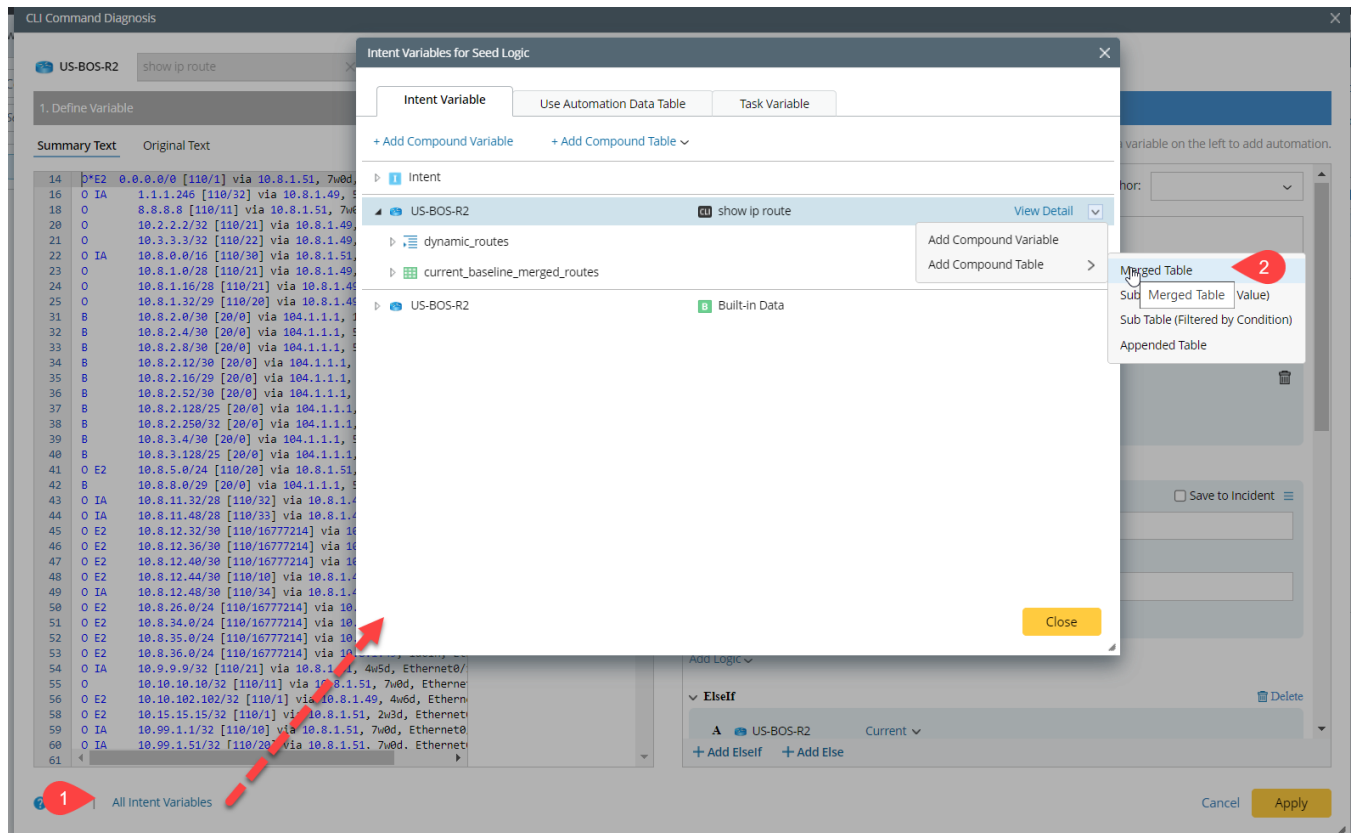
L. Column1	L. Column2	R. Column1	R. Column2
1	a		
2	b	b	x
3	c		
		d	z

So, when you merge the current route table with the past route table using the subnet as the paired key, the entry of a missing route will only have the values in the columns corresponding to the past route table (the values of the columns corresponding to the current route table are empty) in the merged table while an added route will only have the values in the columns corresponding to the current route table.

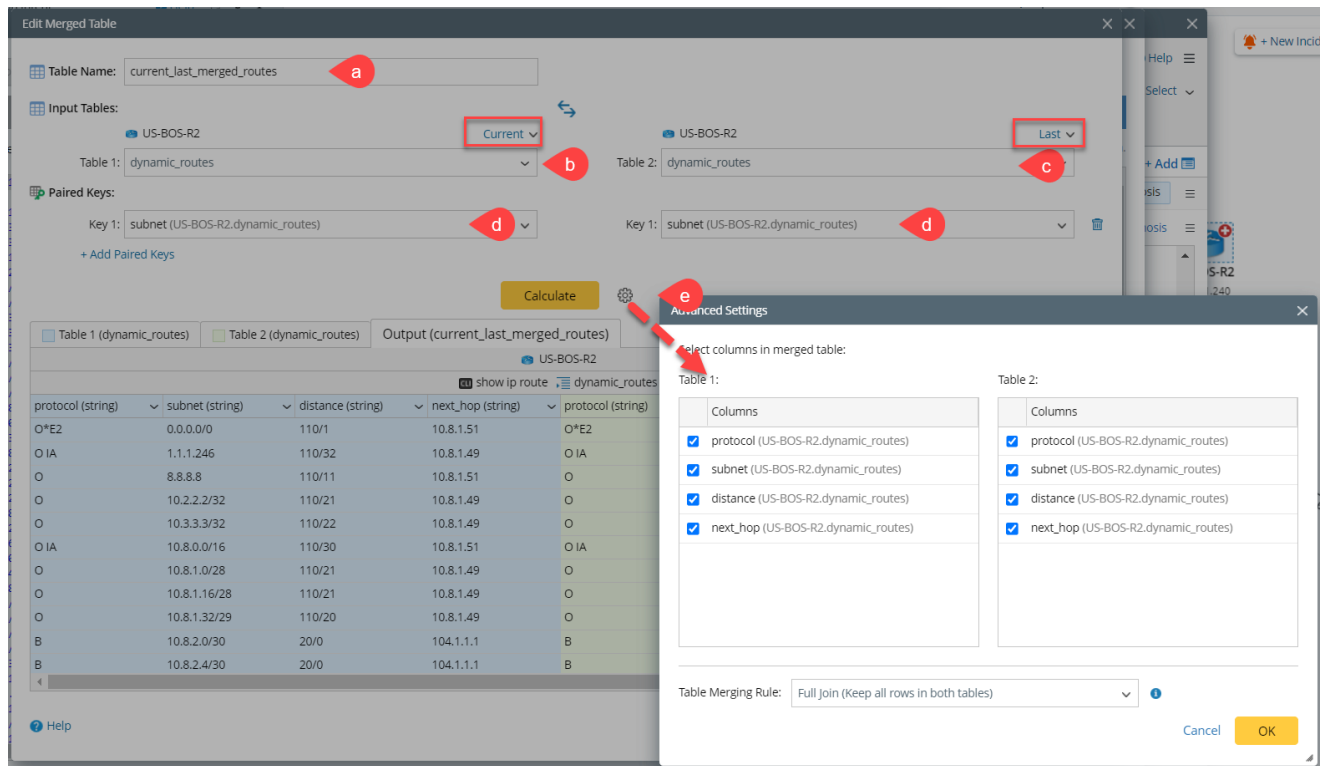
NetBrain R11.1b | 415

To create a merged table,

1. Click the **All Intent Variables** link at the bottom of the Diagnosis window.
2. At the dropdown menu, select **Add Compound Table > Merged Table**.

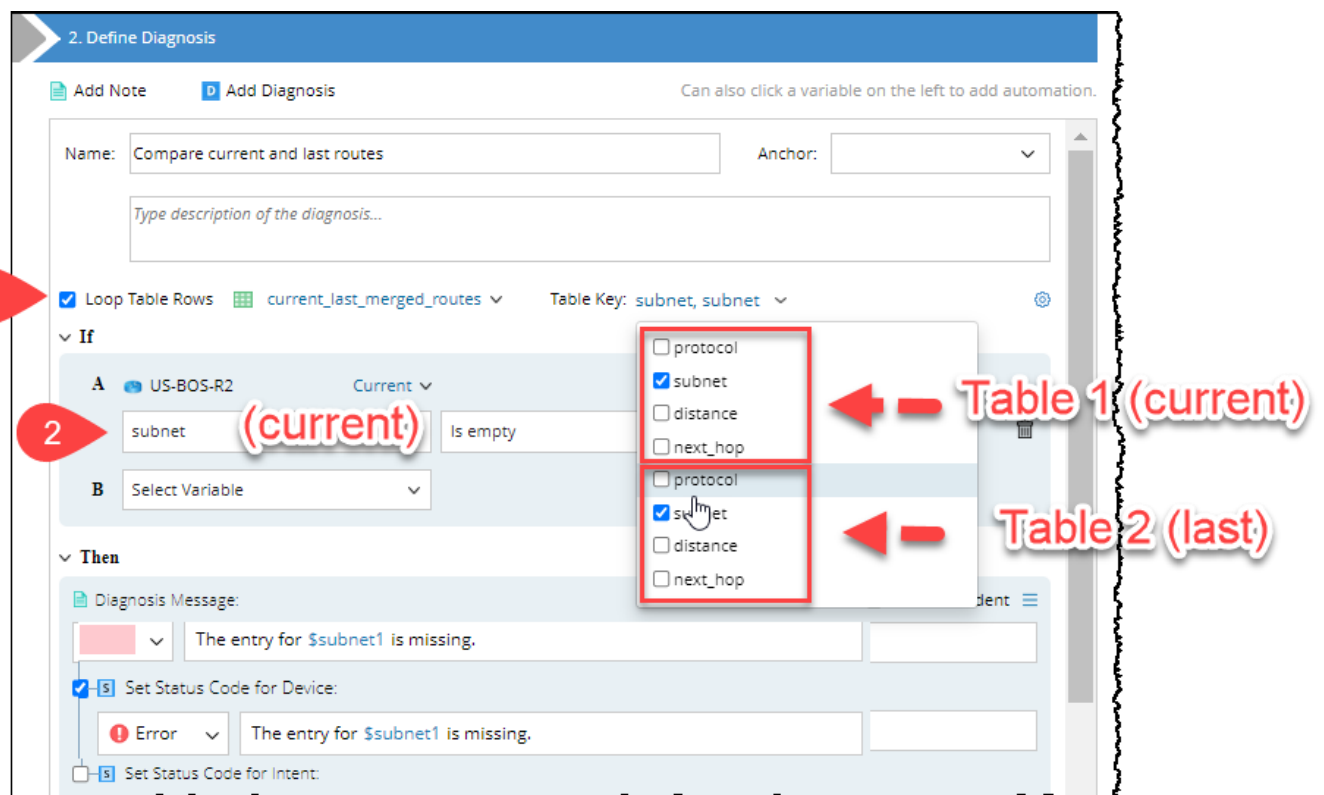


3. In the Edit Merged Table window,
 - a) Enter an easy-to-understand name for the merged table, e.g., **current_last_merged_routes**, implying that this table merges the current and last routes.
 - b) At the right side of **Table 1**, select the table from the pulldown menu, **dynamic_routes**, and then select the **Current** table (which is the default option) above.
 - c) At the right side of **Table 2**, select the table from the pulldown menu, **dynamic_routes**, and then select the **Last** table (which is the default option) above.
 - d) Select the **subnet** as the key for both tables.
 - e) In the **Advanced Setting**, you can select columns in the merged table and select the merging rule. Here, you can use the default value.
 - f) Check the merged table for its accuracy.



Now, you can define the diagnosis by looping through the merged table to find the missing, added, and changed routes:

1. Loop through the merged table, **current_last_merged_routes**, and select both **subnet** fields as the table key. Here, the pulldown options include the column tables from two tables (**table 1** for the **current** route table and **Table 2** for the **last** route table). The column names are identical, with the top half corresponding to **Table 1** and the lower half corresponding to **Table 2**. The same rule applies to all other pulldown options.
2. In the **If** condition, select the **current subnet** (the top one) and check whether it is empty. If so, add a diagnosis message: **The entry for \$subnet1 is missing**. You should use the last subnet in the message.



3. Add an **elseif** condition to check whether a route entry is added.
4. Add another **elseif** condition to check whether the next hop changed.

2. Define Diagnosis

Add Note

Add Diagnosis

Can also click a variable on the left to add automation.

ElseIf

A US-BOS-R2 Current ▼

3 subnet Last ▼ Is empty ▼

B Select Variable ▼

Then

Diagnosis Message: ☐ Save to Incident ≡

▼ A new route for s\$subnet is added

☒ ⓘ Set Status Code for Device:

❗ Error ▼ A new route for s\$subnet is added

☒ ⓘ Set Status Code for Intent:

❗ Error ▼ A new route for s\$subnet is added

[Add Logic](#) ▼

ElseIf

A US-BOS-R2 Current ▼ Current ▼

4 next_hop Current ▼ Does not equal ▼ next_hop Last ▼

B Select Variable ▼

Then

Diagnosis Message: ☐ Save to Incident ≡

▼ The next hop of \$subnet changes from \$next_hop1 to \$next_hop!

☒ ⓘ Set Status Code for Device:

▼

[+ Add Elself](#) [+ Add Else](#)

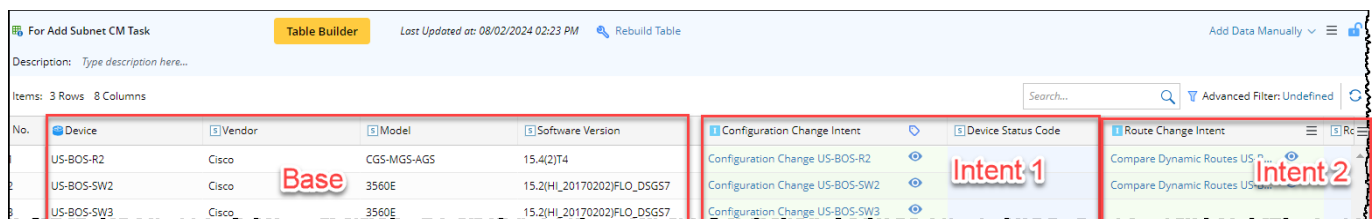
5. Save and run the intent. Check the result for its correctness.

11.2.3 Organize and Run Automations Before Change

After you have all intents ready for a network change task, you need to decode/replicate the intents so that it can be run for the change and possibly affected devices. You can use **Auto Intent Wizard** so that users can replicate and run these intents under the **Auto Intent** pane in the map for the network change. Refer to Chapter 4 for the detailed instructions. Another way (maybe more organized and easier to manage) is to use ADT as follows:

1. Create a device group for the devices that will be changed or affected by the change. You can create the device group from the map directly.
2. Create an ADT, **For Add Subnet CM Task**, for the change management task, and build the base from the device group created in step 1.
3. Use the **Intent Replication Wizard** to replicate the intents (such as the **Configuration Change** you created in 2.1.1 and **Compare Dynamic Routes** you just created in 2.2.3) to the ADT created in step 2. You can select the device group created in step 1 as the intent qualification. And add the additional fields into the ADT such as intent/device status code, and last execution time.

The ADT will be like this:



No.	Device	Vendor	Model	Software Version	Configuration Change Intent	Device Status Code	Route Change Intent
1	US-BOS-R2	Cisco	CGS-MGS-AGS	15.4(2)T4	Configuration Change US-BOS-R2		Configuration Change US-BOS-R2
2	US-BOS-SW2	Cisco	3560E	15.2(HI_20170202)FLO_D5G57	Configuration Change US-BOS-SW2		Compare Dynamic Routes US-BOS-SW2
3	US-BOS-SW3	Cisco	3560E	15.2(HI_20170202)FLO_D5G57	Configuration Change US-BOS-SW3		Compare Dynamic Routes US-BOS-SW3

You can set up an Intent Timer to run all replicated intents of this ADT, or you can just run once before and after the change.

The Auto Test, such as **Batch Ping**, has its own ADT, and you should also set up an Intent Timer to run these auto tests.

To remind yourself or others to run these intents, you can add intent nodes into the CM runbook. To add an intent node into the runbook,

1. Click the + sign before or after an existing node where you want to add a node.
2. In the **Select Action** window, select the **Node Pane** tag.
3. Select Network Intent.
4. In the **Select Intent** window, uncheck all checkboxes at the lower pane and select the intents you want to add. You can select more than one intent.

Test1 Document Execute Note (0) Help X

Select Action

Start

Approved

Define Change

Benchmark Before

Auto Test

Execute

Benchmark After

Compare

Incident: 105XY1

Executed 1 out of 1 Devices By guangdong.liao@netbraintech.com Start 08/01/2024 04:07:01 PM Config Change Rollback

All 1 Commands Sent 1 Export CSV Report Edit Config Change

Filter: All Select All Clear All

US-BOS-R2

Select Action

Node Pane Favorite Network Change Runbook Template

Execute CLI Commands Overall Health View Ping Traceroute Retrieve Live Data Compare Path

Network Intent Data View Template Verify Application Run Qapp Run Gapp Ansible Task Free Text

Name:

Select Intents

Type: Common Intent

Search...

All 10.10.10.1 (6) 192.168.29.62 (4) 3Com (5) 5 f >>

Change Assessment

Auto Test - Batch Ping

Compare Dynamic Routes

Configuration Change

Cookbook examples

ch31 define text

ch41 simple diagnosis

ch412 If Condition

Ch412 Interface errors

ch42 table diagnosis

Ch422 Interface Table Operation

Show Intents Serving as Template Only

Show Intents for Devices on Current Map Only

Show Published Intents Only

Cancel OK

Pre-check Hostname before Execution

5. Since you cannot select the intents of an ADT, so, in the node **description**, you may remind the users of this runbook (which can be yourself or others) that they should run intents of a certain ADT instead of just intents selected here.

The screenshot displays the NetBrain Runbook Editor interface. The top navigation bar includes tabs for Intent, Runbook, and Data View. The main workspace shows a workflow diagram with the following steps: Start, Approved (decision), Define Change (locked), Benchmark Before, Auto Test, Execute, Check Config and Route Change (highlighted with a red circle and the number 5), Benchmark After, and Compare. The right-hand panel, titled 'Check Config and Route Change', contains a description: 'Please go to the ADT For Add Subnet CM Task and run all replicated intents after...'. Below this, there are two tables. The first table, 'Items:2', has columns for Intent Name, Target Dev, Status Code, CSV Report, and Actions, and contains two rows: 'Compare Dyna 1' and 'Configuration C 1'. The second table, 'Items:0', has columns for Intent Name, Target Dev, Status Code, CSV Report, and Diagnosis T, and is currently empty. A 'Run' button is located at the bottom right of the right-hand panel. The bottom status bar shows 'Incident: 105XY1'.

Incident: 105XY1

After adding all intents to the Runbook, you may end up with an intent-based CM flow:



11.2.4 Execute Change

Executing change is straightforward. Click the **Execute** node, select the device, and click the **Execute** button. You can select the execution mode: **one by one automatically, in batch automatically, or one by one manually**. After the execution, you can select a device and view the execution log.

Execute

Note (0) Help

Please input description for this action.

Executed 1 out of 1 Devices By guangdong.liao@netbraintech.com Start 08/02/2024 03:58:57 PM

Config Change Rollback

All 1 Commands Sent 1

Export CSV Report Edit Config Change

Filter: All

US-BOS-R2

Select All Clear All

☐ US-BOS-R2

--- Commands to be sent to device ---
config t
interface loopback101
ip address 192.168.1.101 255.255.255.0
ip ospf 1 area 0
exit
router ospf 1
network 192.168.1.0 0.0.0.255 area 0
exit
exit

--- Login to device and pre-check hostname ---
Pre-check hostname: Successful
Retrieved hostname "US-BOS-R2" matched

--- Login to device and send commands ---
Sending task(containing 9 commands) to Front Server(tid: 22420)
Dispatch the task to live thread
Telnet to device 10.8.1.240

User Access Verification

Username: nb
Password:
US-BOS-R2>enable
Password:
US-BOS-R2#enable
US-BOS-R2#terminal length 0
US-BOS-R2#config t
Enter configuration commands, one per line. End with CNTL/Z.
US-BOS-R2(config)#interface loopback101
US-BOS-R2(config-if)#ip address 192.168.1.101 255.255.255.0
US-BOS-R2(config-if)#ip ospf 1 area 0
US-BOS-R2(config-if)#exit
US-BOS-R2(config)#router ospf 1
US-BOS-R2(config-router)#network 192.168.1.0 0.0.0.255 area 0
US-BOS-R2(config-router)#

0 Devices Selected

Execute

One by One Automatically

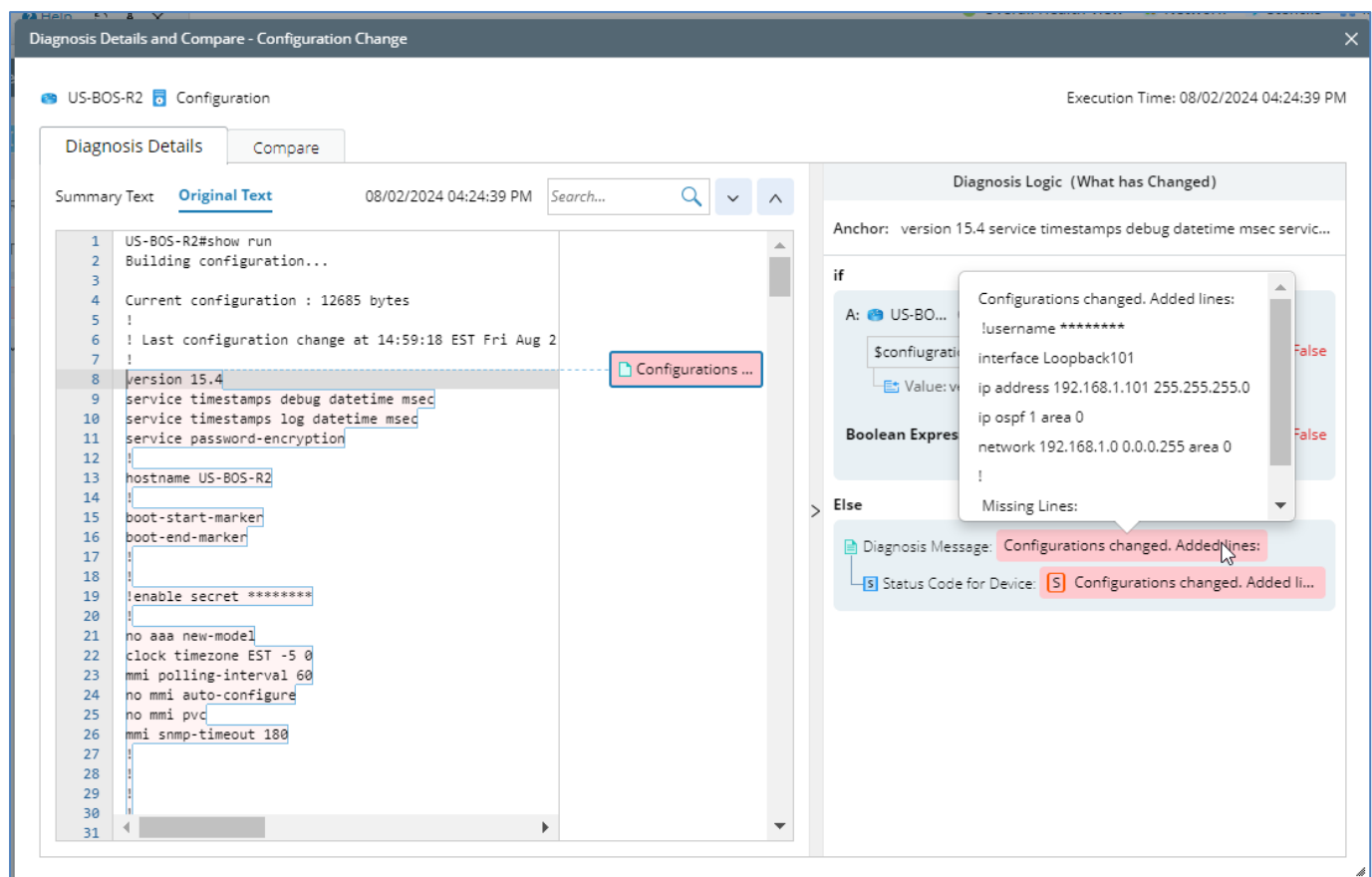
Pre-check Hostname before Execution

11.2.5 Verify Change

After the change, you can run all auto-tests you want to, for example, **Batch Ping**, to verify that all critical servers are still pingable.

In our example, you can open the ADT **For Add Subnet CM Task**, and run all intents to check the configuration change and route change.

First, check the configuration changes in the change device to confirm that the configuration is indeed changed as expected.



Secondly, check the dynamic route change. Here you should not just check the route changes of the change device, but also of the neighbor routers. For example, the following is the result of the intent, **Compare Dynamic Routes**, of a neighbor router, which shows that a new route is added as expected:

Network Intent (View Mode) - All Network Intents/R&D/GD/Change Assessment/Compare Dynamic Routes/Compare Dynamic Routes US-BOS-SW2

1 Compare Dynamic Routes US-BOS-SW2

Open
0
0
Edit

Result: 08/02/2024 04:07 PM

Run
with Live Data

This intent execution is finished at 08/02/2024 04:07 PM with 0 errors. You can View [Execution Log](#)

Dynamic route changes
4

View

US-BOS-SW2
Dynamic route changes dynamic_routes of US...
2
From Seed Device: US-BOS-R2.

show ip route
2 Diagnoses
From Seed Command: show ip route (Device: US-BOS-R2).

1 US-BOS-SW2>show ip route
2 Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
3 D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
4 N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
5 E1 - OSPF external type 1, E2 - OSPF external type 2
6 i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
7 ia - IS-IS inter area, * - candidate default, U - per-user static route
8 o - ODR, P - periodic downloaded static route, H - NHRP, I - LISP
9 a - application route
10 + - replicated route, % - next hop override
11
12 Gateway of last resort is 10.8.1.18 to network 0.0.0.0
13
14 O*E2 0.0.0.0/0 [110/1] via 10.8.1.18, 6w1d, Vlan101
15 [110/1] via 10.8.1.2, 6w1d, Vlan100
16 1.0.0.0/32 is subnetted, 1 subnets
17 O 1.1.1.246 [110/11] via 20.0.5.2, 7w0d, Ethernet0/1
18 8.0.0.0/32 is subnetted, 1 subnets
19 O 8.8.8.8 [110/22] via 10.8.1.18, 6w1d, Vlan101
20 [110/22] via 10.8.1.2, 6w1d, Vlan100
21 10.0.0.0/8 is variably subnetted, 45 subnets, 7 masks
22 O 10.2.2.2/32 [110/2] via 10.8.1.18, 7w0d, Vlan101
23 [110/2] via 10.8.1.2, 7w0d, Vlan100

Dynamic route changes dynamic_routes of US-...

A new route for s192.168.1.101 is added

12 Map and Document Your Network

In previous chapters, you have learned how to create the report and document for your network via the ADT and intents. In this chapter, we will revisit these methods and cover how to create the map with the intent.

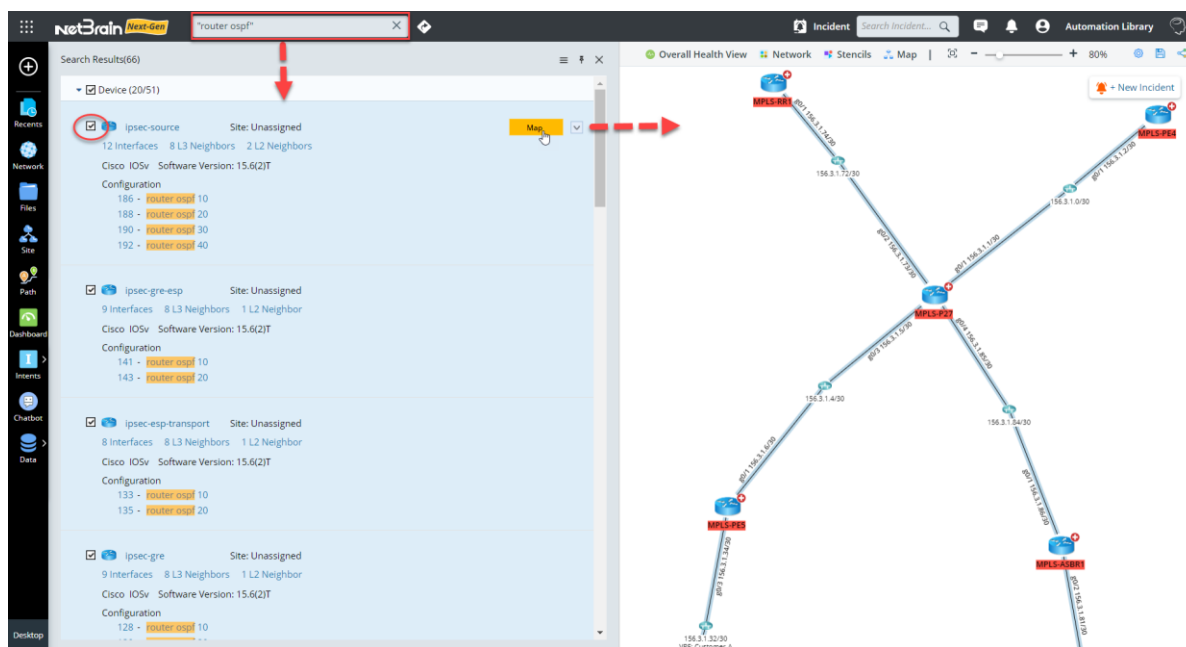
The chapter covers the following topics:

- Create a map with the intent
- Review the built-in inventory report
- Use ADT to build the inventory report
- Create the report from the CLI command results using the pre-replicated intent template
- Create the CSV report from an intent and ADT

12.1 Create a Map

The NetBrain dynamic map provides full visibility of your network. There are many ways to create a map manually. For example, you can map the device group and site, enter two IP addresses to discover and map the path, or search and map the devices from the search results.

For example, you can search all OSPF devices by the keyword ***“router ospf”***. Select all devices or a subset of devices to draw them on the map.



The intent provides a programming way to create the map so that you can create a map according to the CLI command output and have more control over what is mapped, for example,

- Map OSPF neighbors
- Map the multicasting tree
- Map the routing path

12.1.1 Create Map with Intent

In this section, we will create an intent to map the OSPF neighbors with the following steps:

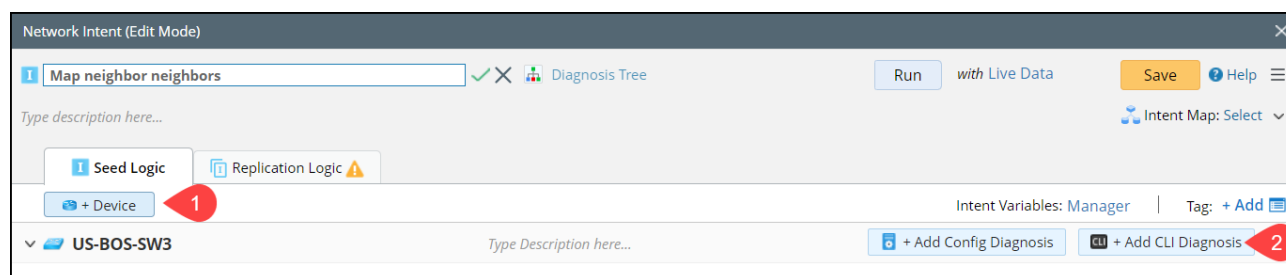
Starting with a seed device, use the command, **show ip ospf neighbor**, to retrieve all ospf neighbors. Use the table parser to parse all ospf neighbors from the CLI command output.

1. In the diagnosis, for each ospf neighbor,
 - a) Add a logic, **draw device**, to draw this device and the interface to the neighbor.
 - b) Add the other Logic, **follow-up self**, to call this intent for the neighbor devices. With this logic, the intent will be run on the neighbors of neighbors recursively.

12.1.1.1 Parse the OSPF Neighbor Table

From the **Intent Manager**, create a new intent and name it **Map neighbor neighbors**:

1. Select an OSPF device as the seed device. You can search the keyword “**router ospf**” (add the double quote around the keyword for an exact match) to find the routers with OSPF configured.
2. Add a CLI diagnosis.



3. In the **CLI Command Diagnosis** window, enter the command, **show ip ospf neighbor**, and click the **Retrieve** button to retrieve the data from the live network.
4. Select the table header and first few lines, and click the **Parse Table** button. The system automatically parses the table and creates a table variable.
5. You may want to change the default table name, **Table1**, to a meaningful name, such as **ospf_nbrs**.

CLI Command Diagnosis

US-BC 3 show ip ospf neighbor Retrieve with Live Data

1. Define Variable 2. Define Diagnosis

Format1

Double-click a variable to parse. Select multiple lines to parse a table. Critical Variable (0)

Current Device 07/17/2024 01:56:36 PM Search...

1 US-BOS-SW1>show ip ospf neighbor

2

3

4 Neighbor ID Pri State Dead Time Address

5 10.3.3.3 1 FULL/DR 00:00:31 10.8.1.19

6 10.3.3.3 1 FULL/BDR 00:00:39 10.8.1.3

7 155.16.178.124 1 FULL/DR 00:00:34 10.8.1.14

8 10.8.1.49 1 FULL/BDR 00:00:32 10.8.1.33

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

US-BOS-SW1 show ip ospf neighbor Retrieve with Live Data

1. Define Variable

Summary Text	Original Text	Search...
3	neighbor ID	
4	10.3.3.3	
5	10.3.3.3	
6	155.16.178.124	
7	10.8.1.49	

Map ospf neighbors

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

Name: Map ospf neighbors 1 Anchor: Select Variable

Type description of the diagnosis...

☒ Loop Table Rows ospf_nbrs Table Key: neighbor_id 2

If

A US-BOS-SW1 Current

neighbor_id Is not empty 3

B Select Variable

Intent Data View

Draw Map 4

Send Email

Follow-up Intent

Set Intent Baseline

Advanced

Add Logic

+ Add Elself + Add Else

Help | All Intent Variables Cancel Apply

- Under the **Draw Device**, select **this_device**. The device will be drawn on the map.
- Select Include **Interface Neighbor** and select the **interface** variable of the table.

1. Define Variable

Summary Text	Original Text	Search...
3	neighbor ID	
4	10.3.3.3	
5	10.3.3.3	
6	155.16.178.124	
7	10.8.1.49	

Map ospf neighbors

2. Define Diagnosis

Add Note Add Diagnosis Can also click a variable on the left to add automation.

☒ Loop Table Rows ospf_nbrs Table Key: neighbor_id

If

A US-BOS-S... Current

neighbor_id Is not empty

B Select Variable

Then

Diagnosis Message: \$intf is down...

☐ Set Status Code for Device:

☐ Set Status Code for Intent:

Pop up

Move Down

Delete 5

Draw Device:

Select Device: this_device a

☒ Include: interface neighbor interface IPv4 L3 Topology b

Add Logic

+ Add Elself + Add Else

5. Optionally, you can delete the message to make your intent more clear.
6. Save and run the intent. Click the icon **View Current Map** to confirm that you have the seed device, and its OSPF neighbors drawn in the map.

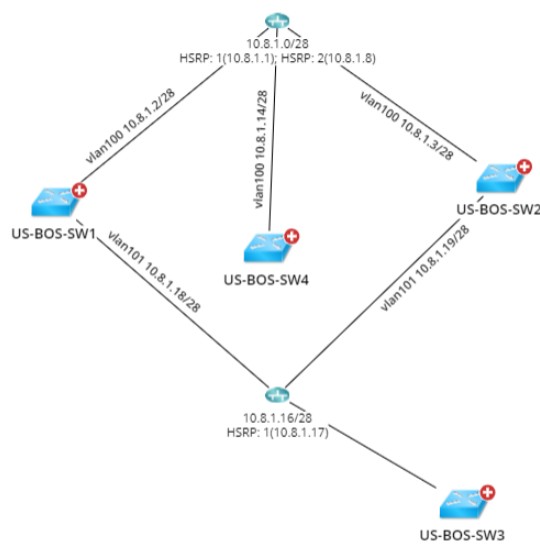
Network Intent (View Mode) - All Network Intents/GD/Map ospf neighbors

I Map ospf neighbors Open 0 0 Edit

Result: 07/17/2024 02:52 PM 👁️ 📄

Run ▼ with Live Data

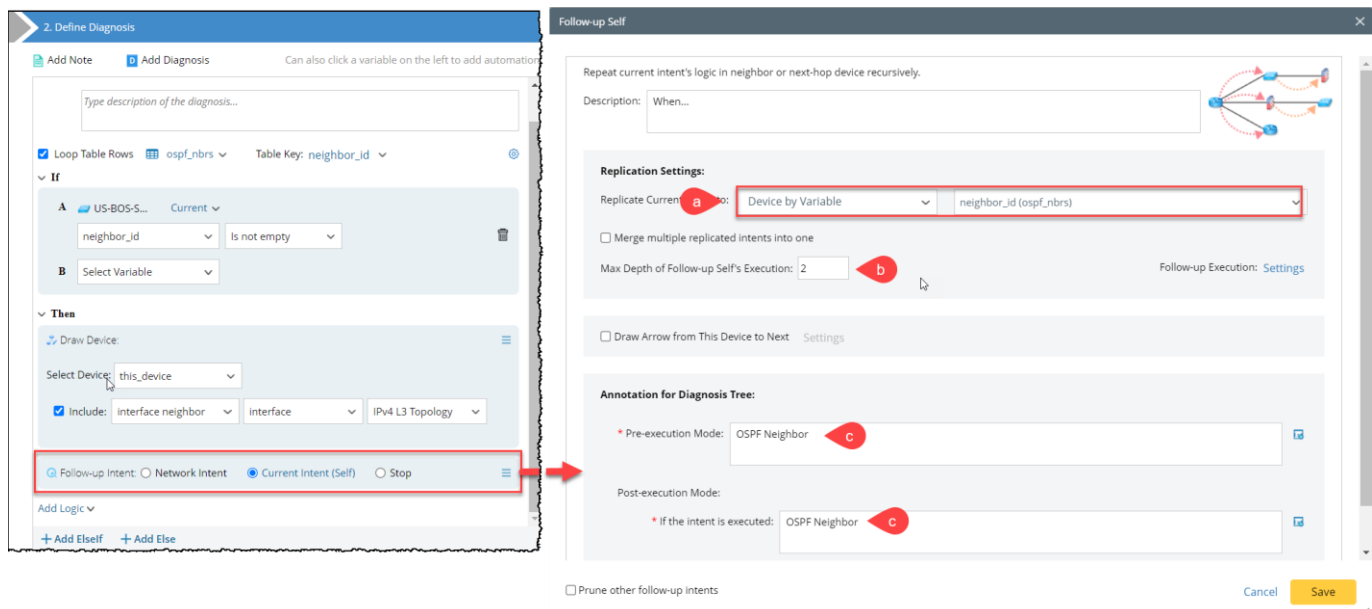
This intent execution is finished at 07/17/2024 02:52 PM with 0 errors. You can View [Execution Log](#)



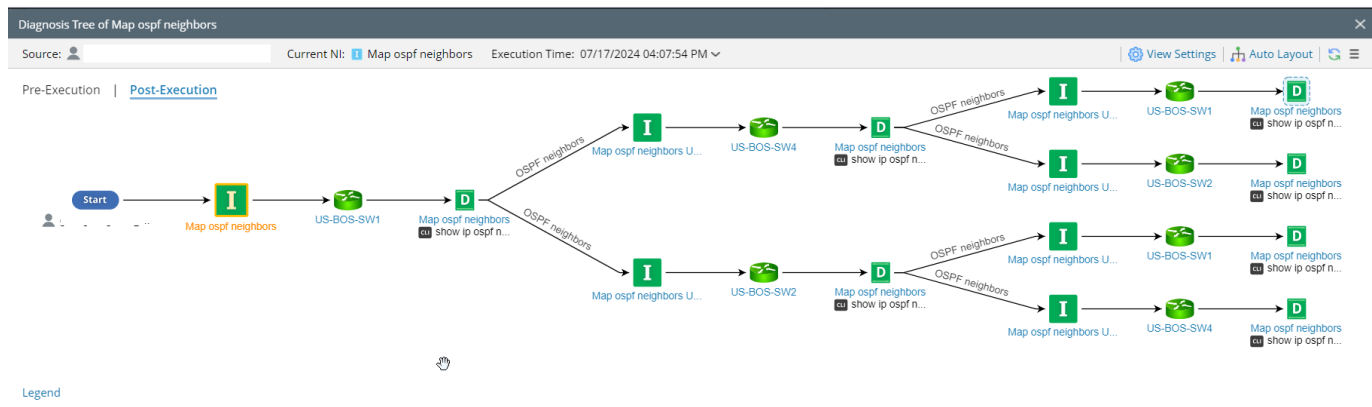
12.1.1.3 Map Neighbor Devices

Let us modify the intent to map the neighbor devices recursively. Add the logic, **follow up intent**, and select **Current Intent (Self)**. Click the **Current Intent (Self)** link.

- Replicate the current intent to the **Device by Variable**, and select the **neighbor_id(ospf_nbrs)**. The system will automatically transfer the IP address (**neighbor_id**) to the hostname.
- Set the maximum depth of the self's execution. The default is 2.
- Change the annotations for the execution tree so that it can be easily understood.

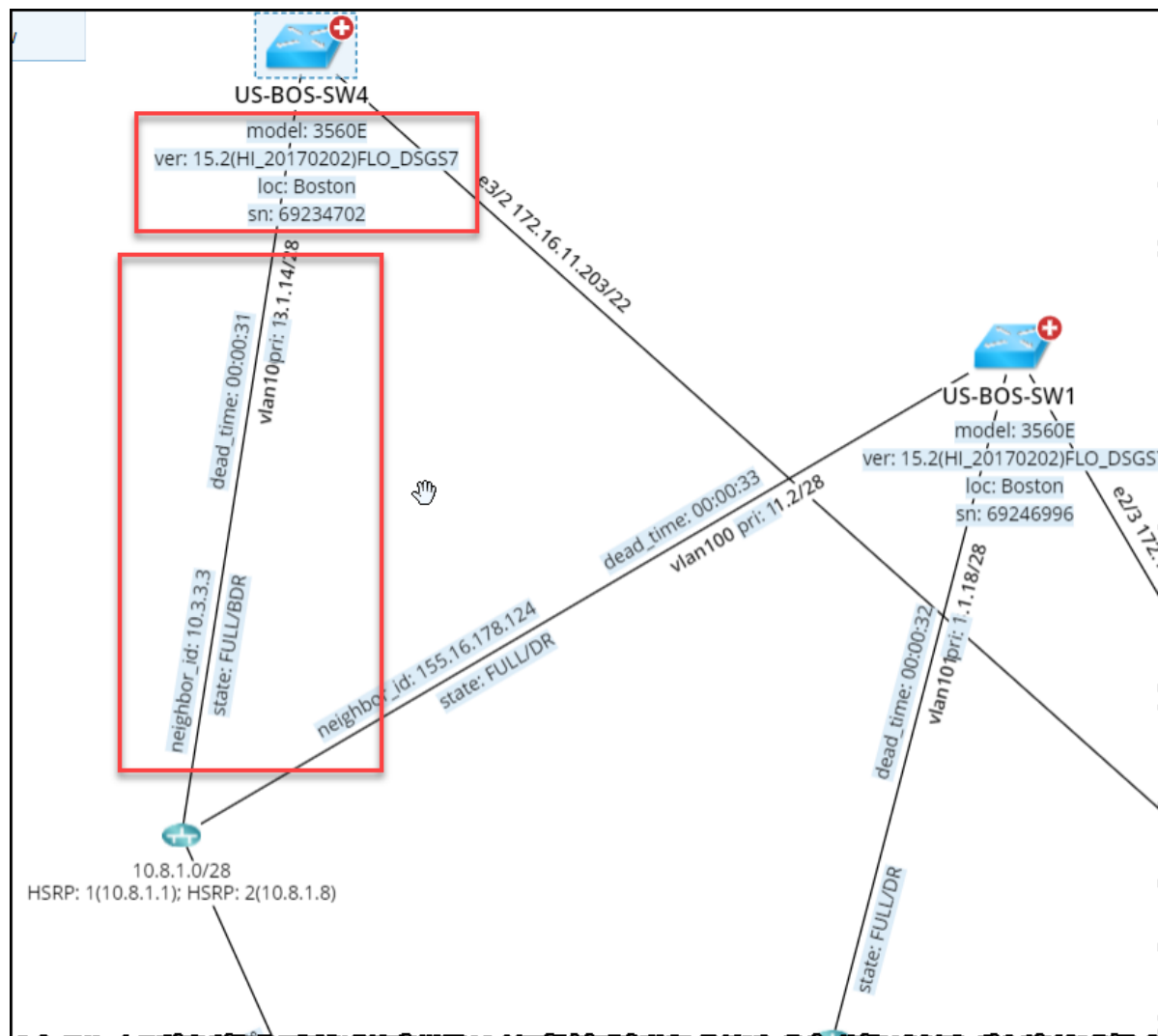


Save and run the intent. Click the icon **View Current Map** to confirm that the OSPF neighbors of the seed device's neighbors are drawn on the map. You can also view the diagnosis tree, which shows the intents are replicated for OSPF neighbors up to two depth levels.



12.1.2 Intent Data View

You can enrich the map with the data view, which can be defined with the logic, **intent data view**. The following diagram is an example of the intent data view. Under the device and along the link, you can select what variables to be displayed. Besides the variables from the intent parser, you can also display the built-in device properties, such as **model**, **software version**, **serial number (sn)**, and **location** displayed here.

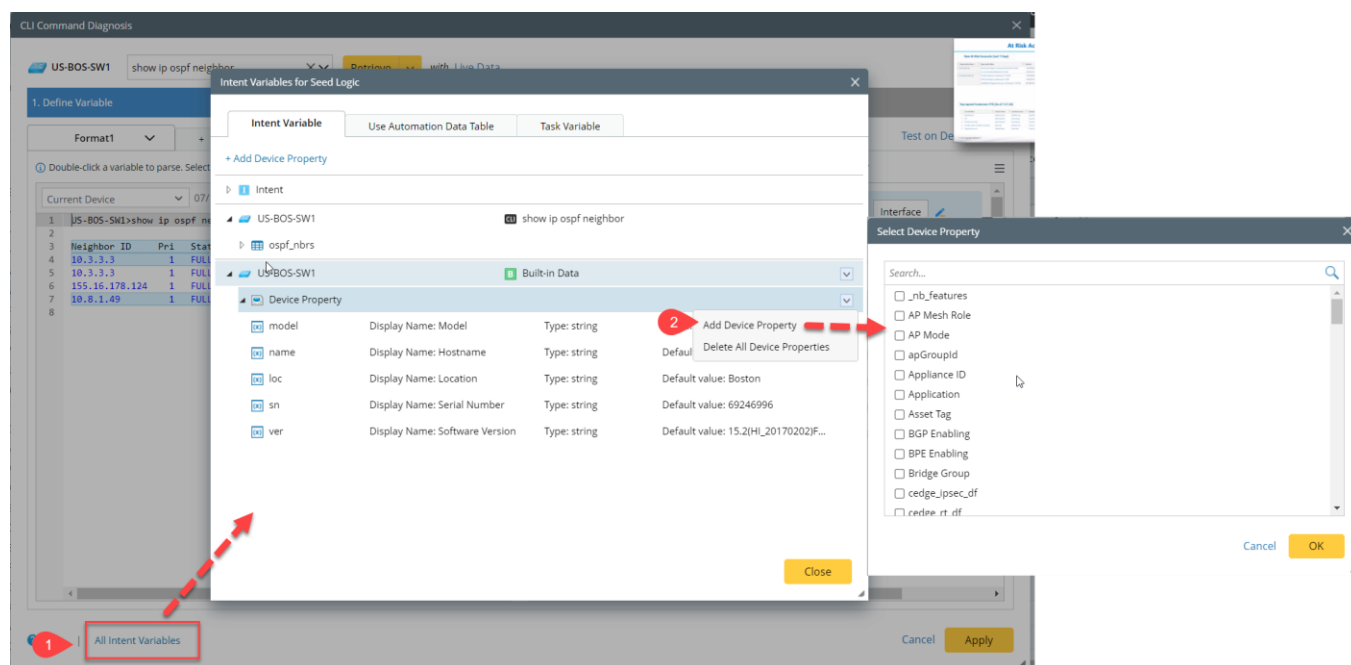


You follow two simple steps to add a data view:

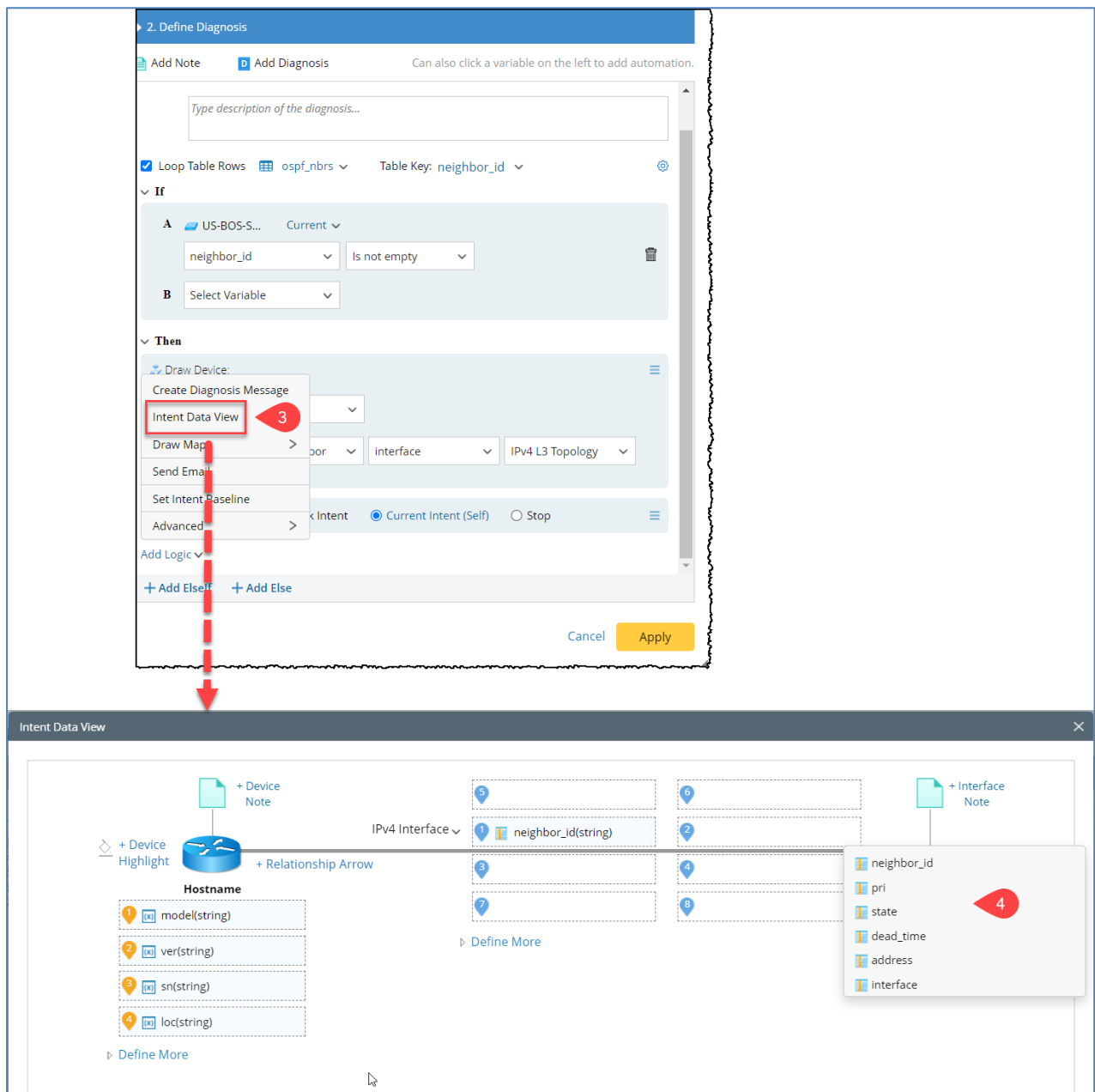
1. Add the device properties to the intent variables so that the intent can use these properties.
2. Add a new logic, Intent Data View.

Let us edit the diagnosis in the previous session:

1. Click the link **All Intent Variables** at the left of the bottom.
2. In the **Intent Variables for Seed Logic** window, click **Add Device Property** and select all device properties you want to add. You can search the properties.



3. Under the **Define Diagnosis** tag, select **Intent Data View** from the **Add Logic** pull-down menus.
4. In the **Intent Data View** window, select the variables or properties you want to display for each position under the device and along the link. While you first add an interface variable along the link, the system may ask you to set the **interface key**, which is the variable **\$interface** of the OSPF neighbor table.

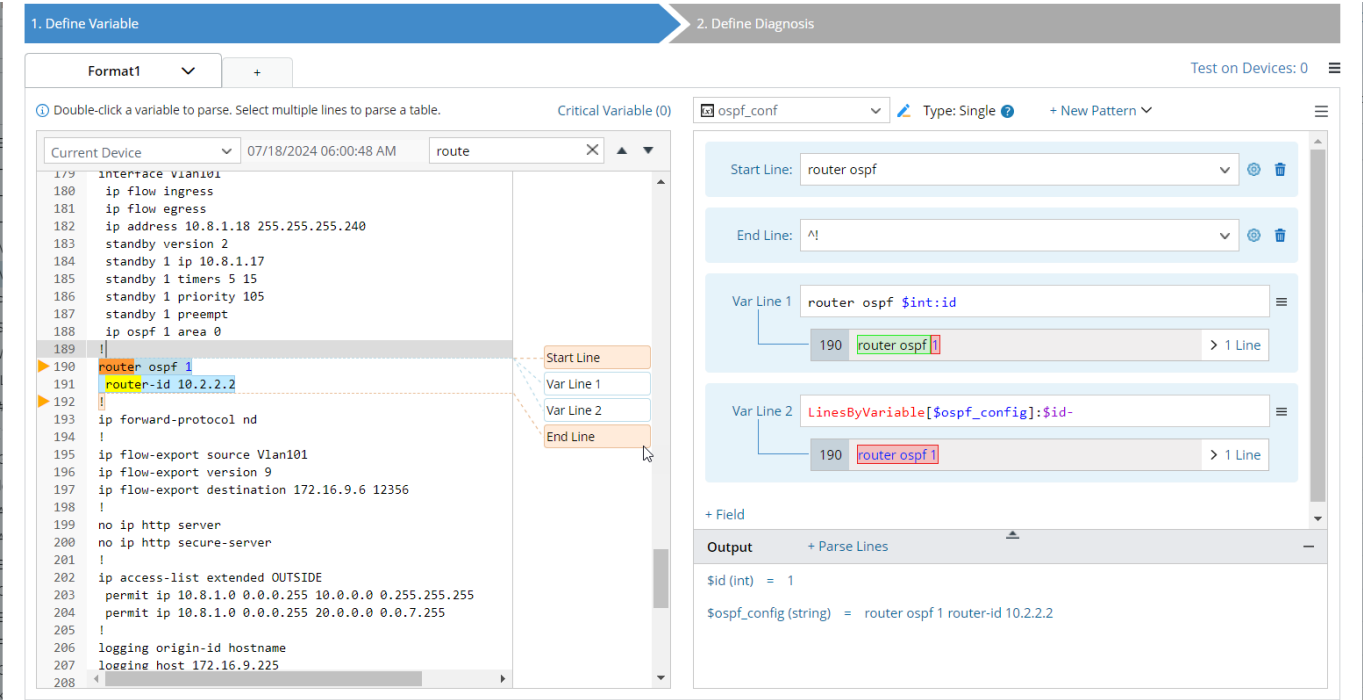


Save and run the intent. Click the icon **View Current Map** to confirm that the data is displayed on the map. You may need to zoom in to see the data along the link.

12.1.2.1 Add OSPF Configurations to Device Note

Often, it is useful to see the relevant configurations while troubleshooting a network problem or documenting a network design, which can be done with the **intent data view**. Before we add the configurations to the intent data view as the device note, you need to parse OSPF configurations with the configuration diagnosis.

Continue editing the intent of the last section. Add a **Configuration Diagnosis**, in which you parse the OSPF configurations as a single variable, **ospf_conf**:



Start Line	<i>router ospf</i>
End Line	<i>^!</i>
Var Line 1	<i>router ospf \$int:id</i>
Var Line 2	<i>LinesByVariable[\$ospf_config]:\$id-</i>

Close this diagnosis, navigate back to the CLI configuration diagnosis created in the last section and click the **Define** link at the left side of the intent data view to edit it.

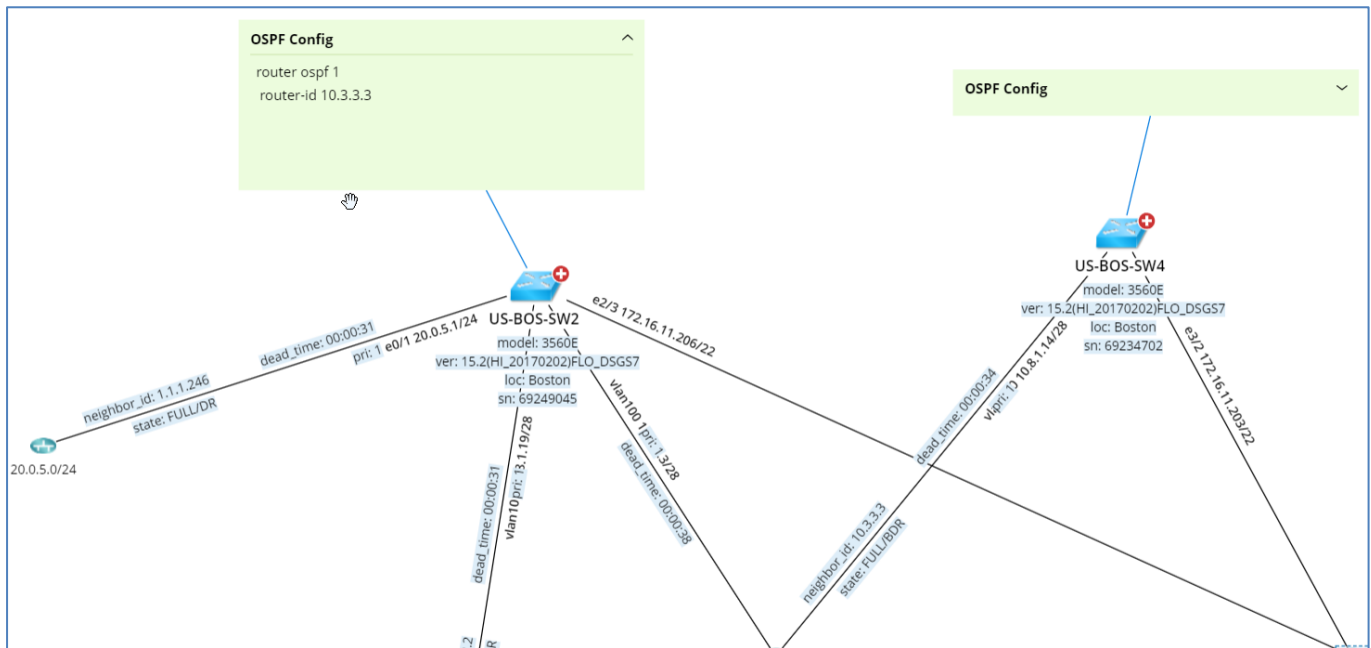
In the Intent Data View window, click the **Device Note** icon and set the following parameters of the device note:

The screenshot shows the NetBrain interface. At the top, there's a network diagram with a router icon labeled "1 Device Notes" and a red arrow pointing down to the "Device Note" configuration panel. The diagram also shows an "IPv4 Interface" with a dropdown menu and a "Relationship Arrow" between the router and the interface. The "Device Note" panel has the following fields:

- Title: OSPF Config
- Content: [Insert Variable](#) (button) followed by a text area containing `ospf_config`
- Note Type: Overwrite (dropdown menu)
- Note View: ☒ Collapsed ☐ Expanded

- Title: enter a meaning title such as **OSPF Config**.
- Content: inset the variable **ospf_config**
- Note Type: can be **Overwrite** or **Append**.
- Note View: can be collapsed or Expanded.

Save and run the intent. Click the icon **View Current Map** to confirm that the device note is displayed on the map. You may expand the note.



12.2 Create the Inventory Report



12.2.1 Built-in Inventory Reports

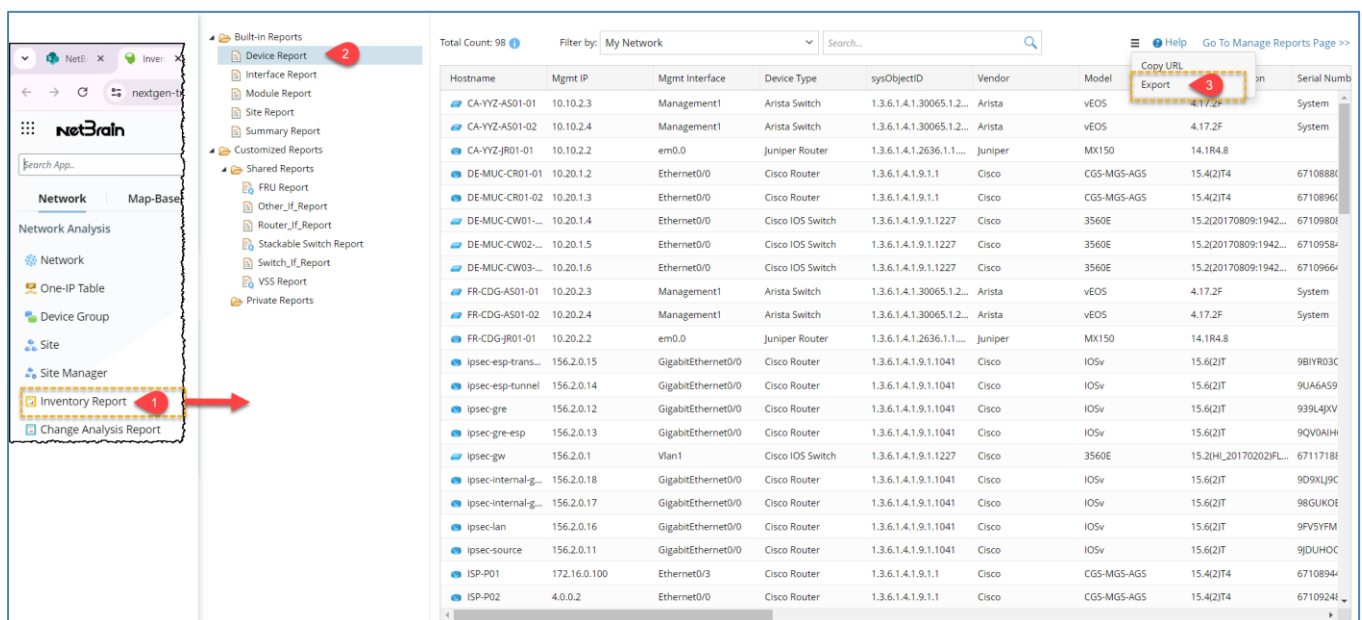
NetBrain system provides the following built-in inventory reports:

- **Device Report:** contains properties of devices in the current domain, such as management IP and device type.
- **Interface Report:** Contains the interface properties for all devices in the current domain, such as IPv4/6 address and bandwidth.
- **Module Report:** Contains the module properties for all devices in the current domain, such as module ports and slot numbers.
- **Site Report:** Contains the summary information for all sites in the current domain, such as location and device count.
- **Summary Report:** Contains the summary information for all devices in the current domain, such as module count and interface count.
- **FRU Report:** Contains the chassis hardware information about Cisco routers, Cisco IOS switches, Juniper routers, and Juniper EX switches in the current domain.

- **Stackable Switch Report:** Contains the stack information about Cisco Routers or IOS Switches in the current domain.
- **VSS Report:** Contains VSS modules in the current domain.
- **Router_If_Report:** Contains the interface properties for routers in the current domain.
- **Switch_If_Report:** Contains the interface properties for switches in the current domain.
- **Other_If_Report:** Contains the interface properties for other device types in the current domain.

It is easy to create a built-in inventory report:

1. Click the **start** menu  and select **Inventory Report**.
2. Select the target report in the left pane to browse asset details in the report. Its asset details are displayed in the right table.
3. Click the  icon and select **Export**.



The screenshot shows the NetBrain interface with the 'Inventory Report' selected in the left sidebar. The main area displays a table of network assets with columns: Hostname, Mgmt IP, Mgmt interface, Device Type, sysObjectID, Vendor, Model, and Serial Number. The table contains 98 entries. In the top right corner of the table, there is a menu with options: 'Copy URL', 'Export', and 'Print'. The 'Export' option is highlighted with a red circle and a red arrow.

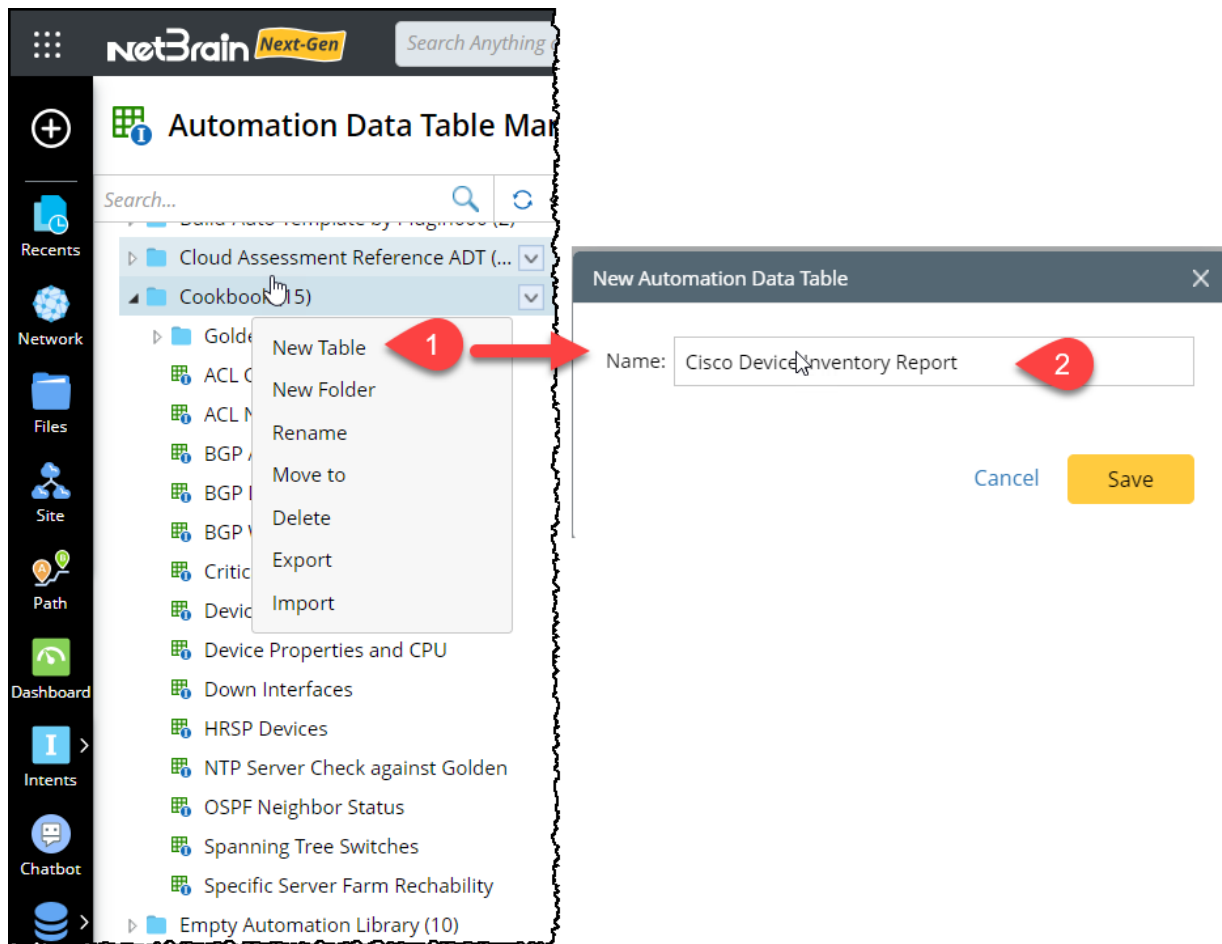
Hostname	Mgmt IP	Mgmt interface	Device Type	sysObjectID	Vendor	Model	Serial Number
CA-YYZ-AS01-01	10.10.2.3	Management1	Arista Switch	1.3.6.1.4.1.30065.1.2...	Arista	vEOS	System
CA-YYZ-AS01-02	10.10.2.4	Management1	Arista Switch	1.3.6.1.4.1.30065.1.2...	Arista	vEOS	System
CA-YYZ-JR01-01	10.10.2.2	em0.0	Juniper Router	1.3.6.1.4.1.2636.1.1...	Juniper	MX150	14.1R4.8
DE-MUC-CR01-01	10.20.1.2	Ethernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1	Cisco	CGS-MGS-AGS	15.4(2)T4
DE-MUC-CR01-02	10.20.1.3	Ethernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1	Cisco	CGS-MGS-AGS	15.4(2)T4
DE-MUC-CW01-...	10.20.1.4	Ethernet0/0	Cisco IOS Switch	1.3.6.1.4.1.9.1.1227	Cisco	3560E	15.2(20170809:1942...
DE-MUC-CW02-...	10.20.1.5	Ethernet0/0	Cisco IOS Switch	1.3.6.1.4.1.9.1.1227	Cisco	3560E	15.2(20170809:1942...
DE-MUC-CW03-...	10.20.1.6	Ethernet0/0	Cisco IOS Switch	1.3.6.1.4.1.9.1.1227	Cisco	3560E	15.2(20170809:1942...
FR-CDG-AS01-01	10.20.2.3	Management1	Arista Switch	1.3.6.1.4.1.30065.1.2...	Arista	vEOS	4.17.2F
FR-CDG-AS01-02	10.20.2.4	Management1	Arista Switch	1.3.6.1.4.1.30065.1.2...	Arista	vEOS	4.17.2F
FR-CDG-JR01-01	10.20.2.2	em0.0	Juniper Router	1.3.6.1.4.1.2636.1.1...	Juniper	MX150	14.1R4.8
ipsec-esp-trans...	156.2.0.15	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-esp-tunnel	156.2.0.14	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-gre	156.2.0.12	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-gre-esp	156.2.0.13	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-gw	156.2.0.1	Vlan1	Cisco IOS Switch	1.3.6.1.4.1.9.1.1227	Cisco	3560E	15.2(WL_20170202)PL...
ipsec-internal-g...	156.2.0.18	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-internal-g...	156.2.0.17	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-lan	156.2.0.16	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ipsec-source	156.2.0.11	GigabitEthernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1041	Cisco	IOSv	15.6(2)T
ISP-P01	172.16.0.100	Ethernet0/3	Cisco Router	1.3.6.1.4.1.9.1.1	Cisco	CGS-MGS-AGS	15.4(2)T4
ISP-P02	4.0.0.2	Ethernet0/0	Cisco Router	1.3.6.1.4.1.9.1.1	Cisco	CGS-MGS-AGS	15.4(2)T4

Though it is easy to export a built-in inventory report, you cannot customize the fields you want to export. To create an inventory report according to your requirements, build an ADT and export the ADT table to a CSV report.

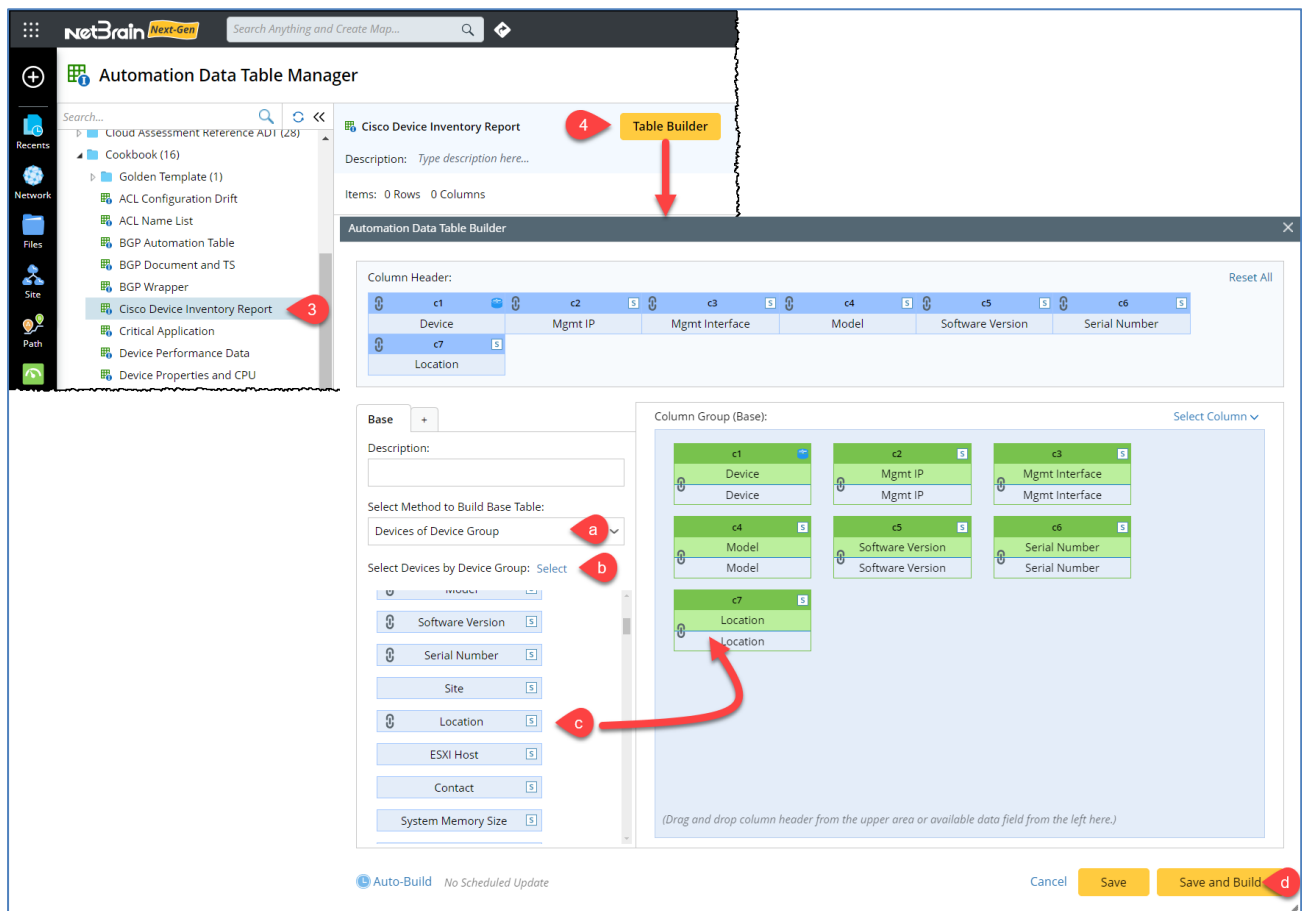
12.2.2 Use ADT to Build Inventory Report

ADT provides a flexible way for you to build the inventory report according to your requirements:

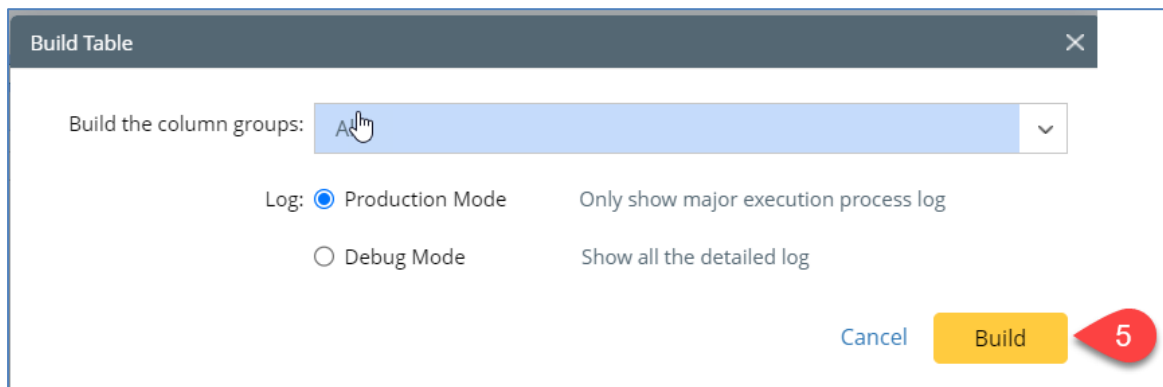
1. In the **Automation Data Table Manager** window, right-click a folder and select the menu item **New Table**.
2. In the **New Automation Data Table** window, enter a name, such as the **Cisco Device Inventory Report** and click the **Save** button.



3. Select the table you just created.
4. Click the Table Builder to open the Automation Data Table Builder window.
 - a) Select the method **Devies of Device Group** as the method to build the base table.
 - b) Select a device group, such as a device group to include all Cisco Devices.
 - c) Drag the fields you are interested into the right pane. You can change the header name.
 - d) Click the Save and Builder.



- Click the **Build** button in the pop-up window **Build Table**.



- Wait for the system to finish building the table. Some data cells may be blank. Click the menu bar at the top right corner and select **Export to CSV Only** to export the data into a CSV file.

Automation Data Table Manager

Search... Table Builder Last Updated at: 07/15/2024 03:11 PM Rebuild Table

Description: Type description here...

Items: 30 Rows 7 Columns

Search...

6 Export to CSV Only

Import from System Table
Lock Settings
Execution log
Export
Export to CSV Only
Export Datasets to File
Dataset Tag Settings
Table Settings

No.	Device	Mgmt IP	Mgmt Interface	Model	Software Version	
1	Berlin-R1	172.16.8.60	Ethernet0/1	CGS-MGS-AGS	15.4(2)T4	
2	DE-MUC-CR01-01	10.20.1.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	
3	DE-MUC-CR01-02	10.20.1.3	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	
4	ISP-PE02	4.0.0.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67109248
5	ISP-PE01	1.0.0.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67109152
6	ISP-PE02	2.0.0.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67109024
7	ISP-PE03	3.0.0.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67108992
8	JP-TYO-CR01-01	10.30.0.2	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67109104 DC-3
9	JP-TYO-CR01-02	10.30.0.3	Ethernet0/0	CGS-MGS-AGS	15.4(2)T4	67109088 DC-3
10	NORI30148WRA1A	10.87.122.5	TenGigabitEthernet0/1/5	ASR1001-HX	16.12.05	TKM21080352 US
11	NORI30148WRC0B	10.87.122.130	TenGigabitEthernet0/1/5	ASR1001-HX	16.12.05	TKM21110248 US
12	RT01-VNDR-SYKE-ASH-V...	10.78.252.1	Loopback0	ISR4431/K9	16.12.05	FJC2332A06R
13	RT01-VNDR-SYKE-CENT...	10.78.253.132	GigabitEthernet0/0/0.100	ISR4431/K9	16.12.05	FJC2332A06Q
14	RT01-VNDR-TTEC-AUS-T...	10.78.16.1	Loopback0	ASR1002-HX	16.12.05	FXS2144Q06j
15	RT01-VNDR-TTEC-CENT...	10.78.24.22	TenGigabitEthernet0/1/...	ASR1002-HX	16.12.05	FXS2144Q065
16	RT02-VNDR-SYKE-ASH-V...	10.78.252.133	GigabitEthernet0/0/1.100	ISR4431/K9	16.12.05	FJC2332A065
17	RT02-VNDR-SYKE-CENT...	10.78.253.2	Loopback0	ISR4431/K9	16.12.05	FJC2332A06T
18	RT02-VNDR-TTEC-AUS-T...	10.78.16.133	TenGigabitEthernet0/1/...	ASR1002-HX	16.12.05	FXS2222Q2XH

Similarly, you can create an inventory report at the interface level. Follow the same steps and select the method **Interfaces of Device Group** at step 6a. In step 6b, you can only select the device group that is created with the static interface or dynamic interface search.

Column Header: Reset All

c1	c2	c3	i1	c4	c5	c6	c7
Device	Interface Name	IPv4 Address	Bandwidth	Speed	Duplex	Inbound ACL	Live Status

Base +

Description:

Select Method to Build Base Table:

Interfaces of Device Group

Select Interfaces by Device Group: Shared Devic...

Bandwidth [U]

Speed [S]

Duplex [S]

Live Status [S]

MAC Address [S]

Slot# [S]

Module Type [S]

Description [S]

Routing Protocol [S]

Tunnel Mode [S]

Multicasting Mode [S]

Column Group (Base): Select Column

c1	c2	c3	i1	c4
Device	Interface Name	IPv4 Address	Bandwidth	Speed
c5	c7			
Duplex	Live Status			
Duplex	Live Status			

(Drag and drop column header from the upper area or available data field from the left here.)

Auto-Build No Scheduled Update Cancel Save Save and Build

You can view the interface data and export it to a CSV file.

Automation Data Table Manager

Interface Level Inventory Report

Description: Type description here...

Items: 325 Rows 8 Columns

Search...

Advanced Filter: Undefined

No.	Device	Interface Name	Pv4 Address	Bandwidth	Speed	Duplex	Live Status	Interface
21	ITE_EXTEND	GigabitEthernet0/45		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
22	ITE_EXTEND	GigabitEthernet0/44		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
23	ITE_EXTEND	GigabitEthernet0/43		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
24	ITE_EXTEND	GigabitEthernet0/42		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
25	ITE_EXTEND	GigabitEthernet0/41		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
26	ITE_EXTEND	GigabitEthernet0/40		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
27	ITE_EXTEND	GigabitEthernet0/39		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
28	ITE_EXTEND	GigabitEthernet0/38		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
29	ITE_EXTEND	GigabitEthernet0/37		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
30	ITE_EXTEND	GigabitEthernet0/36		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
31	ITE_EXTEND	GigabitEthernet0/35		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
32	ITE_EXTEND	GigabitEthernet0/34		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
33	ITE_EXTEND	GigabitEthernet0/33		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
34	ITE_EXTEND	GigabitEthernet0/32		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...
35	ITE_EXTEND	GigabitEthernet0/31		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
36	ITE_EXTEND	GigabitEthernet0/30		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
37	ITE_EXTEND	GigabitEthernet0/29		10000	auto	auto	down/down	ITE_EXTEND - Gigab...
38	ITE_EXTEND	GigabitEthernet0/28		1000000	a-1000	a-full	up/up	ITE_EXTEND - Gigab...

12.3 Create the Report from CLI Command Results

If the built-in data cannot satisfy your need, you may retrieve the data from the CLI commands and create a report from the output of the CLI commands. As you have learned from the previous chapters, there are two ways to create a report from the CLI command results:

- Create a CSV report from an intent and replicate the intent to an ADT. Then from the ADT, you can create a summary report, which combines the CSV reports of all replicated intents.
- Build an ADT base from Pre-Replicated Intent Templates. In the intent replication setting, define all variables you want to export to the ADT as the **signature variables**.

12.3.1 Create CSV Report from CLI Command Results

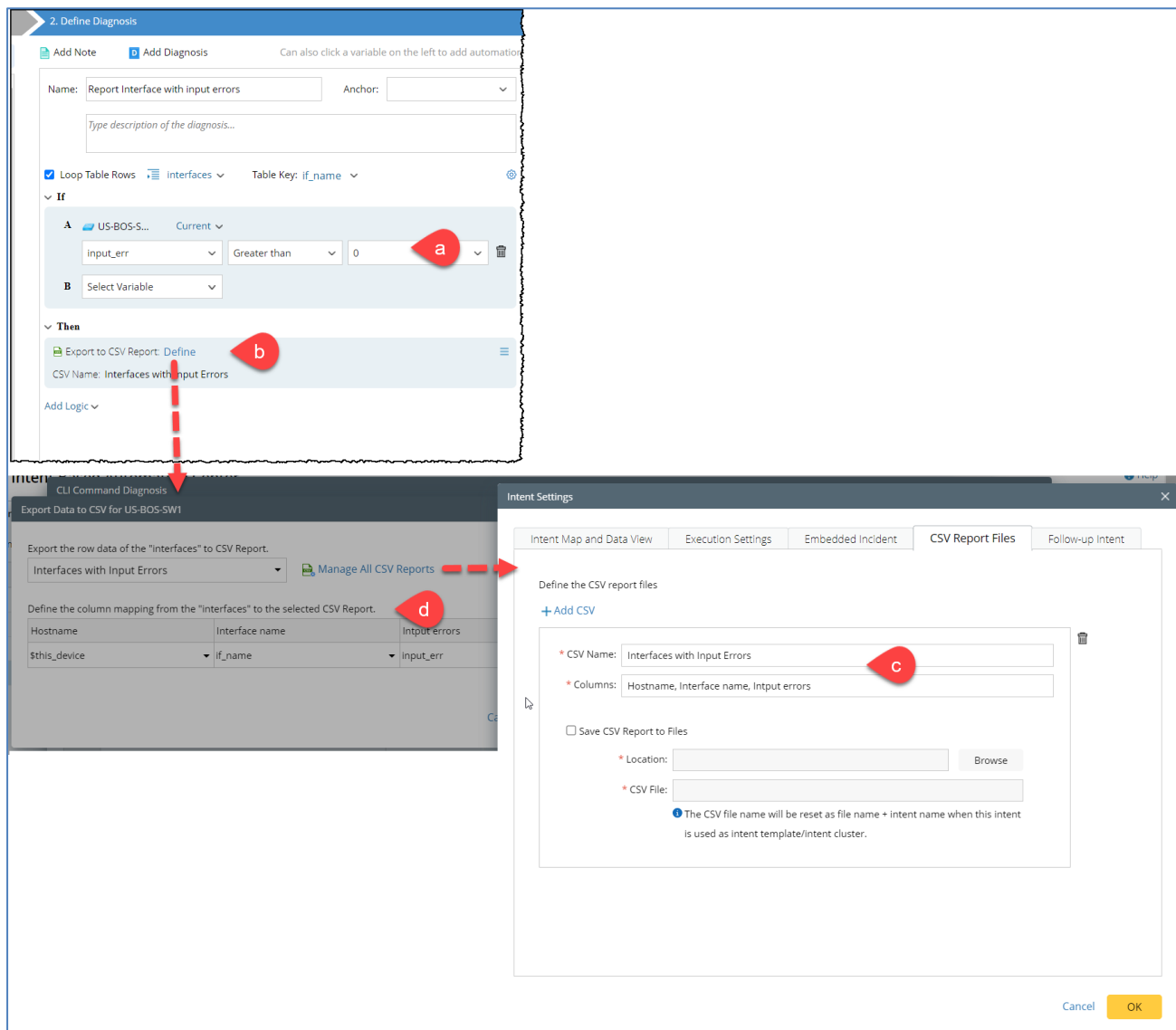
In section 5.2, you have learned how to create a CSV report of all ACL configurations. Let us review the key steps with a simple example: create a report for all interfaces which has input errors larger than 0.

Create an intent **Interface with Input Errors**. Issue the CLI command, **show interfaces**, and parse the interface name, status and input errors with the **Paragraph** Parser. The Paragraph Parser has the ID line, **\$if_name is \$status, line protocol is**, and Var Line 1 as, **\$int:input_err input errors**. You may add other variables such as **input rate, CRC error**, etc.

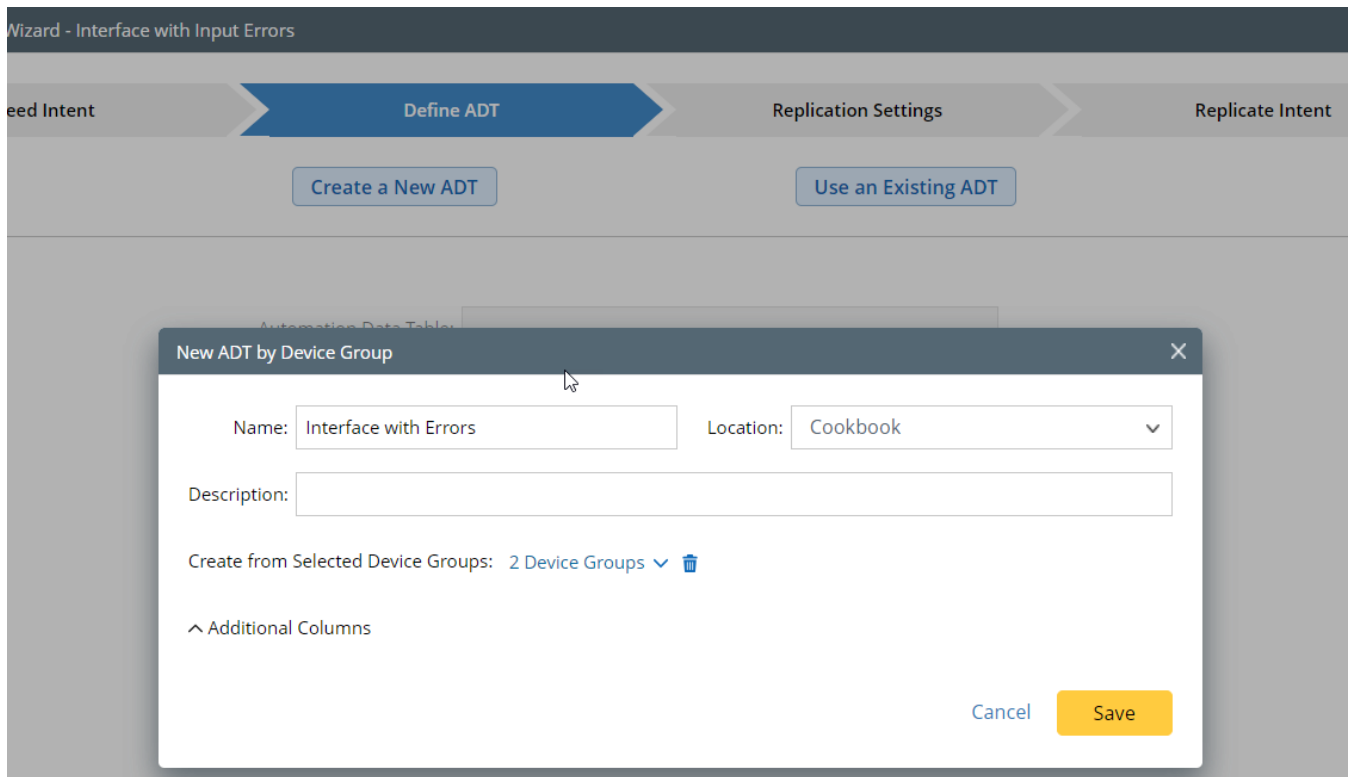
The screenshot shows the 'CLI Command Diagnosis' tool interface. At the top, the command 'show interfaces' is entered, and 'Retrieve' is clicked. Below, the '1. Define Variable' step is active. The 'Current Device' is 'US-BOS-SW1' and the 'Search' field is empty. The 'Critical Variable (0)' section shows the 'ID Line A' as '\$if_name is \$status, line protocol is' and 'Var Line 1' as '\$int:input_err input errors'. The '2. Define Diagnosis' step is also visible. The 'Output' table shows the following data:

\$if_name	\$status	\$input_err
Ethernet0/0	up	0
Ethernet0/1	up	0
Ethernet0/2	up	0
Ethernet0/3	up	0
Ethernet1/0	up	0
Ethernet1/1	up	0

1. Add a diagnosis to export the interfaces with the input errors.
 - a) Loop through the table variables defined in step 1, **interfaces**. Define the condition as, **If the input_errs is greater than 0**, add a logic, **Export to CSV Report**.
 - b) Define the CSV report.
 - c) Click the link **Manage All CSV Reports** and add a CSV report file. Define the file name and column names, which are separated by the comma (.).
 - d) Define the mapping between the variables and column names.



2. Replicate the seed intent to a new ADT. Select the **Intent Replication Wizard** from the intent menu bar and follow the wizard to create a new ADT:
 - a) In 1st step (**Seed Intent**), select the template for **Device-based Replication**.
 - b) In 2nd step (**Define ADT**), select **Create a New ADT**. Define the name and select the device group(s) to include devices for the intent to replicate on.
 - c) In the 3rd step, define the **intent qualification**. You can qualify by the device group and select the same device groups as step b.
 - d) In the 4th step, build and replicate the intents.



3. Open the ADT created in step 3 and wait for the replication process to finish.
 - a) Select the cloned intent column and select **Run Intents Once**. Wait for the system to finish the executions.
 - b) Select the cloned intent column and select **View Summary Report**.

nd Create Map...

Incident Search Incident...

Top3 Solution Lab

ager

Interface Having Input Errors

Table Builder

Last Updated at: 07/18/2024 04:45 PM

Rebuild Table

Add Data Manually

Description: Type description here...

Items: 47 Rows 2 Columns

Search...

Advanced Filter: Undefined

No.	Device	Replicated Intent
29	UK-LHR-CR01-02	Interface with Input Errors UK-LHR-CR01-02
30	UK-LHR-CW01-01	Interface with Input Errors UK-LHR-CW01-01 1
31	UK-LHR-CW01-02	Interface with Input Errors UK-LHR-CW01-02 1
32	US-BOS-CR01-01	Interface with Input Errors US-BOS-CR01-01 1
33	US-BOS-CR01-02	Interface with Input Errors US-BOS-CR01-02 1
34	US-BOS-CW01-01	Interface with Input Errors US-BOS-CW01-01 1
35	US-BOS-CW01-02	Interface with Input Errors US-BOS-CW01-02 1
36	US-BOS-R1	Interface with Input Errors US-BOS-R1
37	US-BOS-R2	Interface with Input Errors US-BOS-R2 1
38	US-BOS-SW1	Interface with Input Errors US-BOS-SW1 1
39	US-BOS-SW2	Interface with Input Errors US-BOS-SW2 1
40	US-BOS-SW3	Interface with Input Errors US-BOS-SW3 1
41	US-BOS-SW4	Interface with Input Errors US-BOS-SW4 1
42	US-NYJ-CR01-01	Interface with Input Errors US-NYJ-CR01-01 1
43	US-NYJ-CR01-02	Interface with Input Errors US-NYJ-CR01-02 1
44	US-NYJ-CW01-01	Interface with Input Errors US-NYJ-CW01-01 1
45	US-NYJ-CW01-02	Interface with Input Errors US-NYJ-CW01-02 1
46	VXLAN-MGMT	
47	Wireless_DHCP_Switch	

Run Intents Once

Run Intents on a Timer

Open Seed Intent

Rebuild Intent-related Column Group

Remove Empty Wrapper Intent

Enable Auto Intent

Export Diagnosis Result to CSV

View Summary Report

Export Intent Output Map

Debug Empty Cells

Tag Current Column

Edit

Delete

Set as Table Key

Submit Related Commands to Benchmark

New Intent Dashboard

The following is an example of the summary report. You can export to a CSV file.

View Summary Report - Replicated Intent (28 intents)

Create summary report of all the CSV reports generated by intents in this column:

Only merge CSV reports generated in: 1 Weeks Create 28 of 28 intent results filtered

Interfaces with Inp...

Items: 5 rows 3 columns

Search...

Hostname	Interface name	Input errors
ISP-PE02	Ethernet1/1	1328
US-BOS-R2	Ethernet0/2	43768
ISP-PE01	Ethernet1/0	2
Berlin-R1	Ethernet0/0	469
Berlin-R1	Ethernet0/3	3013

Export

Close

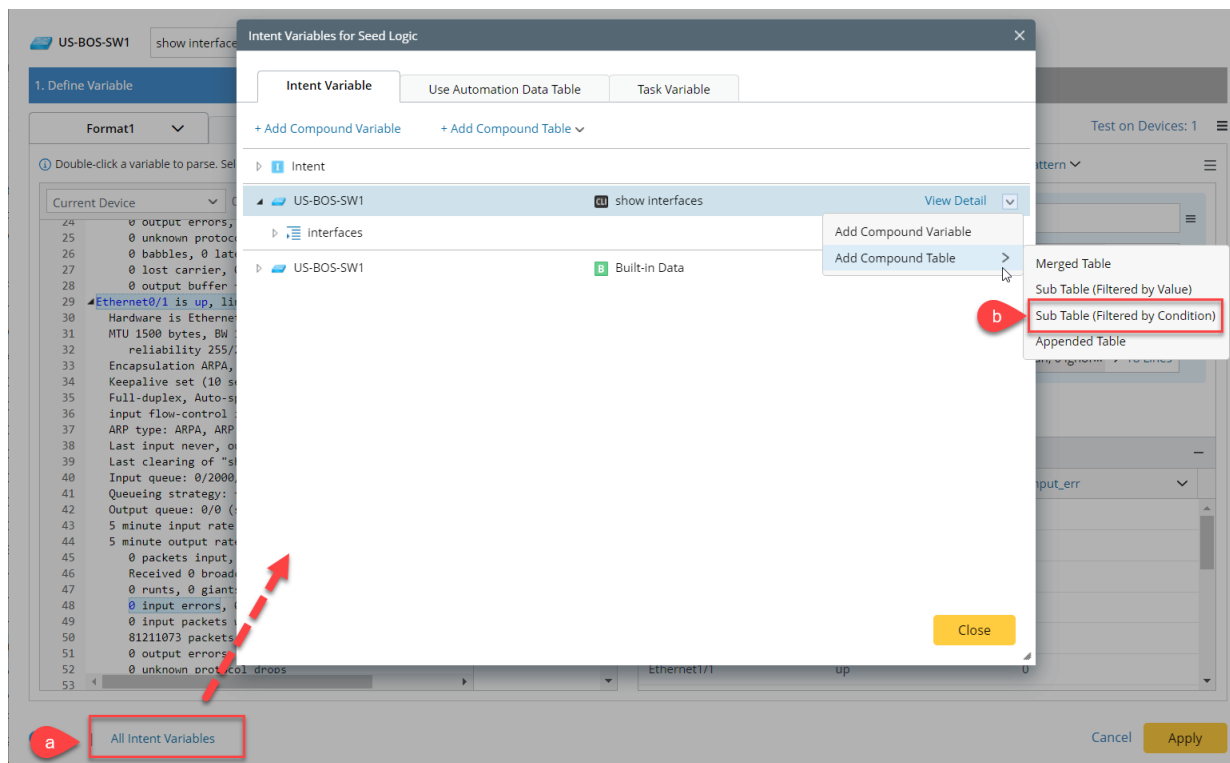
12.3.2 Create Report by Pre-Replicated Intent Template

The other method to create the report is to build an ADT via the method **Pre-Replicated Intent Template**. The key concept is the **signature variables**, which can be defined in the **intent replication setting** and exported to the ADT base. For the signature variables to be imported, you have to install and decode the intent in the **Intent Based Automation Center**.

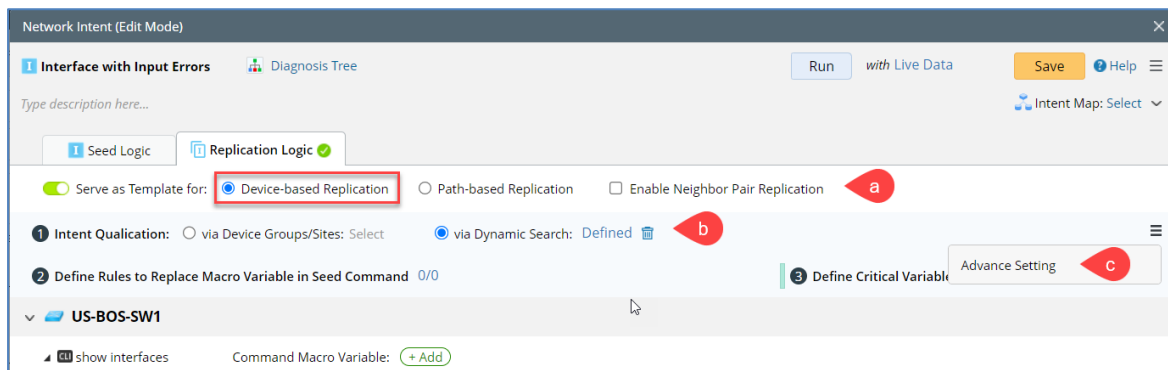
Let us reuse the intent in Section 2.3.1 to create an ADT to display all interfaces with the input errors. Edit the intent, keep the parser and diagnosis (optionally, you can remove the diagnosis to keep the intent clean), and then,

Add a subtable to filter interfaces with input errors.

- Click the **All Intent Variables** link.
- Under the seed device, select Add Compound Table > Sub Table (Filter by Condition).

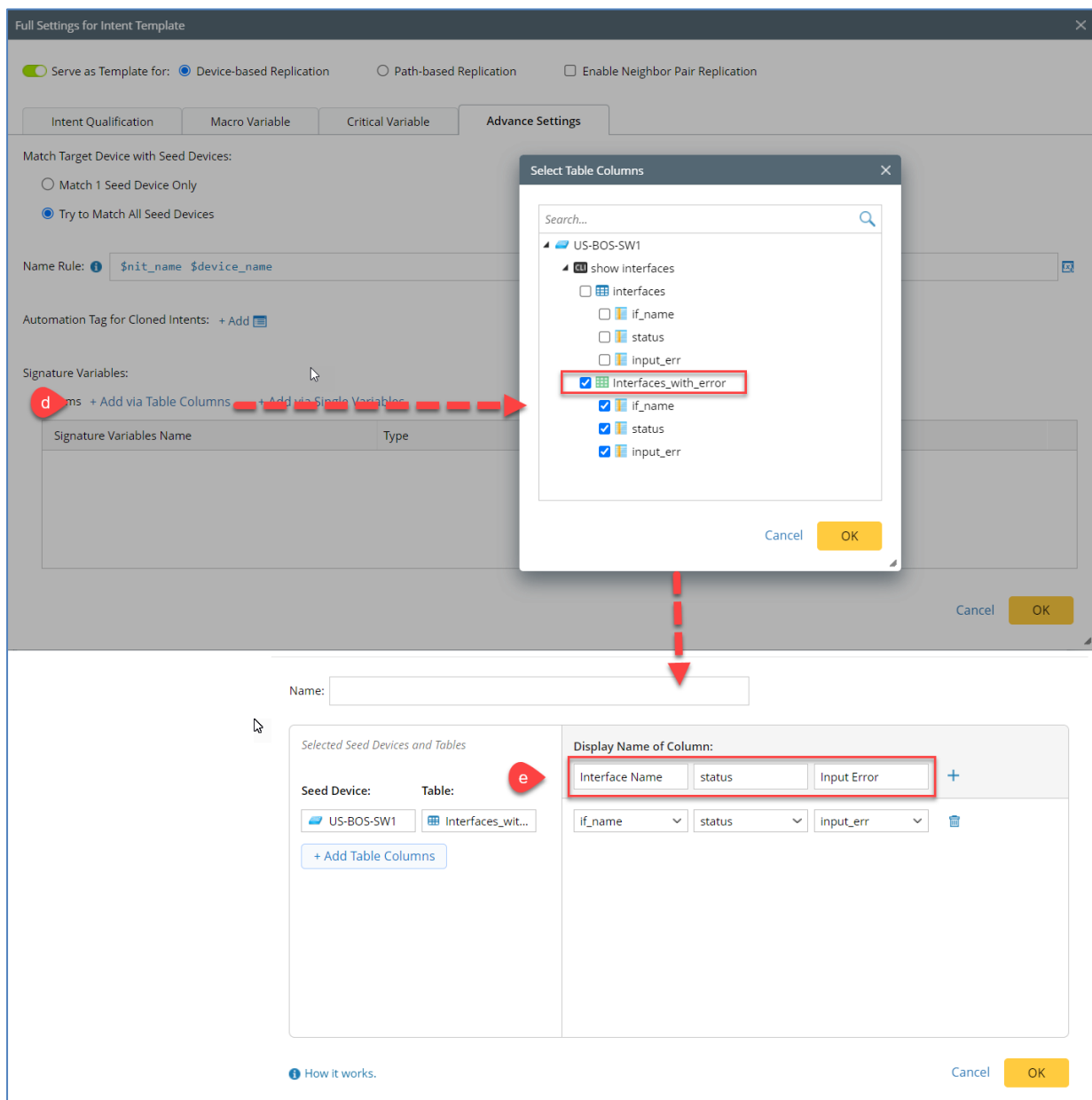


- In the **Add Sub Table** window, enter a table name, select the base table (**interfaces**), and define the filtering logic: **Only Keep** table rows where **input_err > 0**. The filtered table is displayed at the lower half of the window.

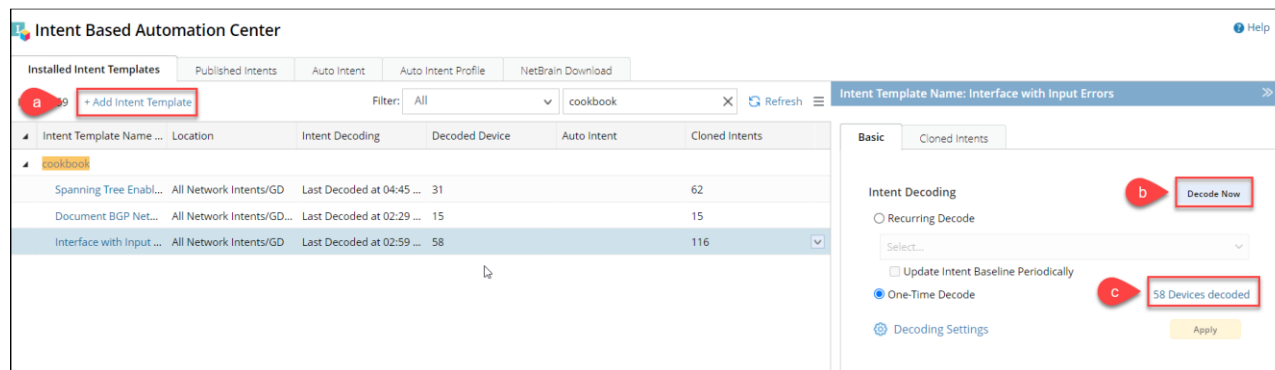


d) Click Add via Table Columns and the subtable you just created, select Interfaces_with_error.

e) In the **Add Signature Variable** window, enter a variable name and edit the display names.



6. Install and decode the intent in the **Intent Based Automation Center**.
 - a) Open the **Intent Based Automation Center** and click **+Add an Intent Template**. Select the intent you just saved.
 - b) Click **Decode Now** to decode the intents.
 - c) Wait for the system to finish replicating and decoding intents. You can see the number of devices already decoded. Click it to view the execution log.



7. Create an ADT from the pre-replicated (installed) intent. Create a new ADT from the **Automation Data Table Manager** and click the **Table Builder**,
 - a) Select the Pre-replicated Intent Template as the method.
 - b) Select the intent template you just saved.
 - c) Drag and drop the signature variables to columns.
 - d) Click **Save and Build** to build the ADT.

Automation Data Table Builder

Column Header:

c1

c2

c3

c4

Device

Interfaces - Interface N...

Interfaces - status

Interfaces - Input Error

Reset All

Base

+

Description:

Select Method to Build Base Table:

Pre-replicated Intent Template

Intent Template: Interface with Input Errors

Built-in Fields:

Device

Intent

Device Signature Variable:

interface_with_error - I...

interface_with_error - s...

interface_with_error - I...

Intent Output:

Intent Message

Intent Status Code

Column Group (Base):

Select Column

c1

Device

Device

c2

Interfaces - Interf...

interface_with_error...

c3

Interfaces - status

interface_with_error...

c4

Interfaces - Input ...

interface_with_error...

(Drag and drop column header from the upper area or available data field from the left here.)

Filter Row

Filter devices of cloned intent and member intent by Device Group/Sites: Select

Auto-Build

No Scheduled Update

Cancel

Save

Save and Build

A sample result is shown as follows. You can create a dashboard from this ADT or export it to a CSV file.

No.	Device	Interfaces - Interface Name	Interfaces - status	Interfaces - Input Error
41	ITE_EXTEND	GigabitEthernet0/9	up	137
42	ITE_EXTEND	GigabitEthernet0/11	up	93
43	ITE_EXTEND	Port-channel2	up	137
44	ITE_EXTEND	Port-channel3	up	93
45	CP-SW3			
46	UK-LHR-CW04-02			
47	DE-MUC-CW01-01			
48	US-BOS-CW04-02			
49	US-BOS-CR01-01			
50	JP-TYO-CW02-01			
51	US-BOS-CW02-02			
52	JP-TYO-CW01-01			
53	SG-SIN-CW02-01			
54	ISP-PE02	Ethernet1/1	up	1337
55	ISP-P02			
56	ISP-PE01	Ethernet1/0	up	2
57	Berlin-R1	Ethernet0/0	up	469
58	Berlin-R1	Ethernet0/3	up	3021