NetBrain® Next-Gen R12.1

NetBrain Network Intent Built-in Functions

Version 12.1 | Last Updated 2025-04-18

Contents

1 Introduction

Network Intent (NI) supports a Built-in functions, which converts the single variable into different formats or calculates new variables using two or more existing single variables.

The following table introduces the Built-in Functions that will assist you in defining compound variables or Formula Column.

Category	Sample Functions	Example Return Value (Depends on input value)
IP Conversion	IPStringToNumber($string Ip)	IPStringToNumber(“192.168.91.1”) Return: 3232258817
	IPNumberToString($number ip)	IPNumberToString(“3232258817”) Return:192.168.91.1
Numeric Conversion	PercentageToNumber($string percent)	PercentageToNumber(“50%”) Return: 50
	ByteConverter($string input)	ByteConverter(“35MB”) Return: 35 MB after conversion woll be 36,700,160 Bytes (35 × 1024 × 1024)
String Transformation	string TrimRight(string original, string trimChars)	TrimRight("4.03, ", " ,") Return: "4.03"
	string TrimLeft($string original, string trimChars)	TrimLeft("-4.03", "-") Return: "4.03"
	string Trim(string original, string trimChars)	Trim($str, " ") Return: "policy-map qos class videopolice 40960000..."
	string Replace(string original, string old, string new)	Replace("2020-8-23", "-", "/") Return: "2020/8/23"
	list<string> SplitString(string input, string SeparatorChars)	SplitString("device1,device2,device3", ",") Return: [“device1","device2","device3”]
Network Utilities	IPToHostname(string ip)	IPToHostname("192.168.91.1") Return: "R1"
	IPToInterfaceName(string ipaddr)	IPToInterfaceName("192.168.91.1") Return: "FastEthernet1/1"

Category	Sample Functions	Example Return Value (Depends on input value)
	InterfaceShortToFull(string hostname, string shortname)	InterfaceShortToFull("R1", "Eth2/1") Return: "Ethernet2/1"
	IPToSubnet(string ip)	IPToSubnet("192.168.100.15") Return: "192.168.100.0/28"
	MACToVendor(string mac_address)	MACToVendor("AAAA.1111.2222.4444") Return: "Cisco"
	FormatMAC(string mac_address)	FormatMAC("D8-67-D9-86-59-40") Return: "d867.d986.5940"
	SNToDevice(string serial_number)	SNToDevice("SNX1992") Return: "BJ*POP"
Mathematical Operations	Round(number number)	Round(36.8732994298449) Return: 36.87
	Find(string str, string substr)	Find("router ospf 100", "ospf") Return: 1 (True) or 0 (False)
	CompareDate(string target_date)	CompareDate("$end_of_sale_date") Return: 0($end_of_sale_date < $now) or 1($end_of_sale_date >= $now)
Table Operations	GetTableRowCount(string table_name)	GetTableRowCount("ospf_nbrs") Return: 32
	Average(table table, string column_name)	Average($bgp, "bgp_routes") Return: 10
	Sum(table table, string column_name)	Sum($bgp, "bgp_routes") Return: 11
	Max(table table, string column_name)	Max($intfs, "traffic") Return: 12
	Min(table table, string column_name)	Min($intfs, "traffic") Return: 13
Device and Interface Management	Device(string Hostname or IP address)	Device("BJ*POP") Return: device
	DoesDeviceExist(string Hostname or ip address)	DoesDeviceExist("BJ*POP")

Category	Sample Functions	Example Return Value (Depends on input value)
		Return: True/False
	Interface(string Hostname, string InterfaceName, string InterfaceType="Interface")	Interface("BJ*POP", "ethernet0/0", "1") Return: interface
	InterfaceByIP(string IPAddress)	InterfaceByIP("10.8.1.30") Return: interface
	DoesInterfaceExist(string Hostname, string InterfaceName)	DoesInterfaceExist("BJ*POP", "ethernet0/0") Return: True/False
	FormalizeIntfStatus(string intf_status)	FormalizeIntfStatus("up") Return: "UP"
Path and Map Management	Path(string ApplicationName, string PathName)	Path("EmailService", "path Name") Return: path
	NewMap(string FullPath)	NewMap("Public\\DeviceNeighbor") Return: map
	Map(string FullPath)	Map("Public\\OpenTopo") Return: map
	DeleteMap(string FullPath)	DeleteMap("Public\\OpenTopo") Return: N/A
	GetSiteOverviewMap()	GetSiteOverviewMap() Return: map
Site Management	NewSite(string Name, string FullPathOfParent, bool IsLeafSite)	NewSite("Netbrain-MultiSource", "My Network\\Traditional-DC", True) Return: site
	Site(string Name, string FullPathOfParent)	Site("Cisco-WLC-Lab", "My Network\\Traditional-DC") Return: site
New Device Group Management	NewDeviceGroup(string Name, string Folder)	NewDeviceGroup("BGPDevices", "BGP") Return: devicegroup
	DeviceGroup(string FullPath)	DeviceGroup("Shared Device Groups/BGP/BGP Devices") Return: devicegroup
	NewFile(string FileName, string Folder, string Type="Text")	NewFile("OpenTopo", "Public", "Text")

Category	Sample Functions	Example Return Value (Depends on input value)
File Management		Return: file
	File(string FullPath, string Type="Text")	File("Public/filename”) Return: file
Time Utilities	GetTimeString(number offset, number timeunit)	GetTimeString(1, 1) Return: "08/09/2023 11:34:00"
Variable Operations	number Delta(string variable_name)	Delta($input_rate) Return: Current Value – Last Value

2 How to use the Functions

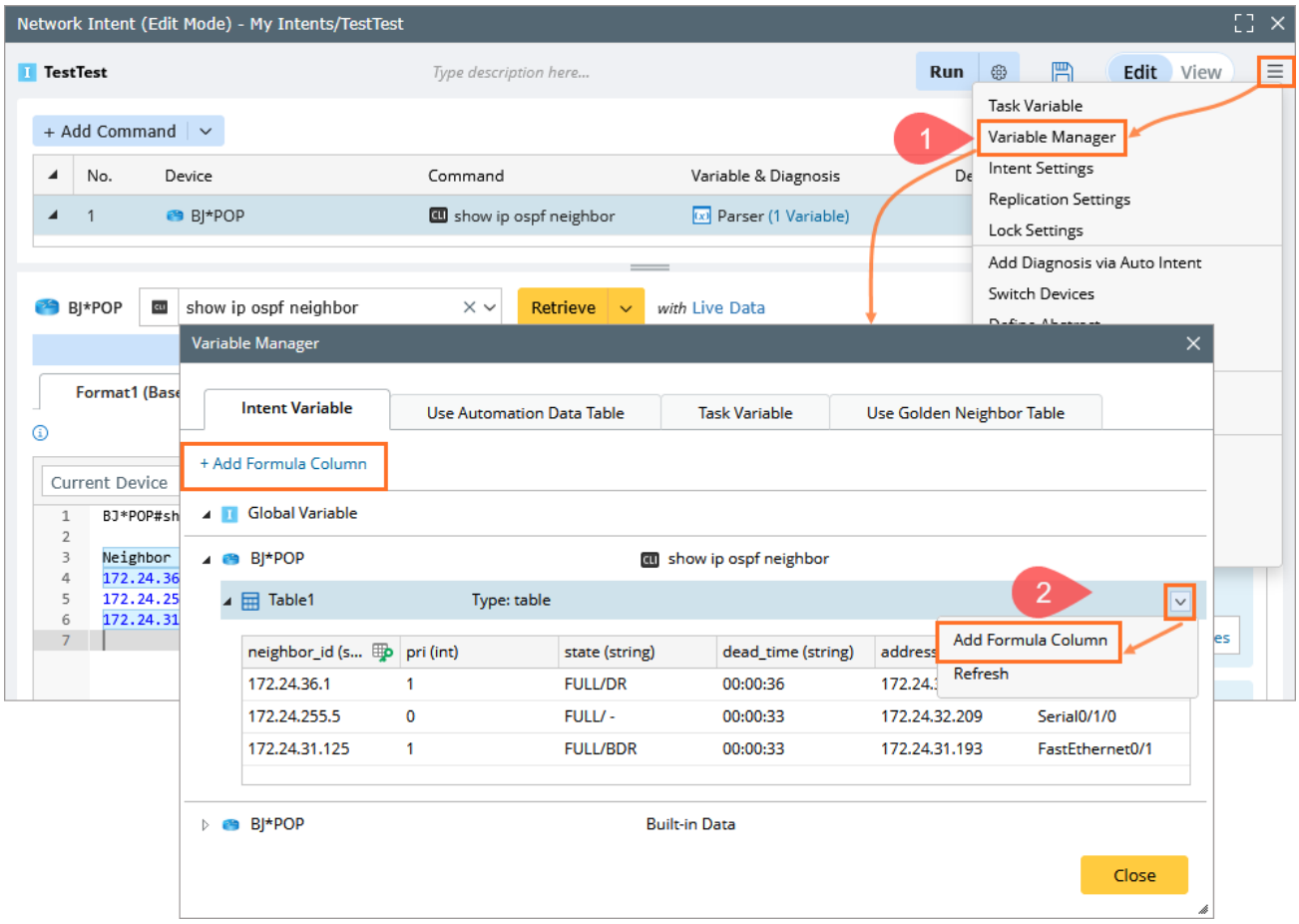
Follow the steps to access the Function in NI edit mode. The Formula columns will be added for the predefined variables that are defined in the Parser.

Note: You can add Functions before defining diagnosis, or during defining diagnosis.

1. In the Network Intent (Edit Mode), navigate to the ☰ menu and select **Variable Manager**.

   Locate the desired variable under the **Device** section.

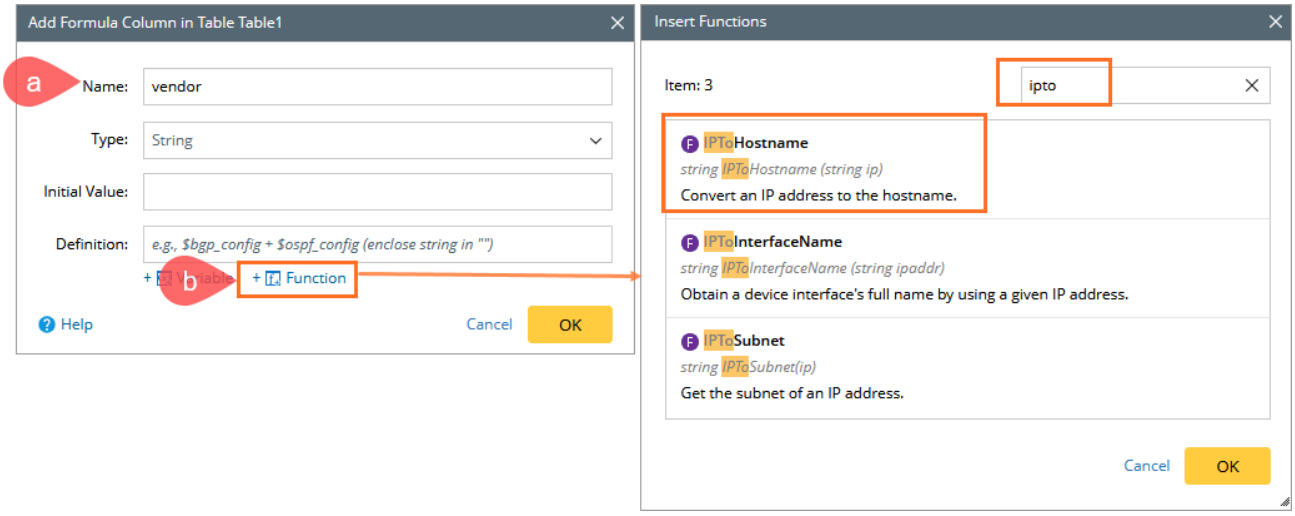
2. Click the arrow next to the variable to open the **Add Formula Column** dialog. Multiple entry options are available for adding a formula column; select any one based on preference.



3. Add Function.

   a) Enter the **Name** and specify the Data **Type** based on the use case.

   b) Click **+Function** and select the relevant function from the dropdown menu.
   Use the search feature by typing the initial letters of the function to locate it quickly.



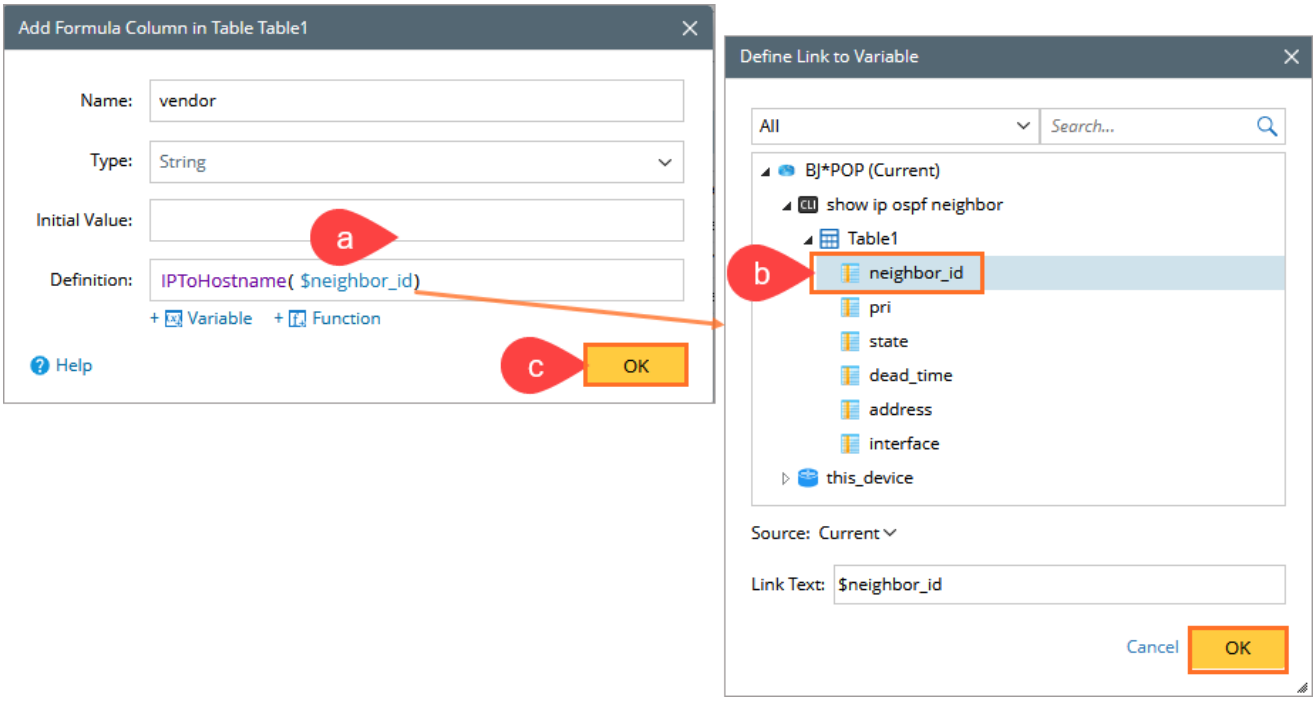
4. Add Varibale to Formula.

   a) Insert a bracket () and enter the dollar symbol (**$**) to display all parsed variables, or click **+Variable** to view the list of available variables.

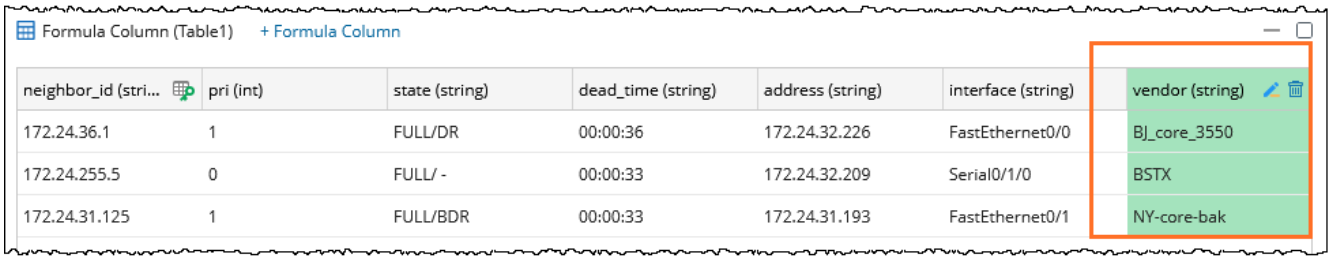
   b) Select the required variables.

   You can choose thr multiple variables, if needed; separate them with commas (,).

   c) Click **OK** to add the formula column to the table.



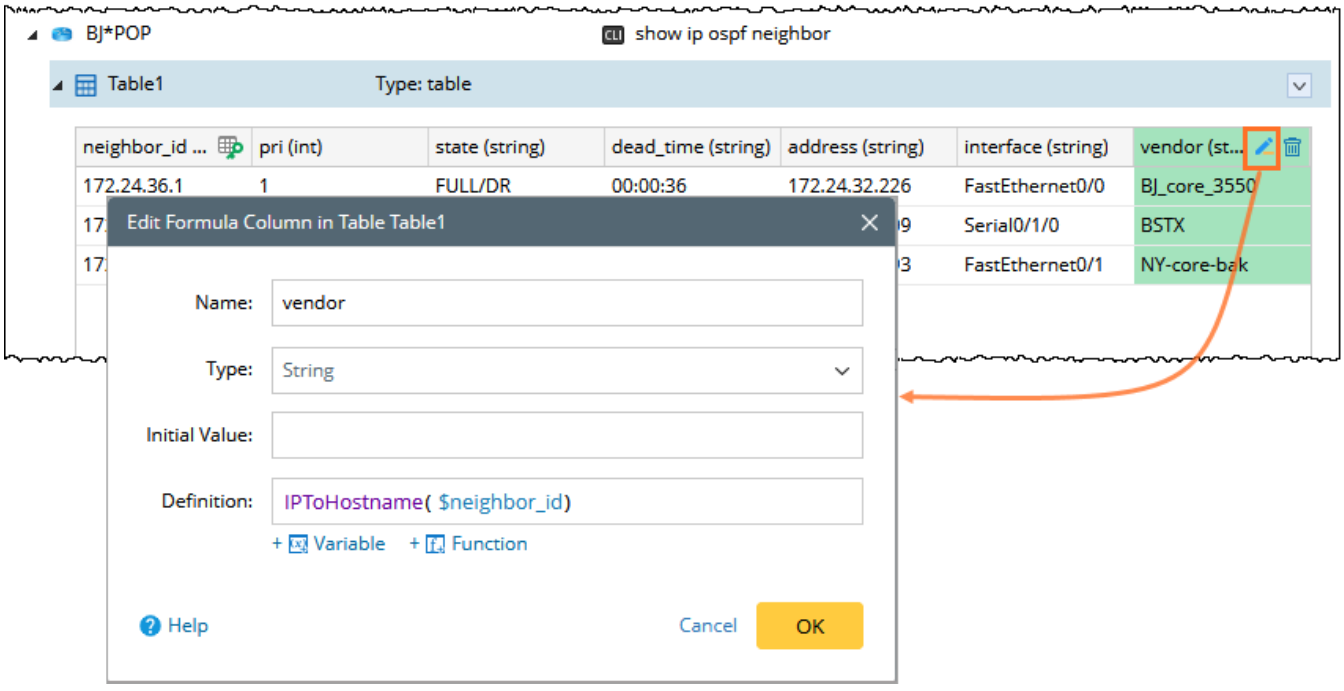
5. Check the return value in the added formula column.



2.1 Edit Formula Column

In the Formula column, locate the pen icon next to the cell or field you wish to modify.

1. Click the pen icon to open the formula editor.
2. Make the necessary changes to the formula as required, and then click OK.



3 NI Built-In Functions - Use Cases

To enhance user experience and streamline navigation, the functions are thoughtfully organized into distinct categories. This structure ensures that users can quickly locate and access the features they need with clarity and ease.

3.1 IP Conversion

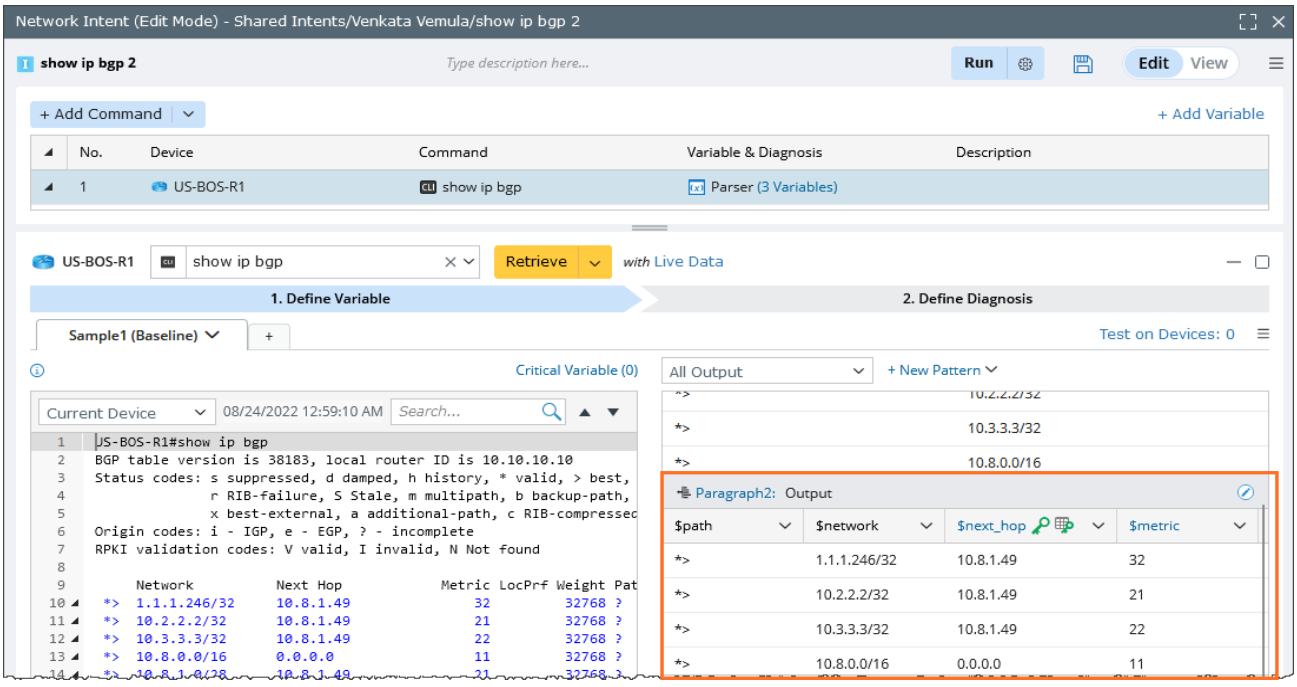
3.1.1 IPStringToNumber

This function is used to convert an IP address from dotted decimal notation to the decimal format stored in our database. It also supports masked IP addresses, such as “192.168.91.1/28.”

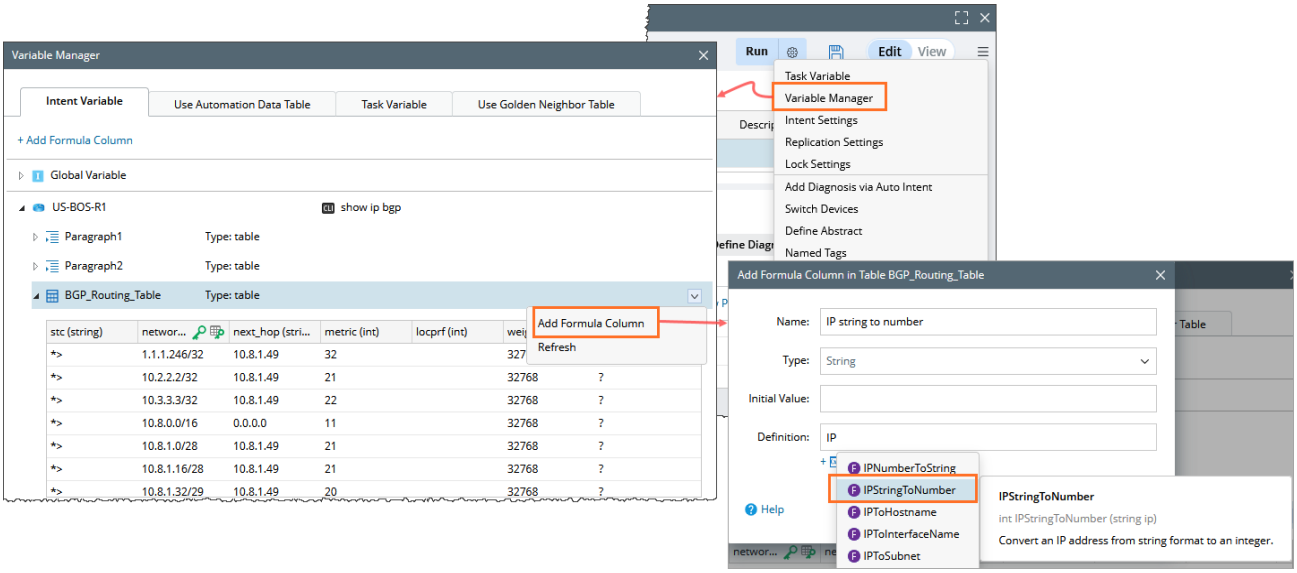
For example, when working with a parsed table that includes the next hop IP address, you can utilize the ***IPStringToNumber($next_hop)*** function to convert the next hop IP address from its string format to a numerical format.

Follow the steps to add the Function to the existing variable table.

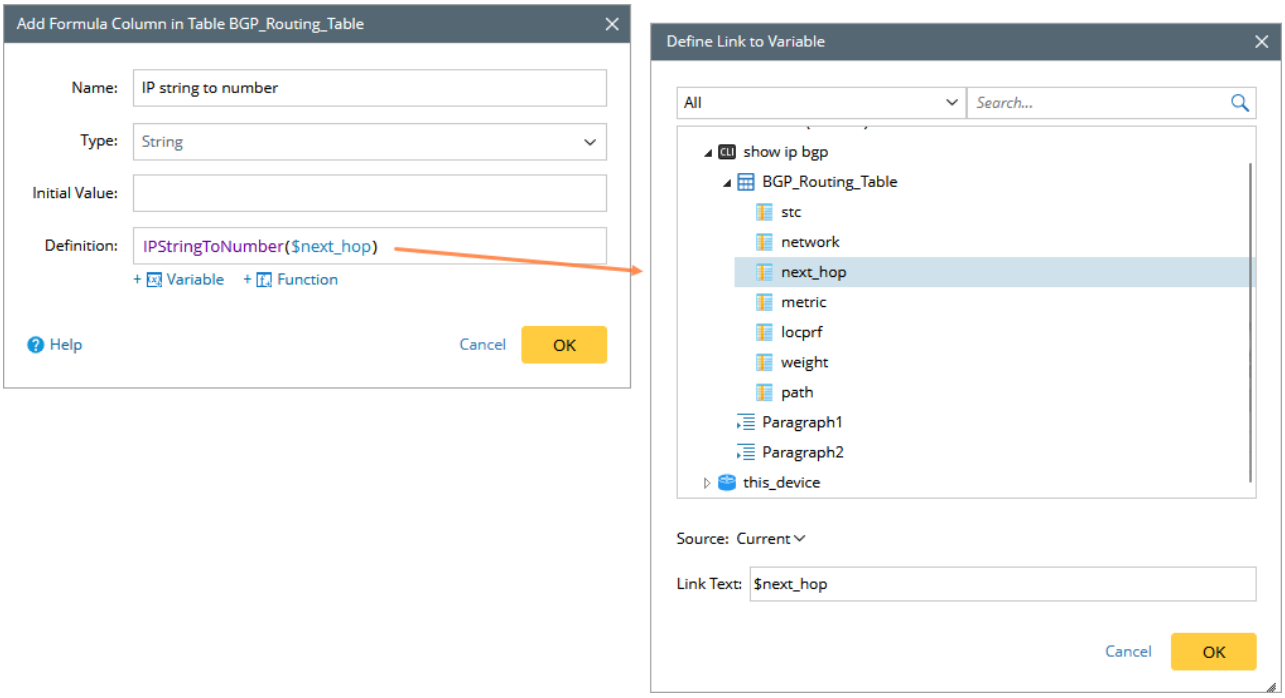
1. Create an Intent and Parse the table Variable.



2. Open **Variable Manager** and add the **Formula Column**.



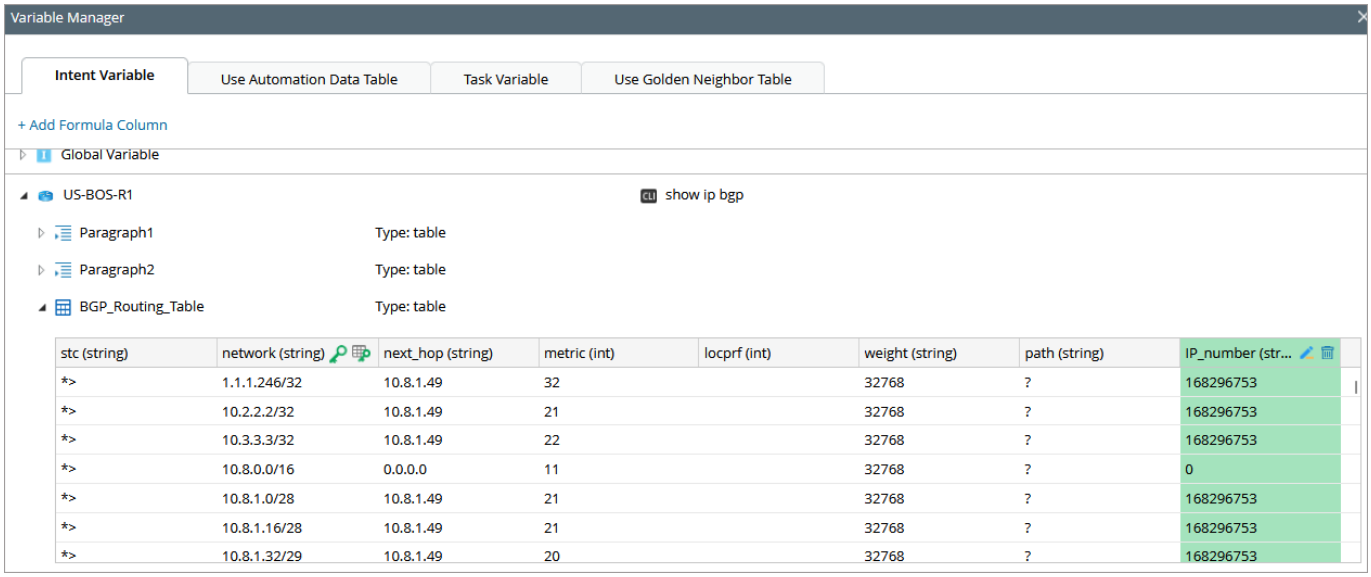
3. Add variable using **$** symbol i.e., ***next_hope***.



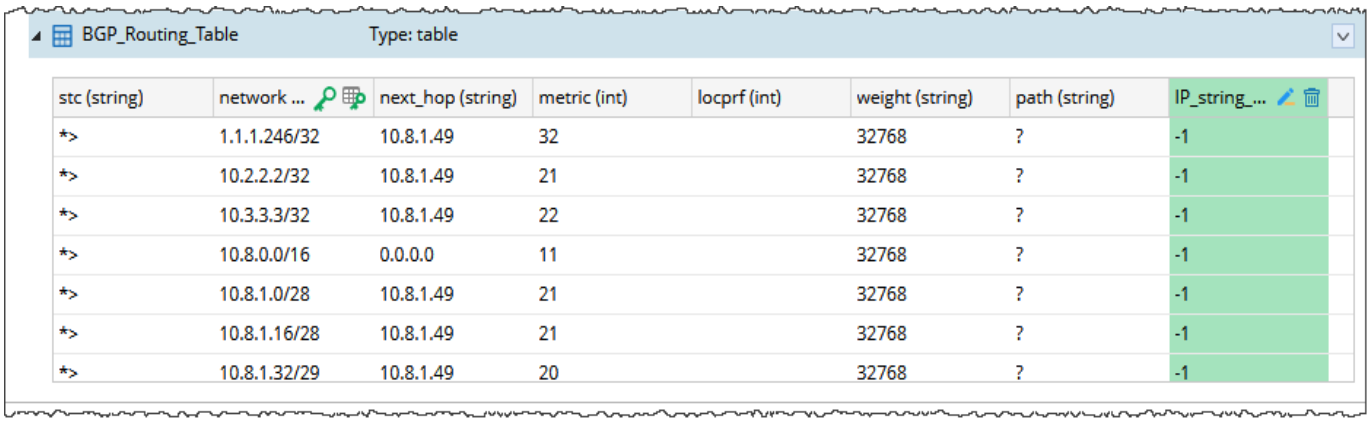
Return Value:

The return value of the function will be displayed in the newly added column.

You will notice that the added formula column displays the numbers format i.e., 168296753.



Return “-1” if the input is not a valid IP address.



In the next section, you will build upon the previous example to convert an IP number back to an IP string. Specifically, you will convert an IP number to a string representing the hub IP.

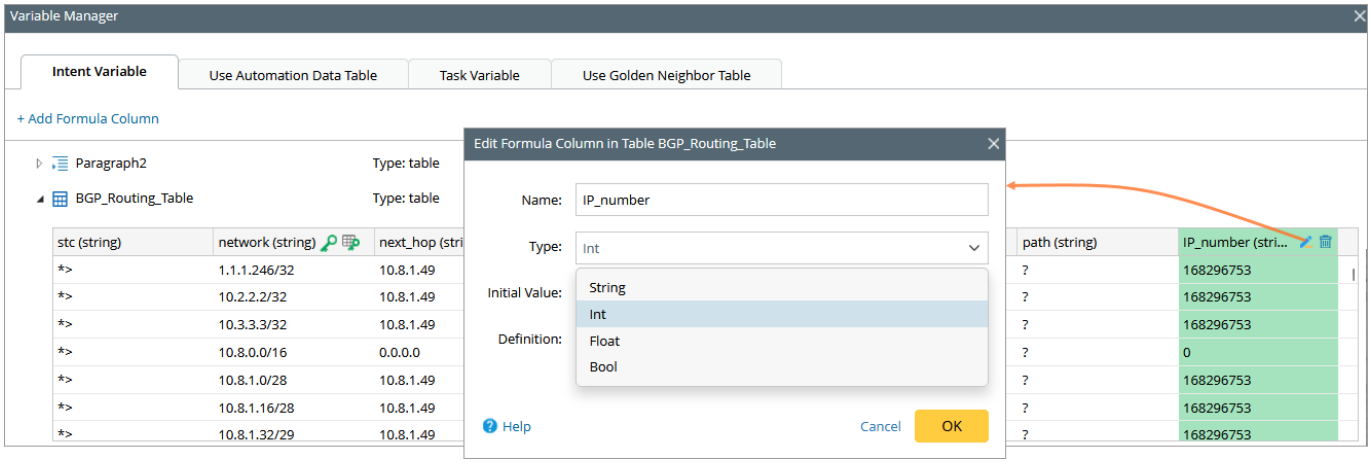
3.1.2 IPNumberToString

Previously, the IP string was converted into a decimal. Now, to retrieve the hub IP, you need to use the formula: ***IPNumberToString($IP_number - 1).***

Conversion Steps:

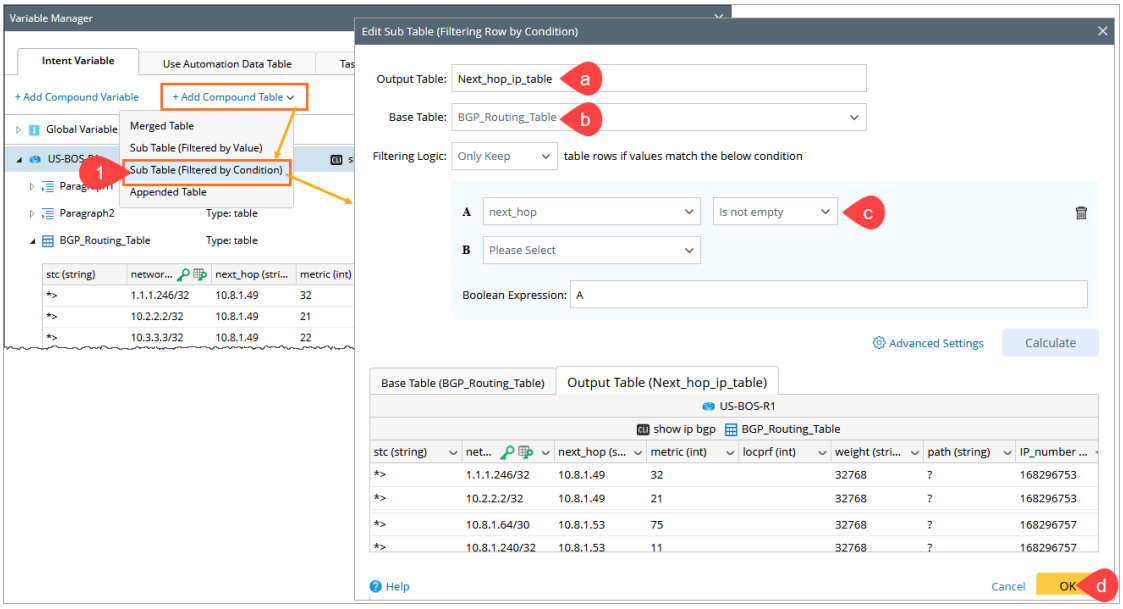
- When converting a **Spoke IP** to a **Hub IP**, subtract **1** from the last digit of the numerical IP.
- Add the **sub table** (**Filtered by Condition**) and reference the previously converted variable (ip_number) from the table.

Note: To use this function explicitly, change the data type from **string** to **integer** before performing the conversion.

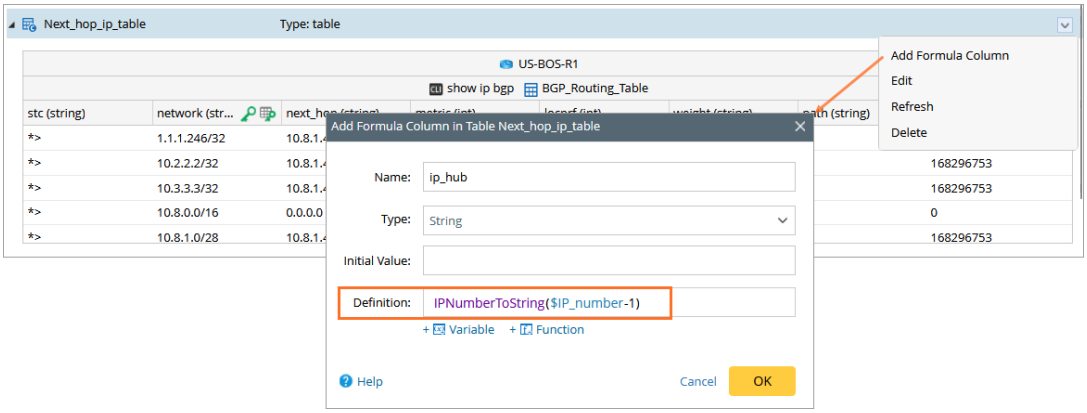


Follow these steps to get the Hub IP.

1. Click **+ Add Compound Table** and select **Sub Table (Filtered by Condition)**.
   a) Enter the **Output Table** name.
   b) Select the **Base Table** from the dropdown (e.g., ***BGP_Routing_Table***).
   c) Set Condition A: **next_hop** is not empty.
   d) Click **OK** to add the sub-table.



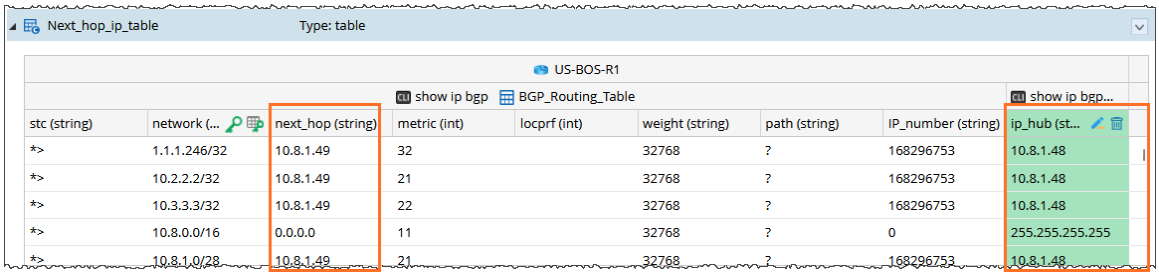
2. In the added Sub Compund table, click **Add Formula Column** and enter the definition ***IPNumberToString($IP_number-1)*** *to get the hup IP.*



Return Value:

The number will be converted into the hub's IP address. Note that the resulting IP address is decremented by one. For example, ***10.8.1.49*** becomes ***10.8.1.48.***

Return null (“”) represents error.

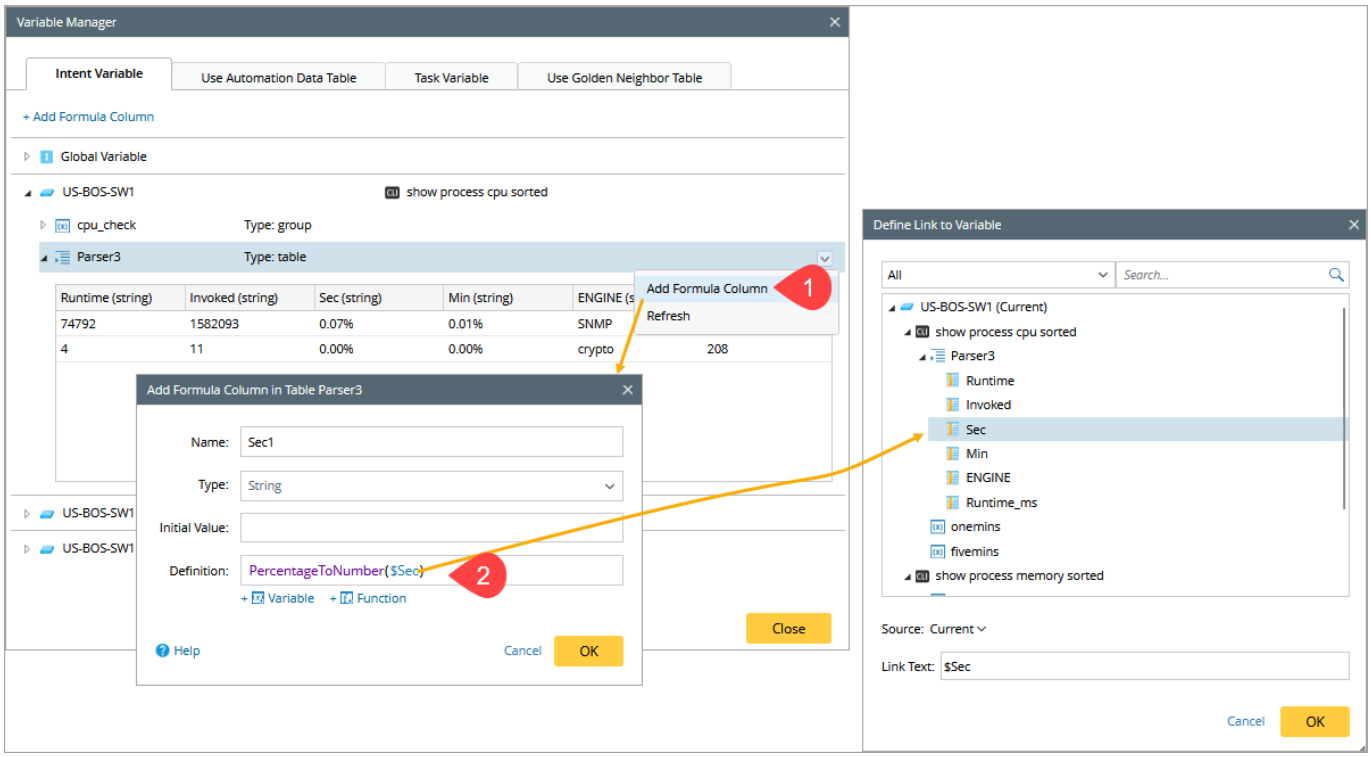


3.2 Numeric Conversion

3.2.1 PercentageToNumber

This function converts a percentage represented as text into a floating-point number. For example, when monitoring CPU memory utilization, you can convert the percentage of time into a plain number using the formula ***PercentageToNumber(string percent).***

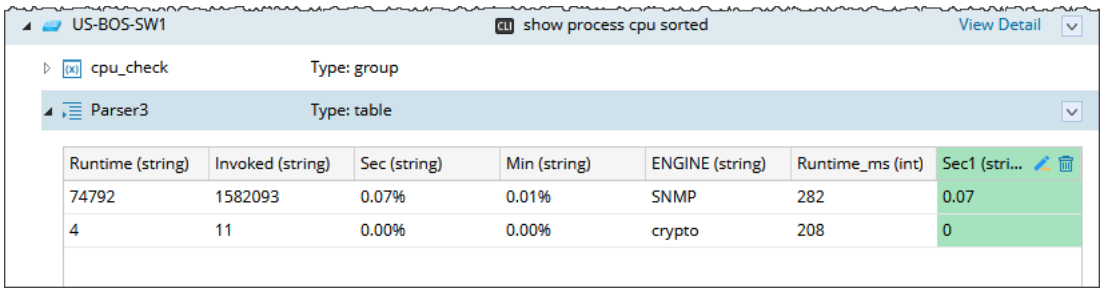
Refer to the image to convert percentages into numerical values.



Return Value:

The return value of the function will be displayed in the newly added column.

Return “-1” if the input is not a valid percentage.

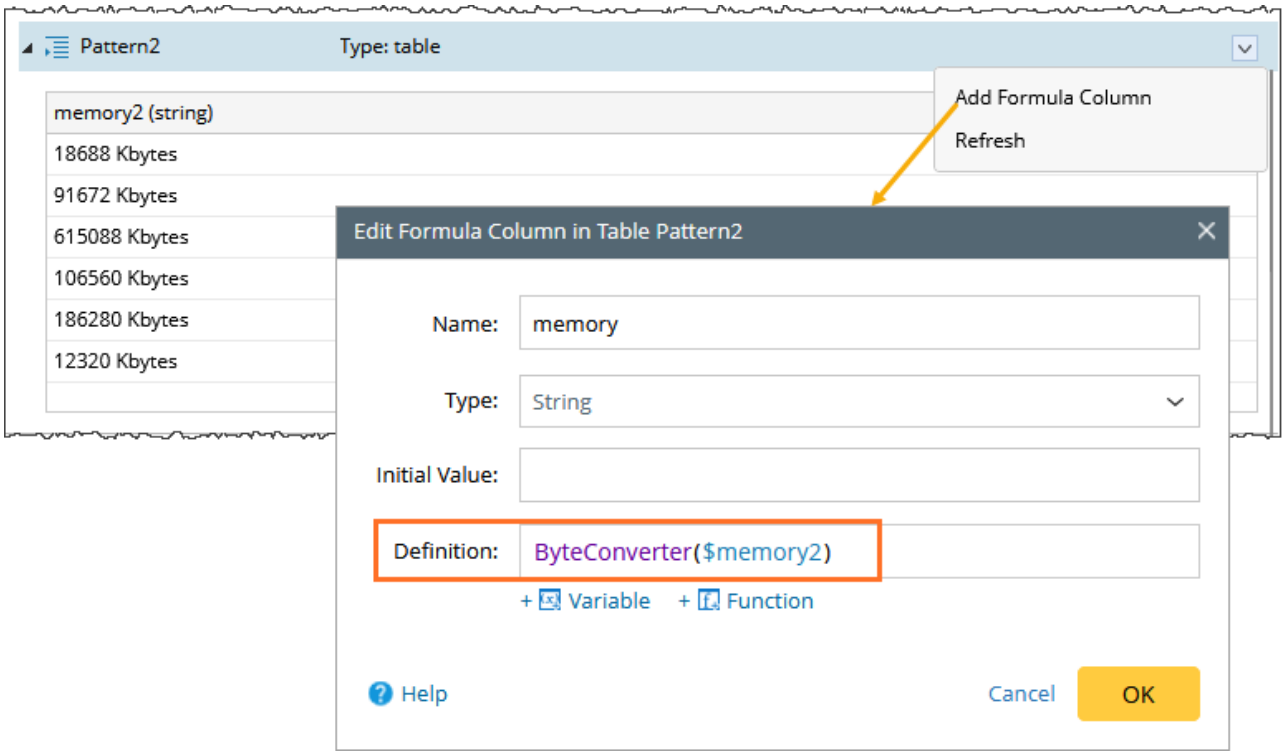


3.2.2 ByteConverter

Convert storage units (such as KB, MB, GB, etc.) to byte.

For example, given is the system memory utilization in the kelobytes. You will use the ***ByteConverter(string input)*** formula to convert this KB to bytes.

Refer to the image to convert KB into the Bytes.

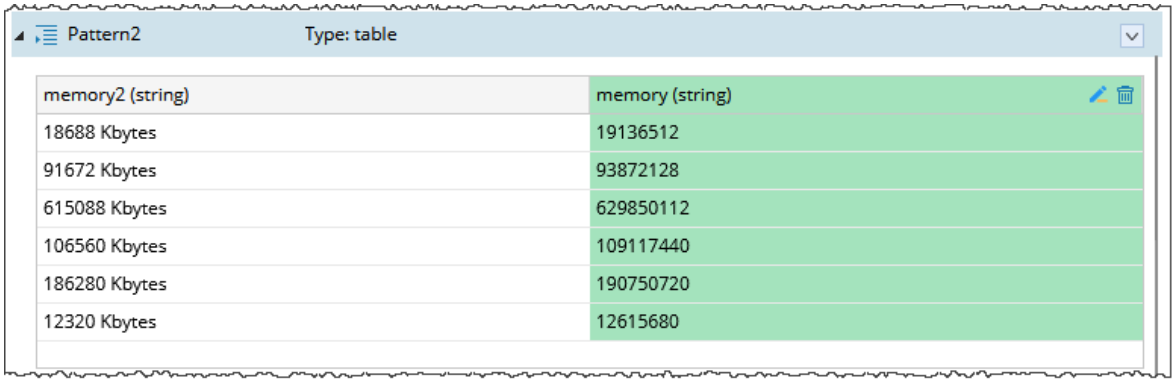


Return Value:

The return value of the function will be displayed in the newly added column.

Note:

- Return "-1" if the input is not a valid storage unit.
- The input parameter is case-insensitive, and abbreviations are allowed, e.g., GB, G, g.



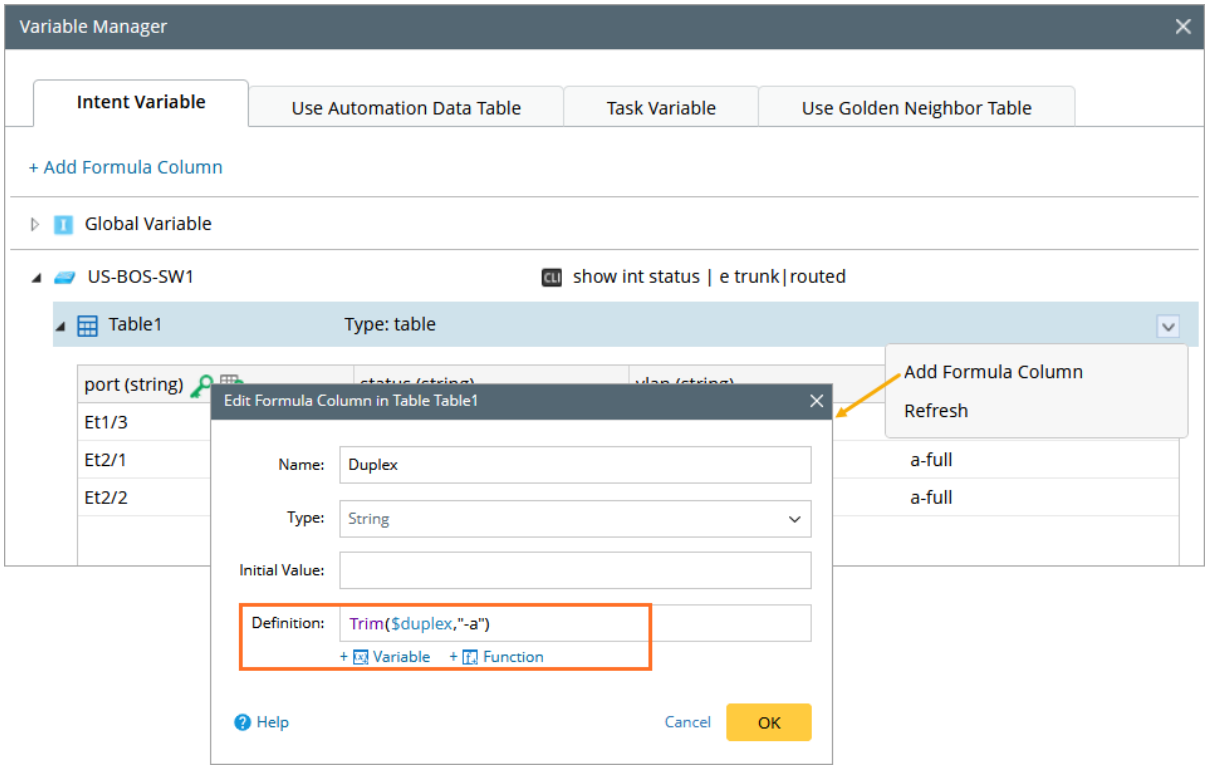
3.3 String Transformation

3.3.1 Trim

This function trims one or more characters from the right or left side of a given string and returns the resulting trimmed string. If no specific characters are provided, the function defaults to trimming whitespace.

To trim characters from the right side, use the ***TrimRight(string original, string trimChars)*** method. To trim characters from the left side, use the ***TrimLeft(string original, string trimChars)*** method. Alternatively, you can use the ***Trim(string original, string trimChars)*** function to simplify the process.

For example, you can trim specific characters from the "auto duplex" column (a-duplex) using the formula ***TrimLeft($duplex, "a-")*** to remove characters from the left side. Similarly, you can trim characters from the right side using the formula ***TrimRight($duplex, "full").***

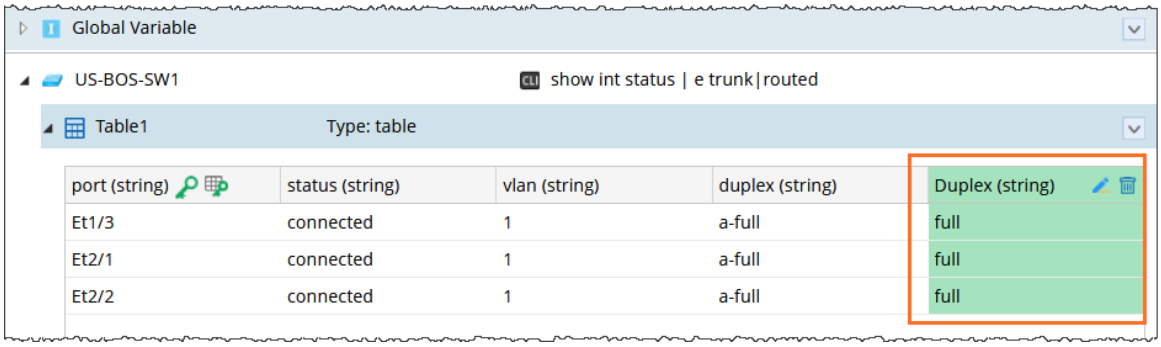


Similarly, you can trim characters from the right side using ***Trim($duples,"full").***

Return Value

The return value of the function will be displayed in the newly added column.

The added formula column displays the trim character. If the trimming fails, the column returns the original value.



3.3.2 Replace

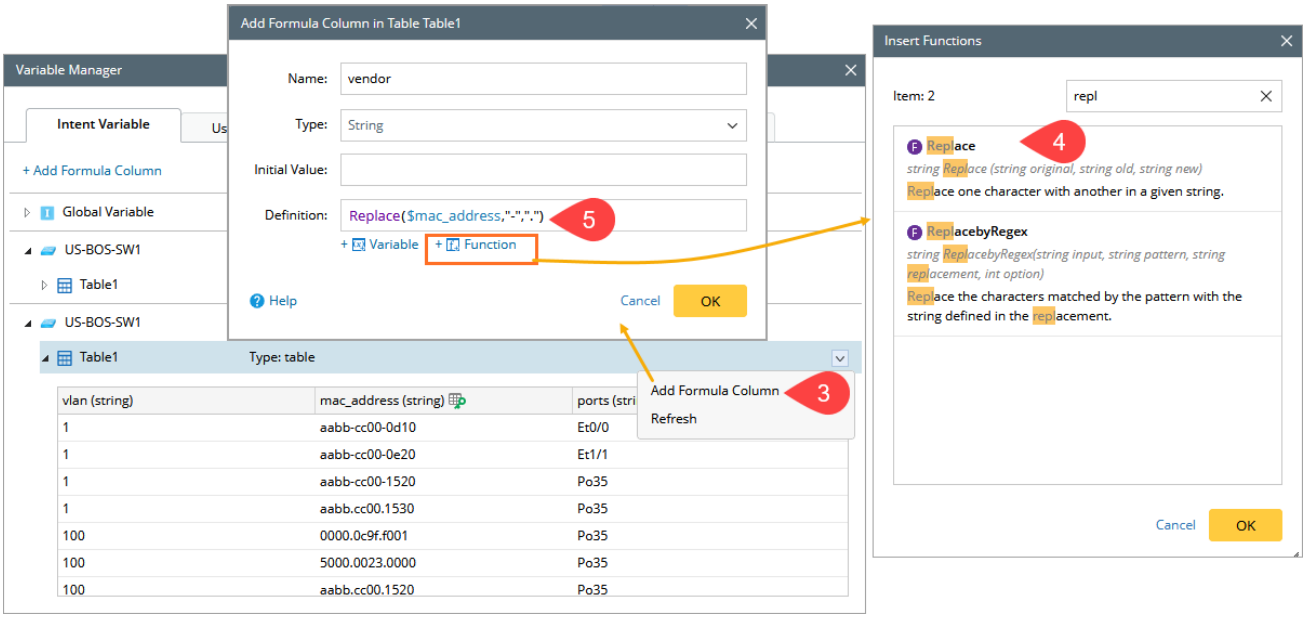
This function allows you to replace one character with another in a given string.

For example, if you have parsed a table containing MAC addresses and need to replace old string hyphens (-) with the new string dots (.), you can achieve this by adding a formula column.

Use the ***Replace($mac_address, "-", ".")*** function to perform the replacement.

Follow the steps to add the Function to the existing variable table.

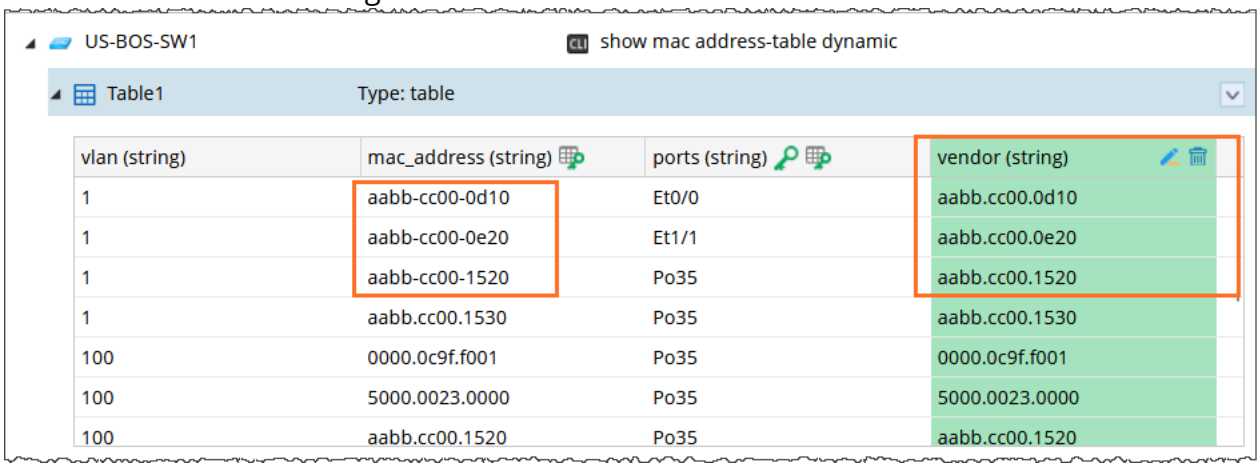
1. Create an Intent and Parse the MAC_Address table as variable.
2. Open Variable Manager and add the Formula Column.
3. Click the arrow to open the **Add Formula Column** dialog.
4. Click **+Function** and select the ***Replace*** function from the dropdown menu
5. Add the variables using the $ symbol (e.g., ***$mac_address***), and insert the formula to define the function: (***$mac_address, "-", "."***).



Return Value

The return value of the function will be displayed in the newly added column.

The added formula column displays the replacement character. If the replacement operation fails, the column returns the original MAC address.

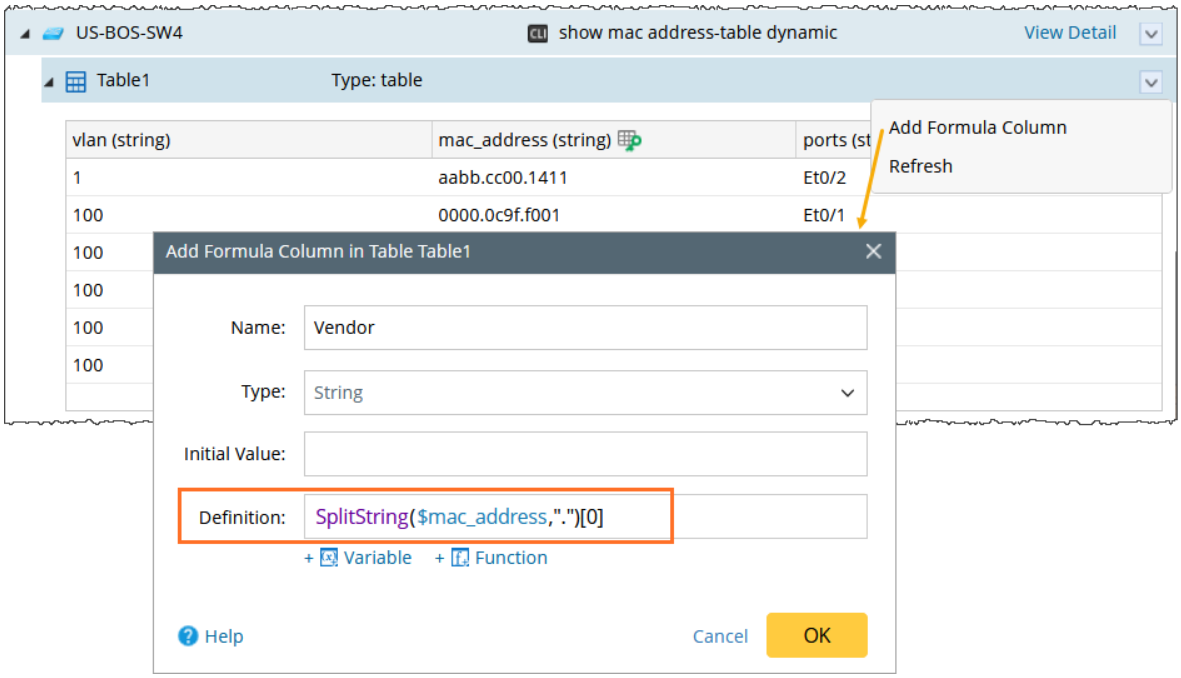


3.3.3 SplitString

This function splits the input into multiple strings based on one or more given separator characters.

For example, a given MAC address can be split into three parts using ***SplitString($mac_address, ".")[0]***. Using **0** extracts the first four strings of the MAC address. Similarly, replacing **0** with **1** or **2** will extract the middle four strings and the last four strings, respectively.

Refer to the image to split strings of the MAC address.



Return Value:

The return value of the function will be displayed in the newly added column.

According to the function definition, the MAC address will be divided into sections.

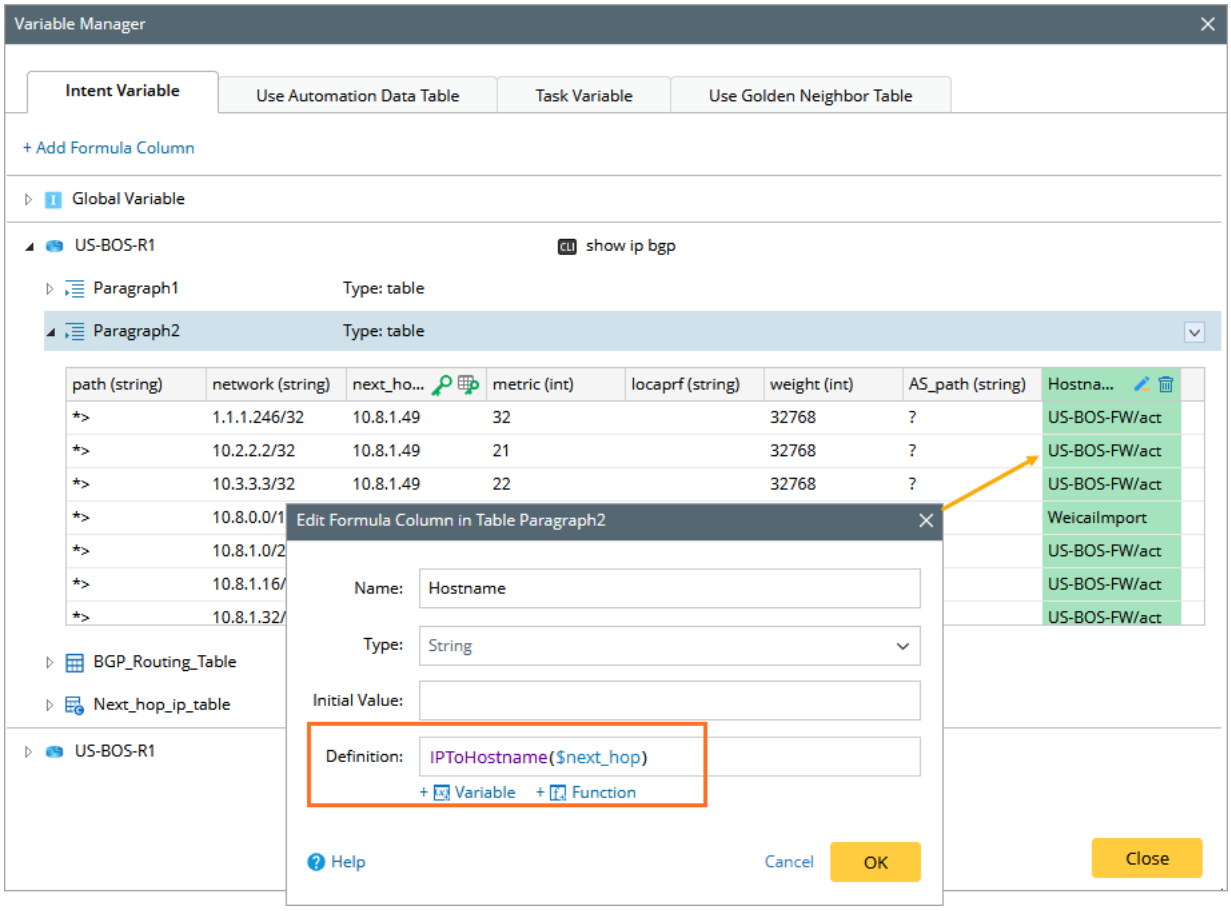
vlan (string)	mac_address (string)	ports (string)	Vendor (string)
1	aabb.cc00.1411	Et0/2	aabb
100	0000.0c9f.f001	Et0/1	0000
100	0000.0c9f.f002	Et0/1	0000
100	aabb.cc00.1500	Et0/1	aabb
100	aabb.cc80.1400	Et0/1	aabb
100	aabb.cc80.1500	Et0/1	aabb

3.4 Network Utilities

3.4.1 IPtoHostname

This function converts an IP address to the hostname. The input can be an IP address or a masked IP address such as “192.168.91.1/28”.

Refer to the image to get the hostname of an IP address.



Return Value:

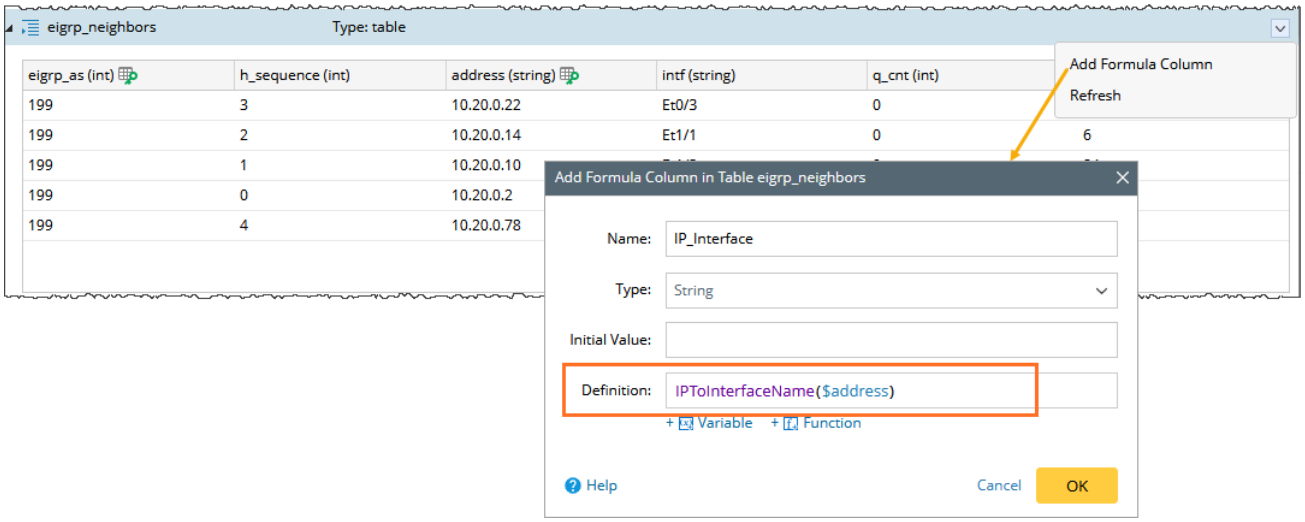
The return value of the function will be displayed in the newly added column

Return NULL("") if the input is not a valid IP address. If an IP address is configured in more than one device, return the hostname of one device randomly.

3.4.2 IPToInterfaceName

This function obtains a device interface full name by using a given IP address. Masked IP address is allowed, e.g., “192.168.91.1/24”.

Refer to the image to convert an IP to interface name.



Return Value:

The return value of the function will be displayed in the newly added column.

Return NULL(“”) if the input is not a valid IP address. If an IP address is configured in more than one interface, return the full name of one interface randomly.

3.4.3 InterfaceShortToFull

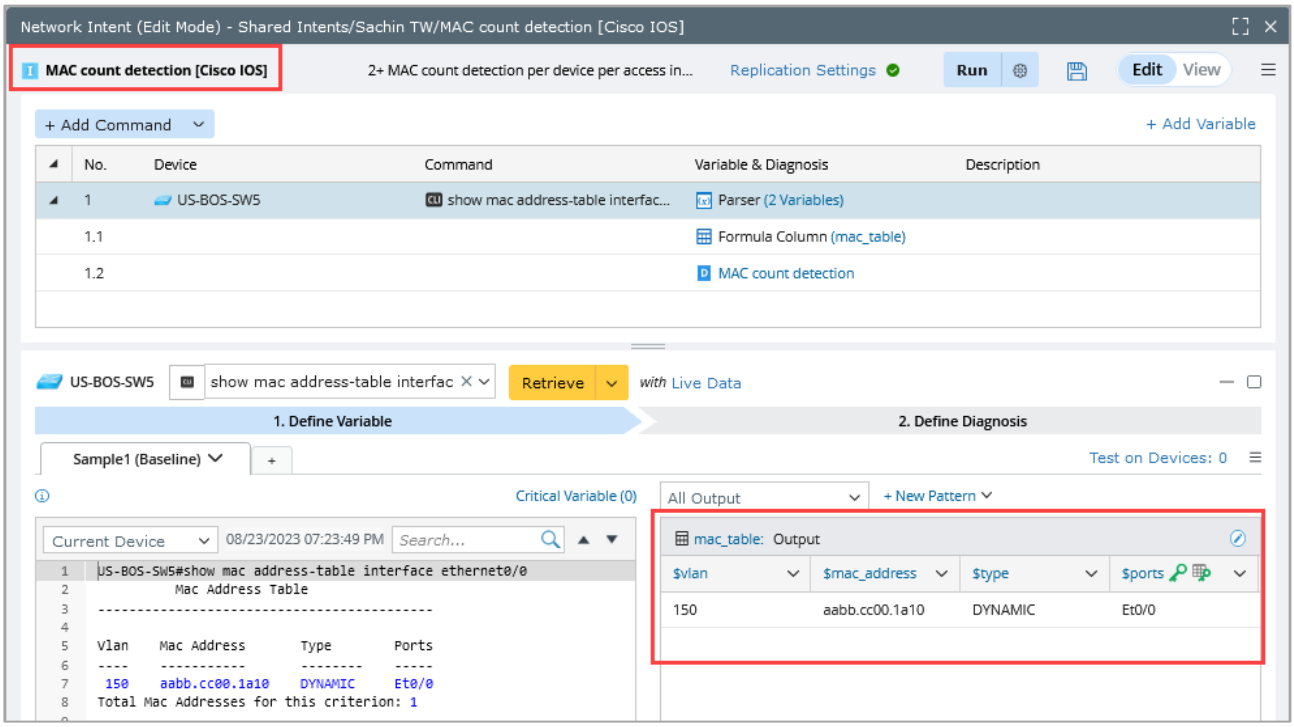
This function converts an interface name from its short form to its full form.

For example, if you have parsed a table containing shortened port or interface names and wish to view the full name, you can use the variable manager to achieve this. By adding a formula column and using the ***InterfaceShortToFull($Hostname, $ports)*** function, you can convert the interface name from its shortened version to its full form.

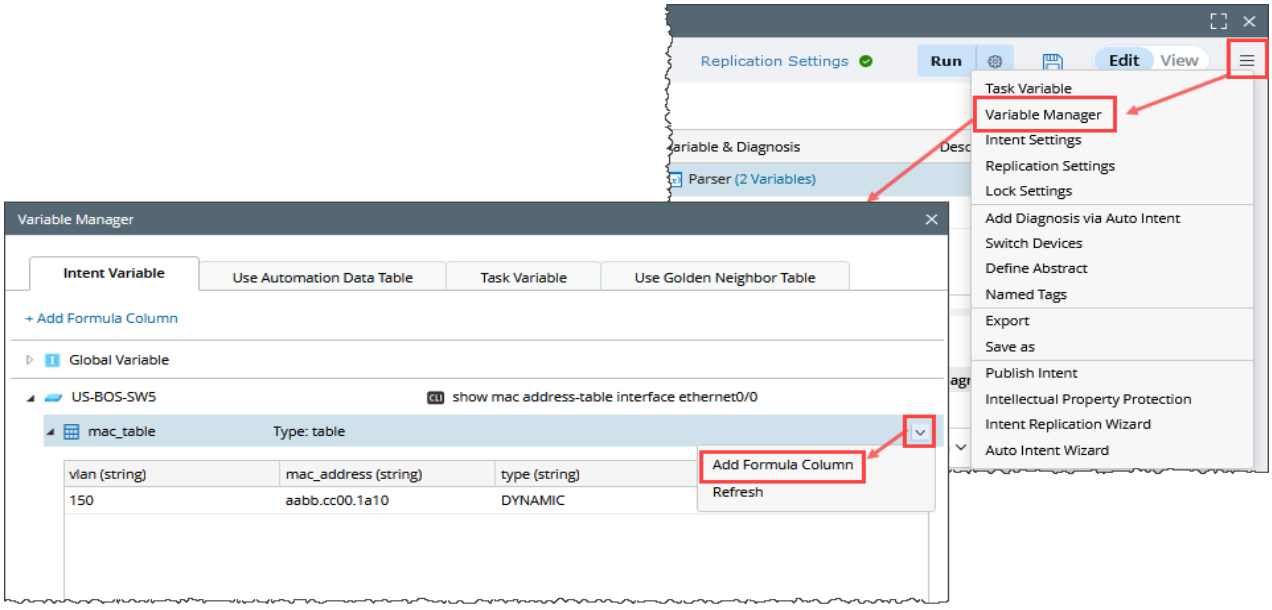
Follow the steps to add the Function to the existing variable table.

1. Create an Intent and Parse the table Variable.

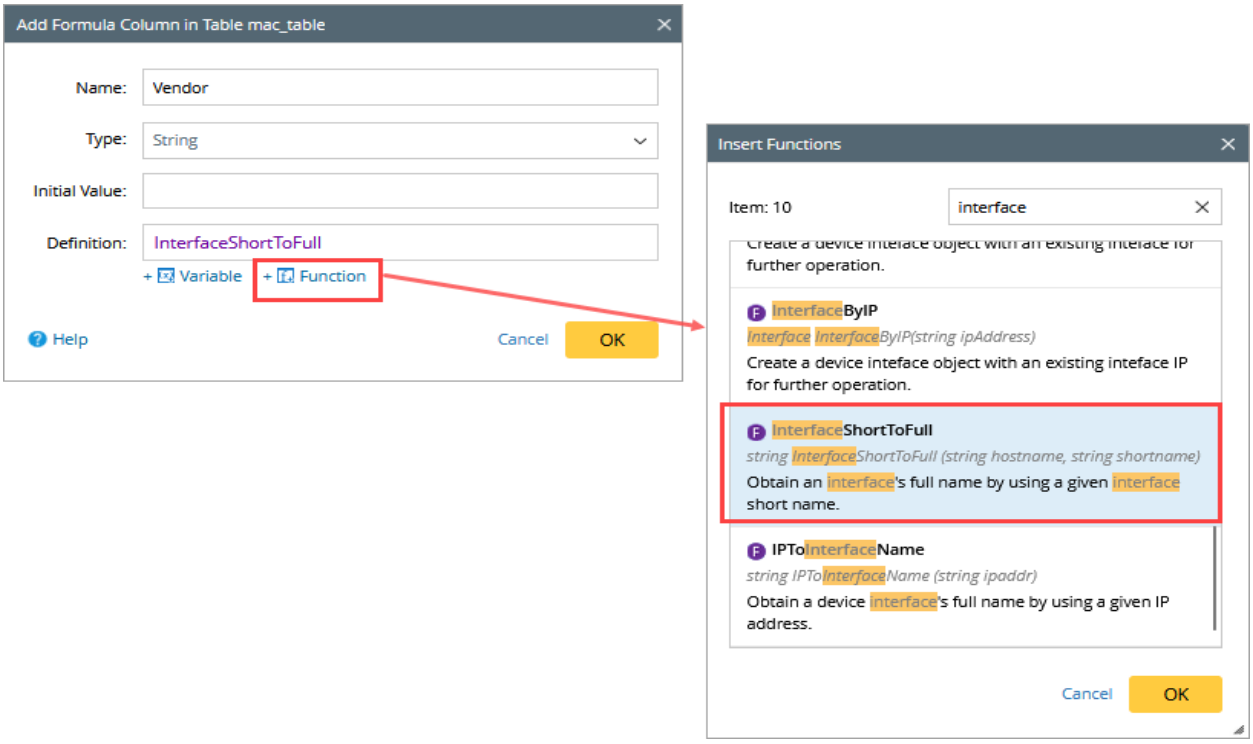
   Create an intent and parse the variable as **mac_table** using the Paragraph Parser. Choose the parsing method based on your use case: Auto Parser, Single Variable Parser, or Multiple Variable Parser.



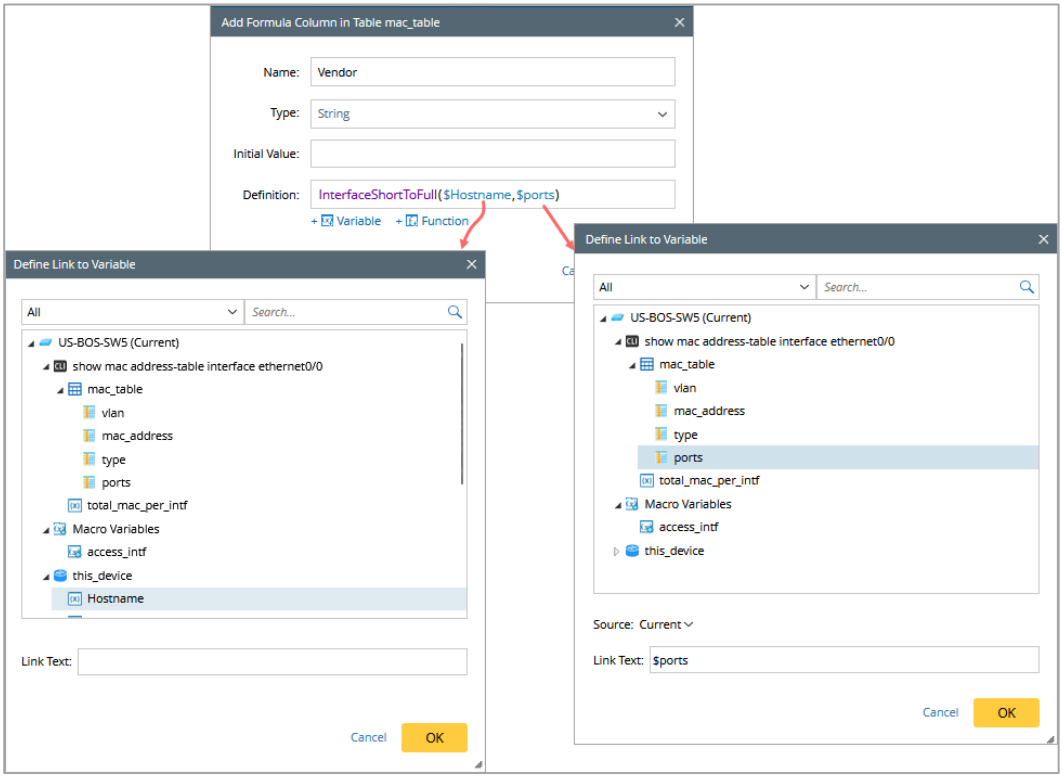
2. Open Variable Manager and add the Formula Column.



3. Click **+Function** and select the relevant function from the dropdown menu.



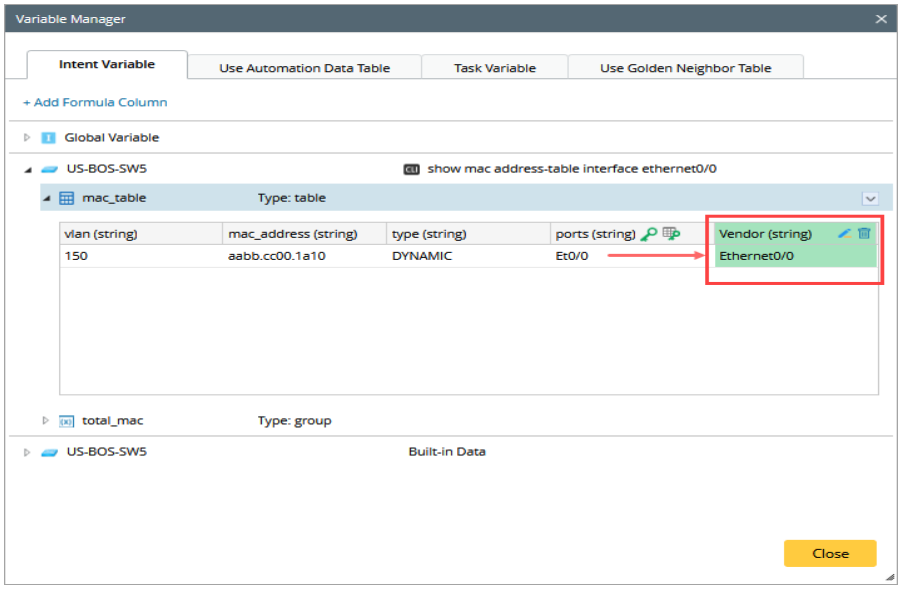
4. Add Variables



Return Value:

The return value of the function will be displayed in the newly added column.

You will notice that the added formula column displays the full port name i.e., ***Ethernet0/0.***

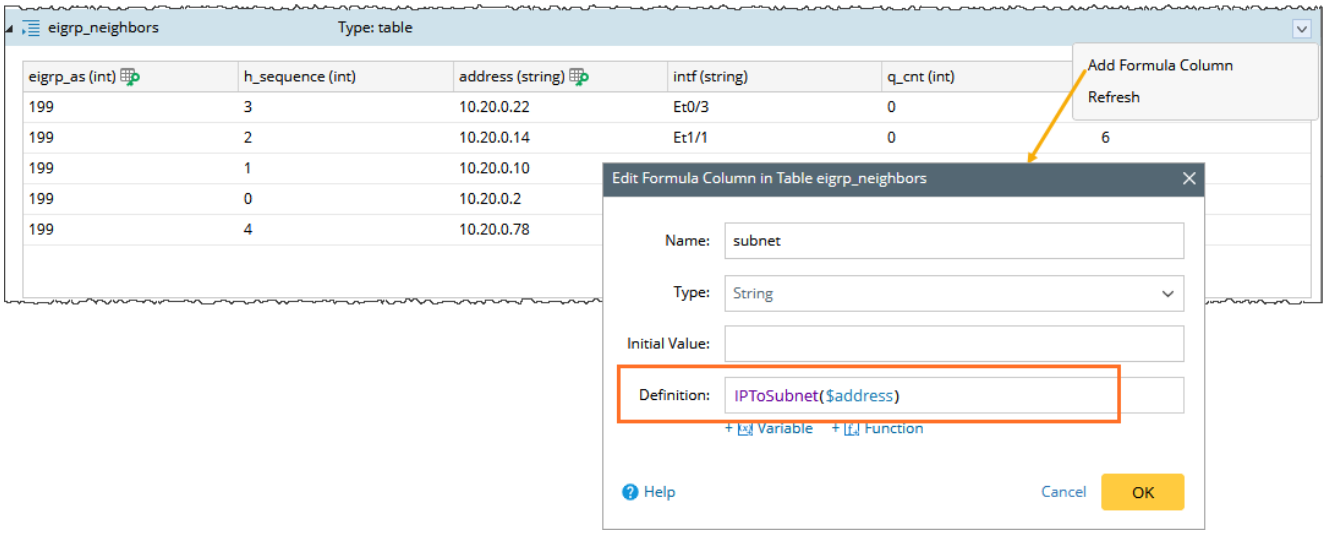


Returning null (“”) if the input is not a valid interface name.

3.4.4 IPToSubnet

This function return the subnet of an IP address. Masked IP address is allowed, e.g., “192.168.91.1/24" , "192.168.91.1 255.255.255.255.0".

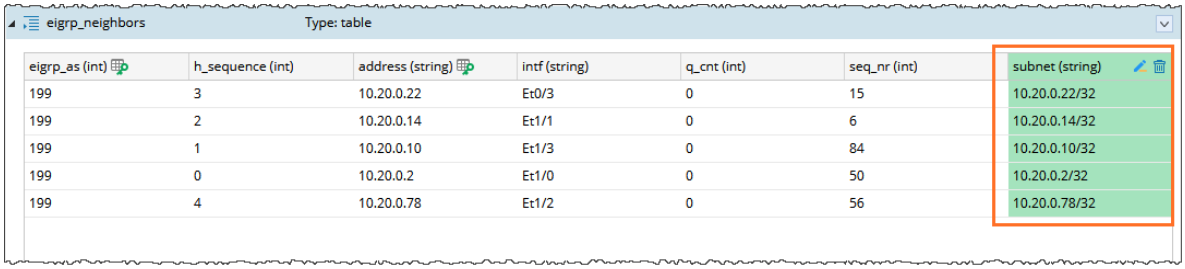
Refer to the image to get the subnet of an IP address.



Return Value:

The return value of the function will be displayed in the newly added column.

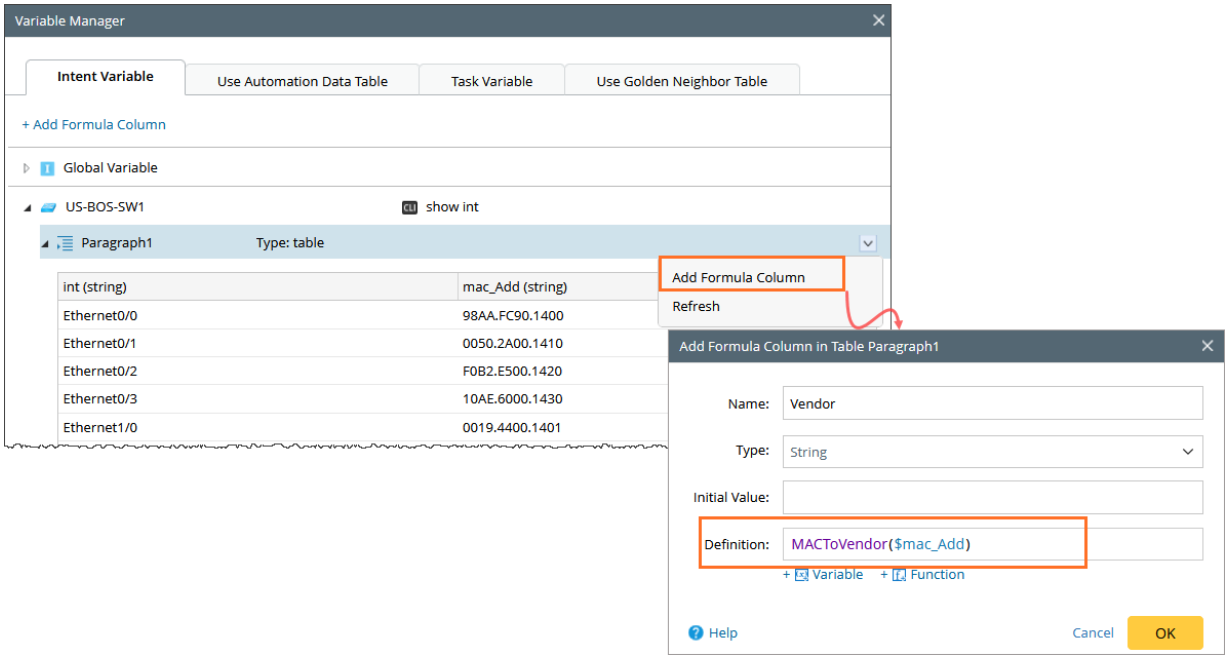
Return NULL("") if the input is not a valid IP address. IP without mask follows class A|B|C boundary.



3.4.5 MACToVendor

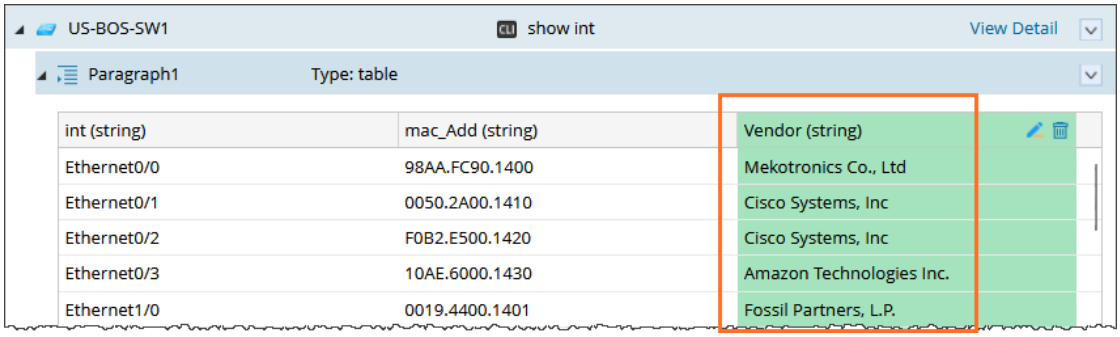
This functions return the vendor information about a device by MAC address.

Refer to the image to get the vendoe information using MAC address.



Return Value:

This function return the Vendor information.

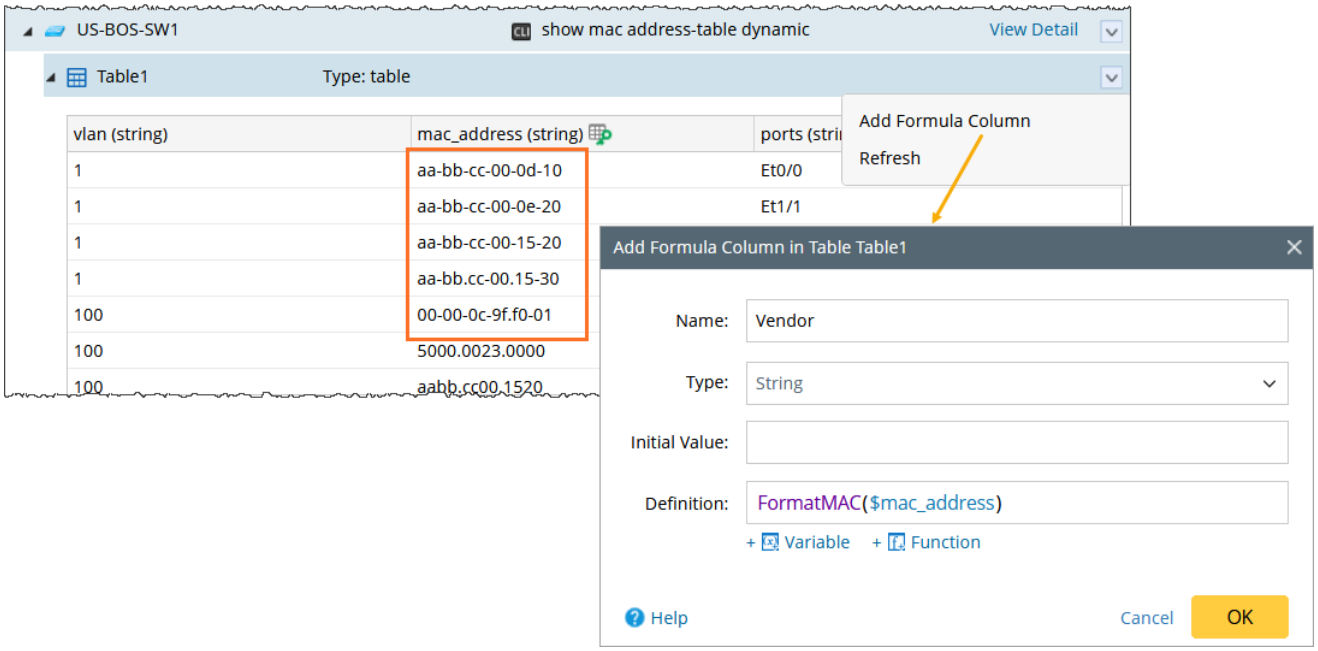


3.4.6 FormatMAC

This function formats MAC address using NetBrain-specific standard (period-separated hexadecimal notation).

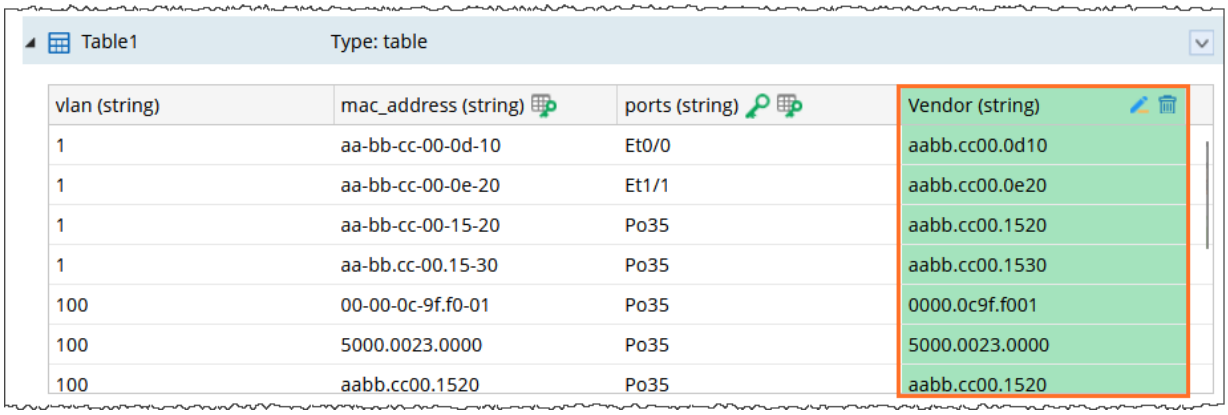
For example, the given MAC address is not as per NetBrain's specific standard. You can use this function to fomat the MAC address as per standard.

Refer to the image to format the Mac Address.



Return Value:

The return value of the function will be displayed in the newly added column. For example: **aabb.cc00.0d10**



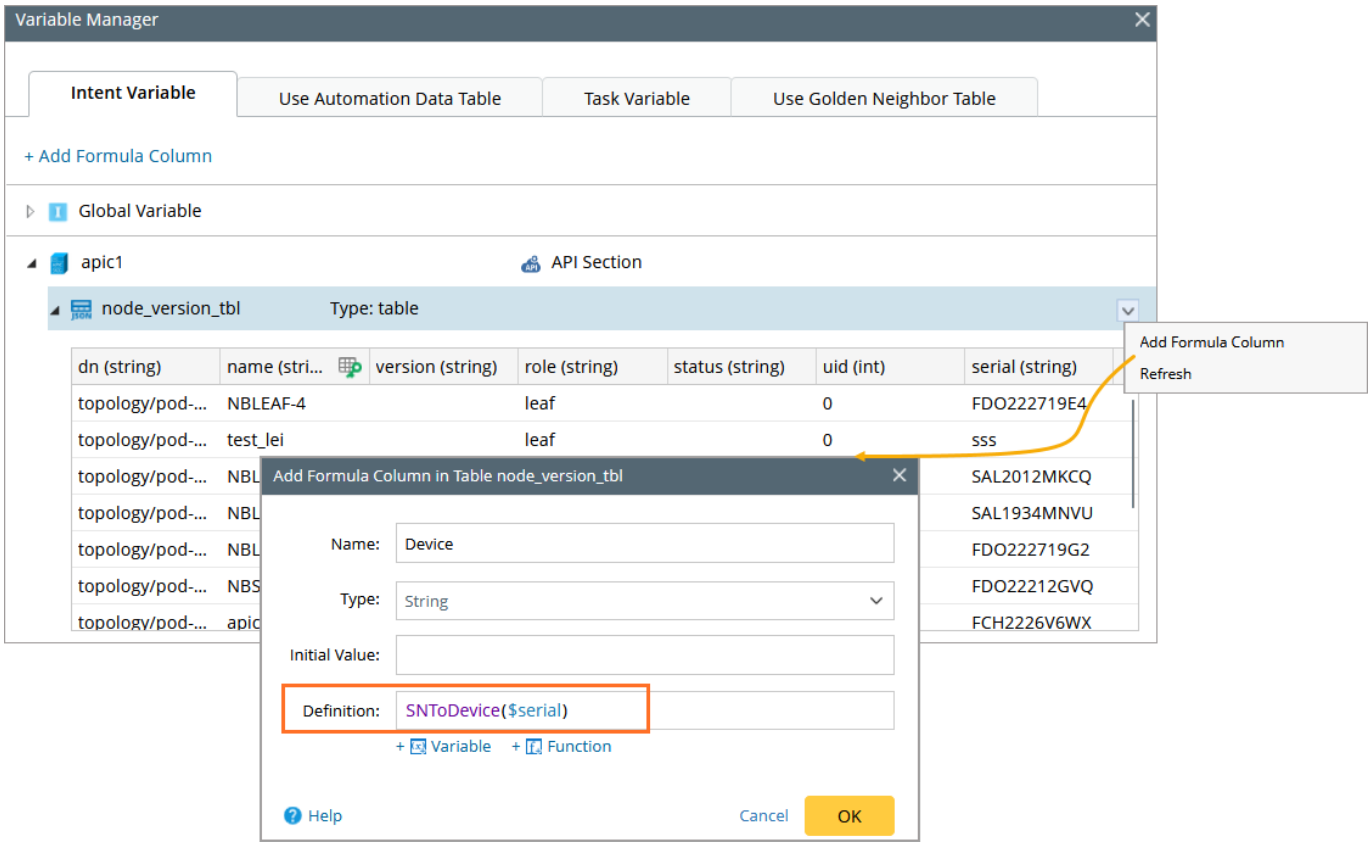
Returns original MAC address if the format fails.

3.4.7 SNToDevice

This function returns a device object based on its serial number.

For example, if you have a serial number and want to retrieve the corresponding device instead of the serial number itself, you can use the ***SNToDevice($serial)*** formula.

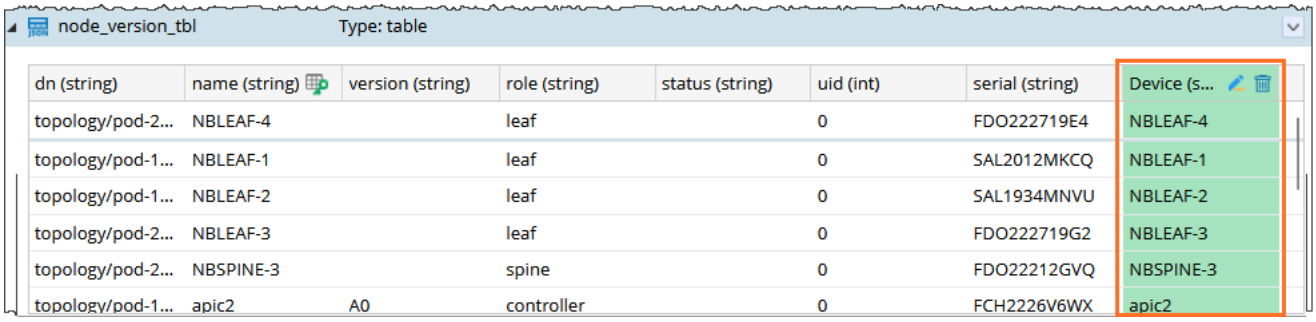
Refer to the image for an example of retrieving a device name using its serial number.



Return Value:

The return value of the function will be displayed in the newly added column.

For example, **NBLEAF-4.** Returns NULL("") if the input is not a valid serial number.



3.5 Mathematical Operations

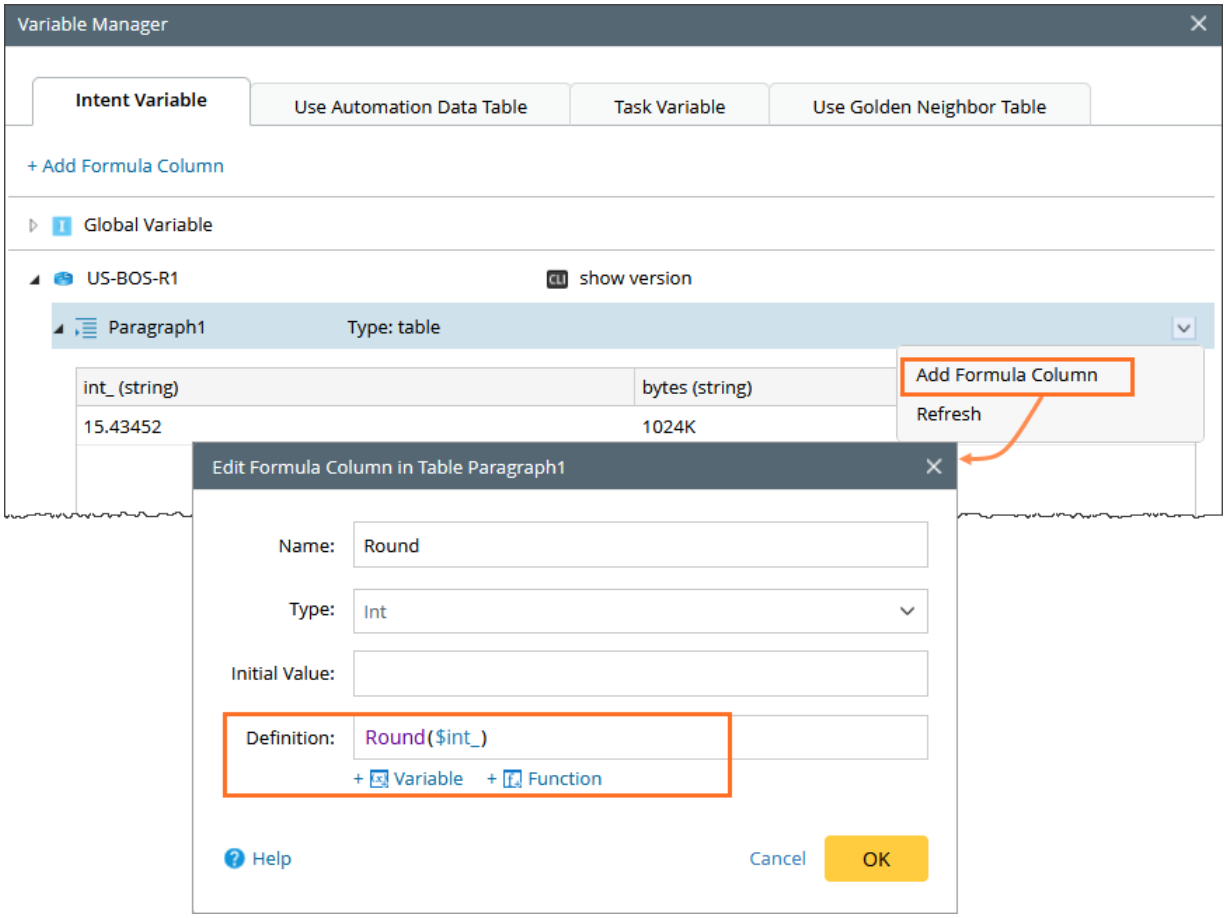
3.5.1 Round

This function returns number rounded to two digits precision after the decimal point.

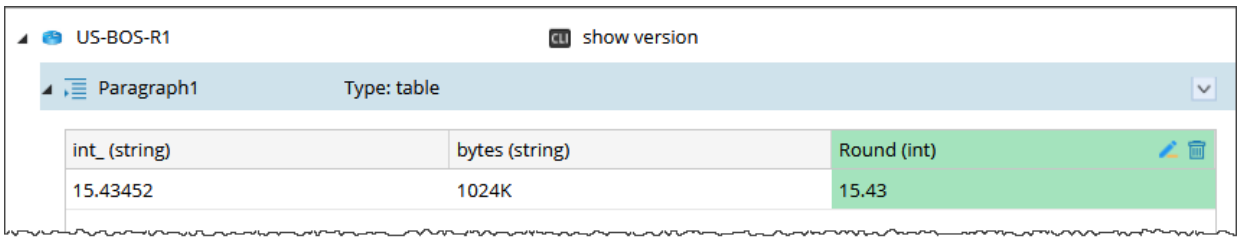
Function: **Round($number_variable)**.

For example, if the variable value is **15.43452**, the function returns **15.43**.

Refer to the image for an illustration of rounding to two decimal points.



Return Value:

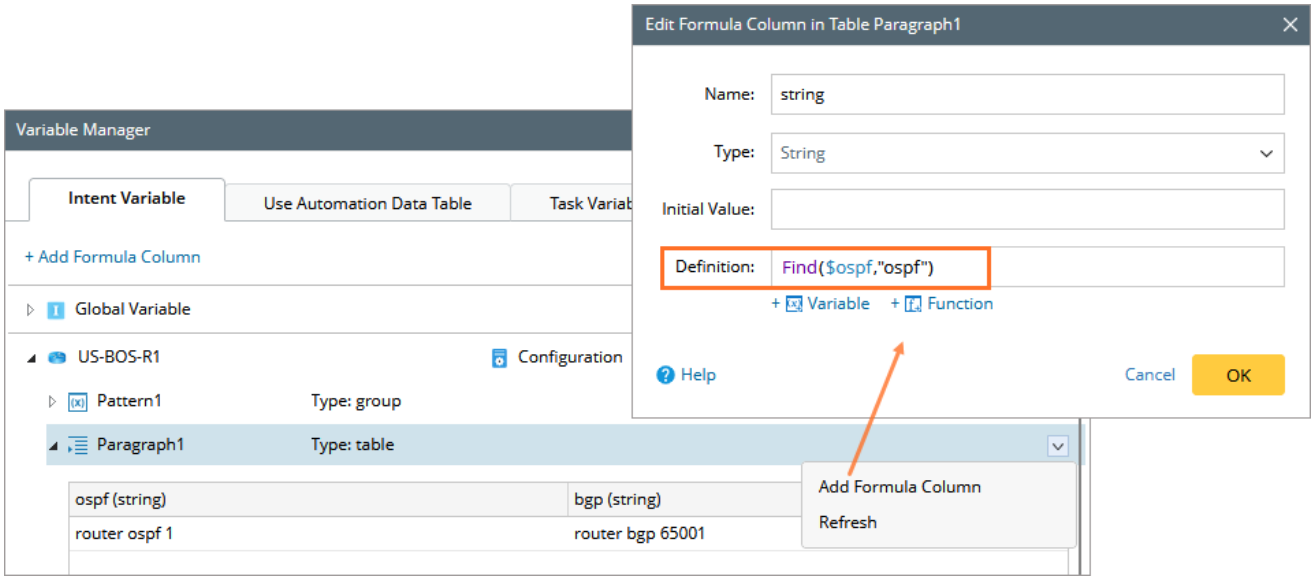


Note: This function returns **-1** if the input is not a valid number.

3.5.2 Find

This function finds one or more characters with another in a given string.

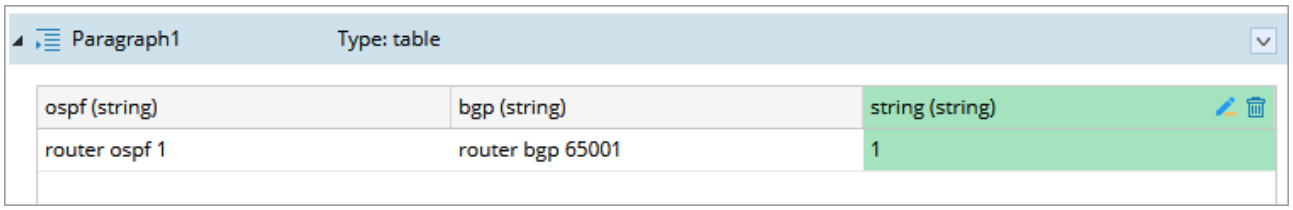
Refer to the image to find a character with given string.



Return Value:

This function returns **1**(True) or **0**(False).

It returns **-1**, if nothing is found.



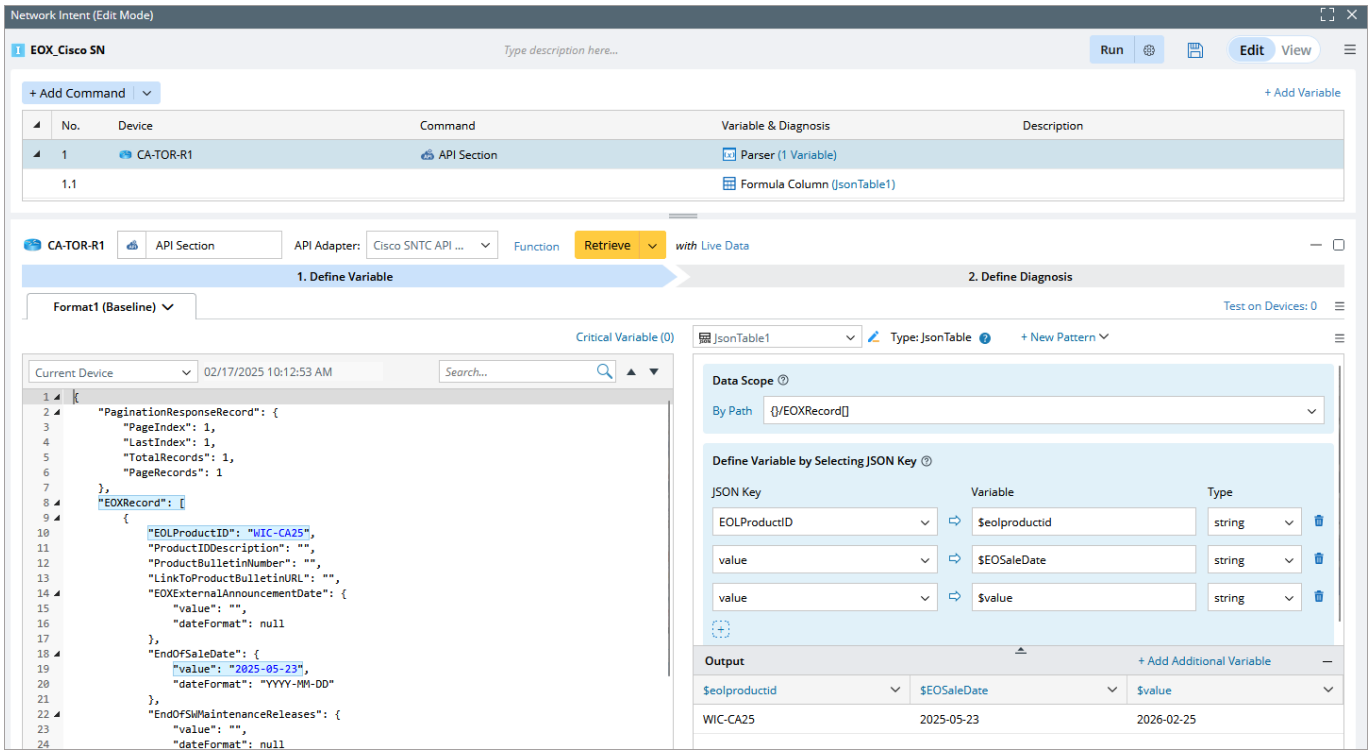
3.5.3 CompareDate

This function compares a target date with the current system time.

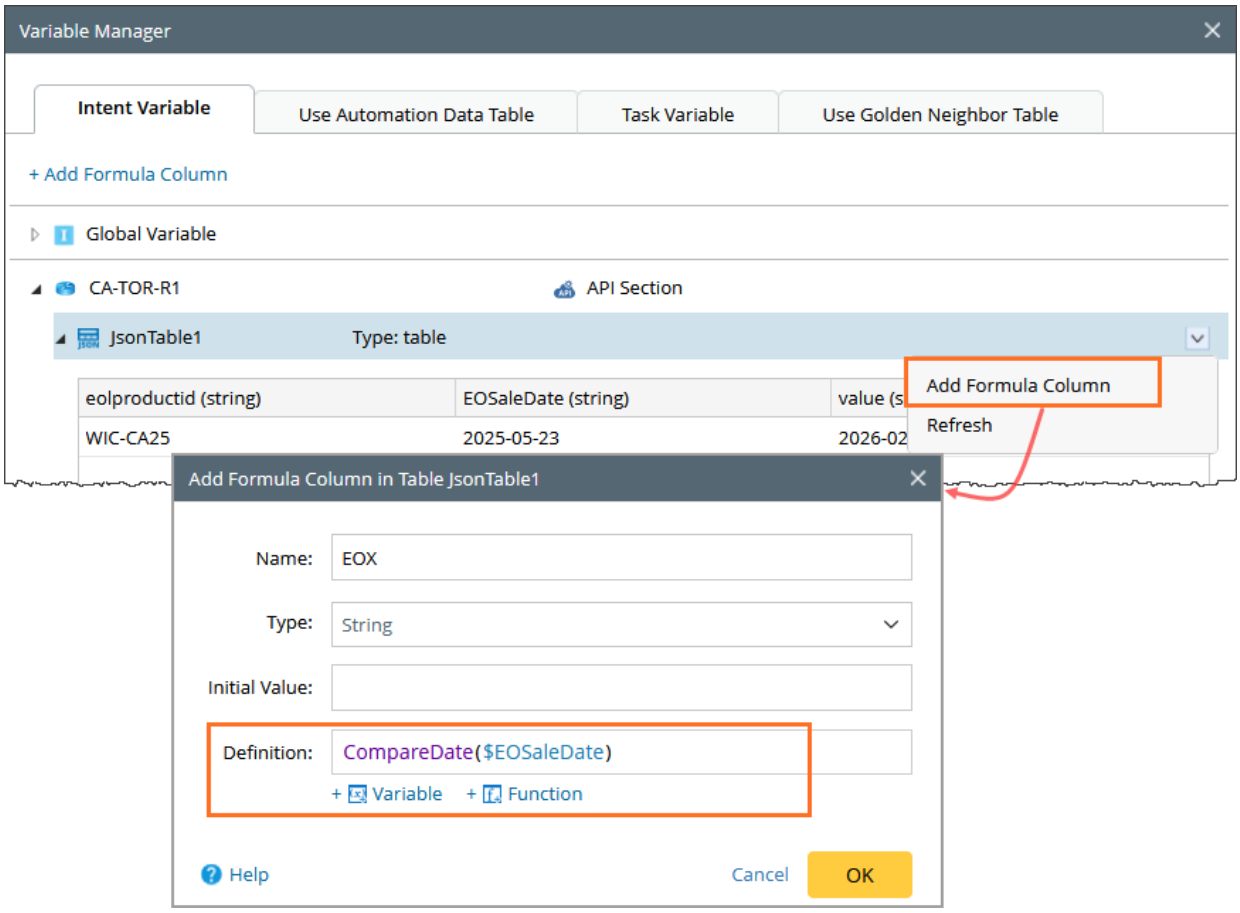
For example, to determine whether a product's **End of Sale (EOS)** date has passed or is approaching, use the ***CompareDate($Variable_name)*** function. This function identifies the specified target date. To use this function, parse the EOS date along with the product ID. This example uses an AP parser, but you can use this function based on your requirements.

If the EOS date is **2025-05-23** and the current date is earlier, the function returns **1**.

Parse the EOS using API parser.

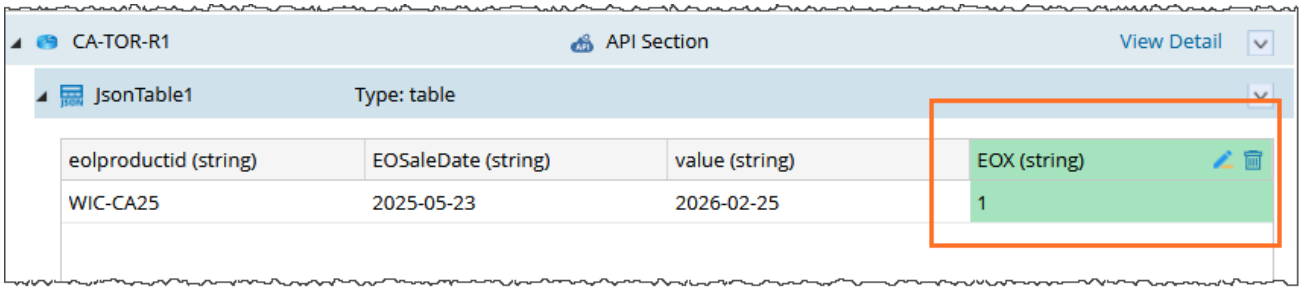


Add Compound variable to use the function.



Return Value:

Return **0** or **1**. Returns **0** if the target data is earlier than the current time; otherwise returns 1.



3.6 Table Operations

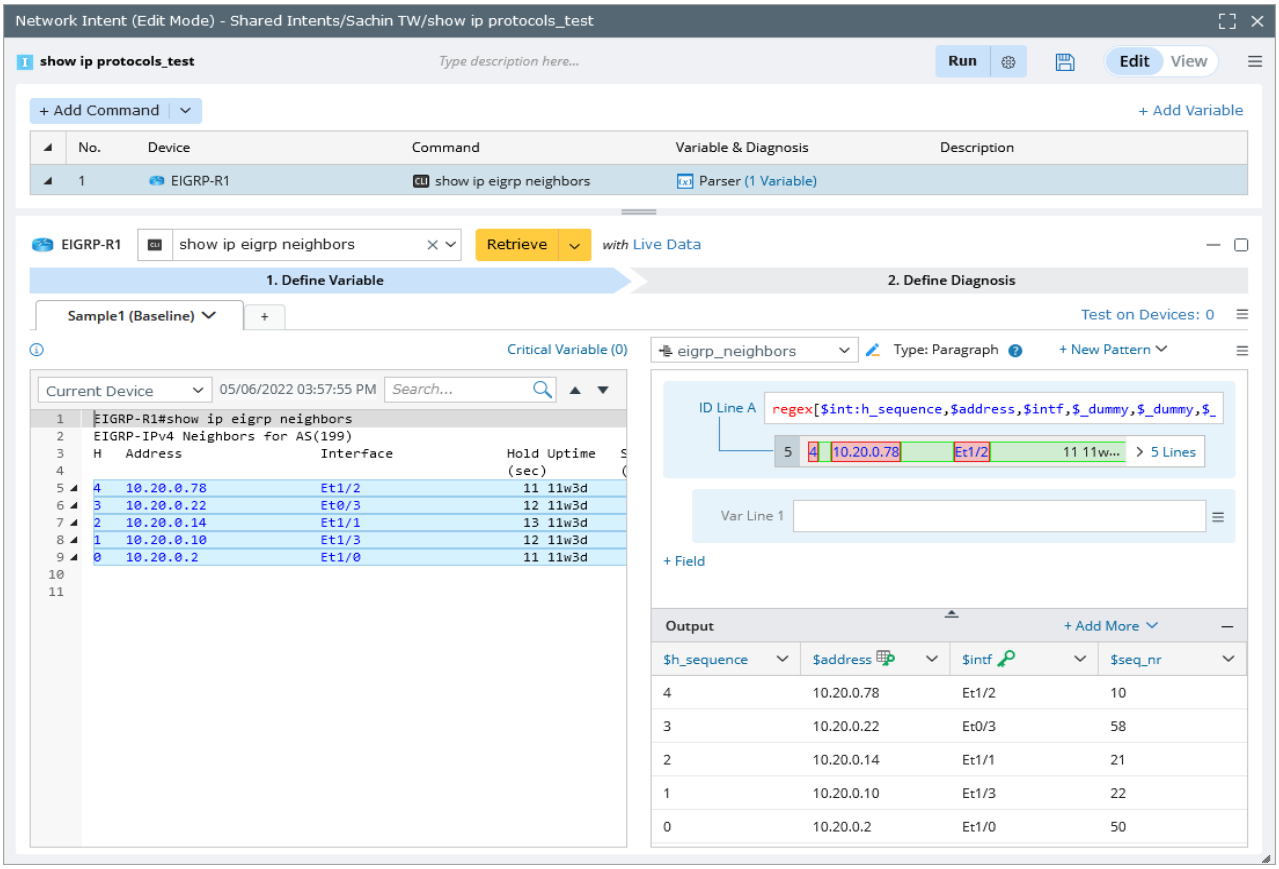
3.6.1 GetTableRowCount

This function retrieves the row count of a specified table by using the provided table name.

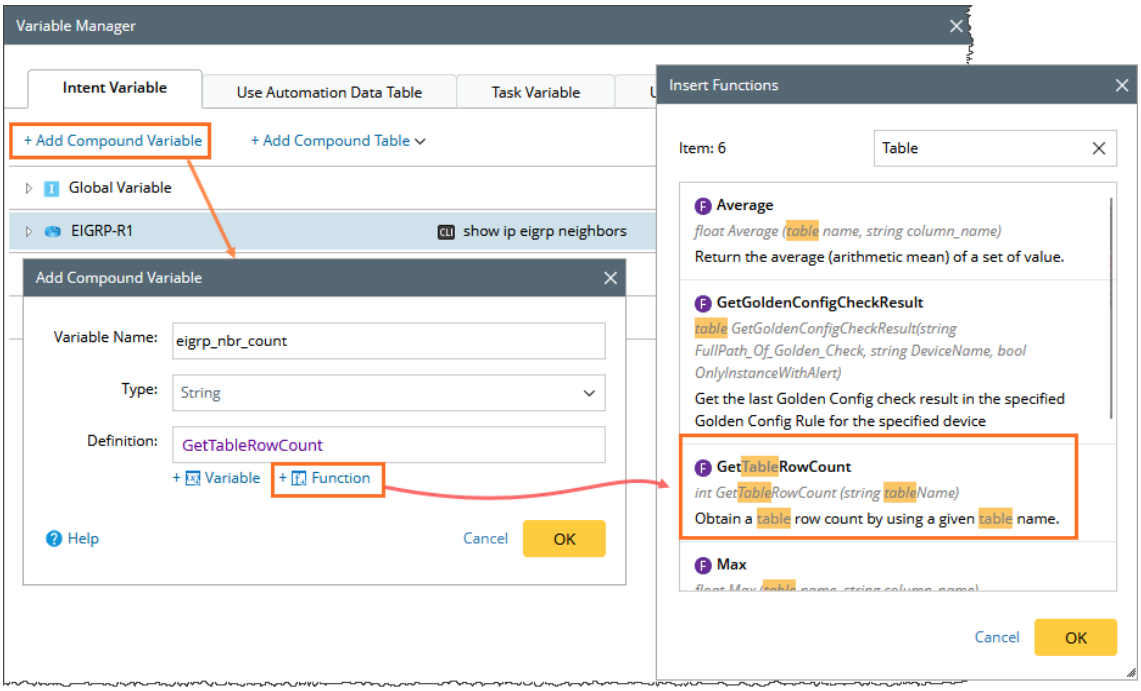
For example, to obtain the row count of the EIGRP Neighbors IP table, the **GetTableRowCount($eigrp_neighbors)** function can be used.

Follow the steps to add the Function to the existing variable table

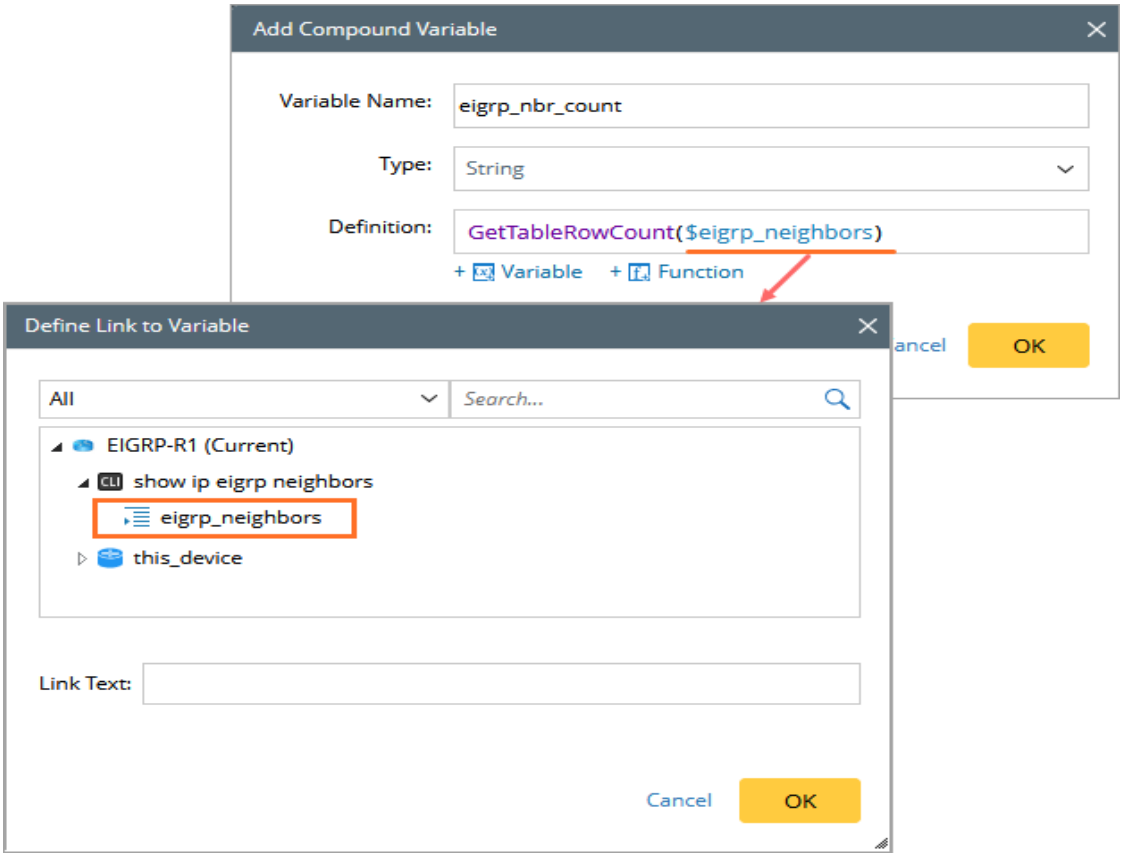
1. Create Intent and Parse eigrp_neighbor_ip table.



2. Open **Variable Manager** and click **+ Add Compound Variable** to add the Function.

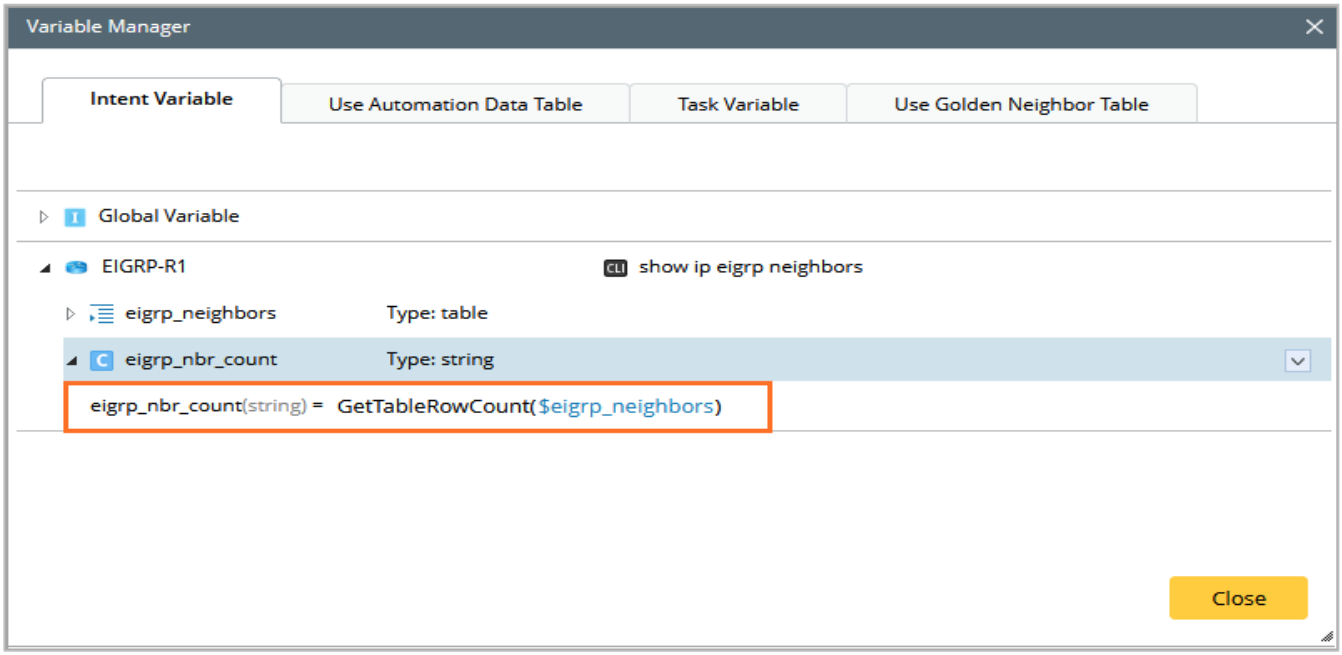


3. Insert a bracket and enter the dollar symbol ($) to Add the parsed table.



Return value:

The function should return the number of table rows if the table is valid and all conditions are met. If the table is invalid, it should return **-1**. The condition applies only when dealing with Compound Variables.



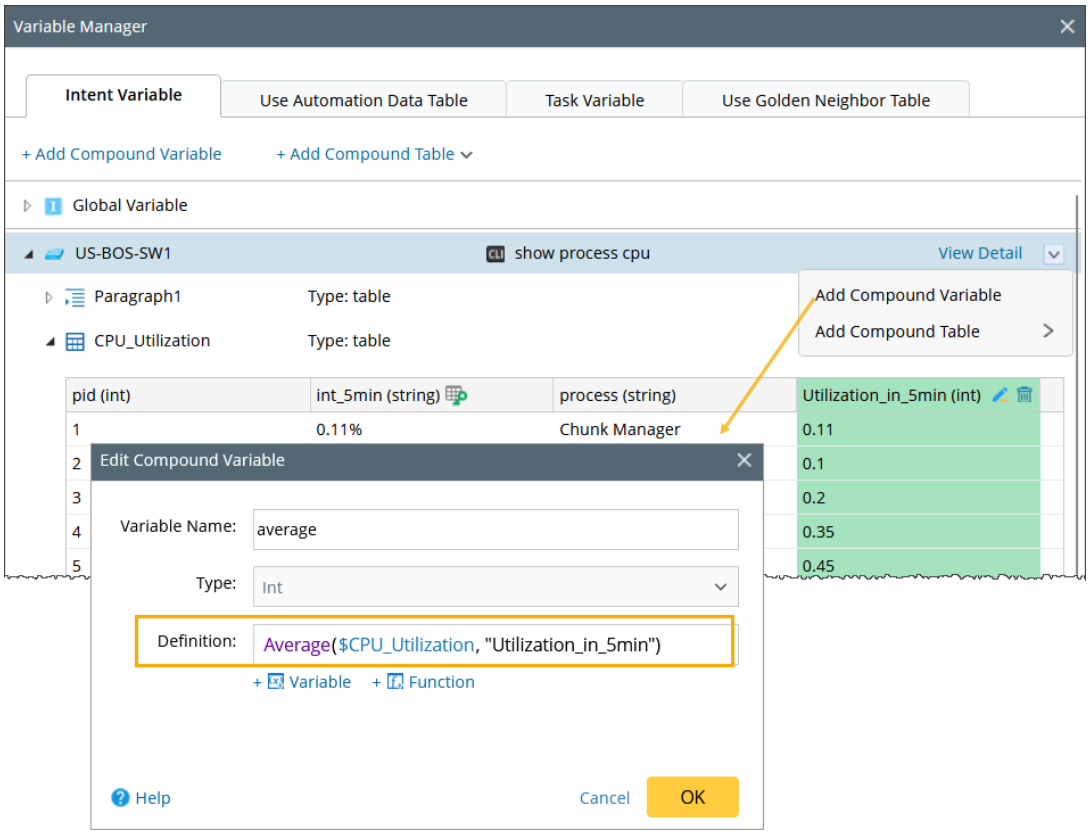
3.6.2 Average

The **Average** function calculates the arithmetic mean of a set of values within a specified column of a table.

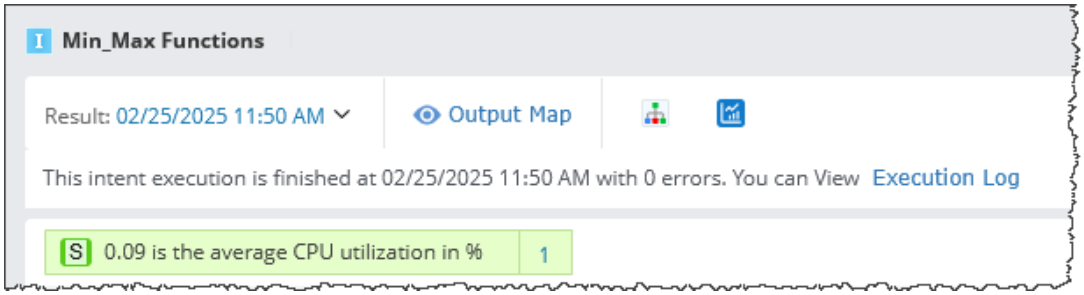
Refer to the previous example of the **Max** function, to find the average CPU utilization over a 5-minute period in the same **CPU_Utilization** table, you would use the function:

Average(CPU_Utilization, "Utilization_in_5min").

Refer to the image for an example obtaining the average value of CPU utilization over 5 minutes.



Print the message, then **Save** and **Run** the Intent to view the output. Return -1 if the input is not a valid number array.



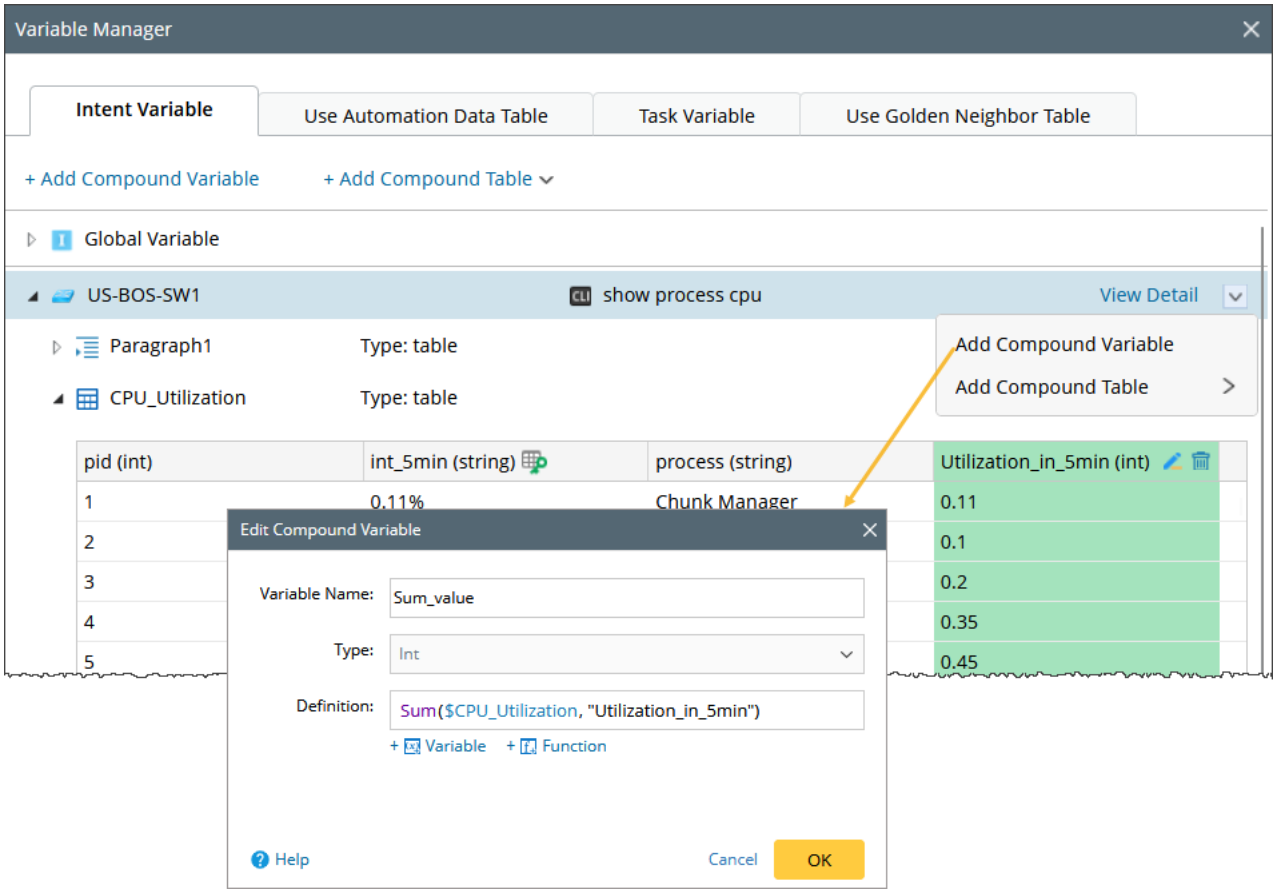
3.6.3 Sum

The **Sum** function returns the sum of values within a specified column of a table.

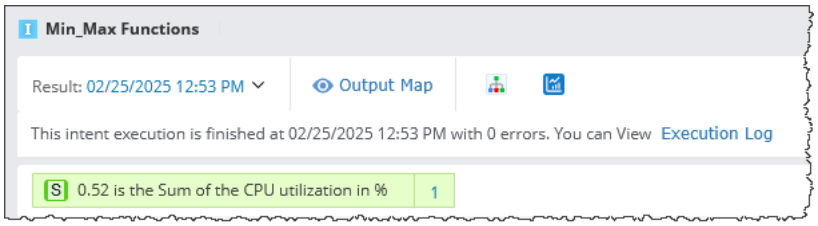
Refer to the previous example of the **Average** function, to find the sumof the CPU utilization over a 5-minute period in the same **CPU_Utilization** table, you would use the function:

Sum(CPU_Utilization, "Utilization_in_5min").

Refer to the image for an example obtaining the sum value of CPU utilization over 5 minutes.



Print the message, then **Save** and **Run** the Intent to view the output. Return -1 if the input is not a valid number array.



3.6.4 Max

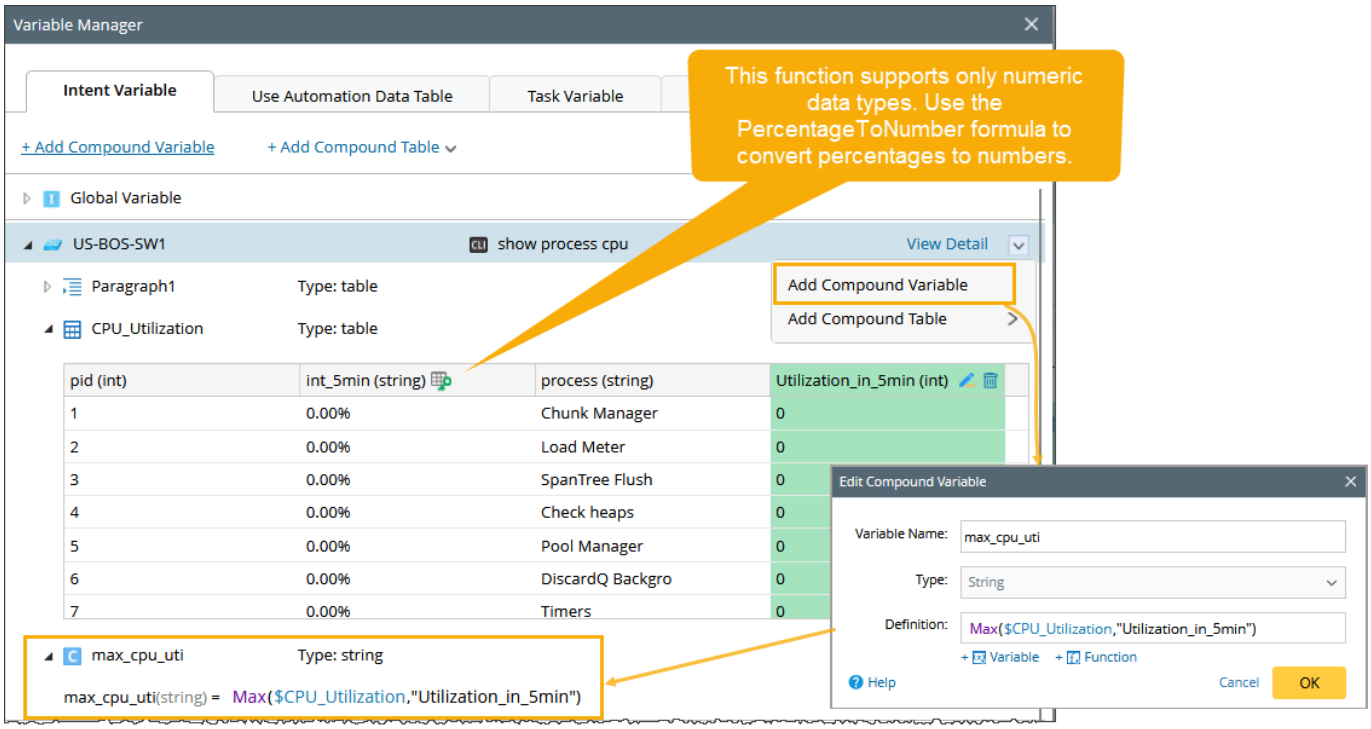
The Max function retrieves the highest value from a specified column within a given table.

Function: ***Max($Table, "Column_Name")***

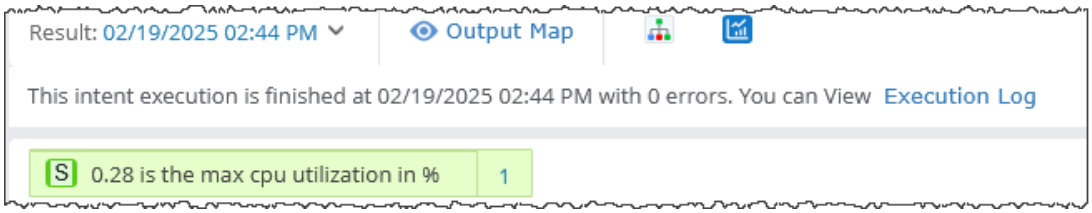
- **$table**: Specifies the table name that holds the data. Example: ***CPU_Utilization***.
- **Column_Name**: Indicates the column from which the maximum value will be calculated. Example: ***Utilization_in_5min***.

If you have a table named CPU_Utilization and you need to find the maximum CPU utilization over a 5-minute period, you would use, **Max(CPU_Utilization, "Utilization_in_5min")** function.

Refer to the image for an example for obtaining the maximum value of CPU utilization over 5 minutes.



Print the message, then **Save** and **Run** the Intent to view the output. Return *"-1"* if the input is not a valid number array.



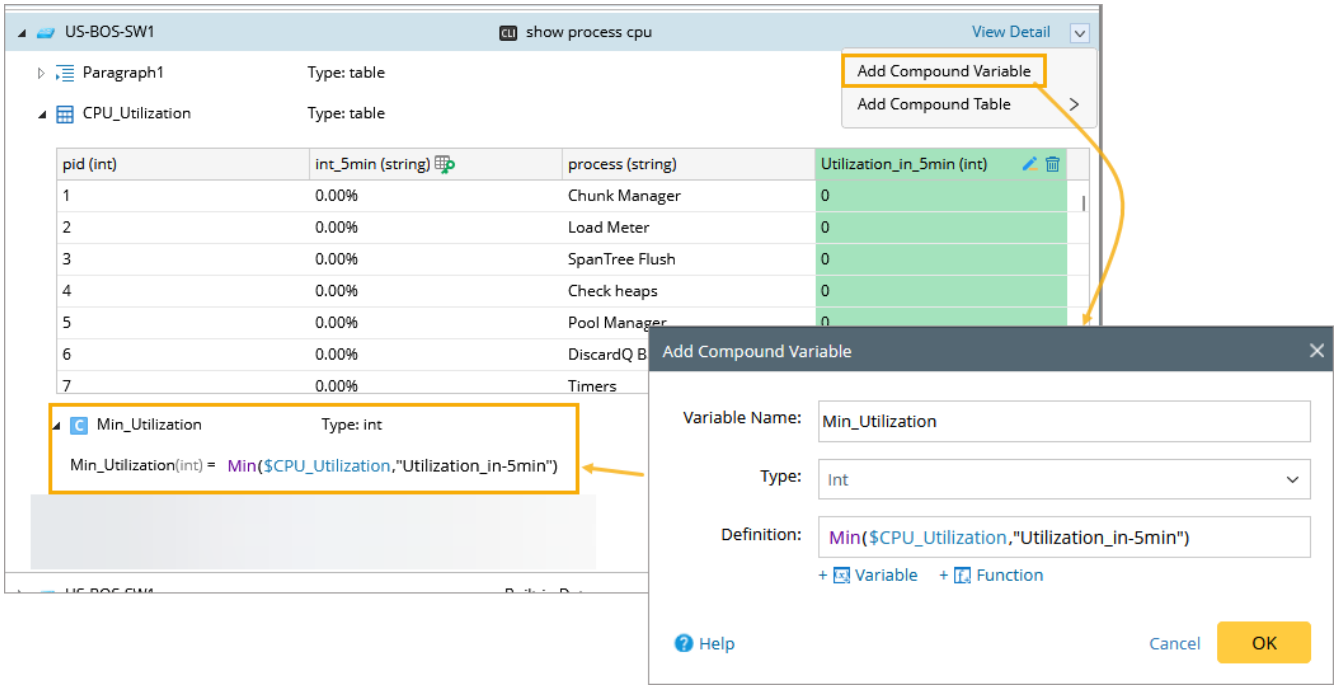
3.6.5 Min

The **Min** function retrieves the lowest value from a specified column within a given table.

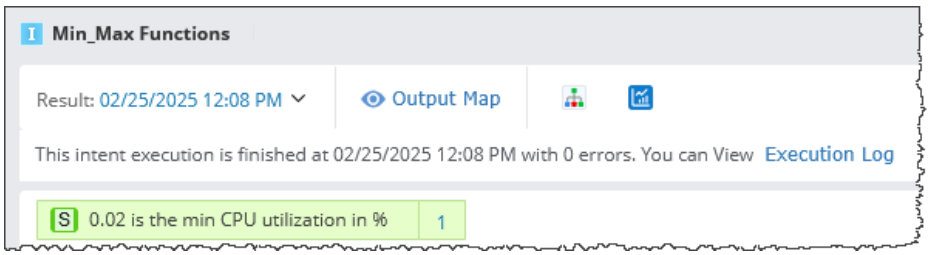
Refer to the previous example of the **Max** function,Tto find the minimum CPU utilization over a 5-minute period in the same **CPU_Utilization** table, you would use the fnction:

Min(CPU_Utilization, "Utilization_in_5min").

Refer to the image for an example obtaining the minimun value of CPU utilization over 5 minutes.



Print the message, then **Save** and **Run** the Intent to view the output. Return -1 if the input is not a valid number array.



3.7 Device and Interface Management

3.7.1 Device

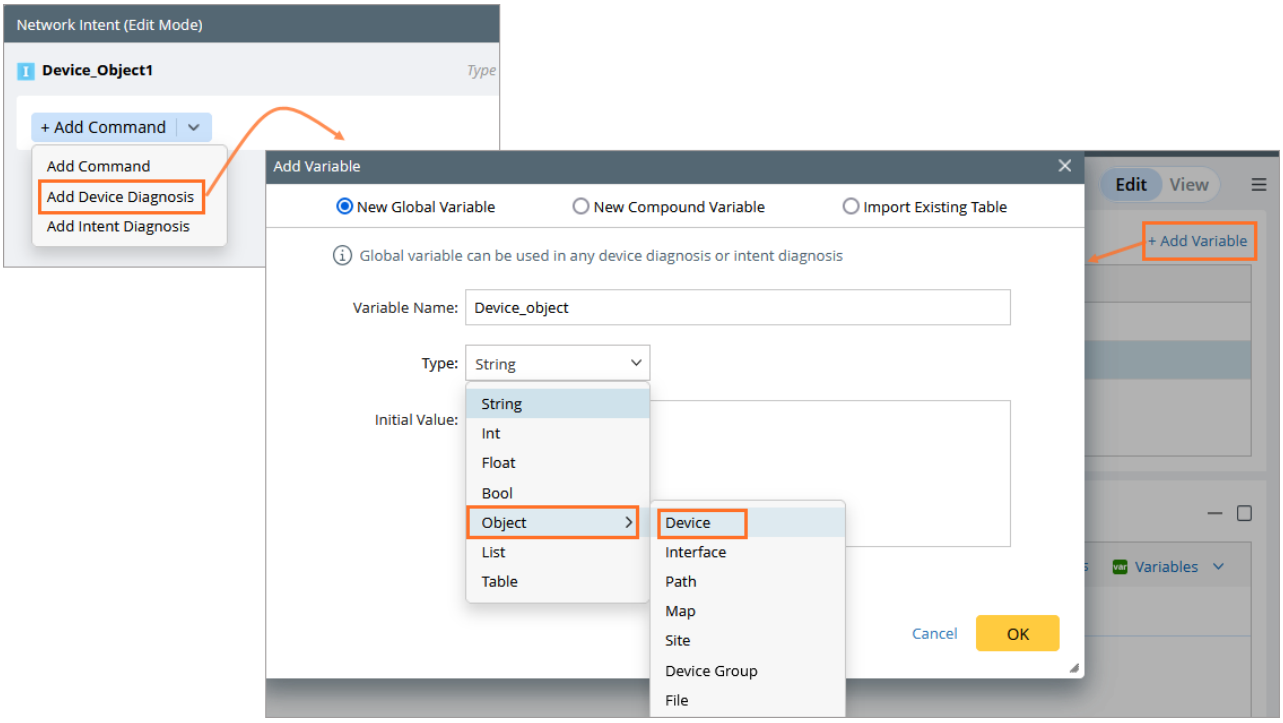
This function create a device object with an existing device for further operation.

Note: You can pass an IP address as the Hostname variable, which will be automatically recognized and converted.

For example, using device diagnosis you can use the function and create a device object.

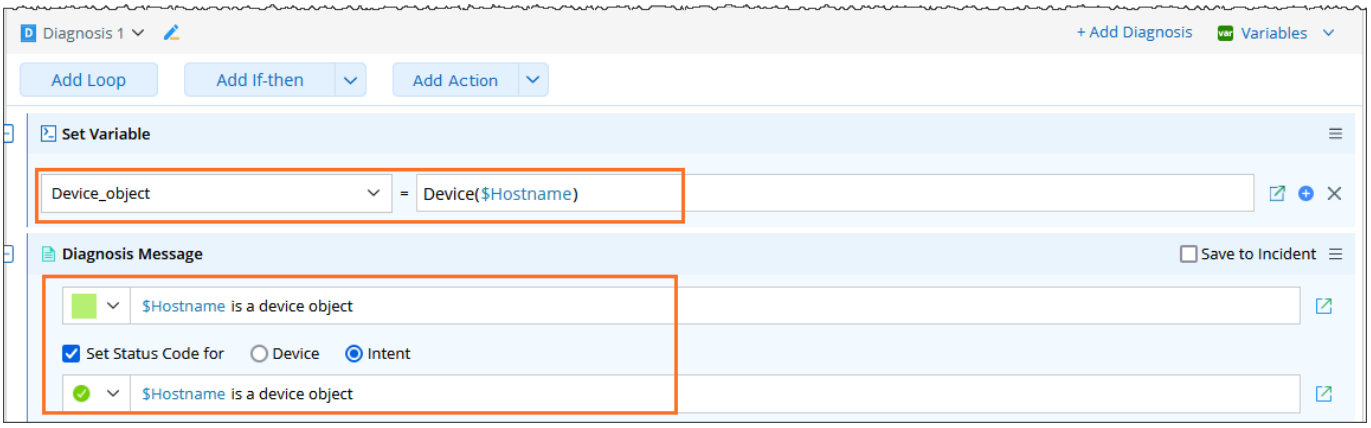
Follow these steps to create a device object using an existing device:

1. In the **Network Intent** edit window, click **+ Add Command,** then select **Add Device Diagnosis**.
2. Click **+ Add Variable** to create a **New Global Variable**.
3. Define the Global Variable as follows:
    a) Enter the Variable Name.
    b) In the **Type** dropdown, select **Object** > **Device,** and then click **OK**.



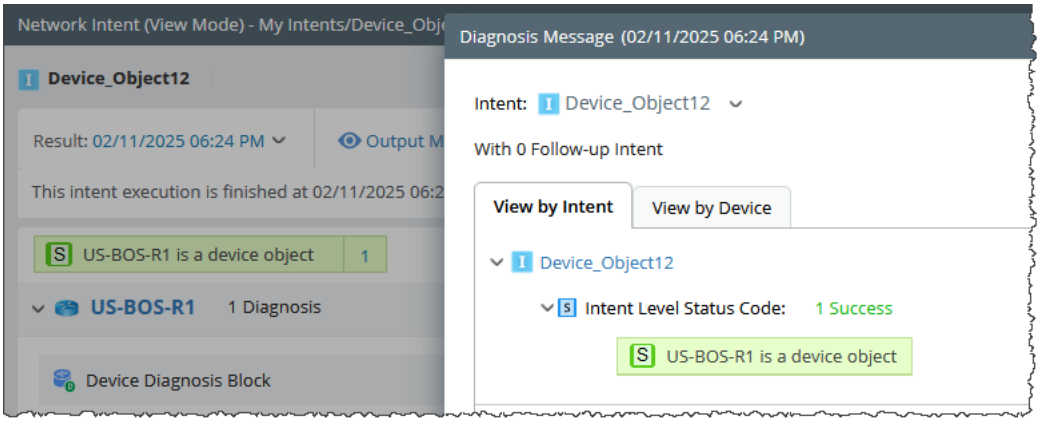
4. Click **Diagnosis 1** to **set veriable** and define the **Diagnosis Message** for creating a device. Refer to the image below.

    Function: **Device($hostname).**



5. Click **Save** and ten **Run** the intent.

   The diagnosis message will print the **Device** name.



Return Value:

This function returns device name.

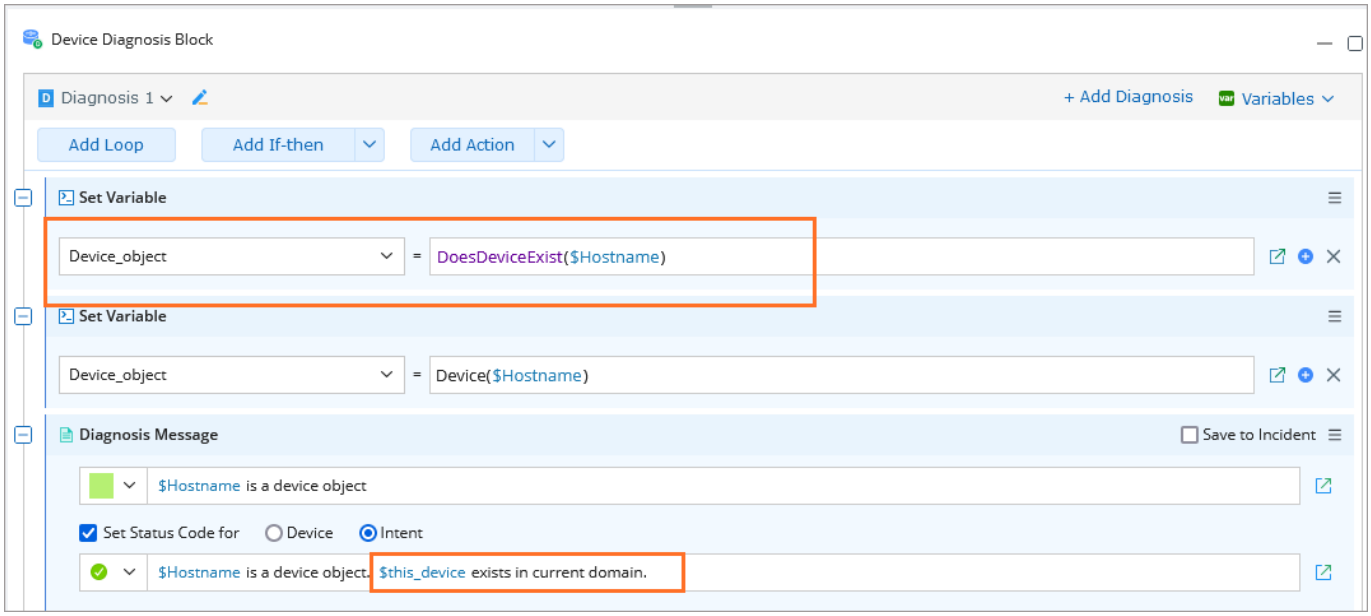
3.7.2 DoesDeviceExist

This function verifies whether a device exists in the current domain using the ***DoesDeviceExist($Hostname)*** formula.

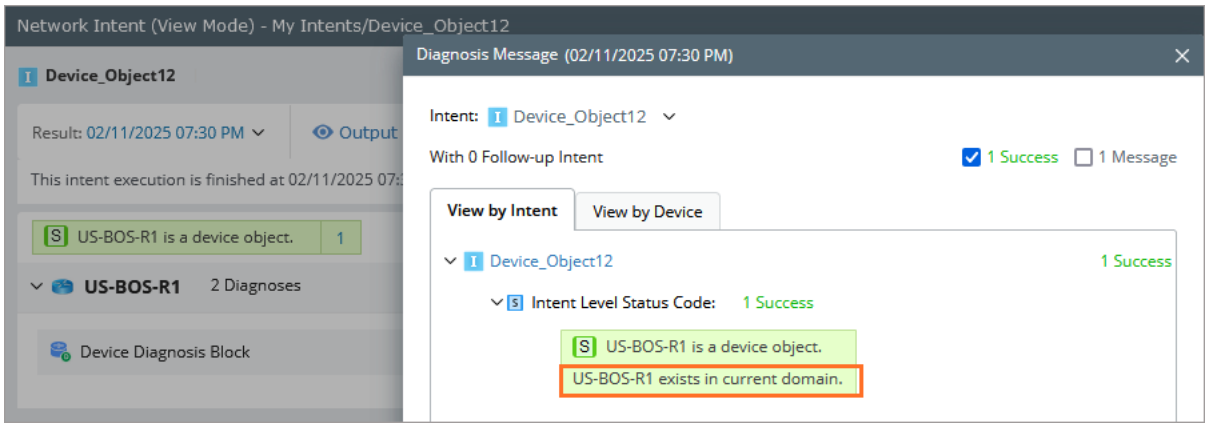
Refer to the image to verify if the device is present within the current domain.

Extend the sample example used above for the **Device** object function.

Note: You can pass an IP address as the **Hostname** variable.



Print the diagnostic message to obtain the return value, indicating whether the device exists in the current domain.



3.7.3 Interface

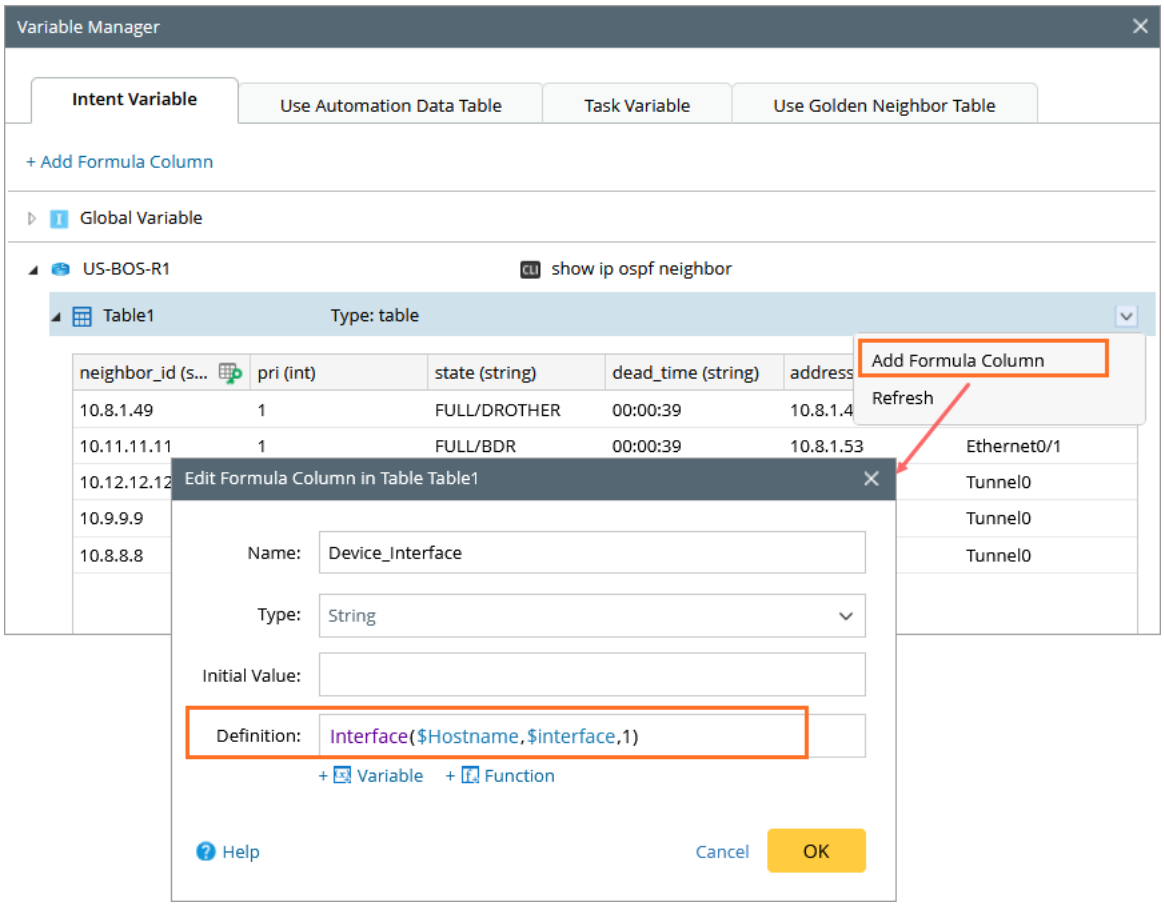
This function creates a device interface object with an existing interface for further operation.

Function: ***Interface($Hostname,$Interface,$Interface_Type)***

You can enter the following string or number as InterfaceType:

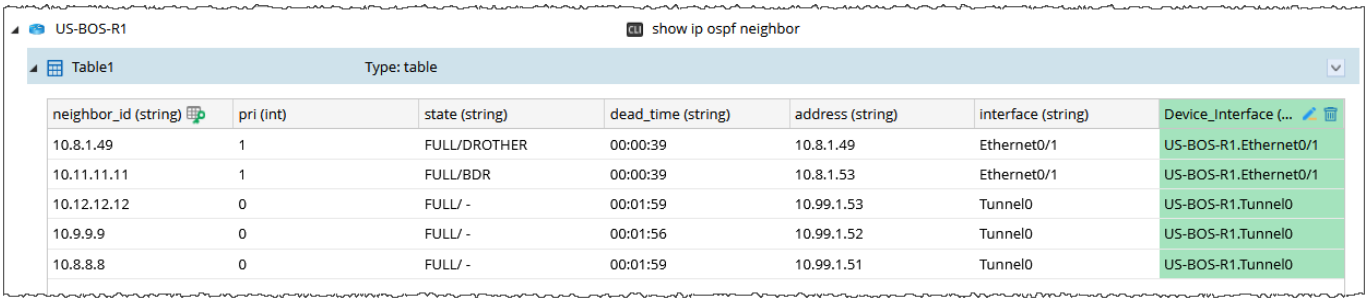
- Interface = 1
- IPv4 Interface = 2
- IPv6 Interface = 3
- IPsec VPN Interface = 4
- GRE VPN Interface = 5

Refer to the image for an example to create a device interface object.



Return Value:

This function returns *device.interface* object.



US-BOS-R1 — show ip ospf neighbor

Table1 — Type: table

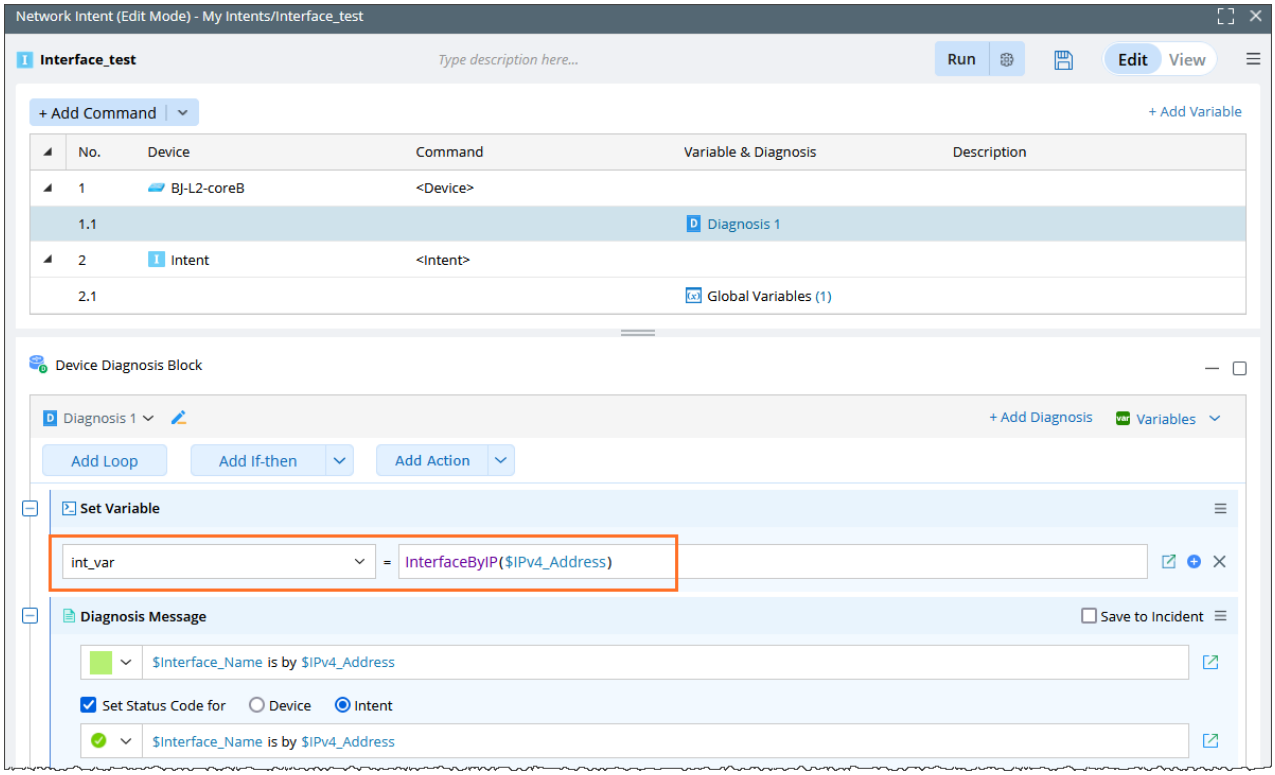
neighbor_id (string)	pri (int)	state (string)	dead_time (string)	address (string)	interface (string)	Device_Interface (...
10.8.1.49	1	FULL/DROTHER	00:00:39	10.8.1.49	Ethernet0/1	US-BOS-R1.Ethernet0/1
10.11.11.11	1	FULL/BDR	00:00:39	10.8.1.53	Ethernet0/1	US-BOS-R1.Ethernet0/1
10.12.12.12	0	FULL/ -	00:01:59	10.99.1.53	Tunnel0	US-BOS-R1.Tunnel0
10.9.9.9	0	FULL/ -	00:01:56	10.99.1.52	Tunnel0	US-BOS-R1.Tunnel0
10.8.8.8	0	FULL/ -	00:01:59	10.99.1.51	Tunnel0	US-BOS-R1.Tunnel0

3.7.4 InterfaceByIP

This function creates a device interface object using an existing interface IP address for further operations. If multiple interfaces have the same IP address, the function selects the first matching entry. The function automatically identifies IPv4 and IPv6 addresses.

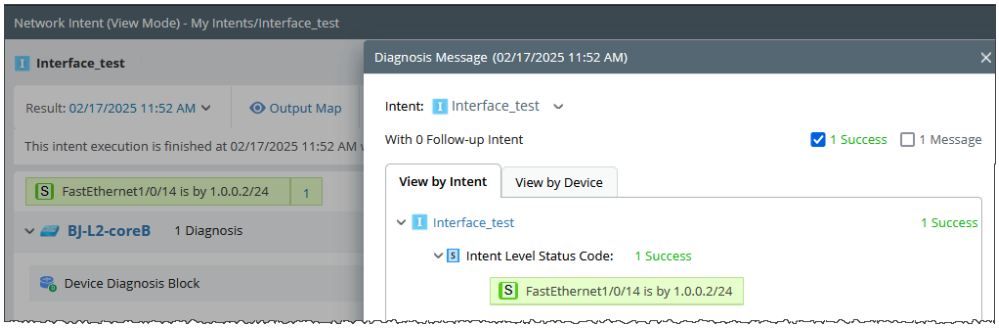
For example, to retrieve the interface associated with a specific IP address, call the ***InterfaceByIP($IPAddress)*** function. Follow the same approach as the Device function. Additionally, perform a device diagnosis, set a variable, and print the diagnosis message.

Refer to the image to creates device interface with an existing interface IP.



Return Value:

This function returns **interface**.

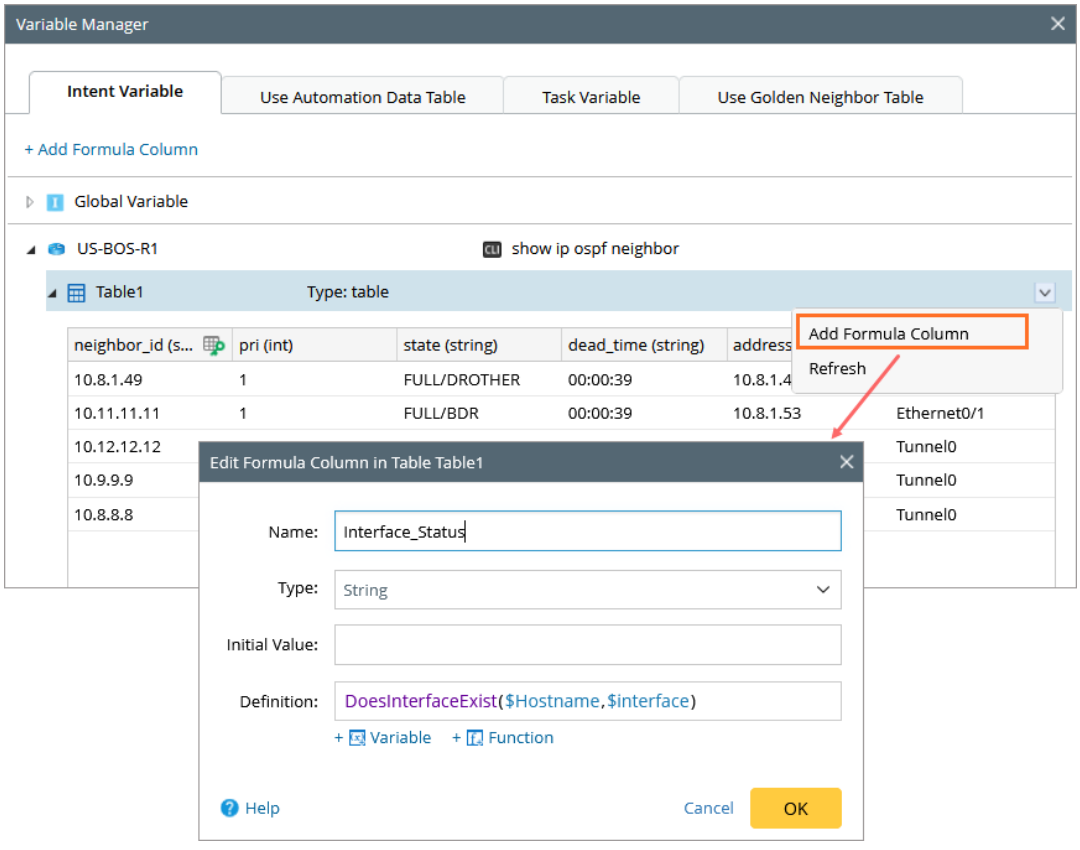


3.7.5 DoesInterfaceExist

This function checks whether an interface still exists in the current domain.

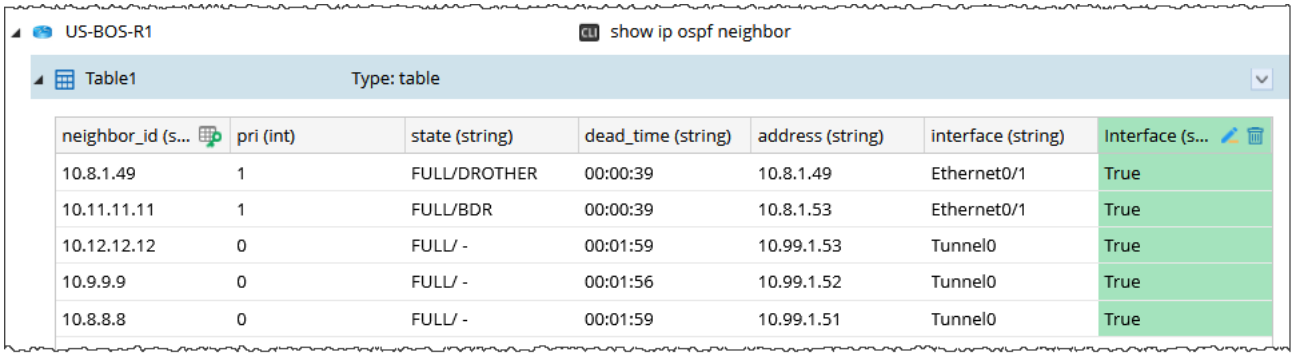
Function: **DoesInterfaceExist($hostname,$interface)**

Refer to the image for an example to check whether an interface still exists.



Return Value:

- **True** – if an interface exists in the current domain.
- **False** – if an interface does not exist in the current domain.



neighbor_id (s...	pri (int)	state (string)	dead_time (string)	address (string)	interface (string)	Interface (s...
10.8.1.49	1	FULL/DROTHER	00:00:39	10.8.1.49	Ethernet0/1	True
10.11.11.11	1	FULL/BDR	00:00:39	10.8.1.53	Ethernet0/1	True
10.12.12.12	0	FULL/ -	00:01:59	10.99.1.53	Tunnel0	True
10.9.9.9	0	FULL/ -	00:01:56	10.99.1.52	Tunnel0	True
10.8.8.8	0	FULL/ -	00:01:59	10.99.1.51	Tunnel0	True

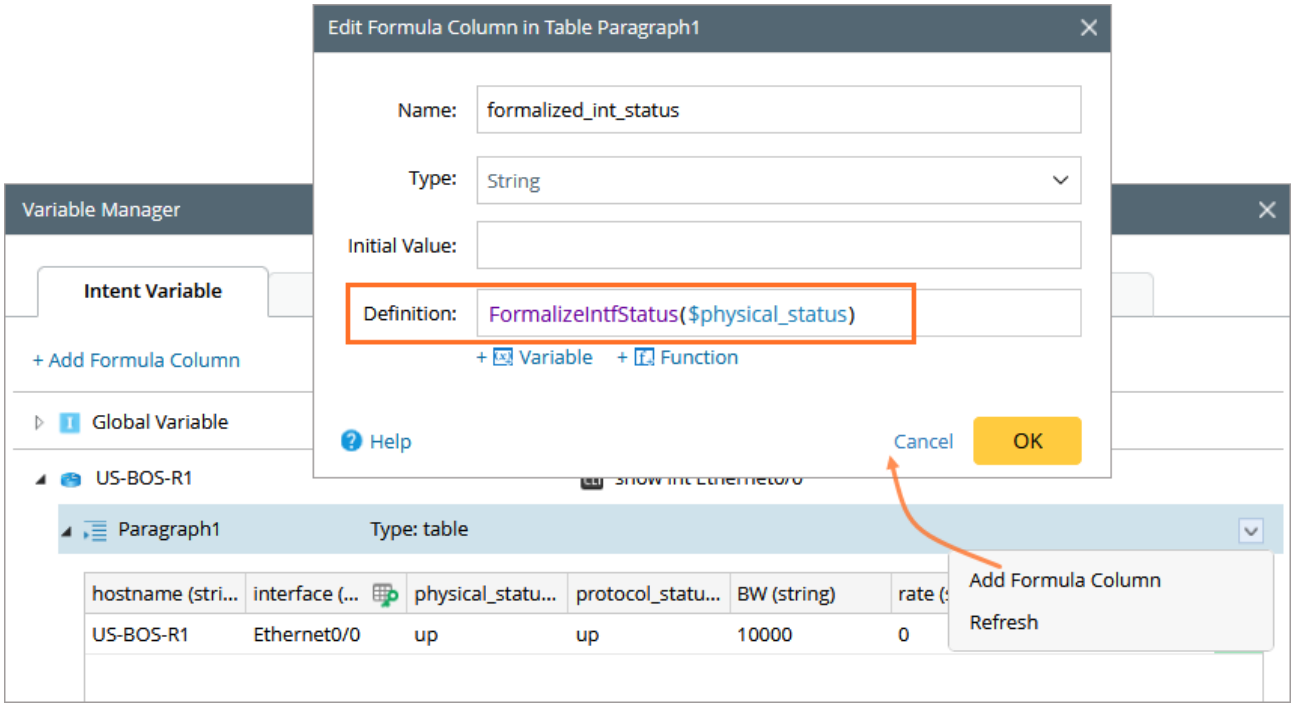
3.7.6 FormalizeIntfStatus

This function returns formalized interface status information based on the given interface status information.

For example, if an entherface physical status is in the lowercase, using this function the status will be in the uppercase.

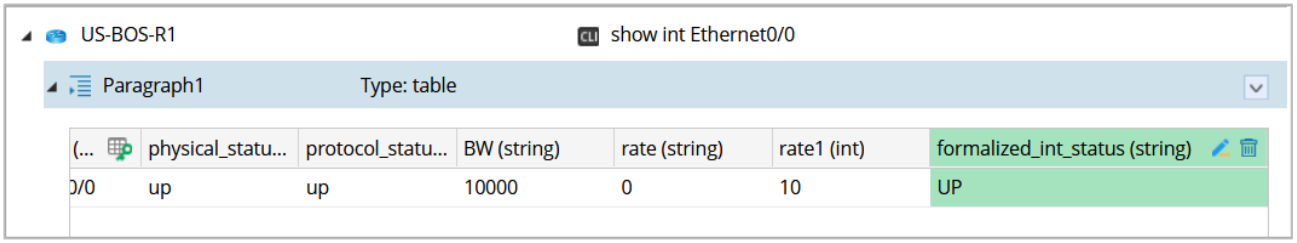
Function: ***FormalizeIntfStatus($physical_status)***

Refer to the image for an example of get the formalized interface status.



Return Value:

It returns **UP**.



3.8 Path and Map Management

3.8.1 Path

This function creates a path object with an existing path defined in domain for further operation.

Note: Creating new paths is not supported.

For example, you need to create an **Intent** and define either a **Global Variable** or a **Local Variable**. In the **Diagnosis** pane, you can set a variable for the created global variable by utilizing the **Function**:

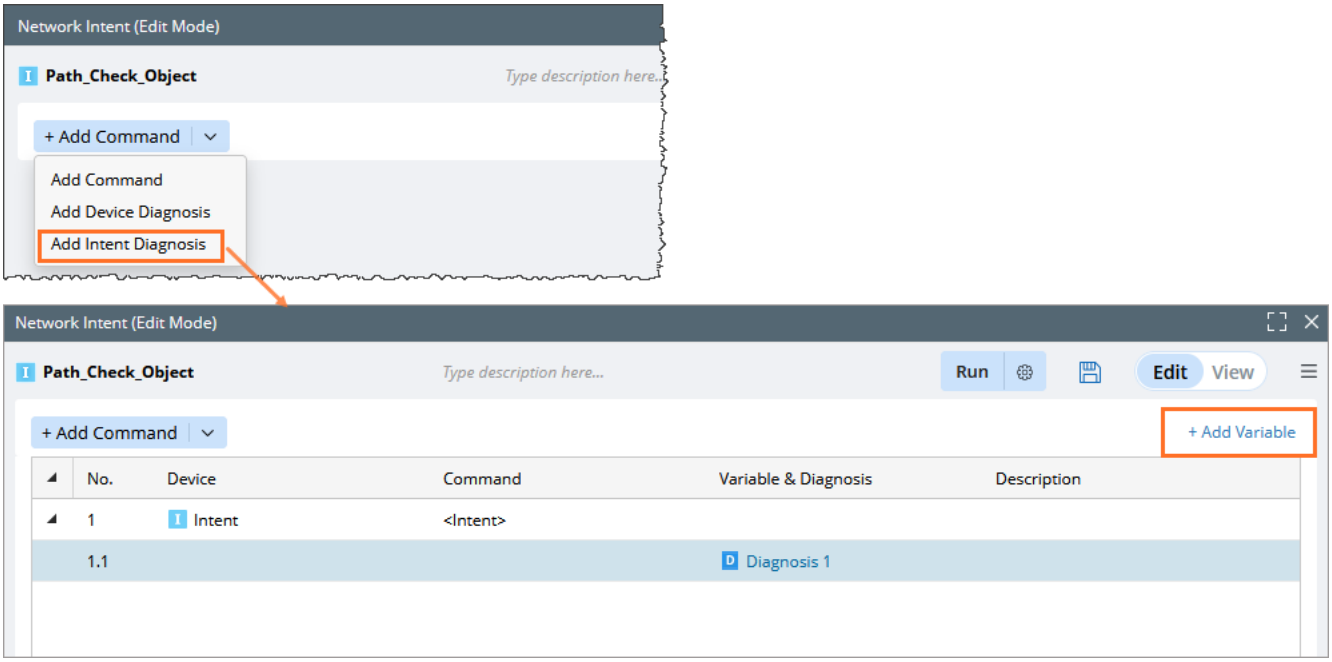
Path("Application Name", "Path Name")

- **DemoDashboard** is the **Application Name**
- **V-7** is the **Path Name**

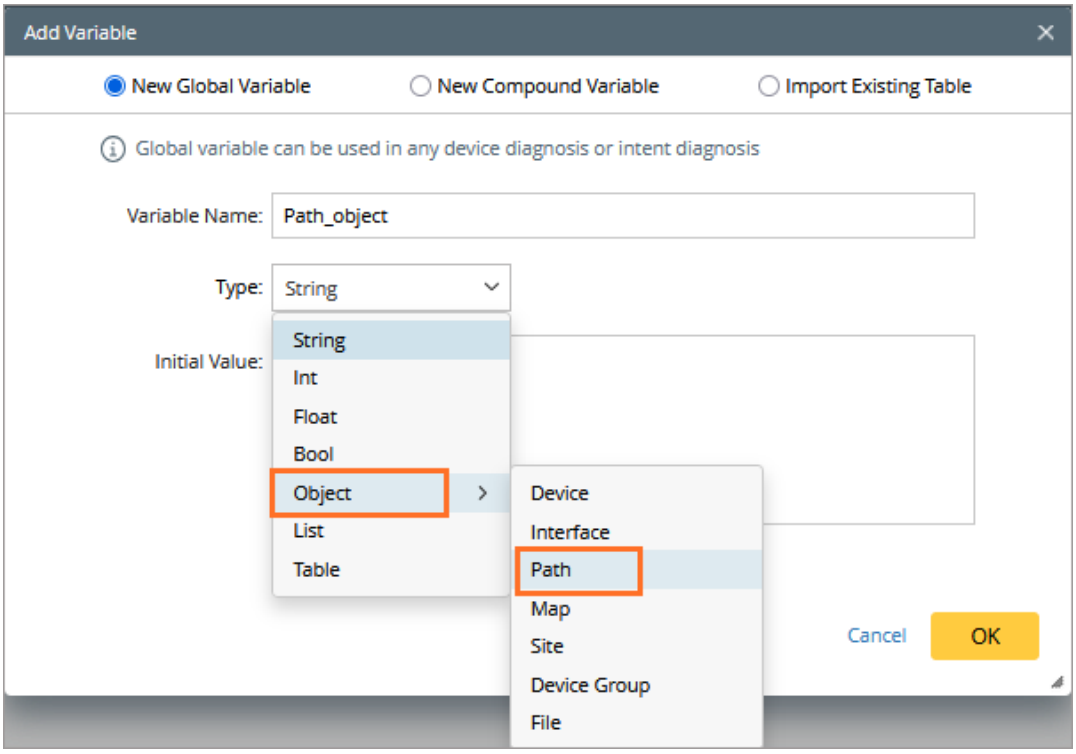
Thus, the function you will use is: **Path("DemoDashboard", "V-7")**

Follow these steps to create a path object using an existing path in the domain:

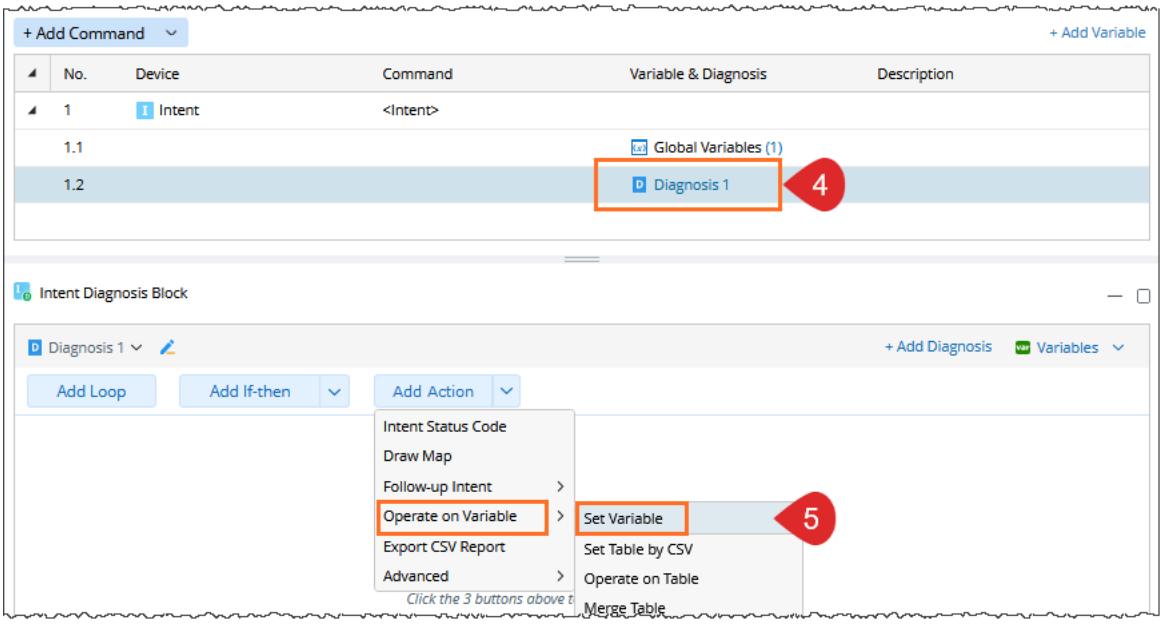
1. In the **Network Intent** edit window, click **+ Add Command,** then select **Add Intent Diagnosis**.
2. Click **+ Add Variable** to create a New Global Variable.



3. Define the Global Variable as follows:
    a) Enter the Variable Name.
    b) In the **Type** dropdown, select **Object** > **Path,** and then click **OK**.

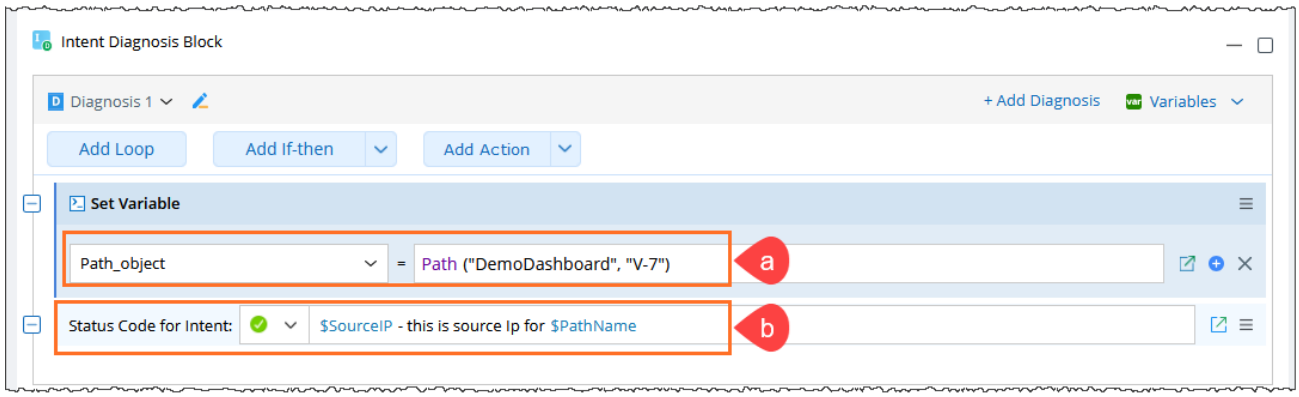


4. Click **Diagnosis 1** to set veriable and define the diagnosis for retrieving the return value.
5. Click the **Add Action** dropdown and select **Operate on variable** > **Set Variable**.



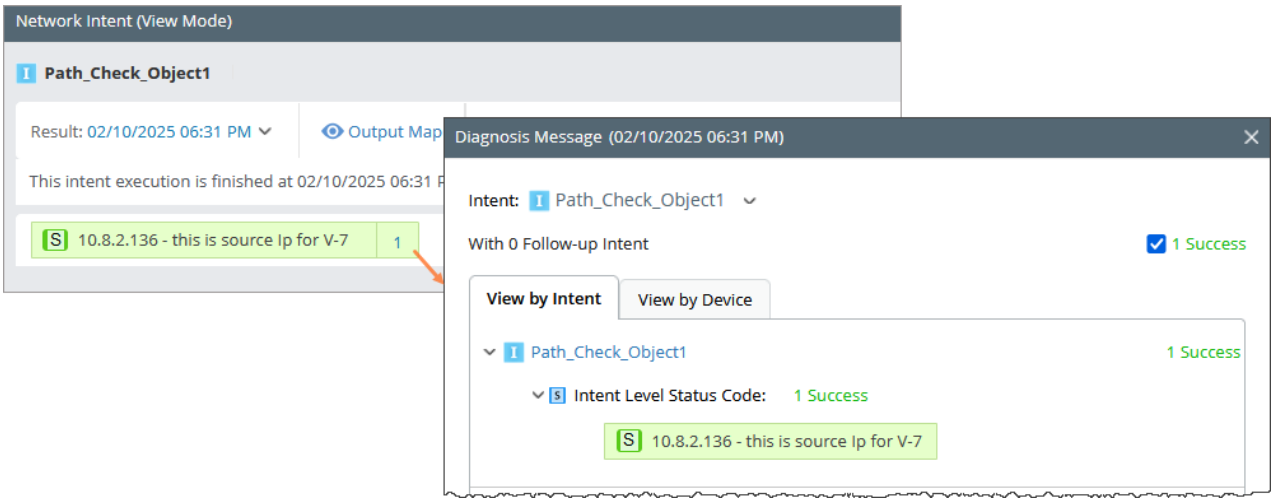
   a) Select the **Path_object** as Global variable and define the function as **Path("DemoDashboard", "V-7").**

   b) Set the **Status code** for the Intent to display a message.



6. Click **Save** and ten **Run** the intent.

   The diagnosis prints the source IP associated with the **V-7** path.



3.8.2 NewMap

This function generates a new map object using the specified full path of the target map file.

Notes:

- Creating a function map is not supported.
- You can create a map using the **Draw Map** feature or by defining the function, device, and other attributes of the map, depending on your requirements.

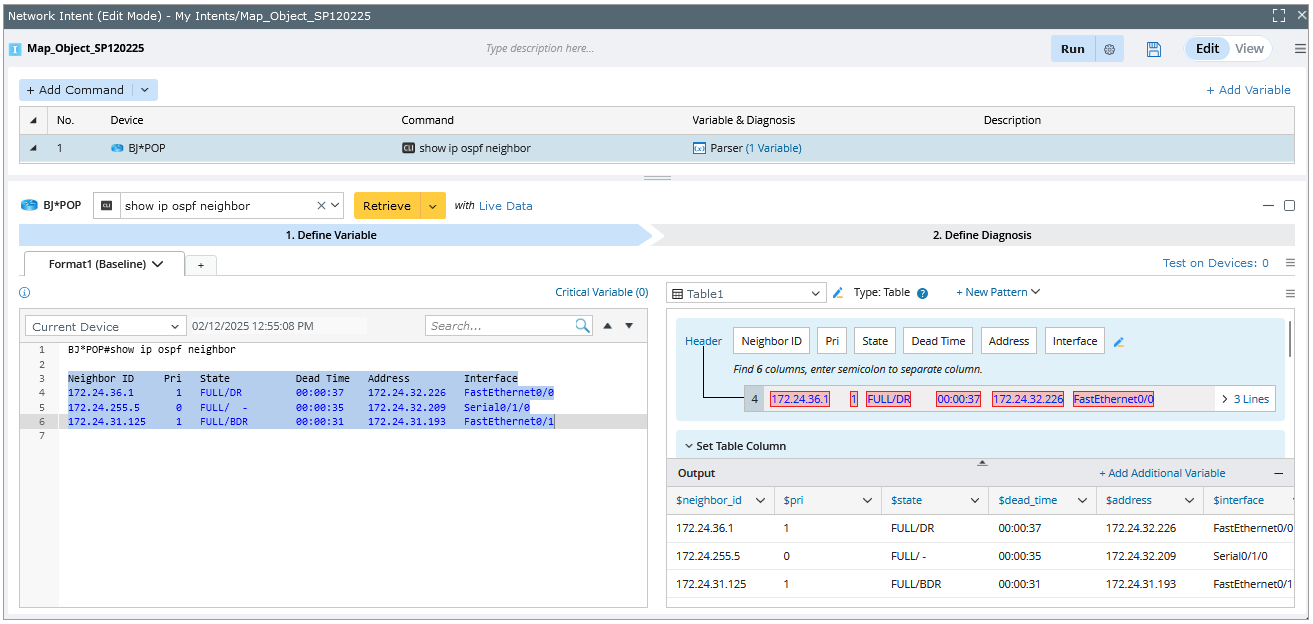
Example:

To create a new map, use the **show ip ospf neighbor** command to parse the OSPF neighbor table. Then:

- Create a local variable for the map object.
- Set the variable using the NewMap function.
- Loop through the parsed table to use its variables for creating the map.

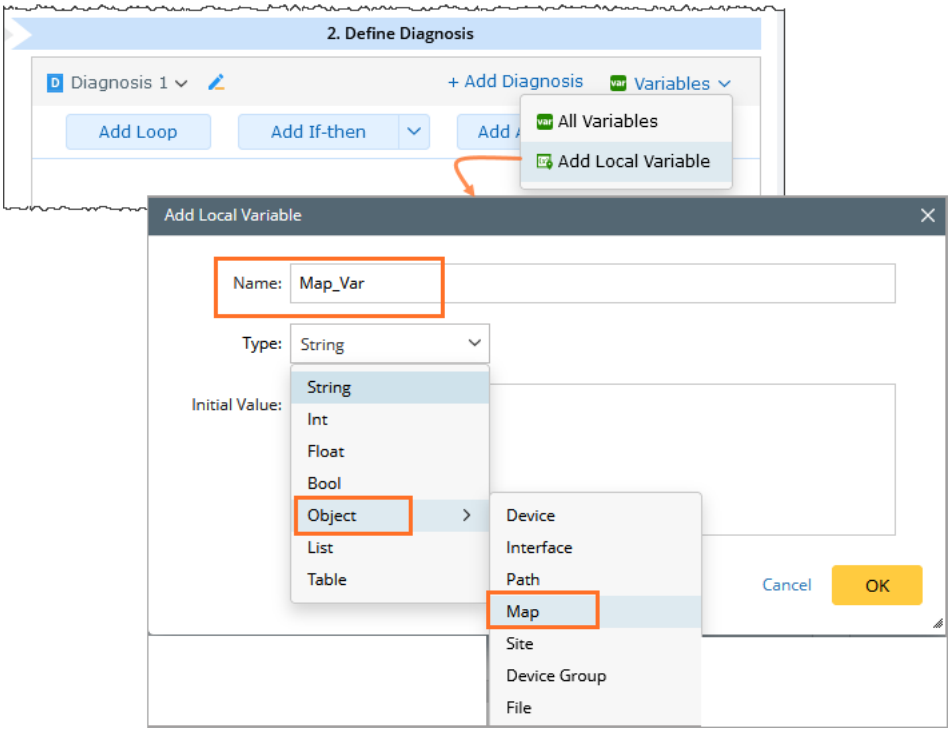
Follow these steps to create a new map using an existing path.

1. Parse the table using the **show ip ospf neighbor** command..

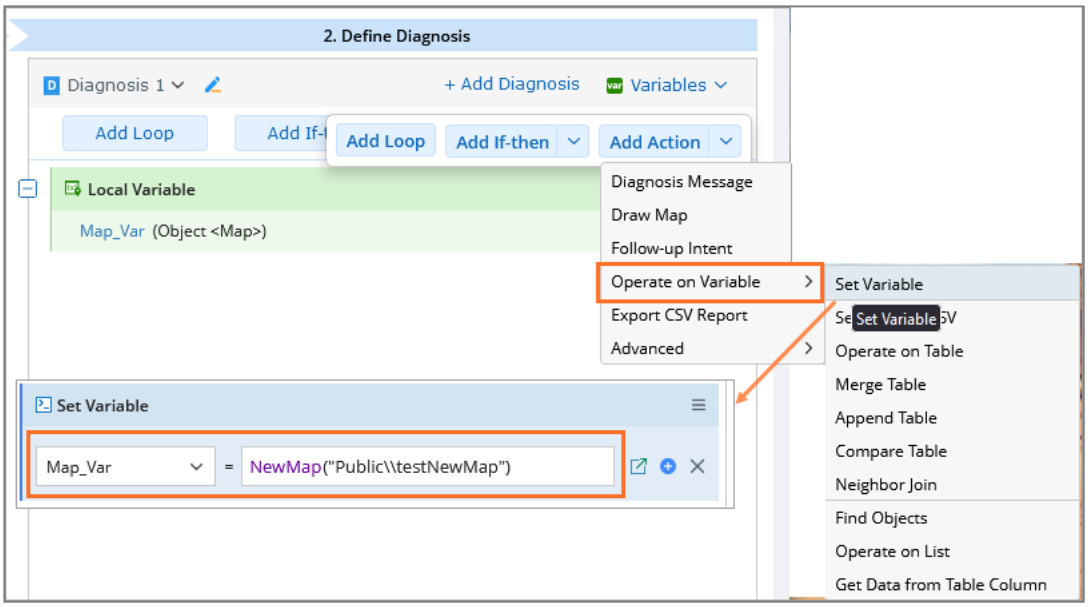


2. Add a **Formula** column to convert the IP address to a hostname using ***IPtoHostname($interface)***, as described in Section 3.4.

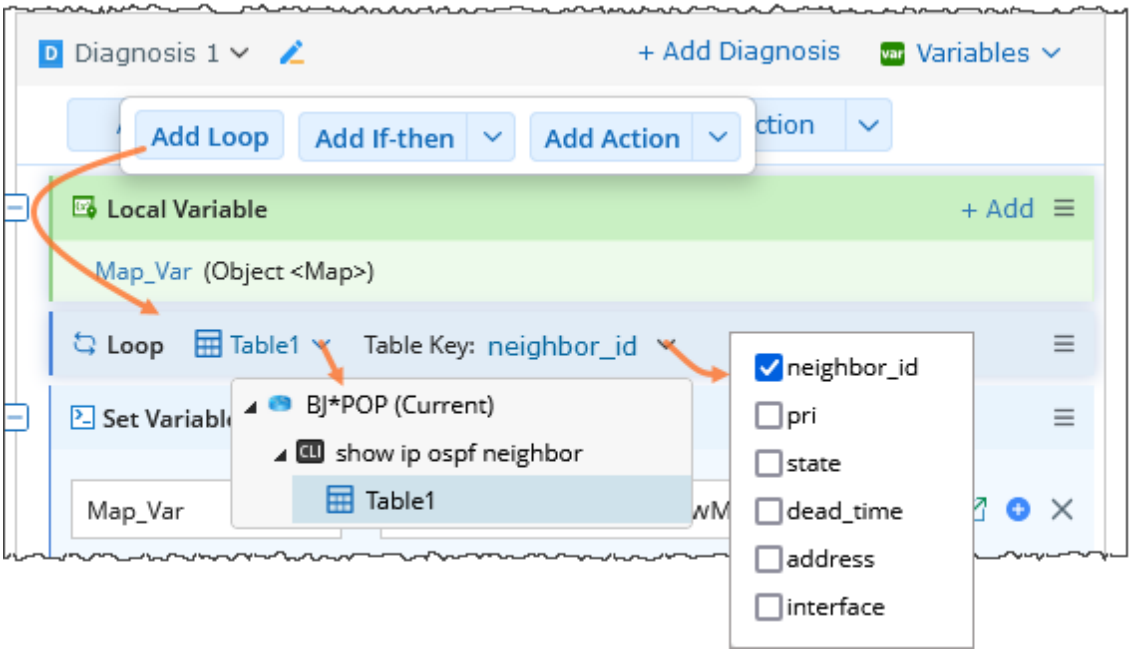
3. Navigate to **Define Diagnosis** and add a **Local Variable** for the **Map Object**.



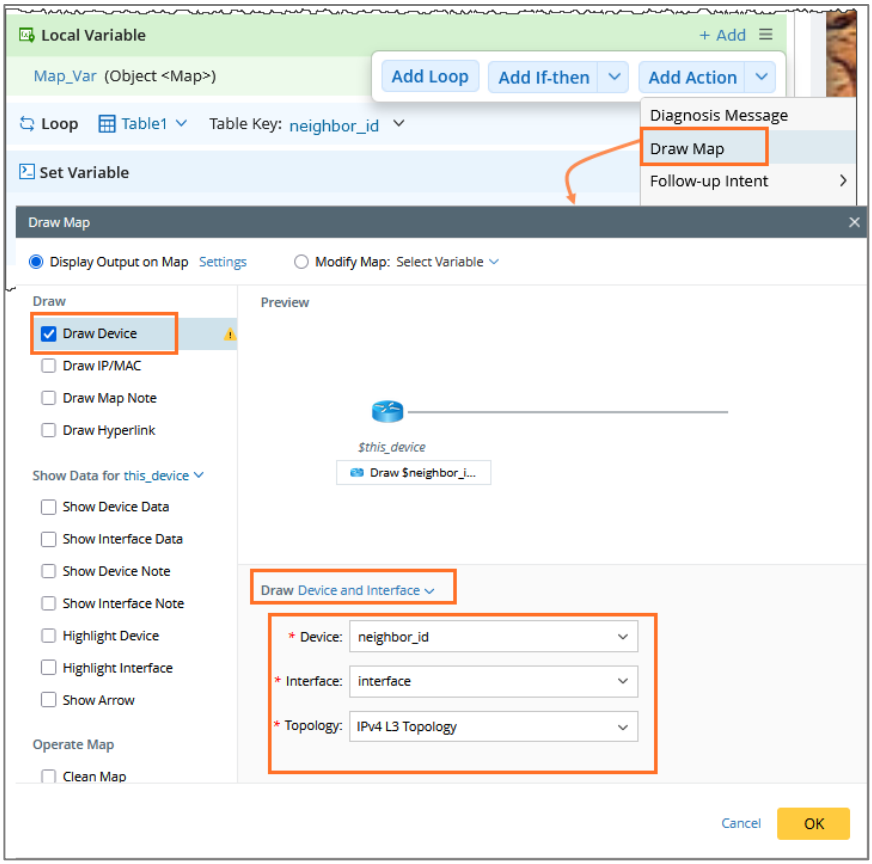
4. Hover over the newly added **Local Variable** and add a **Set Variable** using the **NewMap(Public\\testNewMap)** function.



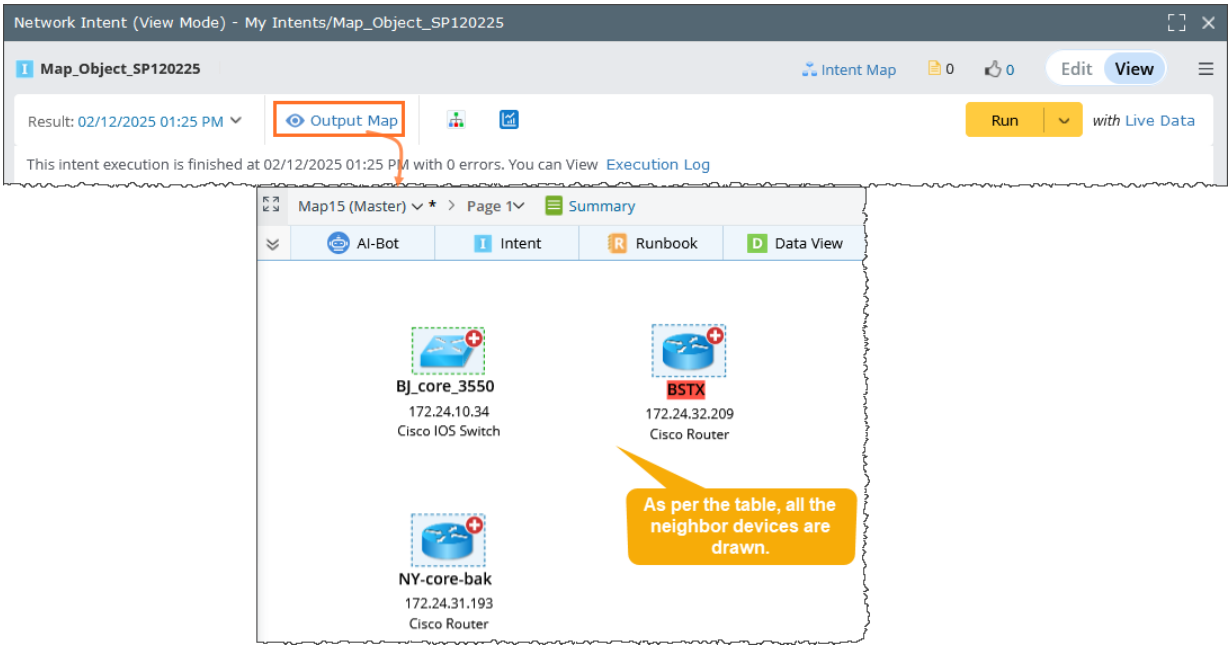
5. Hover over the **Local Variable** again and select **Add Loop** to iterate through the table and add the **Table Key**.



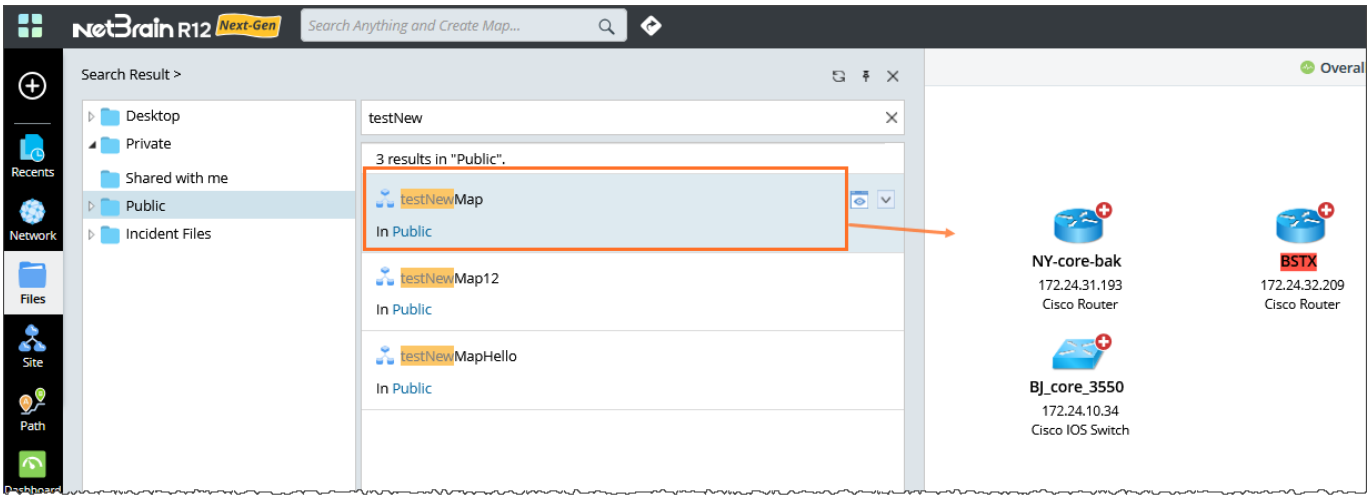
6. Hover over the **Loop**, click **Add Action**, and select **Draw Map** to generate the required map.



7. Click **Save** and Run the **Intent**.
8. Click **Output Map** to view the generated map output.



The newly created map is now available in the **Files > Public** folder for viewing.



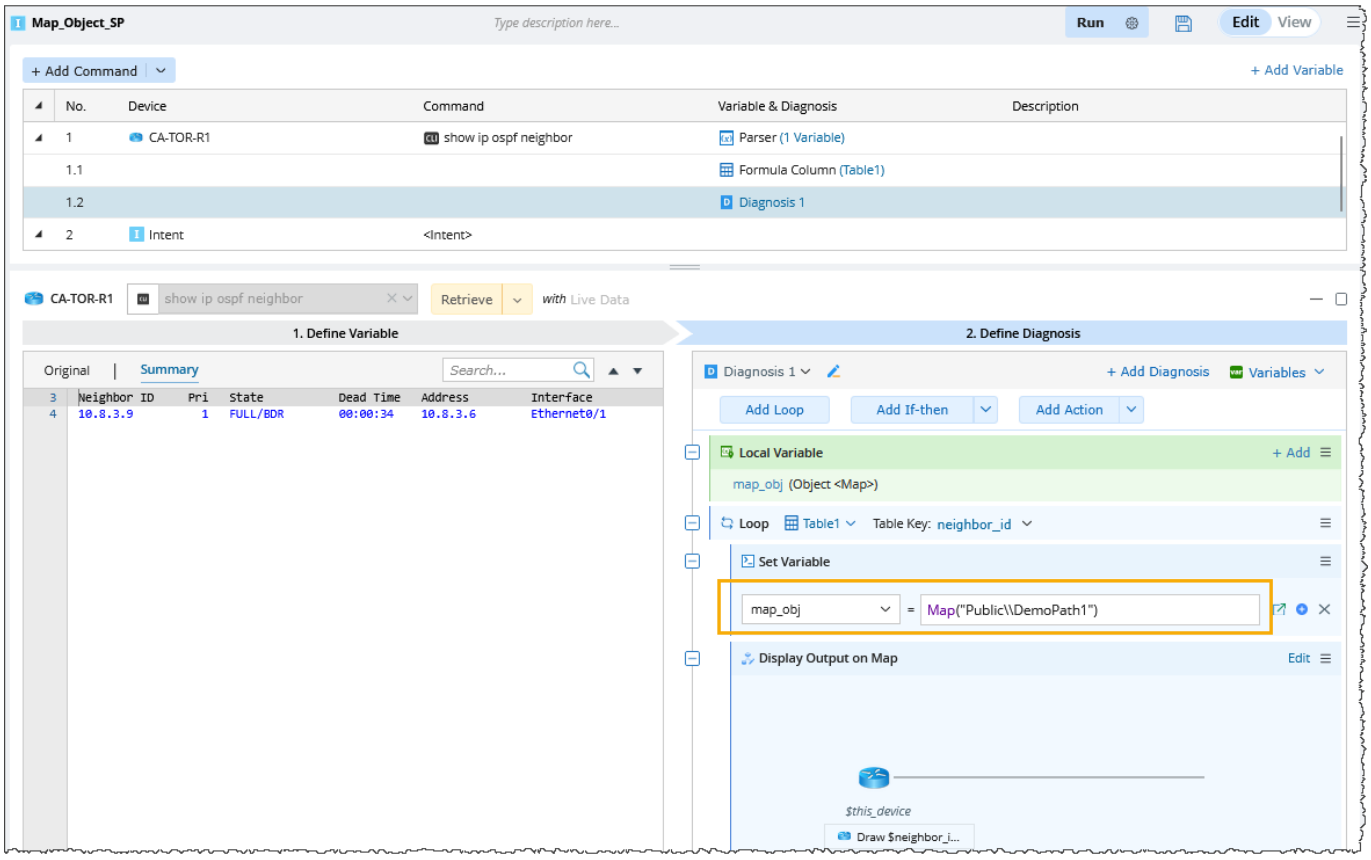
3.8.3 Map

This function open an existing map for further operations. It supports only standard maps and does not support function maps.

Follow the above example of the **NewMap** function for complete execution, including, creating Local Variables, Setting Variables, and Drawing the map.

Formula: ***Map("Public\\DemoPath1")***

Refer to the Diagnosis image which illustrates the example usage of the **Map** function.



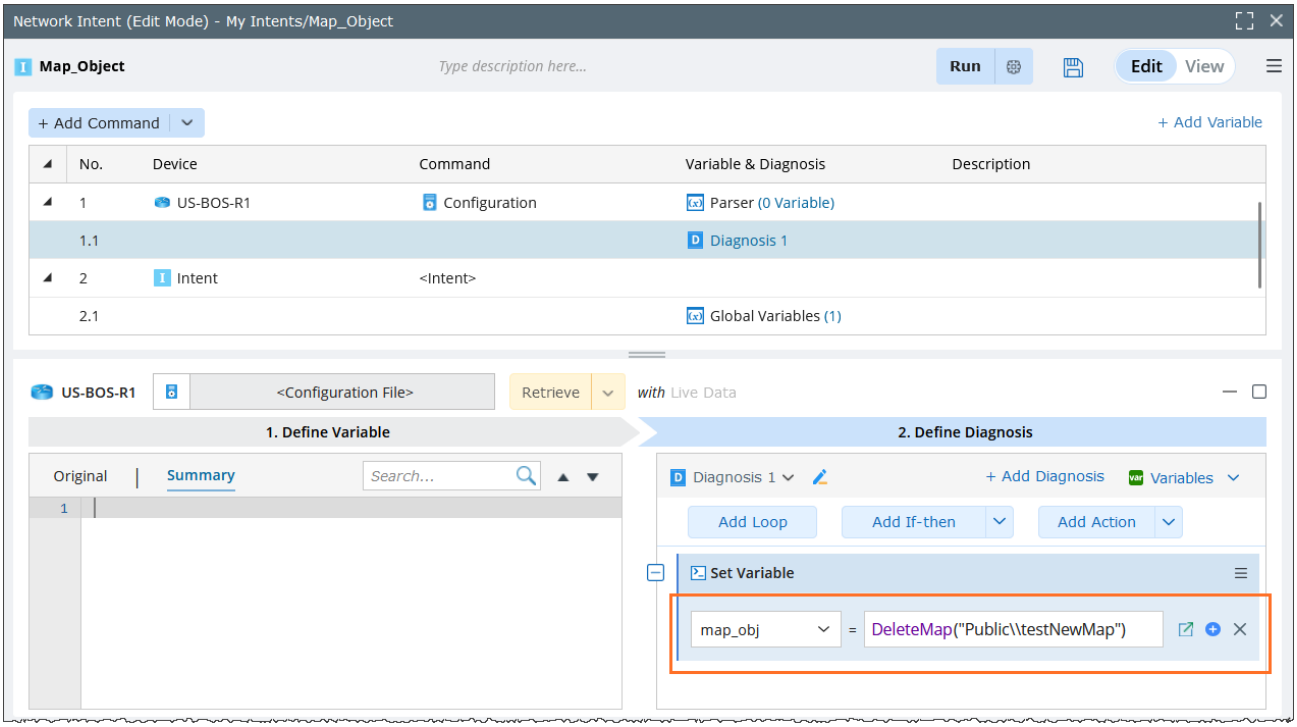
Save and **Run** the Intent, and check the **Output Map**.

3.8.4 DeleteMap

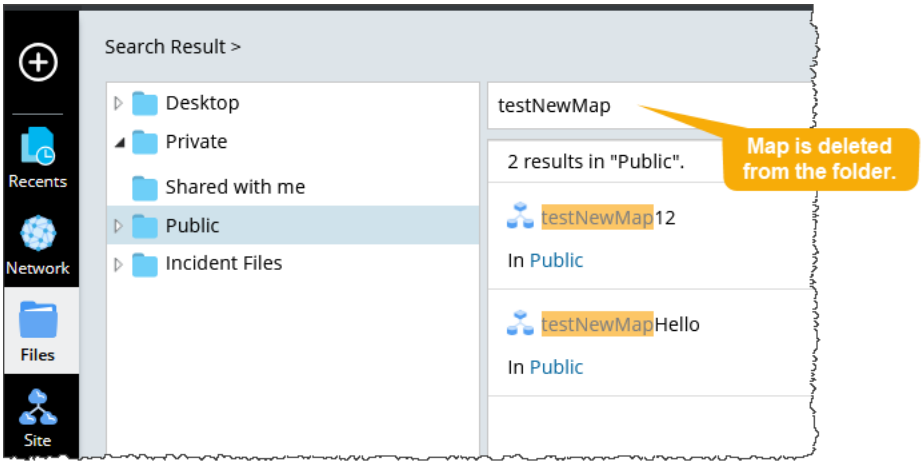
This function deletes an existing normal map specified by the full path of the map file. Note that this function does not support deleting function maps.

If you have previously created a map using the NewMap function, you can delete it or any other map by using function: ***DeleteMap("Public\\testNewMap")***

1. Refer to the image for an example of how to delete a map.

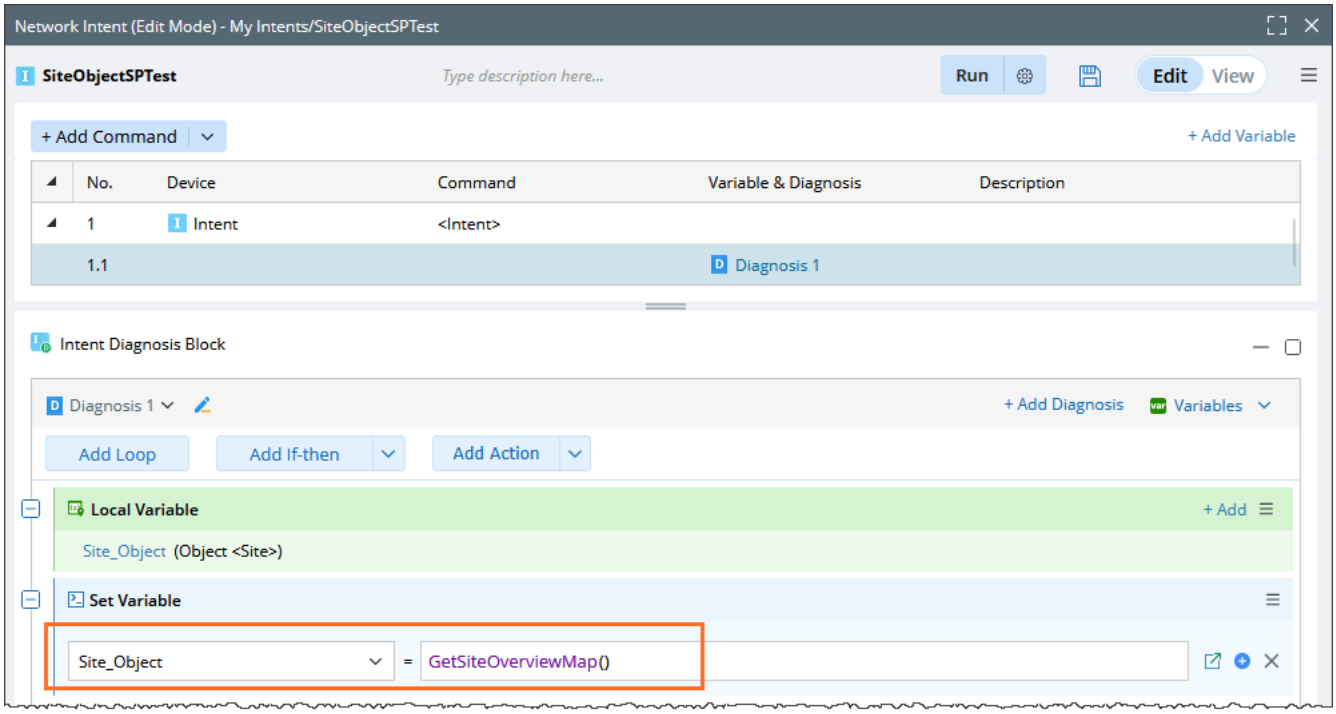


2. Click **Save** and **Run** the Intent to execute the function.
3. Navigate to **Files** > **Public** folder to verify that the map has been deleted..



3.8.5 GetSiteOverviewMap()

The **GetSiteOverviewMap()** function retrieves an interactive overview map displaying all sites, enabling users to visualize and interact with site locations.



3.9 Site Management

3.9.1 NewSite

This function generates a new site object based on the provided parameters. For more details on how to set local variables and assign values, follow the example provided in the **NewMap** function documentation.

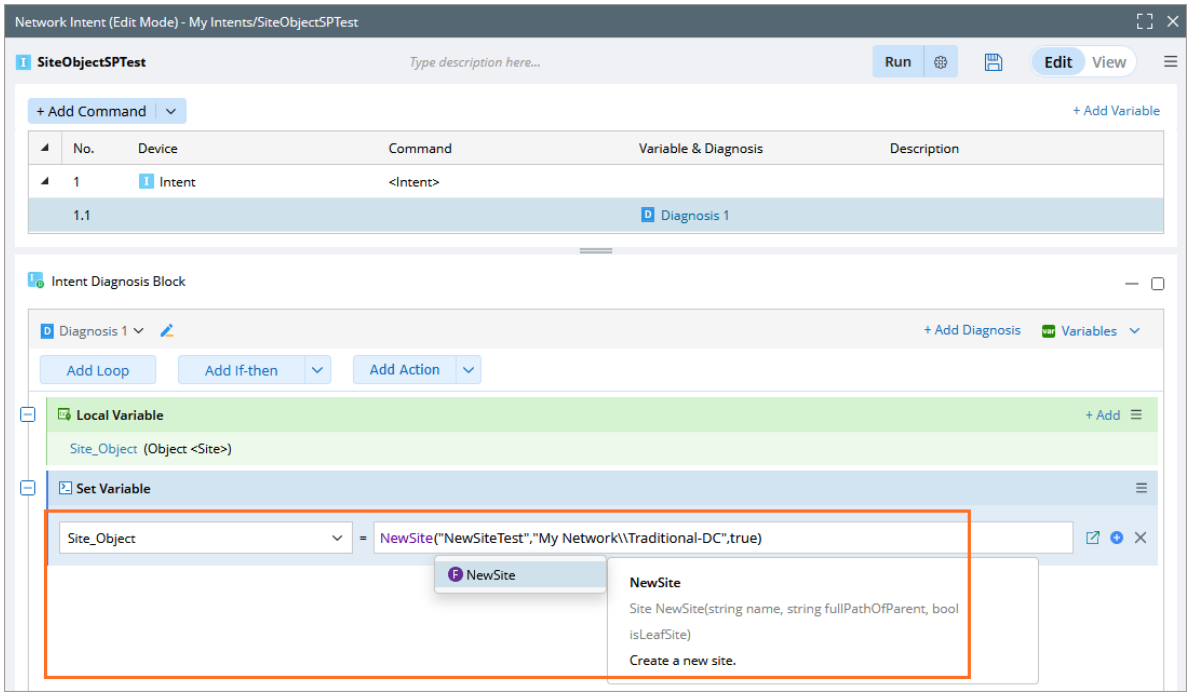
Function: ***NewSite(string name, string fullPathOfParent, bool isLeafSite***)

Parameters:

- **Name**: The name of the new site to be created. Example: **NewSiteTest**.
- **FullPathOfParent**: The full path of the parent site where the new site will be created. Example: **My Network\\Traditional-DC**.
- **IsLeafSite**: Indicates whether the new site is a leaf site (true) or a container site (false). Example: true for a leaf site.

The Final Function is: ***NewSite("NewSiteTest", "My Network\\Traditional-DC", true)***

Refer to the image for an illustration on how to generate the new site for the specified path.



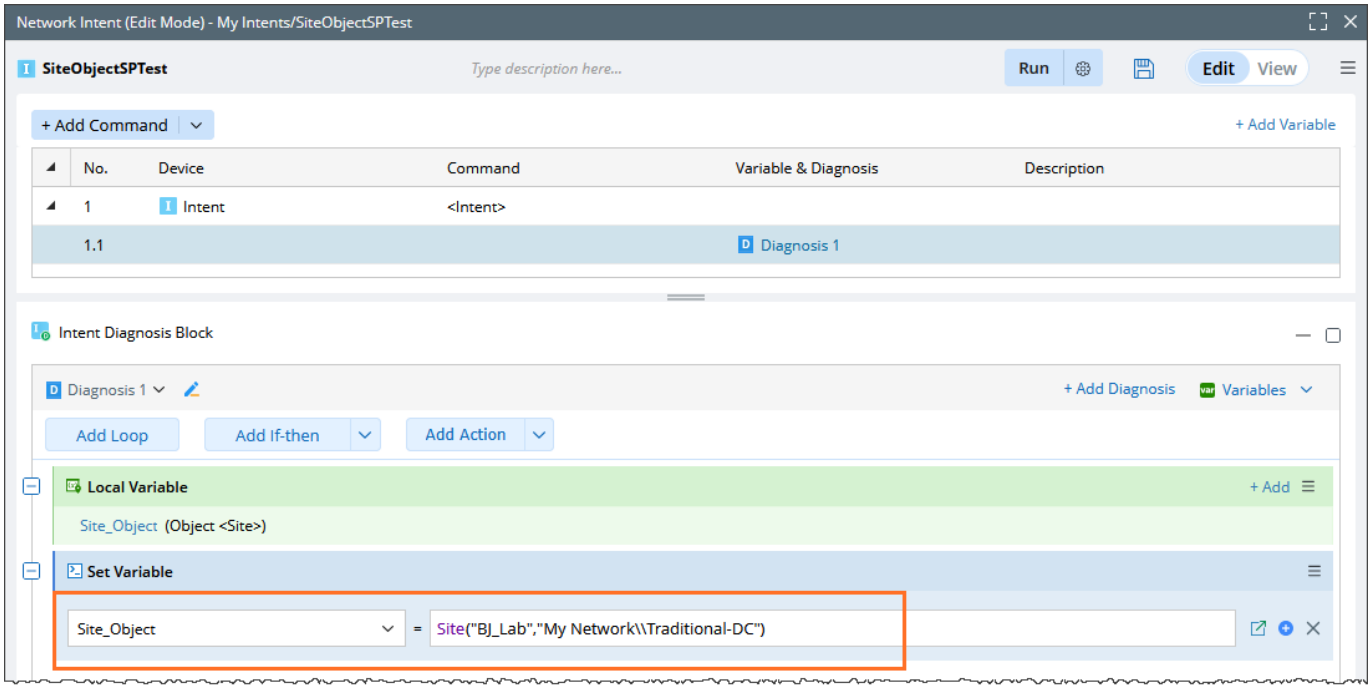
After saving and running the intent, a new site is created. Navigate to the site using the following path: **Sites > My Network > Traditional-DC > NewSiteTest.**

3.9.2 Site

This function generate a site object with an existing site for further operation.

For more details on how to set local variables and assign values, follow the example provided in the **NewSite** function documentation.

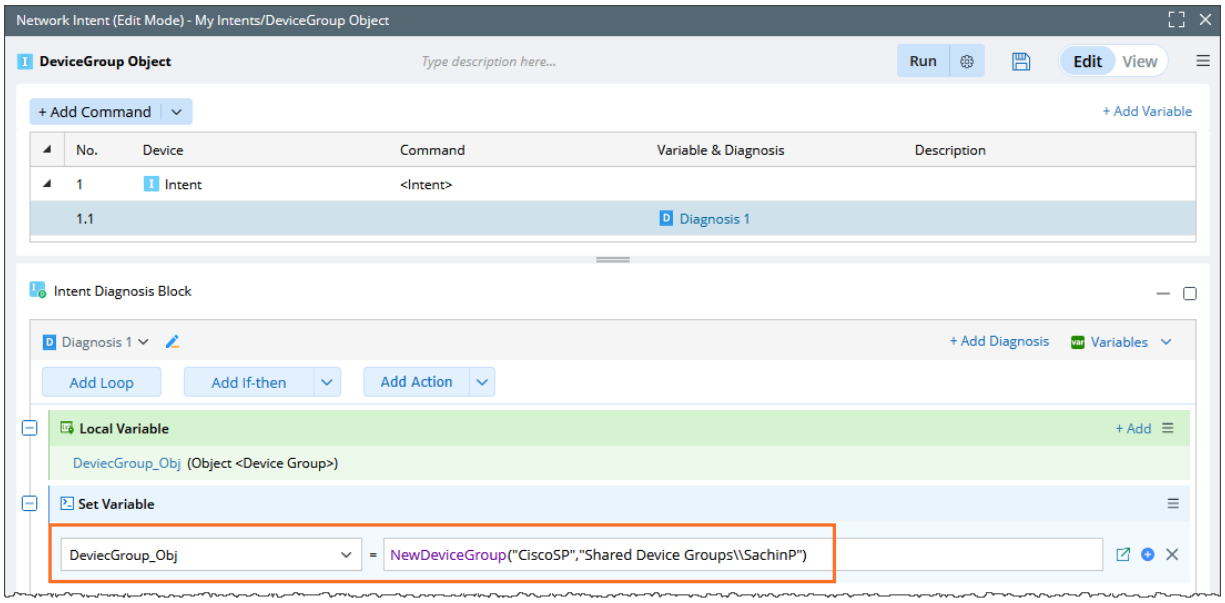
Refer to the image for an illustration on how to generate the site using existing site.



3.10 New Device Group Management

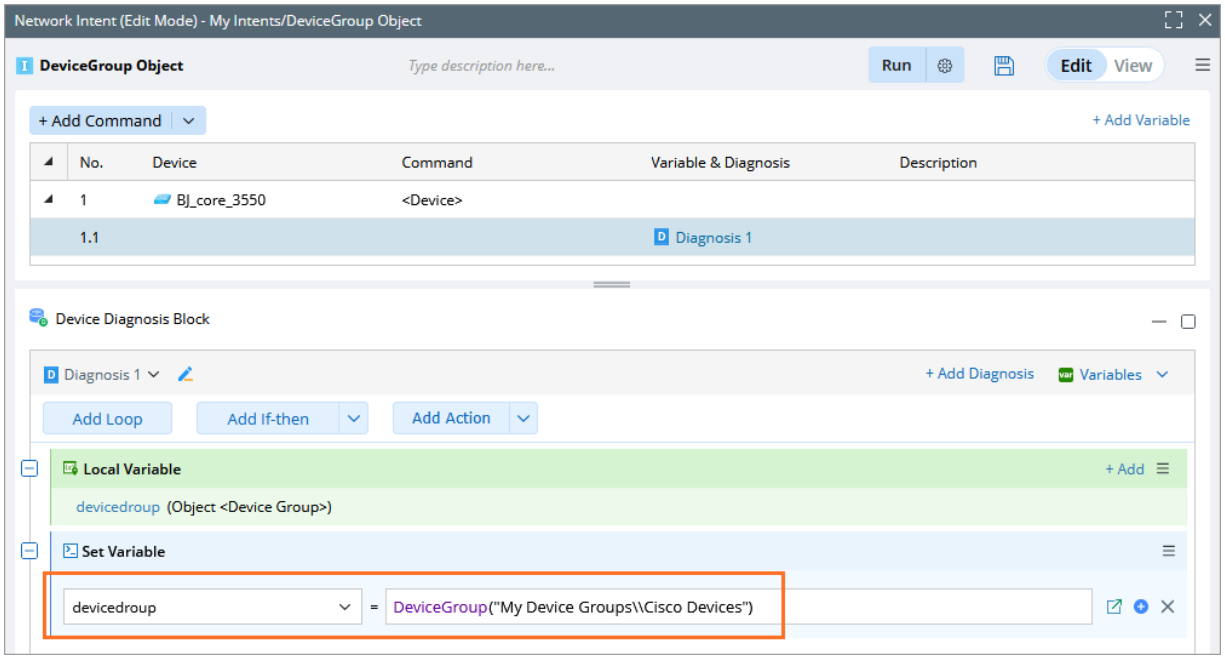
3.10.1 NewDeviceGroup

The **NewDeviceGroup** function creates a device group. Use the returned device group object to add/remove devices and perform other actions.



3.10.2 DeviceGroup

The DeviceGroup function only opens an existing device group.



3.11 File Management

3.11.1 NewFile

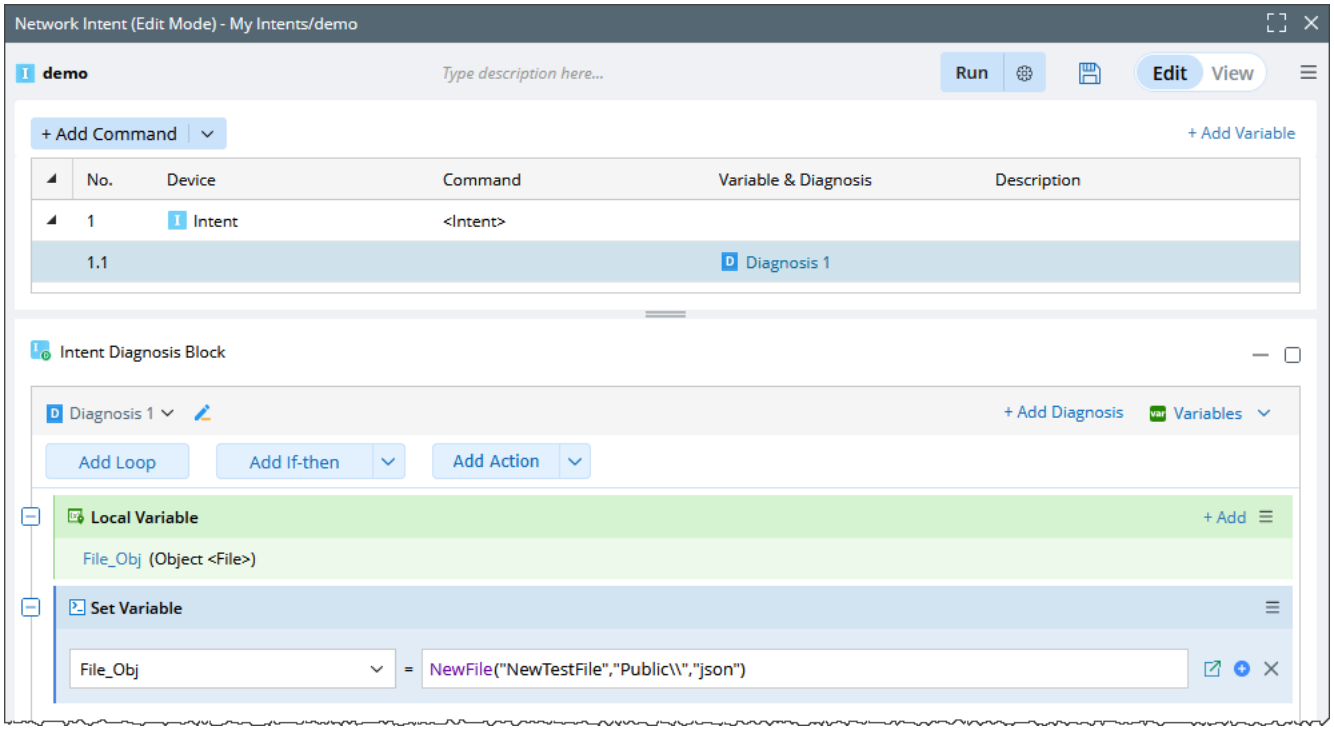
This function creates a new text file (or any given specific) within the specified NetBrain file system. For example, when you want to capture the Intent diagnosis details in the new file, you can use this function.

Parameters:

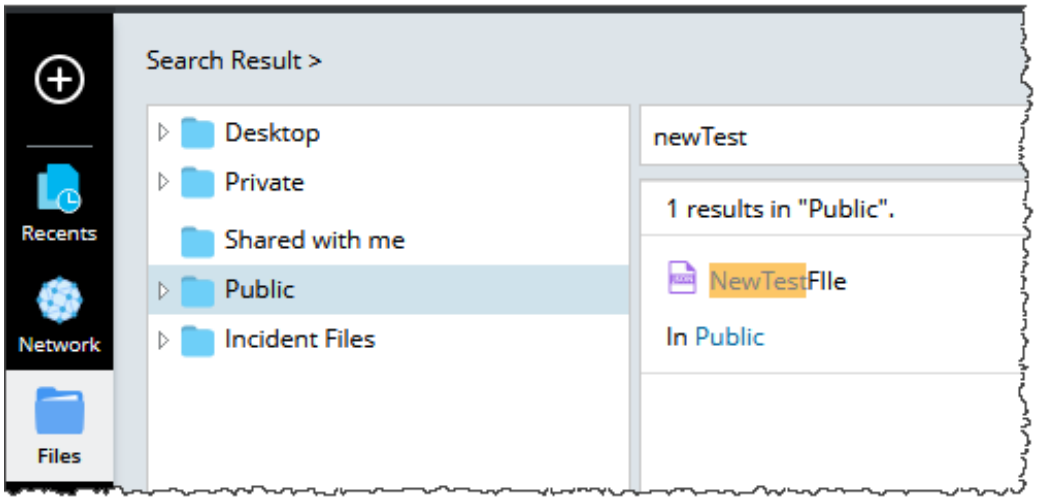
- **FileName (string):** The name of the file to be created. Note: The name should not include the file extension. **Example**: NewTest File
- **Folder (string):** The full path of the directory where the file will be created. Example: ***Public***
- **Type (string, optional)**: Specifies the file type. Valid options are "Text", "CSV", or "Json". The default value is "Text". Example: ***json***

Function: ***NewFile("NewTestFile","Public\\","json")***

Refer to the image for an illustration on how to create anew file in the Files folder.



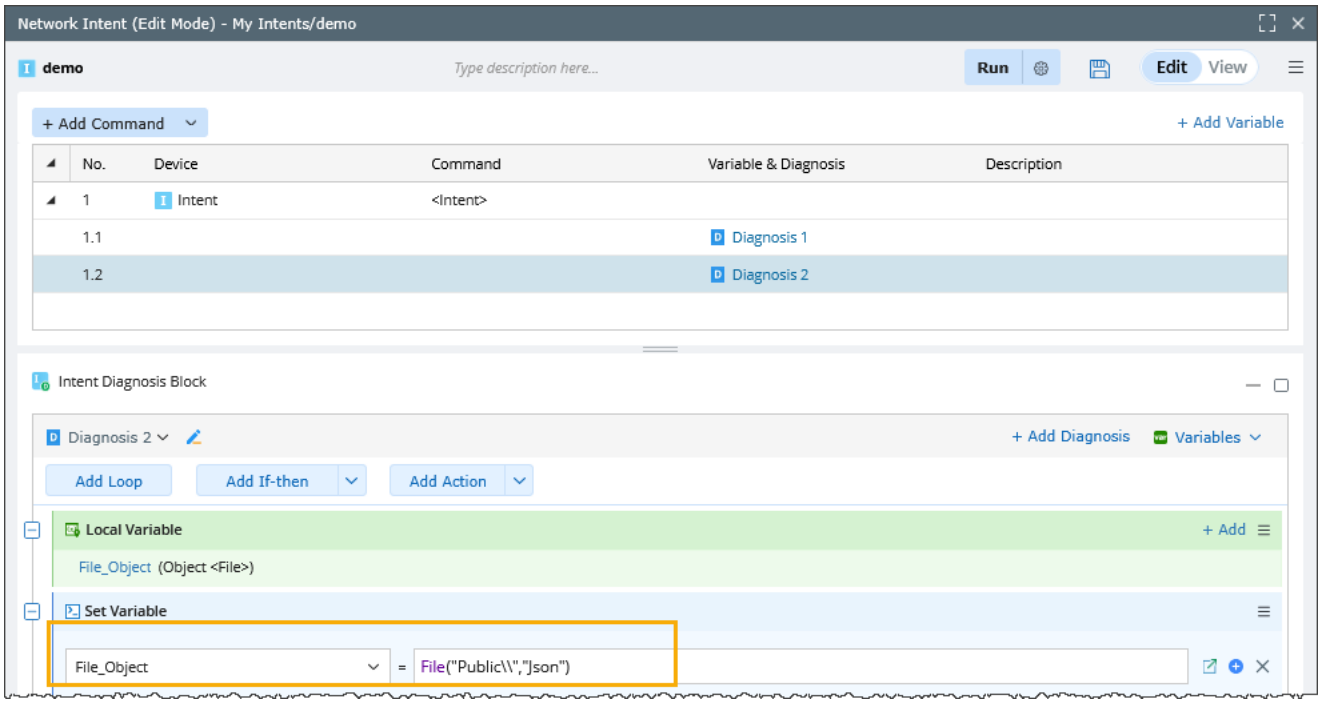
This function return the file as per specified format.



3.11.2 File

This function opens an existing text file (or any given specific) from the the specified NetBrain file system.

Refer to the example of **NewFile** function and enter the existing file path to copy the Intent details in the specified file.



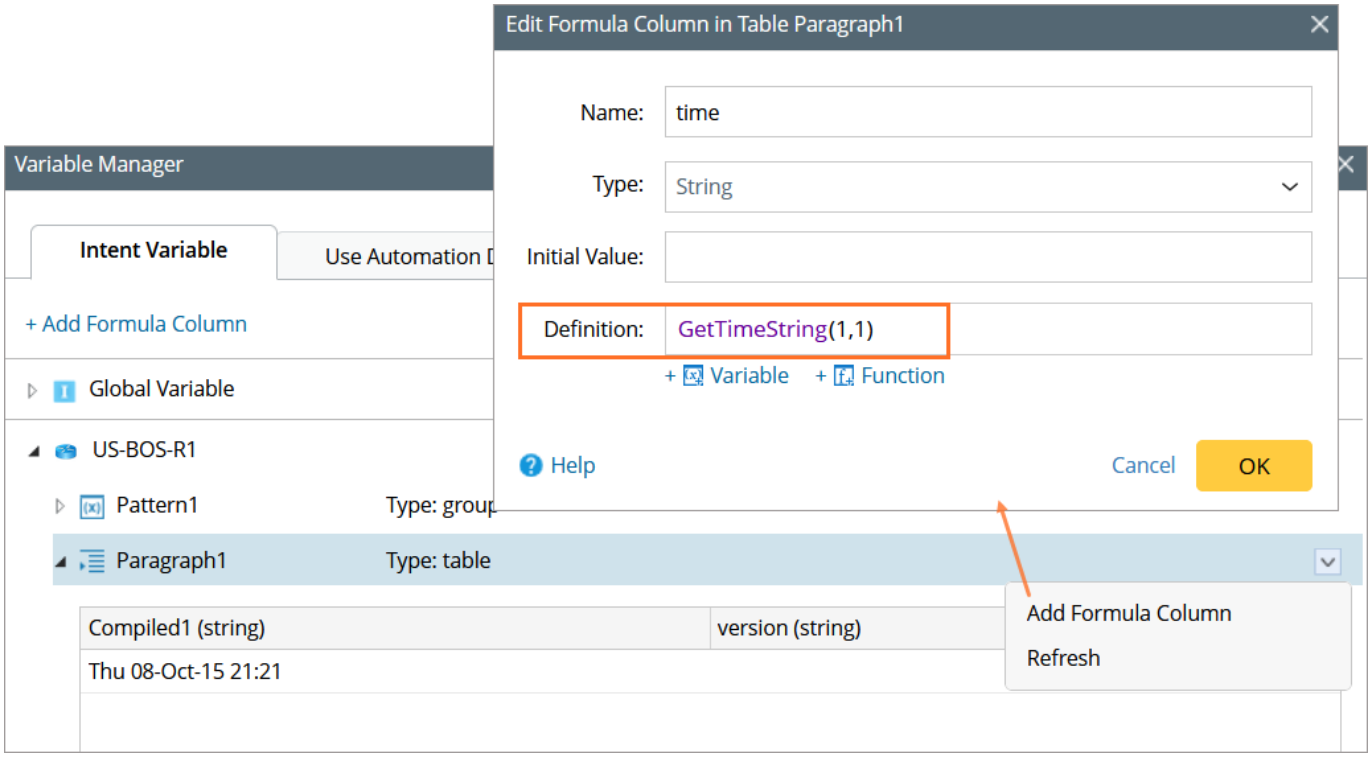
3.12 Time Utilities

3.12.1 GetTimeString

This function gets a certain time before the current time and return the time with string. If the offset value is **0**, the current time will be returned with string. The value of timeunit can be set as 0 or 1 or, 2, respectively representing min/hour/day.

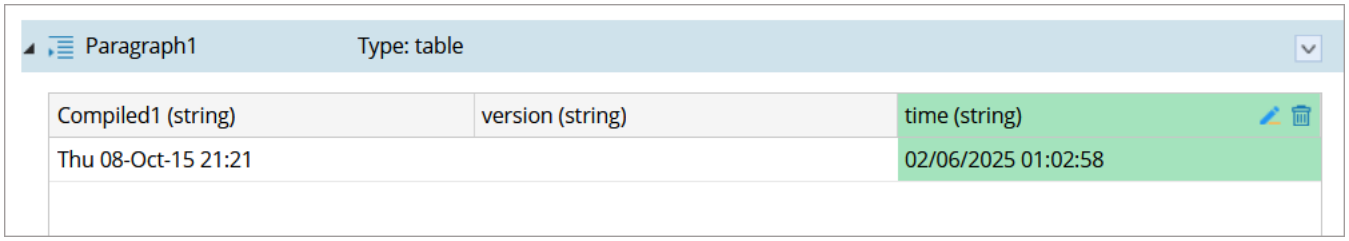
Function: ***GetTimeString(number offset, number timeunit)***

Refer to the image for an example to get the certain time before the current time based on the provided offset value and timeunit value.



Return Value:

This function returns date and time string i.e, ***02/06/2025 01:02:58***.



3.13Variable Operations

3.13.1 Delta

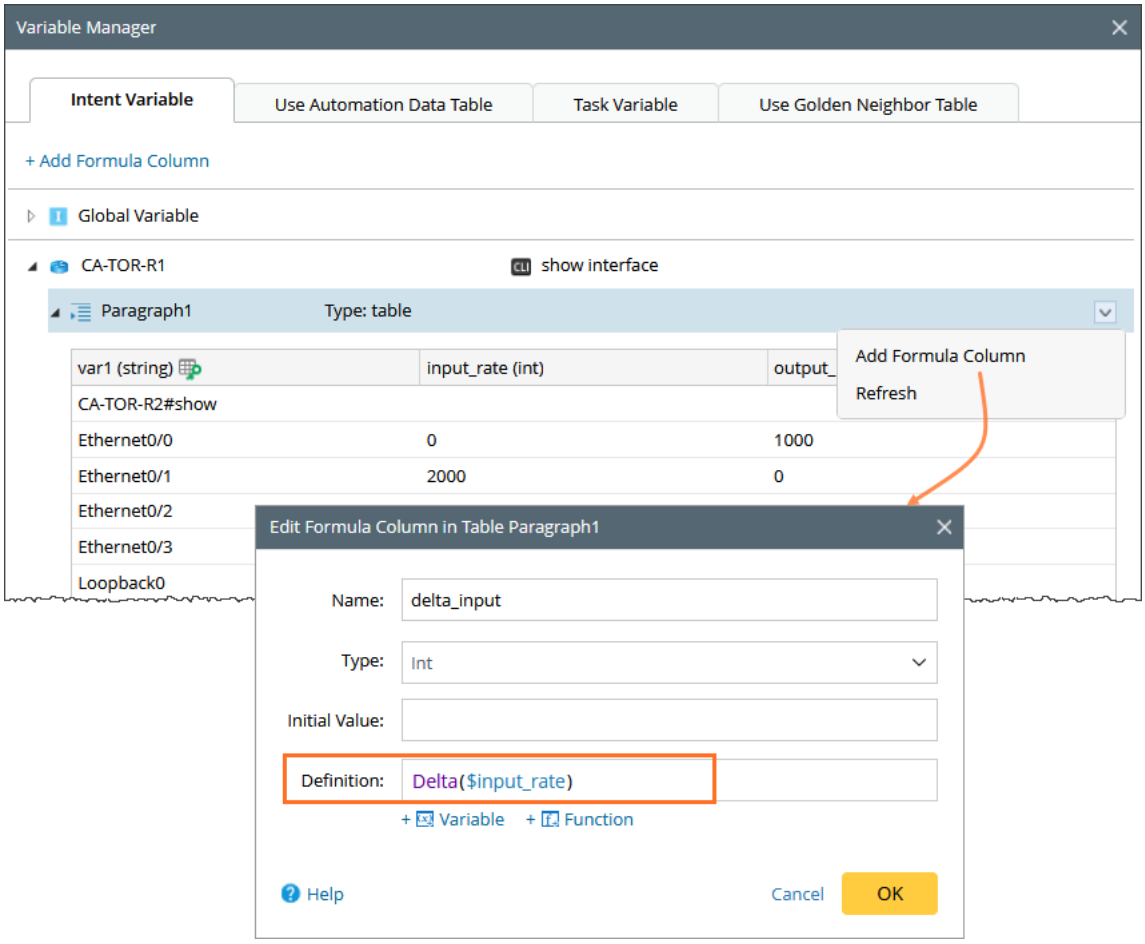
The Delta($) function calculates the difference (delta) between the current and last recorded values of a variable. This function only works for multiple executions of intent. The delta is obtained by subtracting the last value from the latest value.

For example, If the current input rate of Ethernet0/1 is **2000 bits/s**, and the last recorded input rate was **3000 bits/s**, the function calculates the delta as:

Delta = Current Value − Last Value (2000 – 3000 = − 1000)

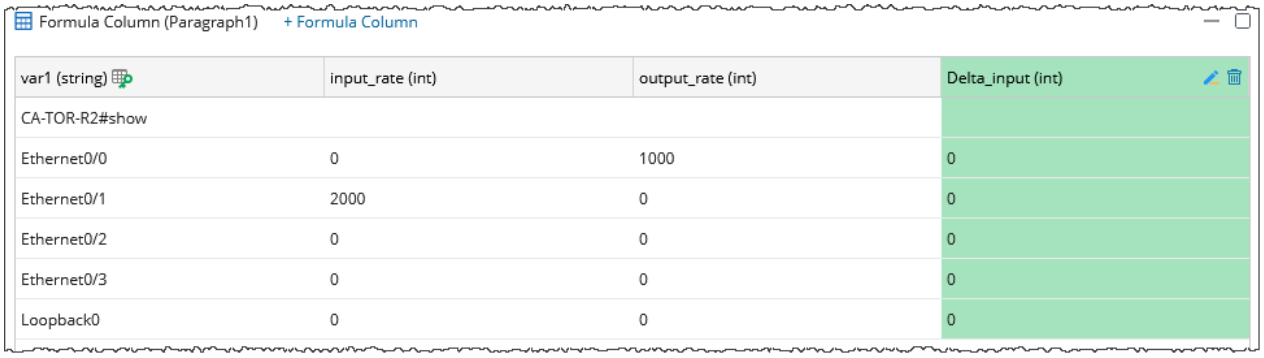
Function: ***Delta($input_rate)***

Refer to the image to observe the delta calculation when a change occurs.



Return Value:

The return value will be 0 in the added column.



var1 (string)	input_rate (int)	output_rate (int)	Delta_input (int)
CA-TOR-R2#show			
Ethernet0/0	0	1000	0
Ethernet0/1	2000	0	0
Ethernet0/2	0	0	0
Ethernet0/3	0	0	0
Loopback0	0	0	0

After printing the message, you will observe the delta between variables if there is a change in the input rate. In this case, a value of **-1000** indicates that the previous input rate was **3000 bits/s**.