

AOS-CX 10.08 Network Analytics Engine Guide

6200, 6300, 6400, 8320, 8325, 8360, 8400

Switch Series

The Aruba logo consists of the word "aruba" in a lowercase, rounded, orange sans-serif font. The letters are closely spaced, and the 'a' and 'u' have a distinctive shape with a slight curve.

a Hewlett Packard
Enterprise company

Copyright Information

© Copyright 2022 Hewlett Packard Enterprise Development LP.

Open Source Code

This product includes code licensed under the GNU General Public License, the GNU Lesser General Public License, and/or certain other open source licenses. A complete machine-readable copy of the source code corresponding to such code is available upon request. This offer is valid to anyone in receipt of this information and shall expire three years following the date of the final distribution of this product version by Hewlett Packard Enterprise Company. To obtain such source code, send a check or money order in the amount of US \$10.00 to:

Hewlett Packard Enterprise Company
6280 America Center Drive
San Jose, CA 95002
USA

Notices

The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Links to third-party websites take you outside the Hewlett Packard Enterprise website. Hewlett Packard Enterprise has no control over and is not responsible for information outside the Hewlett Packard Enterprise website.

Contents	3
About this document	7
Applicable products	7
Latest version available online	7
Command syntax notation conventions	7
About the examples	8
Identifying switch ports and interfaces	8
Identifying modular switch components	10
Aruba Network Analytics Engine Introduction	11
AOS-CX Web UI for Analytics introduction	11
Aruba Network Analytic Engine scripts introduction	11
Aruba Network Analytic agents introduction	12
Built-in scripts and agents	12
Aruba-certified scripts	13
Aruba Solutions Exchange (ASE) introduction	13
User-created scripts	13
Developer communities for the Network Analytics Engine	14
Aruba Solution Exchange (ASE)	14
GitHub	14
Airheads Community	14
Aruba Network Analytics Engine supported maximums	14
AOS-CX Network Analytics Engine (NAE) Design considerations ...	16
NAE factors that impact system resources	16
NAE deployment considerations	16
Considerations when writing an NAE agent	17
Aruba Network Analytics Engine framework	18
Configuration and state database	18
Time series database	18
AOS-CX REST API	19
Python and the REST API for scripts	19
"Sandboxes" for agent actions	20
Managing scripts	21
Managing NAE scripts across switch software updates	21
Viewing script details using the Web UI	22
Downloading a script using the Web UI	23
Uploading a script using the Web UI	23
Updating a script using the Web UI	25
Deleting a script using the Web UI	25
Getting information about scripts using the REST API	26
Uploading a script using the REST API	27
Updating a script using the REST API	27
Deleting a script using the REST API	28
Downloading a script from the switch using the REST API	28

show running-config command output for agents and scripts	29
Script status	30
Managing agents	31
Viewing a list of agents installed on a switch using the Web UI	31
Viewing agent information using the Web UI	31
Finding alert details using the Web UI	35
Working with an Analytics time series graph	37
Customizing data displayed on the graph	38
Zooming in on the graph	39
Downloading the graph as an image or .csv file	39
Viewing an alert on the graph	40
Enabling a disabled agent using the Web UI	43
Disabling an agent using the Web UI	43
Deleting an agent using the Web UI	44
Creating an agent from an existing script using the Web UI	44
Changing the configuration of an agent using the Web UI	45
Getting information about agents using the REST API	46
Creating an agent from an existing script using the REST API	47
Enabling an agent using the REST API	48
Disabling an agent using the REST API	48
Changing the configuration of an agent using the REST API	49
Deleting an agent using the REST API	50
Procedure	50
Agent status	50
Behaviors when multiple agents monitor the same resource	51
Troubleshooting agent and script issues	52
Showing the current and maximum number of agents, monitors, and scripts	52
High switch CPU and memory usage are affecting switch performance	54
Downloading NAE support files	54
NAE support file content	55
Error: "Switch time and browser time are not in sync"	55
Analytics time series graph displays message instead of data: "Agent data not found, please verify..."	56
Inaccurate or no data displayed in analytics time series graph	57
URI errors	58
Error: "The NAE Agent is not created...DB constraint violation errors"	58
Error: "The NAE Agent has Python errors."	59
Error: "Timeseries data cannot be generated...The URI is invalid or not configured"	59
Error: "The script syntax is invalid"	61
Error: "The script agent syntax is invalid"	61
Error: "Sandbox timed out while running script"	61
Error: "The agent instantiated sandbox has timed out"	62
Error: "Unable to parse condition expression..."	62
Error: "The CLI command is invalid"	63
Error: "Command failed: non-zero exit status"	64
Error: "The action is invalid"	65
ActionShell output error: "not available in enhanced secure mode"	65
Using the Aruba Solutions Exchange (ASE)	66
Finding NAE scripts on the ASE website	66
Finding NAE scripts on the ASE using the Web UI	66
Viewing recent changes to existing NAE solutions	67
Downloading or installing a script from the ASE using the Web UI	67
Downloading a solution from the ASE website to your workstation	68

NAE scripts repository on GitHub	70
Scripts and security	71
Scripts	72
Python version and library support	72
Python modules available	72
Third-party Python libraries available	74
REST API version support	74
Rules for script files	74
Script example	74
Parts of a script	76
Header	77
Import statements	78
Manifest	78
Required items	78
Optional items	78
Parameter definitions	79
ParameterDefinitions description	80
Agent class constructor	81
Graph	82
Title	83
on_agent_re_enable	84
on_agent_restart	84
on_parameter_change	85
Baselines for dynamic thresholds for monitors	85
Baseline workflow and considerations	86
Example of baselines in a time series graph	87
Example of a script that uses baselines	88
Baseline	89
ADCs	93
ADCList class	94
ADCEntry class	95
Monitors	97
URIs for monitors	98
Path component of the URI	98
Query component of the URI	99
Wildcard characters in monitored URIs	99
Parameters in monitored URIs	100
Slash (/) characters in monitored URIs	101
Attribute filters in monitored URIs	102
Constructing a URI using the AOS-CX REST API Reference	102
Aggregate operators	104
Aggregate functions	105
Nested aggregate functions and operators	107
Aggregate functions in monitors and conditions	107
Rules	108
Conditions	108
Clear conditions	109
Condition expression syntax	110
Durations, evaluation delays, and pauses in condition expressions	110
Conjunction (AND), disjunction (OR), and multiple subconditions	111
Function behavior when monitored URI does not contain wildcards	112
Function behavior when monitored URI has wildcards	113
Actions	114
Predefined actions	114

ActionCLI, CLI action	114
ActionCustomReport	116
ActionShell, SHELL action	117
ActionSyslog, Syslog action	118
Callback actions	119
Clear actions	120
Alert level functions	121
Logging functions	122
Agents	123
Agents and user authority	123
Rules for agent names	123
Network Analytics Engine commands	124
show nae-agent	124
show nae-script	125
show running-config nae	127
HTTPS server commands	129
https-server rest access-mode	129
https-server session close all	130
https-server vrf	130
show https-server	132
Support and Other Resources	134
Accessing Aruba Support	134
Accessing Updates	134
Aruba Support Portal	134
My Networking	135
Warranty Information	135
Regulatory Information	135
Documentation Feedback	135

This document describes features of the AOS-CX network operating system. It is intended for administrators responsible for installing, configuring, and managing Aruba switches on a network.

Applicable products

This document applies to the following products:

- Aruba 6200 Switch Series (JL724A, JL725A, JL726A, JL727A, JL728A)
- Aruba 6300 Switch Series (JL658A, JL659A, JL660A, JL661A, JL662A, JL663A, JL664A, JL665A, JL666A, JL667A, JL668A, JL762A)
- Aruba 6400 Switch Series (JL741A, R0X26A, R0X27A, R0X29A, R0X30A)
- Aruba 8320 Switch Series (JL479A, JL579A, JL581A)
- Aruba 8325 Switch Series (JL624A, JL625A, JL626A, JL627A)
- Aruba 8360 Switch Series (JL700A, JL701A, JL702A, JL703A, JL706A, JL707A, JL708A, JL709A, JL710A, JL711A)
- Aruba 8400 Switch Series (JL375A, JL376A)

Latest version available online

Updates to this document can occur after initial publication. For the latest versions of product documentation, see the links provided in [Support and Other Resources](#).

Command syntax notation conventions

Convention	Usage
<code>example-text</code>	Identifies commands and their options and operands, code examples, filenames, pathnames, and output displayed in a command window. Items that appear like the example text in the previous column are to be entered exactly as shown and are required unless enclosed in brackets ([]).
example-text	In code and screen examples, indicates text entered by a user.
Any of the following: <ul style="list-style-type: none">■ <code><example-text></code>■ <code><example-text></code>■ <i>example-text</i>■ <i>example-text</i>	Identifies a placeholder—such as a parameter or a variable—that you must substitute with an actual value in a command or in code: <ul style="list-style-type: none">■ For output formats where italic text cannot be displayed, variables are enclosed in angle brackets (< >). Substitute the text—including the enclosing angle brackets—with an actual value.■ For output formats where italic text can be displayed, variables might or might not be enclosed in angle brackets. Substitute the text including the enclosing angle brackets, if any, with an actual value.

Convention	Usage
	Vertical bar. A logical OR that separates multiple items from which you can choose only one. Any spaces that are on either side of the vertical bar are included for readability and are not a required part of the command syntax.
{ }	Braces. Indicates that at least one of the enclosed items is required.
[]	Brackets. Indicates that the enclosed item or items are optional.
... or ...	Ellipsis: <ul style="list-style-type: none"> ■ In code and screen examples, a vertical or horizontal ellipsis indicates an omission of information. ■ In syntax using brackets and braces, an ellipsis indicates items that can be repeated. When an item followed by ellipses is enclosed in brackets, zero or more items can be specified.

About the examples

Examples in this document are representative and might not match your particular switch or environment. The slot and port numbers in this document are for illustration only and might be unavailable on your switch.

Understanding the CLI prompts

When illustrating the prompts in the command line interface (CLI), this document uses the generic term `switch`, instead of the host name of the switch. For example:

```
switch>
```

The CLI prompt indicates the current command context. For example:

```
switch>
```

Indicates the operator command context.

```
switch#
```

Indicates the manager command context.

```
switch (CONTEXT-NAME)#
```

Indicates the configuration context for a feature. For example:

```
switch (config-if) #
```

Identifies the `interface` context.

Variable information in CLI prompts

In certain configuration contexts, the prompt may include variable information. For example, when in the VLAN configuration context, a VLAN number appears in the prompt:

```
switch (config-vlan-100) #
```

When referring to this context, this document uses the syntax:

```
switch (config-vlan-<VLAN-ID>) #
```

Where `<VLAN-ID>` is a variable representing the VLAN number.

Identifying switch ports and interfaces

Physical ports on the switch and their corresponding logical software interfaces are identified using the format:

On the 6200 Switch Series

- *member*: Member number of the switch in a Virtual Switching Framework (VSF) stack. Range: 1 to 8. The primary switch is always member 1. If the switch is not a member of a VSF stack, then member is 1.
- *slot*: Always 1. This is not a modular switch, so there are no slots.
- *port*: Physical number of a port on the switch.

For example, the logical interface 1/1/4 in software is associated with physical port 4 in slot 1 on member 1.

On the 6300 Switch Series

- *member*: Member number of the switch in a Virtual Switching Framework (VSF) stack. Range: 1 to 10. The primary switch is always member 1. If the switch is not a member of a VSF stack, then member is 1.
- *slot*: Always 1. This is not a modular switch, so there are no slots.
- *port*: Physical number of a port on the switch.

For example, the logical interface 1/1/4 in software is associated with physical port 4 on member 1.

On the 6400 Switch Series

- *member*: Always 1. VSF is not supported on this switch.
- *slot*: Specifies physical location of a module in the switch chassis.
 - Management modules are on the front of the switch in slots 1/1 and 1/2.
 - Line modules are on the front of the switch starting in slot 1/3.
- *port*: Physical number of a port on a line module.

For example, the logical interface 1/3/4 in software is associated with physical port 4 in slot 3 on member 1.

On the 83xx Switch Series

- *member*: Always 1. VSF is not supported on this switch.
- *slot*: Always 1. This is not a modular switch, so there are no slots.
- *port*: Physical number of a port on the switch.

For example, the logical interface 1/1/4 in software is associated with physical port 4 on the switch.



If using breakout cables, the port designation changes to x:y, where x is the physical port and y is the lane when split to 4 x 10G or 4 x 25G. For example, the logical interface 1/1/4:2 in software is associated with lane 2 on physical port 4 in slot 1 on member 1.

On the 8400 Switch Series

- *member*: Always 1. VSF is not supported on this switch.
- *slot*: Specifies physical location of a module in the switch chassis.
 - Management modules are on the front of the switch in slots 1/5 and 1/6.
 - Line modules are on the front of the switch in slots 1/1 through 1/4, and 1/7 through 1/10.
- *port*: Physical number of a port on a line module

For example, the logical interface 1/1/4 in software is associated with physical port 4 in slot 1 on member 1.

Identifying modular switch components

- Power supplies are on the front of the switch behind the bezel above the management modules. Power supplies are labeled in software in the format: *member/power supply*:
 - *member*: 1.
 - *power supply*: 1 to 4.
- Fans are on the rear of the switch and are labeled in software as: *member/tray/fan*:
 - *member*: 1.
 - *tray*: 1 to 4.
 - *fan*: 1 to 4.
- Fabric modules are not labeled on the switch but are labeled in software in the format: *member/module*:
 - *member*: 1.
 - *member*: 1 or 2.
- The display module on the rear of the switch is not labeled with a member or slot number.

The Aruba Network Analytics Engine is a first-of-its-kind built-in framework for network assurance and remediation. Combining the full automation and deep visibility capabilities of the AOS-CX operating system, this unique framework enables monitoring, collecting network data, evaluating conditions, and taking corrective actions through simple scripting agents.

This engine is integrated with the AOS-CX system configuration and time series databases, enabling you to examine historical trends and predict future problems due to scale, security, and performance bottlenecks. With that information, you can create software modules that automatically detect such issues and take appropriate actions.

With the faster network insights and automation provided by the Aruba Network Analytics Engine, you can reduce the time spent on manual tasks and address current and future demands driven by Mobility and IoT.

AOS-CX Web UI for Analytics introduction

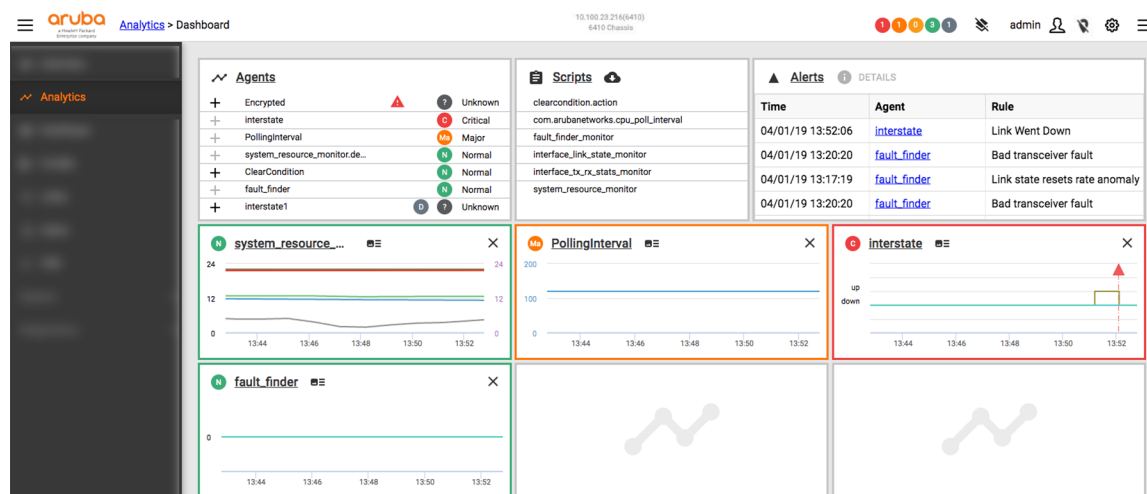
The AOS-CX Web UI provides access to information related to Aruba Network Analytics Engine agents, scripts, and alerts on the switch, including time-series data graphs and other information generated by the enabled agents.

From the AOS-CX Web UI, select **Analytics** in the navigation pane to view information in the Analytics Dashboard. You can display time-series graph panels for up to nine agents on the dashboard. However, many more agents can be enabled on a switch.

From the Analytics Dashboard, you can open Analytics detail pages. Analytics detail pages allow you to enable, disable, create, delete, or edit agents, upload scripts, and view detailed data about monitored resources. Administrator rights are required for actions that modify an agent.

For more information about the Web UI, see the *Introduction to the Web UI Guide*.

Figure 1 Analytics Dashboard in the Web UI



Aruba Network Analytic Engine scripts introduction

An Aruba Network Analytics Engine script is a Python script that defines which switch resources to monitor and, optionally, rules that define what actions to take when certain conditions are true.

Examples of tasks a script can define include the following:

- Monitoring average CPU usage and sending a system log message when the CPU usage is greater than 70% for 5 minutes.
- Monitoring the connection state of a particular BGP neighbor and executing a CLI command when the connection state transitions from UP to DOWN.

A script is used to create an agent, which is a specifically-configured executable instance of a script on a switch. A single script can be the template for many different agents.

Aruba Network Analytic agents introduction

An Aruba Network Analytics Engine agent is a specifically-configured executable instance of an NAE script on a switch. When the agent is enabled, it performs the tasks defined by the script. Agents have administrator rights.

For scripts that provide configuration parameters, different agents can be configured to use specific parameter values when performing the tasks defined by the script.

For example, a script that monitors the connection state of a particular BGP neighbor can define a parameter to specify which BGP neighbor to monitor. From that script, you can do the following:

- You can create an agent to monitor a specific BGP neighbor by specifying that BGP neighbor as the value of the parameter.
- You can create multiple agents—one for each BGP neighbor you want to monitor.
- When a new BGP neighbor is added, you can create an agent to monitor that neighbor without having to change the script itself.

Some parameters that are integers—such as a CPU utilization threshold—can be changed for a given agent after that agent has been created. Network administrators can change such parameters easily through the Web UI—no programming skill is required.

Built-in scripts and agents

Built-in scripts and agents are installed on the switch before the switch is shipped from the factory. All scripts and agents have information about their origin associated with them. Built-in scripts and agents have an origin of "system" and are marked `System Created` in the Web UI.

Built-in scripts and agents cannot be deleted. You can enable, disable, and change the configuration of built-in agents.

If you create an additional agent from a built-in script, that agent is considered a user-created agent, which can be deleted.

In the output of the `show running-config` command:

- Built-in scripts are not displayed
- Built-in agents are displayed only if one of more parameters has been changed and saved.

The current software release includes a single built-in script and agent that monitors several system resources:

- Built-in script: `system_resource_monitor`
- Built-in agent: `system_resource_monitor.default`

Aruba-certified scripts

Aruba-certified Network Analytics Engine scripts are written and tested by Aruba.

On the Aruba Solution Exchange, Aruba-certified scripts have the following tag: `nae-aruba-certified`

Figure 1 Script tags as shown on the Aruba Solution Exchange

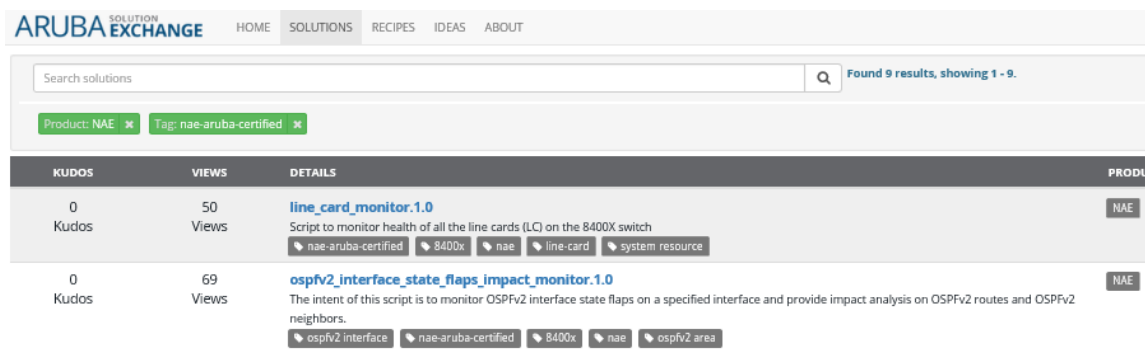


Figure 2 Script tags as shown in the Web UI

Installed	Name	Tags	Last Modified
<input checked="" type="checkbox"/>	daemon_resource_monitor.1.0	nae-aruba-certified, 8400x, nae, daemon, system resource	10/17/17 08:57:21
<input checked="" type="checkbox"/>	interface_link_flap_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, link-flaps, port	10/17/17 08:56:46
<input checked="" type="checkbox"/>	interface_link_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:57:00
<input checked="" type="checkbox"/>	interface_state_stats_monitor.1.0	interface, nae-aruba-certified, nae, 8400x, link, tx, rx, port	10/17/17 08:51:44
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_impact_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:57:53
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:54:27
<input checked="" type="checkbox"/>	port_admin_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:56:21
<input type="checkbox"/>	stp_bpdu_tcn_rate_monitor.1.0	8400x, nae, nae-aruba-certified, stp, spanning-tree, tcn, topology-cha	10/17/17 08:56:03

Aruba Solutions Exchange (ASE) introduction

Aruba Solution Exchange (ASE) is a utility that allows users to dynamically create configurations for many Aruba products. The exchange offers a set of solutions certified by Aruba for specific use cases.

The collective knowledge of the Aruba community powers the Solution Exchange. All users are encouraged to create solutions, modify existing ones, and suggest ideas for new solutions.

Access to the Aruba Solution Exchange is available at:

<https://ase.arubanetworks.com/>

No login is required to browse the exchange or look at information about solutions. Downloading solutions or viewing solution source code is free for anyone with a valid user account with the Aruba SSO authentication system. If you do not have an existing account, register for a free Airheads Social account at:

<http://community.arubanetworks.com>

For more information about using the Aruba Solution Exchange, see <https://ase.arubanetworks.com/docs>

User-created scripts

The Aruba Network Analytics Engine is highly extensible through the use of scripts. You can create scripts or find scripts created by other developers in the community to solve specific network issues.

Network Analytic Engine scripts must be written in Python. Although Python programming is beyond the scope of this document, this document provides information about the structure and functions of Network Analytics Engine scripts.

For more information about Network Analytics Engine scripts, see [Scripts](#).

Developer communities for the Network Analytics Engine

There are several communities that facilitate the development, use, and sharing of scripts for the Aruba Network Analytics Engine:

Aruba Solution Exchange (ASE)

The Aruba Solution Exchange (ASE) is the primary portal for Network Analytic Engine scripts. Aruba-certified public scripts are posted to ASE. Developers can create their own Network Analytics Engine solutions and post them for either private or public use. See the Aruba Solution Exchange at:

<https://ase.arubanetworks.com/>

GitHub

The `NAE_scripts` repository on GitHub includes Aruba-certified Network Analytics Engine (NAE) scripts and examples organized by feature or protocol and then by supported hardware platform. Developers can fork and customize or enhance these scripts for their own use. See the repository at:

<https://github.com/aruba/nae-scripts>

Airheads Community

The Airheads community provides a place for members or participants to search for information, read and post about topics of interest, and learn from each other. Guests (unregistered visitors) can browse or search the community for information. Members (registered users) can post messages or comments, track discussions, and get email notifications on posting activity and other community actions.

Within the Airheads community, there is a Developer Community group that is specific to APIs, programming, and automation. The Developer Community is the recommended place to post questions about the Aruba Network Analytics Engine (NAE).

See the Developer community at:

<https://community.arubanetworks.com/t5/Developer-Community/bd-p/DeveloperCommunity>

See the Airheads community at <http://community.arubanetworks.com/>

Aruba Network Analytics Engine supported maximums

8400 Aruba Switch Series

- Maximum number of scripts per switch: 50
- Maximum number of agents per switch: 100
- Maximum number of monitors per switch: 300
- Maximum script file size: 256 KB
- Number of days of time-series data to store: 400
- Amount of switch storage allocated to time-series data: 18 GB

Aruba 8320 and Aruba 8325 Switch Series

- Maximum number of scripts per switch: 25
- Maximum number of agents per switch: 50
- Maximum number of monitors per switch: 150
- Maximum script file size: 256 KB
- Number of days of time-series data to store: 400
- Amount of switch storage allocated to time-series data: 9 GB

Aruba 6300 and Aruba 6400 Switch Series

- Maximum number of scripts per switch: 10
- Maximum number of agents per switch: 10
- Maximum number of monitors per switch: 130
- Maximum script file size: 256 KB
- Number of days of time-series data to store: 45
- Amount of switch storage allocated to time-series data: 3.1 GB

Aruba 6200 Switch Series

- Maximum number of scripts per switch: 10
- Maximum number of agents per switch: 10
- Maximum number of monitors per switch: 130
- Maximum script file size: 256 KB
- Number of days of time-series data to store: 15
- Amount of switch storage allocated to time-series data: 1 GB

The Network Analytics Engine (NAE) feature within AOS-CX switches is a framework for automating the detection of issues and automating root cause analysis. This document covers design considerations when deploying NAE agents within a network.

NAE factors that impact system resources

There are several factors that need to be considered when deploying NAE within a network:

- **CPU and memory:** The number of agents and what the agents do within their callbacks can impact the CPU and memory of the system. NAE agents are not running at all times and will only run when one of their conditions are met for, at most, 30 seconds. Agents will only consume, at most, 10% of CPU and 64 MB of memory as well while running.
- **Wildcards within monitors:** The use of wildcards within an NAE monitor can increase the amount of resource consumption. Wildcards allow the agent to monitor an entire set of resources, but the number of resources covered can be a large amount. For example, monitoring the `tx_dropped` statistics for all interfaces will result in a metric for each interface:

```
/rest/v1/system/interfaces/*?attributes=statistics.tx_dropped
```

Each NAE metric results in a unique time series for this statistic. In this particular case, a switch with 500 interfaces equates to 500 time series metrics. In order to provide a rich history of each time series metric, all NAE time series data is written to persistent storage. As noted within the manual, time series data is collected and stored every 5 seconds.

- **Alerts:** Every ActionShell and ActionCLI output is written to persistent storage to provide a history of events encountered by an NAE agent. This is useful for troubleshooting events that occurred in the past and collecting relevant details associated with when those events occurred. These actions consume storage resources. Also, the amount of data collected by these commands must be considered.
- **ADC's:** Application Data Collections (ADCs) are used to monitor traffic rates which can be utilized within an NAE agent, but consume hardware resources used by ACLs, classifier policies, and other traffic identifiers and filters. Hardware resources are limited based on platform. Please refer to the "show resources" CLI command for a breakdown of hardware resource consumption.

NAE deployment considerations

It is highly recommended that any agent be thoroughly tested and vetted on a single or limited number switches within the network prior to a wider deployment of such agent. NAE agents published by Aruba have been tested for many customer deployments, but do not cover all cases. Customer networks can be unique and may exhibit differences compared to a controlled lab environment.

All new NAE agents deployed within a customer network should be initially monitored to ensure that it is performing according to expectations. The switch CPU and memory utilization can be reviewed through the built-in System Resource Monitor NAE agent which provides the current and running history of resource consumption. Storage usage can be monitored by reviewing the endurance utilization under the `show system resource-utilization` command over time.

There have been instances where the initial deployment of an NAE agent within a customer network helped identify network oddity or misconfiguration from the start. The initial review of any NAE alerts issued by this agent would quickly point out these misconfigurations. After making the necessary switch or network changes, any future alerts would be reduced. As an example, a high number of RX drops may indicate a bad fiber cable. Replacement of this cable would then result in less or zero RX drops and the NAE agent will stop issuing alerts.

NAE agents are published through the Aruba Solution Exchange ([ASE](#)) and, much like switch firmware, can be updated at any time. It is recommended that customers periodically check the ASE website for any updates to NAE agents that are used within their network.

Considerations when writing an NAE agent

Aruba publishes agents which take the design recommendations covered in this document into consideration, but NAE agents can be written by 3rd party developers and customers. As a result, care should be taken when writing an NAE agent. Areas to consider include, and are not limited to:

- The use of wildcards should be used sparingly in order to reduce the memory and storage used by the agent.
- Alerts should not be triggered frequently in order to reduce the wear of the storage device.
- The thresholds on conditions shouldn't trigger often in order to reduce the number of alerts and CPU time needed to act on the conditions.
- Any outbound network connections can only access the default VRF.

The AOS-CX operating system has been built for programmability. With the database-centric design and a programmatic interface to the entire database schema, network operators have access to every network function and state within the switch.

Central to this design is the Aruba Network Analytics Engine framework, which includes the following:

- Full integration with the AOS-CX network operating system for logging events, maintaining high availability during switch failover events, and interacting through the Web UI, REST API, and CLI.
- A REST API that can access every switch resource and state.
- Scripts and agents that can be programmed not only to monitor switch resources, but also automatically execute actions when certain conditions are met over time.
- A built-in Python interpreter.
- A Python sandbox that exists only while an agent executes an action. The sandbox has full access to the internal database and the time series database.

Configuration and state database

The AOS-CX operating system is a modular, database-centric operating system. Every aspect of the switch configuration and state information is modeled in the AOS-CX switch database and is accessible through the REST API.

Developers can access a specific configuration, statistic, or status result for any aspect of the switch through its REST URI. Switches that are members of a stack are treated as a single switch.

For example:

- The following URI accesses the link state information about interface 1/1/5:
`/rest/v1/system/ports/1%2F1%2F5?attributes=link_state`
- The following URI accesses the link state of all interfaces on the switch:
`/rest/v1/system/interfaces/*?attributes=link_state`
- The following URI accesses the number of received packets on interface 1/1/5:
`/rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.rx_packets`
- The following URI accesses the link state information about interface 2/1/5 (stack member 2, slot 1, port 5):
`/rest/v1/system/ports/2%2F1%2F5?attributes=link_state`

Time series database

The Aruba Network Analytics Engine includes a built-in time series database. Time-series data about the resources monitored by agents are automatically collected and presented in graphs in the switch Web UI. The database makes the data seamlessly available to agents that use rules that evaluate network conditions over time.

Old time-series data is removed automatically either as the storage space on the switch is used, or as the maximum number of days of data is reached. The amount of storage space consumed at any given time

depends on the number of switch resources being monitored at that time. Each monitored resource creates one time series. Each time series consumes approximately 240 KB of storage for each day.

When creating a script, software developers do not interact with this database directly. Use of the database is automatically handled by the `Monitor` and `Rule` functions.

AOS-CX REST API

Switches running the AOS-CX software are fully programmable with a REST (REpresentational State Transfer) API, allowing easy integration with other devices both on premises and in the cloud. This programmability—combined with the Aruba Network Analytics Engine—accelerates network administrator understanding of and response to network issues.

The AOS-CX REST API enables programmatic access to the AOS-CX configuration and state database at the heart of the switch. By using a structured model, changes to the content and formatting of the CLI output do not affect the programs you write. And because the configuration is stored in a structured database instead of a text file, rolling back changes is easier than ever, thus dramatically reducing a risk of downtime and performance issues.

The AOS-CX REST API is a web service that performs operations on switch resources using HTTPS `POST`, `GET`, `PUT`, and `DELETE` methods.

A switch resource is indicated by its Uniform Resource Identifier (URI). A URI can be made up of several components, including the host name or IP address, port number, the path, and an optional query string. The AOS-CX operating system includes the AOS-CX REST API Reference, which is a web interface based on the Swagger UI. The AOS-CX REST API Reference provides the reference documentation for the REST API, including resources URIs, models, methods, and errors. The AOS-CX REST API Reference shows most of the supported read and write methods for all switch resources.

Python and the REST API for scripts

Network Analytics Engine scripts are written in Python, and the Network Analytics Engine provides a built-in Python interpreter that is used when validating scripts and creating agents from scripts.

Python is the go-to language for network engineers:

- Python is high-level and human-readable.
- Python is popular with an active development community.
- There are many libraries (code written for you that you can use in your programs) available.

Python and the REST API to the AOS-CX database provide powerful tools to support network automation. By using Python and the REST API, you can move far beyond CLI scripting in network automation.

In the past, a network engineer might use Perl scripts to automate show and configure CLI commands. This scheme provided some automation, but it was inefficient because it still used the CLI to interact with the switch. CLI inputs and outputs are in an unstructured, human-readable format. You must use text processing based on specific CLI output to extract the information you want.

For example, you would have to write code to detect a MAC address in the large continuous string of text that is the CLI output. The CLI output might have many things to make it human-readable, such as table formatting, column headings, introductory text, and so forth, and the way MAC addresses are presented can vary by CLI, operating system version, and sometimes even by individual command output.

In contrast, by using programmatic APIs such as the AOS-CX REST API, you can get the information you are looking for in a predictable and structured way. For example, you can ask for and get just the MAC address.

Although Python programming is beyond the scope of this document, this document provides information about the structure and functions of Network Analytics Engine scripts.

"Sandboxes" for agent actions

When an agent performs an action, the action is performed in a "sandbox" that is created when the action starts and removed when the action completes. The sandbox is in the default VRF, so it does not have access to the management network.

A sandbox is an isolated, tightly controlled environment in which programs can be run. Sandboxes restrict what a program can do, giving it the appropriate permissions and computing resources without allowing it access to the entire computing environment.

This design has the following benefits:

- Agents coexist and are prevented from using an excessive amount of CPU resources.
- Agents can benefit from the high-availability features of AOS-CX. During a switch failover event, the daemon that handles the sandbox can recover its state information and continue operations as before.
- Agents are prevented from accessing sensitive information—such as certificate files—in the switch operating system.

You can view a list of scripts and information about specific scripts on the switch using any of the following interfaces:

- Web UI **Analytics** panel on the **Overview** page

The **Analytics** panel shows the total number of scripts, agents, and monitors compared to the total number supported on the switch.

- Web UI **Analytics** dashboard
- REST API GET method on the `nae_scripts` resource
- CLI `show nae-script` command

Managing NAE scripts across switch software updates

After you update the switch software to a new release or install an older software release on the switch, some Aruba Network Analytics Engine (NAE) scripts might not be valid. A script might not be valid because the script uses a URI or API function that is not valid on that software release. The existing scripts might generate errors, and the NAE data might not be valid until you upload the new scripts and clear the NAE data.

Prerequisites

The new switch software has been installed on the switch.

Procedure

1. Clear the NAE data by entering the `clear nae-data` command from the manager context.

```
switch# clear nae-data
```

2. For each script that is not marked `System Created` in the Web UI, locate the version of the script that supports the software release running on the switch.
 - For Aruba-certified NAE scripts, the Aruba Solutions Exchange (ASE) includes tags that indicate the minimum and maximum supported software release.
 - For switches running 10.01 and later software releases, when you select the ASE download button in the Web UI, the Web UI displays only the Aruba-certified NAE scripts that are supported on the software release running on the switch.
 - For scripts that are not Aruba-certified NAE scripts downloaded from the ASE, see the information provided by the script author about which script version is supported.
3. Follow the instructions for updating a script.

The steps to update a script depend on the switch software release version that was running when the script was installed.

4. On the Analytics Dashboard, locate and close any time series graph panels for the default agent from the previous software release.

The default agent is created from the built-in script. The name of the agent is based on the name of the built-in script. Built-in scripts and agents have an origin of "system" and are marked `System Created` in the Web UI. Built-in scripts and their agents are updated automatically. However, sometimes the time series graph panel for the previous default agent is not closed during the update. After the software update, instead of graphed data, such panels show an error message beginning with: `Agent data not found.`

Viewing script details using the Web UI

Prerequisites

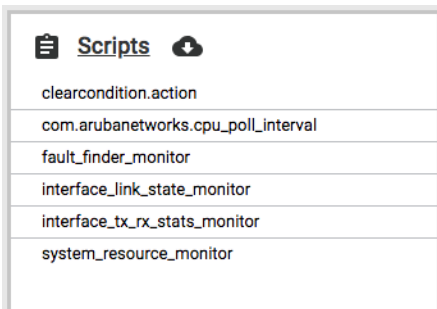
You must be logged in to the AOS-CX Web UI.

Procedure

1. From the Overview page, look at the **Analytics** panel to view the total number of scripts, agents, and monitors compared to the total number supported on the switch.

For example, `Scripts: 7/25` indicates that there are total of seven scripts out of a maximum of 25 scripts supported on this switch.

2. Select **Analytics** from the AOS-CX Web UI navigation pane. The Analytics Dashboard is displayed.
3. The **Scripts** panel in the Analytics Dashboard shows a list of the scripts available on the switch.



4. From the Scripts panel, select a link to a specific script to display the Script Details page.

```
## -*- coding: utf-8 -*-
#
# (c) Copyright 2017-2019 Hewlett Packard Enterprise Development LP
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied. See the License for the
# specific language governing permissions and limitations
# under the License.

import math

Manifest = {
    'Name': 'system_resource_monitor',
    'Description': 'System Resource (CPU/Memory) Monitoring agent',
    'Version': '1.2',
    'Author': 'Aruba Networks'
}

ParameterDefinitions = {
    'short_term_high_threshold': {
        'Name': 'Average CPU/Memory utilization in percentage '
        'in a short period of offense to set Minor alert',
        'Description': 'When average CPU/Memory utilization exceeds '
        'this value, agent status will be set to Minor '
        'and agent will log all system daemons '
        'CPU/Memory utilization details with CoPP statistics',
        'Type': 'integer',
        'Default': 90
    },
    'short_term_normal_threshold': {
        'Name': 'Average CPU/Memory utilization in percentage in '
        'a short period of offense to unset Minor alert',
        'Description': 'When average CPU/Memory utilization is below '
        'this value, Minor status will be unset.',
        'Type': 'integer',
        'Default': 80
    },
    'short_term_time_period': {
        'Name': 'Time interval in minutes to consider average CPU/Memory '
        'utilization for Short Term thresholds',
        'Description': 'Time interval to consider average CPU/Memory '
        'utilization for Short Term thresholds',
        'Type': 'integer',
        'Default': 480
    }
}
```

You can view the following script details.

- **Script Details panel:** Shows script information, and if the script is system created. You can select the + sign to create an agent from the script.
- **Script Parameters panel:** Shows a list of parameters configured in the script. Select a parameter to display a dialog box with a description of the parameter.
- **Script Contents panel:** Shows the programmatic contents of the script. You can click the down arrow to download the script.
- **Script Details panel:** Shows script information, and if the script is system created. You can select the + sign to create an agent from the script.
- **Script Parameters panel:** Shows a list of parameters configured in the script. Select a parameter to display a dialog box with a description of the parameter.
- **Script Contents panel:** Shows the programmatic contents of the script. You can click the down arrow to download the script.

Downloading a script using the Web UI

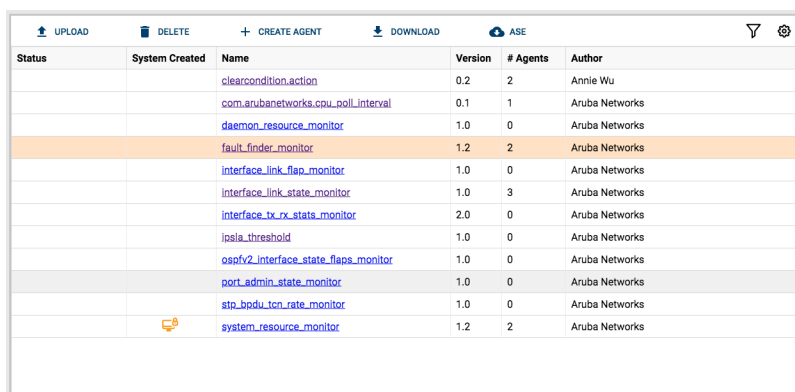
You may want to change a downloaded script and upload it again after making changes. You can download a script and save the file to the specified location.

Prerequisites



You must be logged in to the AOS-CX Web UI.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Scripts panel, click **Scripts** to display the Script Management page.



Status	System Created	Name	Version	# Agents	Author
		clearcondition.action	0.2	2	Annie Wu
		com.arubanetworks.cpu_poll_interval	0.1	1	Aruba Networks
		daemon_resource_monitor	1.0	0	Aruba Networks
		fault_finder_monitor	1.2	2	Aruba Networks
		interface_link_flap_monitor	1.0	0	Aruba Networks
		interface_link_state_monitor	1.0	3	Aruba Networks
		interface_tx_rx_stats_monitor	2.0	0	Aruba Networks
		ipsla_threshold	1.0	0	Aruba Networks
		osofv2_interface_state_flaps_monitor	1.0	0	Aruba Networks
		port_admin_state_monitor	1.0	0	Aruba Networks
		stp_bpdu_tcn_rate_monitor	1.0	0	Aruba Networks
		system_resource_monitor	1.2	2	Aruba Networks

3. Select a script row (not the script link) and click  **Download** to download the script.
You can also download a script from the Script Details page, in the Script Contents panel, where you see the  button.
4. In the download dialog box, you can either open the script file or save the file.



Saving will overwrite any existing file in the specified location with the same name.

To download the script, click **OK**. To cancel this operation, click **Cancel**.

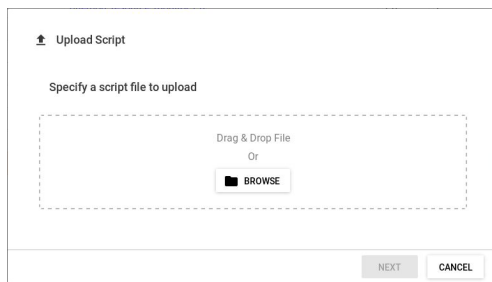
Uploading a script using the Web UI

Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

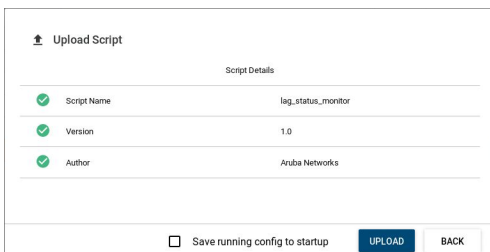
1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Scripts panel, click **Scripts** to display the Script Management page.
3. In the Script Management page, click **Upload**.
4. The Upload Script dialog box is displayed where you can specify a script file to upload. You can either drag and drop a file or browse to select the file to upload. Click **Next** to continue with the upload or click **Cancel**.



5. In the Upload Script dialog box, do the following:
 - a. Verify that the correct script is displayed.
 - b. Optionally, select **Save running config to startup**.

If you do not save the running configuration to the startup configuration, the change is not preserved when the switch is rebooted.

- c. To upload the script, click **Upload**. To cancel this operation or to upload a different script than the script that is displayed, click **Back**.



6. If the name of the script as shown in the script manifest matches the name of the existing script, when you upload the script, the Aruba Network Analytics Engine (NAE) does the following:

If the name of the script as shown in the script manifest matches the name of the existing script, a confirmation dialog box is displayed. Confirm that you want the existing script to be updated.

 - Displays a message that a script with the same name exists, and asks you to confirm that you want to update the existing script.
 - Deletes the existing script and agents
 - Replaces the existing script with the new script
 - Recreates the agents with the same parameters that were used before the script was updated.

If the upload is successful, a success message is displayed. The new script is added to the list of available scripts, or the existing script is updated.

If the upload is unsuccessful, an error is displayed indicating what parts of the script manifest are invalid.

If the script has a syntax error but it contains the required manifest, the upload will complete. However, the status field will show an error indicator. If you go to the Script Details page, you can get more details on the error.

Updating a script using the Web UI

Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. If the new script has the same name as the existing script, and the existing script was installed while the switch was running AOS-CX software version 10.03 or later, upload the new script:

The name in the manifest of the script must match the name of the script in the Scripts panel of the Analytics Dashboard exactly. For example, a script with a version number in the name does not match a script that does not include a version number in the name. The script file name is not required to match the script name in the manifest. For example, the script file name can include a version number.

If the name of the script as shown in the script manifest matches the name of the existing script, when you upload the script, the Aruba Network Analytics Engine (NAE) does the following:

- Displays a message that a script with the same name exists and asks you to confirm that you want to update the existing script.
- Deletes the existing script and agents
- Replaces the existing script with the new script
- Recreates the agents with the same parameters that were used before the script was updated.

If the monitors in the new script and the existing script are the same, previously collected time-series data is preserved when the script is updated. Clearing the NAE data is not required.

2. If the name of the new script does not match the name of the existing script, or the existing script was installed while the switch was running an AOS-CX software version earlier than 10.03, do the following:
 - a. Delete the existing script. When you delete a script, the agents are deleted automatically.
 - b. Upload the new script.
 - c. Clear the NAE data by entering the `clear nae-data` command from the manager context.

```
switch# clear nae-data
```

- d. Create agents associated with the new script.
- e. Add time series graph panels for the agents to the Analytics Dashboard.

Deleting a script using the Web UI



Deleting a script also deletes all agents associated with that script.

Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Scripts panel, click **Scripts** to display the Script Management page.
3. Select a script row (not the script link) and click **Delete**.



4. In the confirmation dialog box, do the following:
 - a. Optionally, select **Save running config to startup**. If you do not save the running configuration to the startup configuration, the change is not preserved when the switch is rebooted.
 - b. To delete the script, click **Confirm**. To cancel, click **Cancel**.

Getting information about scripts using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

You must be logged in to the switch REST API.

Procedure

- To get detailed information about a specific script, use the GET method on the URI of the script.

This example gets information about the script named

`com.arubanetworks.mac_arp_count_monitor`:

`GET https://192.0.2.5/rest/v1/system/nae_scripts/com.arubanetworks.mac_arp_count_monitor`

The following example is the response body for the request. The entire script is returned in base64 format. Because of the length of the script, the example shows only part of the script.

```
{
  "author": "Aruba Networks",
  "description": "MAC address learned on VLAN ID and number of neighbors learned
using ARP",
  "expert_only": false,
  "nae_parameters": {
    "vlan_id": "/rest/v1/system/nae_scripts/mac_arp_count_monitor/nae_
parameters/Vlan_Id"
  },
  "name": "mac_arp_count_monitor",
  "origin": "user",
  "script": "IyAtKi0gY29XBUCAgICpdGRvcih1cmky...LCAnQVJQIFRhYmx1IENvdW50JykK",
  "status": {
    "state": "VALIDATED"
  },
  "tags": ['bridging', 'vlan', 'mac', 'neighbors', 'arp']
  "version": "1.0"
}
```

- To get a list of all the scripts, use the GET method on the URI of the `nae_scripts` collection of the script.

For example:

`GET https://192.0.2.5/rest/v1/system/nae_scripts`

Example response:

```
[
  "/rest/v1/system/nae_scripts/system_resource_monitor",
  "/rest/v1/system/nae_scripts/mac_arp_count_monitor",
  "/rest/v1/system/nae_scripts/ospf_neighbor"
]
```

Uploading a script using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.
- The script must be encoded in base64 format.

Scripts you download from the Aruba Solution Exchange website (instead of through the switch Web UI) or the GitHub repository are not in base64 format. You must encode those scripts before you upload them.

Procedure

Use the POST method to upload the script to the `nae_scripts` resource collection.

For example:

```
POST https://192.0.2.5/rest/v1/system/nae_scripts
{
  "name": "port_admin_state_monitor",
  "expert_only": false,
  "script": "<script_content_encoded_in_base64_format>"
}
```

In the example:

- The name of the script is: `port_admin_state_monitor`
- The `expert_only` parameter indicates whether it is recommended that someone using the script have expert knowledge about the script. Typically, this parameter is `false`.
- `<script_content_encoded_in_base64_format>` is the text string of the script after it is encoded in base64 format. For an example of a script in base64 encoded format, see [Script example](#).

If the script name in the Manifest of the new script exactly matches the name of an script that has already been uploaded, the upload fails with a message that informs you that the script already exists. You can use the PUT method to update the existing script.

If you are uploading a script to replace an existing script, but the script name in the Manifest is not an exact match, you must delete the script you want to replace.

Updating a script using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

1. If the name of the script in the Manifest is an exact match to the name of the existing script, and the existing script was installed while the switch was running AOS-CX software version 10.03 or later, use the PUT method to update the script.

When the PUT method is used, the existing script and agents are deleted, the script is replaced, and the agents are automatically recreated with the same parameters that were used before the script was updated. If the monitors in the new script and the existing script are the same, previously collected time-series data is preserved when the script is updated.

2. If the existing script was installed while the switch was running an AOS-CX software version earlier than 10.03, or the script name in the Manifest does not match the name of the existing script—for example one has a version number in the name and the other does not—you must delete the existing script and then upload the new script:
 - a. Use the DELETE method to delete the script.

When you delete a script, all associated agents are deleted automatically.
 - b. Use the POST method to upload the modified script.
 - c. Clear the NAE data.
 - d. Create agents associated with the new switch.

Deleting a script using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.



Deleting a script also deletes all agents associated with that script.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

To delete a script, use the DELETE method on the URI of the script.

The following example deletes the script: `com.myco.bgp_mon`.

```
DELETE https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon
```

Downloading a script from the switch using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

You must be logged in to the switch REST API.

Procedure

Use the GET method on the URI of the script.

This example downloads script named `com.arubanetworks.mac_arp_count_monitor`.

```
GET https://192.0.2.5/rest/v1/system/nae_scripts/com.arubanetworks.mac_arp_count_monitor
```

The following example is the response body for the request. The entire script is returned in base64 format. Because of the length of the script, the example shows only part of the script.

```
{
  "author": "Aruba Networks",
  "description": "MAC address learned on VLAN ID and number of neighbors learned
using ARP",
  "expert_only": false,
  "nae_parameters": {
    "Vlan_Id": "/rest/v1/system/nae_scripts/mac_arp_count_monitor/nae_
parameters/Vlan_Id"
  },
  "name": "mac_arp_count_monitor",
  "origin": "user",
  "script": "IyAtKi0gY29XBUCAgICpdGRvcihlcmky...LCAnQVJQIFRhYmx1IENvdW50JykK",
  "status": {
    "state": "VALIDATED"
  },
  "version": "1.0"
}
```

show running-config command output for agents and scripts

Built-in scripts are not included in the output of the `show running-config` command.

A built-in agent is included in the output of the `show running-config` command only if one or more of the agent parameters has been modified and saved.

User-created scripts and agents are included in the output of the `show running-config` command.

The output of the `show running-config` command includes the following information for scripts and agents:

- The name of the script.
- The value of the `expert_only` parameter. Typically, this parameter has a value of `false`.
- The script code in base64 format.
- The name of the agent.
- The value of the `enabled` parameter.
- The name of each agent parameter, followed by its value in base64 format.

The following example shows the output of `show running-config` for a user-created NAE script and one agent that monitors a route count. The name of the script is: `route_count_monitor`. The name of the agent is: `route_count_instance1`.

The entire script is returned in base64 format. Because of the length of the script, the example shows only part of the script.

```
switch# show running-config
...
nae-script route_count_monitor false IyAtKi0gY29kaW5nOiBldGYtOCAtK
i0NCiMNCiMgQ29weXJpZ2h0IChjKSAyMDE3IEhld2xldHQgUGFja2FyZCBFbnRlcnB
...
yaXNlIERldmVsb3BtZW50IEExQDQojDQojIEExpY2Vuc2VkIHVuZGVyIHRoZSBBcGFja
nae-agent route_count_monitor route_count_instance1 false upper_co
unt_threshold:MTAwMDA= vrf_name:ZGVmYXVsdA== lower_count_threshold
:OTUwMA==
...
```

Script status

A script can have the following status values:

CREATED

The script has been uploaded to the switch, but script validation has not begun.

ERROR

The script validation process detected an error that would result in execution errors. Resolve the error by modifying the script.

VALIDATING

The script syntax and components (manifest, parameters, monitor, condition, and action) are in the process of being validated.

VALIDATED

The script syntax and components (manifest, parameters, monitor, condition, and action) have been validated and no errors have been found.

You can view a list of agents and information about specific agents on the switch using any of the following interfaces:

- Web UI **Analytics** panel on the **Overview** page

The **Analytics** panel shows the total number of scripts, agents, and monitors compared to the total number supported on the switch.

- Web UI **Analytics** dashboard
- REST API GET method on the `nae_agents` resource
- CLI `show nae-agent` command

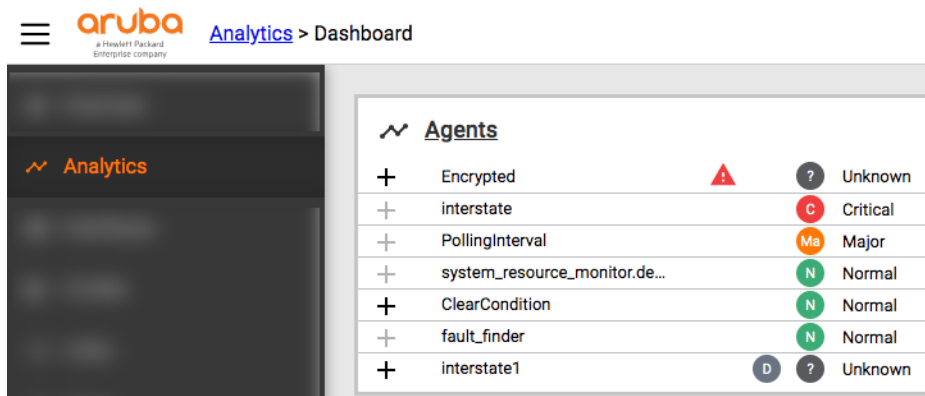
Viewing a list of agents installed on a switch using the Web UI

Prerequisites

You must be logged in to the AOS-CX Web UI.

Procedure

1. Select **Analytics** from the ArubaOS-CX Web UI navigation pane. The Analytics Dashboard is displayed.
2. The **Agents** panel in the Analytics Dashboard shows a list of the agents installed on the switch, along with the status of each agent.



Viewing agent information using the Web UI

You can view Analytics agent information including: agent status, script information, agent parameters, one or more time series graphs, and any alerts generated.

Prerequisites

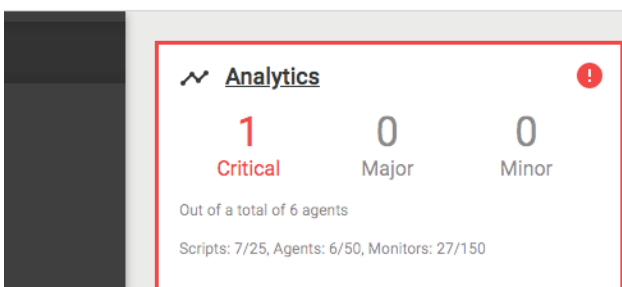
- You must be logged in to the Web UI.
- Ensure that the switch and the client where Web UI is running are set to use NTP or to a time zone based on UTC time. Otherwise, NAE agent data might be incorrect or missing.

For example, if the time on switch is set to 2 hours ahead of the client manually instead of by changing the time zone offset, the agent data is populated according to the new time on switch. If the switch time is set back to match client time later, the Time Series Database does not overwrite the old data. Therefore the client Web UI shows inaccurate data.

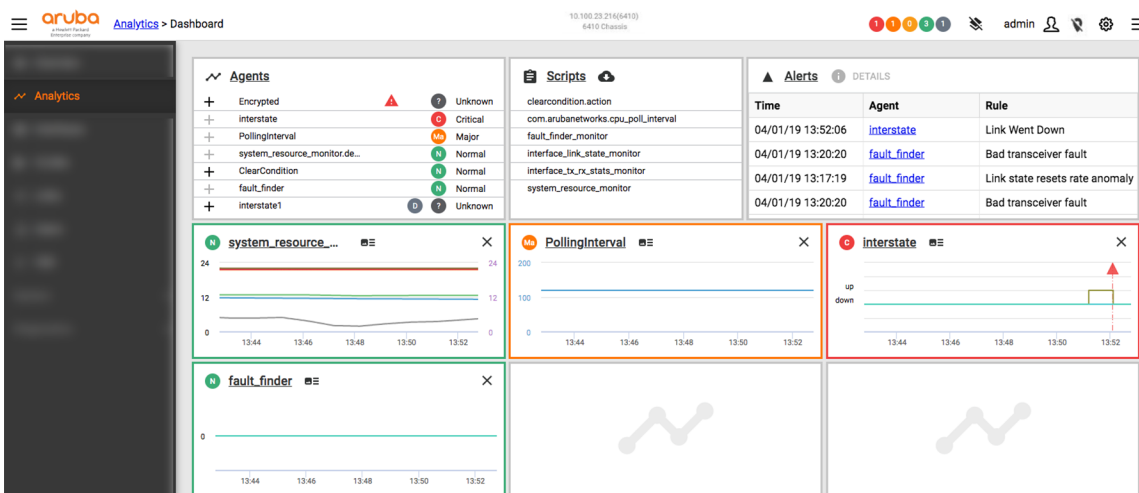
Procedure

1. From the Overview page, look at the **Analytics** panel to see the total number of agents in critical, major, and minor status. If the panel is outlined in red, it indicates agent status issues.

Overview

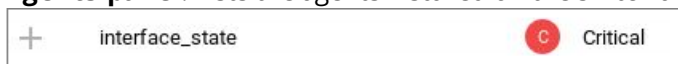



2. To go to the Analytics Dashboard, select the **Analytics** link in the Analytics panel on the Overview page.



View the following information on the Analytics Dashboard:

- **Top banner:** Shows the number of agents with each type of status.
- **Agents panel:** Lists the agents installed on the switch and indicates the status of each agent.



If there is an error in an agent, the Agents panel shows an error icon next to the agent status.  Optionally, you can add an Analytics agent time series graph to the Analytics Dashboard by clicking the + plus sign next to any agent listed in the Agents panel.

The time series graph shows data collected by the Analytics agent. If an agent has multiple time series graphs, the graph displayed on the Analytics Dashboard is specified by the script. You cannot choose

which graph to display on the Analytics Dashboard, but you can see all the graphs in the Agent Details page.

Click the **Agents** link to display the Agent Management page. On this page you can create, edit, delete, enable, and disable an agent.

- **Scripts panel:** Lists available scripts.

Select a script from the list to display the Script Details page where you can view script details, create an agent to run the script, and download the script.

Click the **Scripts** link to display the Script Management page. On this page you can upload, download or delete a script, create an agent, and access the Aruba Solution Exchange to find more scripts.

- **Alerts panel:** Lists alerts generated by all agents.

Select an alert in the list and click **Details** to display the Alert Details dialog box.

Click the **Alerts** link to display a list of the alerts with information on the rule and actions for each alert.

- **Time series graphs:** If an agent time series graph has been added to the Analytics Dashboard, the graph is outlined in the agent status color. Agents can have more than one time series graph, but only one graph for the agent is displayed in the Analytics dashboard. Click the link in the graph to display the Agent Details page.

3. From the Analytics Dashboard, Agents panel, select the link to a specific agent. The Agent Details page is displayed.



View the following information from the Agent Details page:

- **Agent Details panel:** Shows information about the agent.

Select the Edit button to enable or disable an agent and modify agent parameters.

Select the View Script button to display the Script Details page where you can view script information, create an agent to run the script, and download the script.

- **Status panel:** Shows the status of the agent and when the status was last updated. For some agents, you may see additional information. For example:

System Created

If the Status panel includes the statement `System Created`, the agent cannot be deleted.

Baseline Thresholds

If the Status panel includes Baseline Thresholds, the agent can learn about the activity being measured and set low thresholds and high thresholds based on what it learns. The Baseline Thresholds information shown in the Status panel includes the number of thresholds in the following states:

Active

When a baseline threshold is in the active state, the agent has learned and established the high and low thresholds, and the agent executes actions and generates alerts based on those low or high thresholds.

Inactive

When agent is disabled, the baseline stops collecting data and updating thresholds. After the agent is re-enabled, the baseline goes into the learning state again.

Learning

While a baseline threshold is in the learning state:

- The agent gathers data related to that baseline until the initial learning period completes. Low and high thresholds are determined using the learning algorithm defined in the script, and are set only after learning state is completed.
- Default thresholds (if specified in the script) are used to determine whether to execute actions or generate alerts.

Baseline thresholds remain in the learning state for a script-specified period of time after the agent is enabled.

Selecting **Baseline Thresholds** displays a dialog box that shows additional information about all the baselines for the agent, including the name, the associated monitor, state, and the current learned low and high thresholds.

Baseline Title	Monitor	State	Low	High
Baseline for Interface rx Packets	Rx Packets (packets)	active	14177973	23629956
Baseline for Interface tx Packets	Tx Packets (packets)	active	866849100	1444748500

If there is an agent error, an error indicator is shown and you can hover over it for more information.

Status

Normal

Agent Error
Timeseries data cannot be generated...

Last Status
Never

Parameters

ospf_area_id: 0.0.0.0
OSPFv2 area id

ospf_interface_id: 1/1/2

Timeseries data cannot be generated due to the following agent error: The URI is invalid or not configured. Either please upload new script with valid URI or configure the resource, disable the instance and enable again: /rest/v1/system/xrfs/default/ospf_routers/1/areas/0.0.0.0/ospf_interfaces/1/%2F1%2F2?attributes=ifm_state

- **Parameters panel:** Shows the parameters used by the agent. For example, a parameter can be a threshold value that, when breached, causes the agent status to change and an alert to be generated. Selecting a parameter displays the description in a dialog box.

- Time Series graph:** Graphs the data collected by the agent over time. Agents might have more than one time series graph. Alert indicators and configuration checkpoints are overlaid on the graph.

Alert indicators can include: a red or yellow triangle for an alert, a green triangle for return to normal, a blue triangle for an alert on several resources being monitored. An example of an alert on several resources: when monitoring multiple interfaces (wildcard), if an interface goes down, a red alert is generated. If another interface goes down, then a blue alert is generated. A green alert will not be generated until all the interfaces are back up.

Configuration checkpoints are shown as purple diamonds on the graph.

Clicking an alert indicator on the graph displays the Alert Details dialog box.
- Alerts panel:** Lists alerts.

Select the **Alerts** link to display a list of the alerts with information on the rule and actions for each alert.

Select an alert and click **Details** to display the Alert Details dialog box.

Select an alert and click **Navigate** to change the time series graph to show the time period with this alert.

Finding alert details using the Web UI

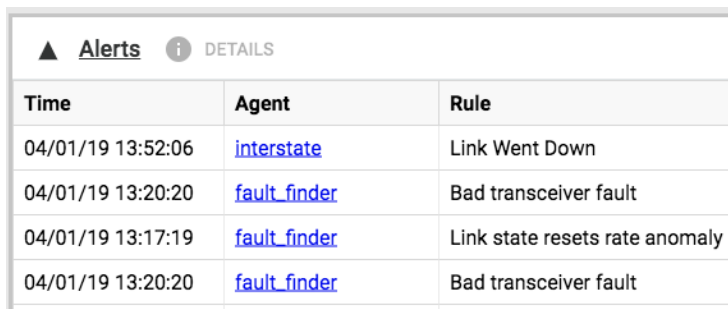
You can view details on the alerts displayed in the Web UI.

Prerequisites

You must be logged in to the Web UI.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, the Alerts panel lists the alerts for all agents.



Time	Agent	Rule
04/01/19 13:52:06	interstate	Link Went Down
04/01/19 13:20:20	fault_finder	Bad transceiver fault
04/01/19 13:17:19	fault_finder	Link state resets rate anomaly
04/01/19 13:20:20	fault_finder	Bad transceiver fault

3. To see the alerts for a specific agent, in the Analytics Dashboard Agents panel or Alerts panel, select an agent.
4. In the Agent Details page, the agent alerts are listed in the Alerts panel.



- In the Alerts panel, select an alert and click **Details** to view the Alert Details dialog box. To close the dialog box, click **Close**.

You can also access alert details directly from the Analytics Dashboard by selecting an alert in the Alerts panel and clicking **Details**.

- The **Action Result(s)** in Alert Details dialog box might include additional details about actions and links to the action result output.

▲ Alert Details

Agent	interstate
Rule	Link Went Down
Time	04/01/19 11:16:51
Action(s)	ALERT_LEVEL,CLI(2),SYSLOG
Monitors:	Interface Link status
Time Series:	Interface_link_state
Resources:	Interface=1/1/3
Action Result(s):	
Alert Level Changed	C Critical
SYSLOG	[local] Interface 1/1/3 Link gone down
CLI (show interface 1/1/3 extended)	Output - SUCCESS ✓
CLI (show lldp configuration 1/1/3)	Output - SUCCESS ✓

To view the Action Result Output dialog box for an action, click the **Output** link.

▲ Action Result Output

Time
04/01/19 11:16:59

 **SUCCESS**

Commands
show lldp configuration 1/1/3

Output

```
switch# show lldp configuration 1/1/3

LLDP Global Configuration
=====
LLDP Enabled           : Yes
LLDP Transmit Interval : 30
LLDP Hold Time Multiplier : 4
LLDP Transmit Delay Interval : 2
LLDP Reinit Time Interval : 2

LLDP Port Configuration
=====
PORT      TX-ENABLED  RX-ENABLED
-----
1/1/3     Yes         Yes
```

Working with an Analytics time series graph

Data collected by an Analytics agent is displayed in the Web UI in one or more time series graphs. An agent has at least one graph. An agent can have multiple graphs as specified in the script, but only one graph represents the agent on the Analytics dashboard. The graph that represents the agent on the Analytics dashboard is specified in the script.

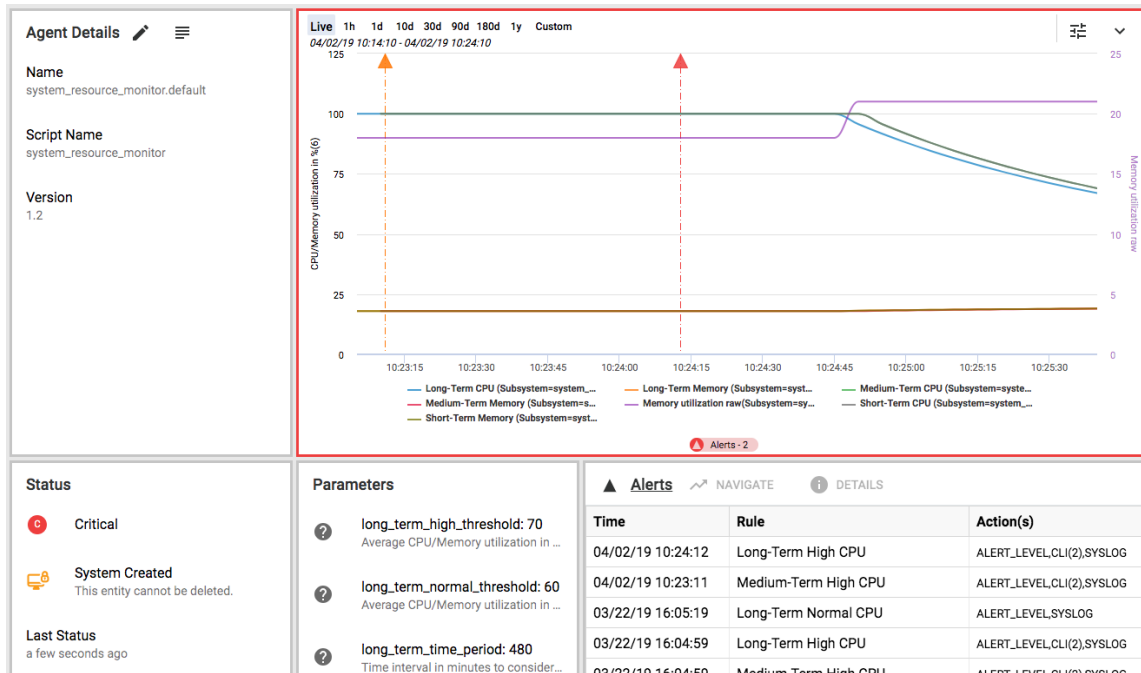
If the Analytics dashboard does not include a graph for an agent, you can add the graph that represents that agent from Analytics dashboard. Graphs on the Analytics dashboard represent a live view only. The graph customization toolbar is not available from the Analytics dashboard.

The Agent Details page displays all the graphs for an agent, with each graph displayed in a panel.

Configuration checkpoints and alert indicators are overlaid on the graph. Configuration checkpoints are shown as purple diamonds. Alert indicators can include the following:

- A red or yellow triangle for an alert
- A green triangle for a return to normal
- A blue triangle for an alert on several resources being monitored.

The graph displays alerts for all metrics being monitored. However, time series graphical information can be shown for a maximum of eight metrics. The metrics that are being shown on the graph are listed at the bottom of the graph.



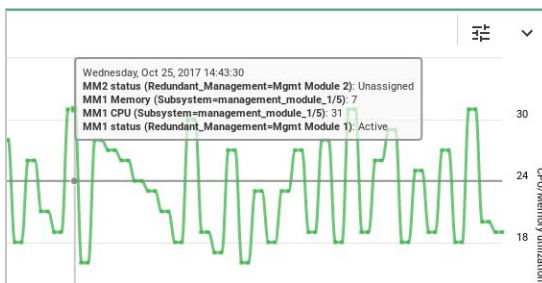
- [Customizing data displayed on the graph on page 38](#)
- [Zooming in on the graph on page 39](#)
- [Downloading the graph as an image or .csv file on page 39](#)
- [Viewing an alert on the graph on page 40](#)

Customizing data displayed on the graph


There are several ways you can customize the data displayed on a time series graph to show more or less data.

Procedure

1. View a tooltip for each data point on the graph by hovering the cursor over the data point.
The tooltip displays the date, time range, and min-max range.



2. Hover over a specific item in the legend following the graph to show only that specific data line on the graph. The other data will be less visible.

From the graph shown on the Agent Details page, click the  Configure Chart button to open the Customize Chart dialog box.

- The default mode is Automatic Monitoring, where the most meaningful monitors and resources are automatically selected to display on the agent graph. To customize what data (monitors and resources) you want displayed on the agent time series graph, select **Customize Monitoring**.

- You can sort and filter the Show column. If a monitor is a wildcard type, then you see a different icon from the check box, where you can click and select subresources under that monitor.

The graph displays alerts for all metrics being monitored. However, the graph can show graphed data for a maximum of eight metrics at a time. The metrics that are being shown on the graph are listed at the bottom of the graph. You can choose which metrics to show. To remove the metric from the graph, clear the box in the Show column of the metric you want to remove. To show a metric in the graph, select the box in the Show column of the metric you want to display.

- The Resources Selected column shows how many total resources are selected out of the total available resources.
- If a monitor can have an aggregation function, that function is displayed in the Aggregation column.

Show	Monitor	Aggregation	Time Series	Alert Total	Resources Selected
<input checked="" type="checkbox"/>	Short-Term Memory	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	0	N/A
<input checked="" type="checkbox"/>	Short-Term CPU	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	1	N/A
<input checked="" type="checkbox"/>	Memory utilization raw		Subsystem_resource_utilization	0	N/A
<input checked="" type="checkbox"/>	Medium-Term Memory	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	0	N/A
<input checked="" type="checkbox"/>	Medium-Term CPU	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	1	N/A
<input checked="" type="checkbox"/>	Long-Term Memory	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	0	N/A
<input checked="" type="checkbox"/>	Long-Term CPU	AVG_OVER_TIME	AverageOverTime_Subsystem_resource_utiliz	2	N/A

Zooming in on the graph

There are several different ways to zoom in on a specific time period on the time series graph.

Procedure

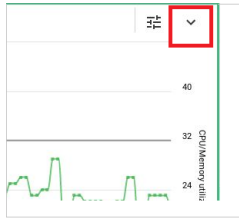
1. Zoom in and out on the graph by selecting a zoom level from the options displayed at the top of the time series graph: 1 hour, 1 day, 10 days, 30 days, 90 days, 180 days, 1 year. You can also select **Custom** to enter a specific date and time range.

Live 1h 1d 10d 30d 90d 180d 1y Custom
06/30/17 14:44:37 - 06/30/17 14:54:37

2. Or you can highlight a custom range of data on the graph as follows:
 - a. Position the cursor on the time axis of the graph until a vertical line appears through the time.
 - b. Drag the vertical line to the left or right to the beginning or end of the time period you want to view.
 - c. The selected time period is highlighted and the begin and end dates are displayed next to the Custom zoom level.
 - d. Release the mouse button and the graph is redrawn for just the time period selected.
3. Reset the graph to the default by selecting the **Live** zoom level.

Downloading the graph as an image or .csv file

You can download the graph either as an image or represented as a set of comma-separated values that can be opened in spreadsheet programs. The download options are accessed from the down arrow in the top right corner of the time series graph:



- To download the graph as an image, click the down arrow and select **Download Chart**. The graph is downloaded in a file in `.png` format.
- To download the graph as a set of comma-separated values, click the down arrow and select **Export to CSV**.

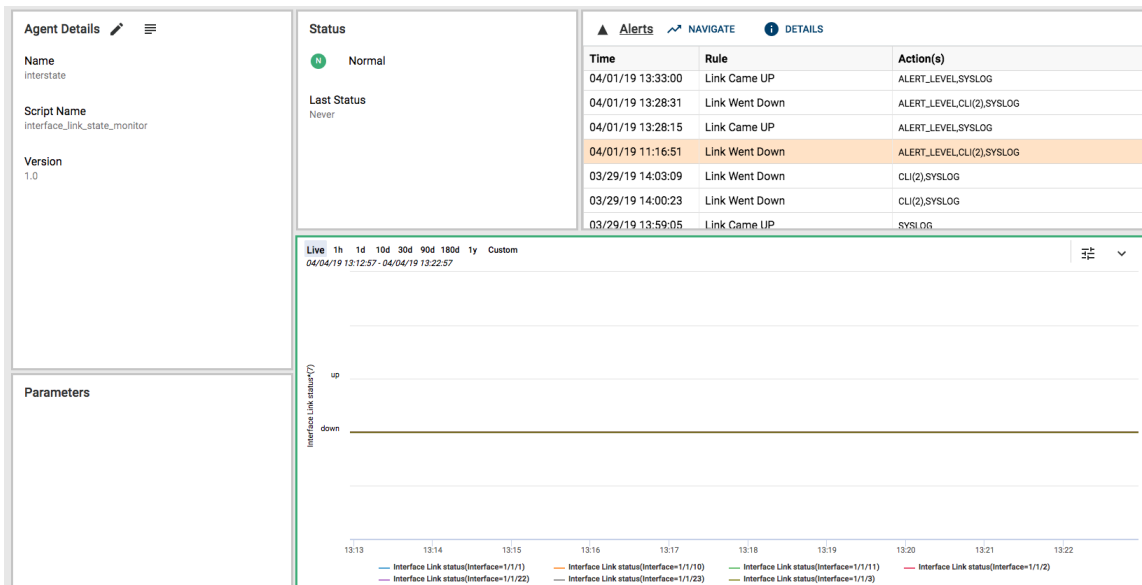
The graph is downloaded in a file in `.csv` format.

Viewing an alert on the graph

The graph shown on the Agent Details page might not show the time period or resource associated with a specific alert. Use this procedure to change the graph to show the alert and the associated metric.

Procedure

1. From the alerts panel on the Agent Details page, select an alert and click **Navigate**.



The graph is changed to display the time period containing the alert. However, the alert might be for a metric that is being monitored but that is not being shown in the graph.

The graph displays alerts for all metrics being monitored. However, the graph can show graphed data for a maximum of eight metrics at a time. The metrics that are being shown on the graph are listed at the bottom of the graph.



2. To adjust the graph display to show the metrics for the alert, do the following:
 - a. Locate the alert on the graph and click the alert triangle flag. The Alert Details dialog box is displayed.

▲ Alert Details

Agent	Interstate
Rule	Link Went Down
Time	04/01/19 11:16:51
Action(s)	ALERT_LEVEL,CLI(2),SYSLOG
Monitors:	Interface Link status
Time Series:	Interface_link_state
Resources:	Interface=1/1/3
Action Result(s):	
Alert Level Changed	C Critical
SYSLOG	[local] Interface 1/1/3 Link gone down
CLI (show interface 1/1/3 extended)	Output - SUCCESS ✔
CLI (show lldp configuration 1/1/3)	Output - SUCCESS ✔

VIEW ON GRAPH
CLOSE

- b. Click **View on Graph**.

3. If the graph is showing eight metrics and the metric you want to display is the ninth metric, you must choose an existing metric to clear so that the graph can show the metric associated with the alert.

For example:

▲ Alert Details

The configuration cannot exceed 8 enabled resources. Please deselect at least one resource to add Interface=1/1/3.

	Monitor	Resource ID
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/1
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/10
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/11
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/2
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/22
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/23
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/33
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/9

- a. Clear the selection box for the metrics you no longer want to show.
For example:

▲ Alert Details

The configuration cannot exceed 8 enabled resources. Please deselect at least one resource to add Interface=1/1/3.

	Monitor	Resource ID
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/1
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/10
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/11
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/2
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/22
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/23
<input checked="" type="checkbox"/>	Interface Link status	Interface=1/1/33
<input type="checkbox"/>	Interface Link status	Interface=1/1/9

- b. Click **View on Graph**.
The graph is changed to show the metric associated with the alert.
For example:



- You can reset the graph to the default by selecting the **Live** zoom level.

Enabling a disabled agent using the Web UI

Prerequisites

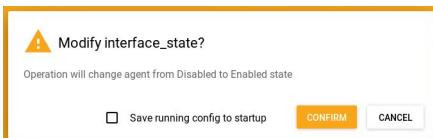
You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

- Select **Analytics** from the navigation pane.
- In the Analytics Dashboard, in the Agents panel, click **Agents**.
- The Agent Management page is displayed. To enable a disabled agent, select the agent row (not the link) for an agent that is disabled and click **Enable**.

Error	System Created	Name	Disabled	Status	Last Status	Script
⚠		ClearCondition		🔍 Unknown	Never	clearcondition.action
		PollingInterval		🟡 Major	an hour ago	com.arubanetworks.cpu_poll_interval
		interstate		🔴 Critical	an hour ago	interface_link_state_monitor
		interstate1	🔴	🔍 Unknown	2 hours ago	interface_link_state_monitor
	🔧	system_resource_monitor.default		🟢 Normal	Never	system_resource_monitor

- In the confirmation dialog box, do the following:
 - Optionally, select **Save running config to startup**.
If you do not save the running configuration to the startup configuration, the change is not preserved when the switch is rebooted.
 - To enable the agent, click **Confirm**. To cancel, click **Cancel**.



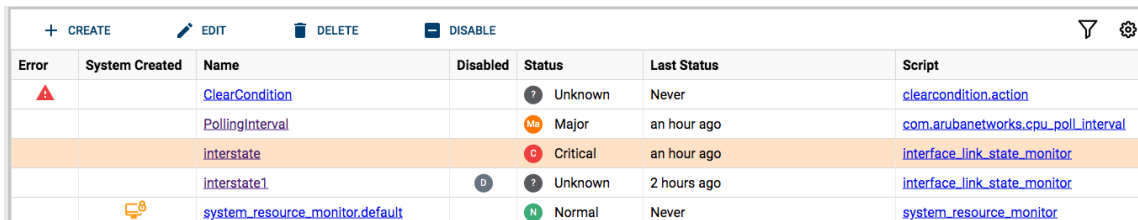
Disabling an agent using the Web UI









Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Agents panel, click **Agents**.
3. The Agent Management page is displayed. To disable an agent, select the agent row (not the link) in the list and click **Disable**.



Error	System Created	Name	Disabled	Status	Last Status	Script
		ClearCondition		 Unknown	Never	clearcondition.action
		PollingInterval		 Major	an hour ago	com.arubanetworks.cpu_poll_interval
		interstate		 Critical	an hour ago	interface_link_state_monitor
		interstate1		 Unknown	2 hours ago	interface_link_state_monitor
		system_resource_monitor.default		 Normal	Never	system_resource_monitor

4. In the dialog box, click **Confirm** to disable the agent or **Cancel**. Optionally, check the box to **Save the running config to startup**. If you do not select **Save the running config to startup**, the change is not preserved when the switch is rebooted.

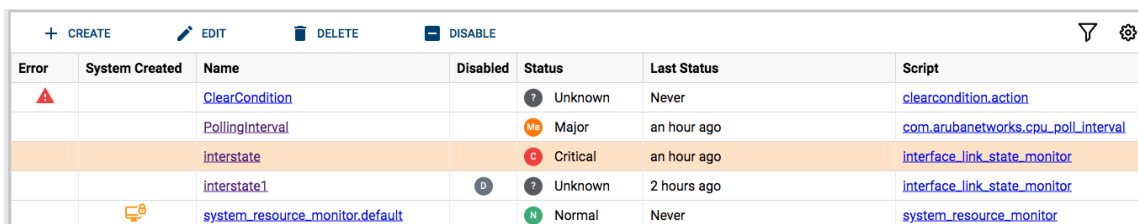
Deleting an agent using the Web UI









Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Agents panel, click **Agents**.
3. The Agent Management page is displayed. To delete an agent, select the agent row (not the link) in the list and click **Delete**.



Error	System Created	Name	Disabled	Status	Last Status	Script
		ClearCondition		 Unknown	Never	clearcondition.action
		PollingInterval		 Major	an hour ago	com.arubanetworks.cpu_poll_interval
		interstate		 Critical	an hour ago	interface_link_state_monitor
		interstate1		 Unknown	2 hours ago	interface_link_state_monitor
		system_resource_monitor.default		 Normal	Never	system_resource_monitor

4. In the dialog box, click **Confirm** to delete the agent or **Cancel**. Optionally, check the box to **Save the running config to startup**. If you do not select **Save the running config to startup**, the change is not preserved when the switch is rebooted.

Creating an agent from an existing script using the Web UI

Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Agents panel, click **Agents**.
3. The Agent Management page is displayed. To create an agent, click **+ Create**.

You can also create an agent from the Scripts Management page by selecting a script row and clicking **+ Create Agent**. Access the Scripts Management page from the Analytics Dashboard by selecting the Scripts link in the Scripts panel.

4. In the Create Agent dialog box, enter the agent information.
 - a. To see a list of scripts available on the switch, click the down arrow next to the Script field.
 - b. Select a script for the agent to run.
 - c. Enter a name for the agent.
 - d. Based on the script you selected, a list of parameters is displayed without values in the **Value** column. Enter the values for the parameters.

The **More Info** column displays additional information about the parameter, such as the default value.

If the **More Info** column includes the lock icon, the value of the parameter is encrypted. The characters you enter are masked on the screen with bullet characters.

If you do not enter a value and there is a default value defined, the default value is used.

If the **Value** column labels the parameter as **Required** and that parameter does not include a default value, you must supply a value before you can create the agent.

+ Create Agent

Script
configuration_change_email

Agent Name

Must start with an alphanumeric character, may consist of a combination of alphanumeric, period (.), dash (-) and underscore (_) characters, length of 3 - 80 characters

Type	Name	Description	More Info	Value
STRING	email_subject	[Optional] Subject of the email notification	Default: NAE detected a config change event	
STRING	recipient_email_address	[Optional] Comma separated list of email addresses to which the notification is to be sent	Required	Missing required parameter value
STRING	sender_email_address	[Optional] Email address from which the alert is sent	Default:	
STRING	smtp_server_address	[Optional] IP address/hostname and port number of the SMTP server	Default:	
STRING	smtp_server_user_id	[Optional] User name for SMTP server if protected with credentials	Default:	
STRING	smtp_server_user_password	[Optional] Password for SMTP server if protected with password	Default: [Lock icon]	
STRING	switch_host_name	[Optional] Hostname for the switch to be used in SMTP hello/ehelo message	Default:	

Some parameter values are missing or invalid

Save running config to startup **CREATE** **CANCEL**

- e. Optionally, check the box to **Save running config to startup**.
If you do not save the running configuration to the startup configuration, the change is not preserved when the switch is rebooted.
- f. To create an agent running the specified script with the parameter values entered, click **Create**.
To cancel the operation, click **Cancel**.

Changing the configuration of an agent using the Web UI

Prerequisites

You must be logged in to the AOS-CX Web UI with administrator rights.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Agents panel, click **Agents**.
3. The Agent Management page is displayed. To edit an agent, select the agent row (not the link) in the list and click **Edit**.
4. In the Edit dialog box, change parameter values you want to change.
Only integer values (threshold or rate) can be changed.
You can also disable or enable the agent by clicking the **Enabled** slider.
5. Optionally, select **Save running config to startup**.
If you do not save the running configuration to the startup configuration, the change is not preserved when the switch is rebooted.
6. To save your changes, click **Save**. To cancel, click **Cancel**.

Getting information about agents using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

You must be logged in to the switch REST API.

Procedure

- To get detailed information about a specific agent, use the GET method on the URI of the agent.

This example gets information about the agent named `com.myco.bgp_mon.Agent1`:

```
GET https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_agents/com.myco.bgp_mon.Agent1
```

The following response is an example of an agent that has an error:

```
{
  "name": "com.myco.bgp_mon.Agent1",
  "disabled": false,
  "parameters_values": {
    "threshold": "10"
  }
  "status": {
    "error_at": "1490134092",
    "error_description": "The URI is invalid or not supported",
    "executed_at": "1490134092",
  }
}
```

- To get a list of the agents of a specific script, use the GET method on the URI of the `nae_agents` collection of the script.

This example gets a list of the agents of the script `com.myco.bgp_mon`:

```
GET https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_agents
```

The following is an example of a response for a script, `port_admin_state_monitor`, that has multiple agents:

```
[
  "/rest/v1/system/nae_scripts/port_admin_state_monitor/nae_agents/port-1_1_1",
  "/rest/v1/system/nae_scripts/port_admin_state_monitor/nae_agents/port-1_1_2"
]
```

- To get detailed information about all the agents on the switch, use the GET method and use wildcards for the script name and agent name:

For example:

```
GET https://192.0.2.5/rest/v1/system/nae_scripts/*/nae_agents/*
```

Creating an agent from an existing script using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.
- The script you will use to create the agent must be in a `VALIDATED` state.

Procedure

1. Use the POST method on the URI of the `nae_agents` collection of the script. In the request body, provide the required information in JSON format.

This example creates an agent with the following characteristics:

- The name of the agent is: `com.myco.bgp_mon.Agent1`
- The name of the script is: `com.myco.bgp_mon`
- The agent is created in a disabled state (`"disabled":true`).
- The `threshold` parameter of the script is set to the value 10.

```
POST https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_agents
{
  "name": "com.myco.bgp_mon.Agent1",
  "disabled":true,
  "parameters_values":{
    "threshold":"10"
  }
}
```

- Agents must have a unique name. If an agent that has the same name exists, the request fails with response code 400 and the message:
NAE agent `<agent_name>` exists already

- If you supply a parameter value that is not the expected type for that parameter, the request fails with response code 400 and the message:

Value type mismatch for the parameter `<parameter_name>`

- If the JSON request body does not include a parameter, but the script defines a default value for that parameter, the default value is used.

If the JSON request body does not include a parameter that is required by the script, and the script does not define a default for that parameter, the agent is created, but the agent has the following error message:

The NAE Agent has Python errors. Please check hpe-policyd logs for Python errors.

Enabling an agent using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

- To enable the agent at the time the agent is created, set the `disabled` parameter to `false` when you create the agent.

For example:

```
POST https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_agents
{
  "name": "com.myco.bgp_mon.Agent1",
  "disabled":false,
  "parameters_values":{
    "threshold":"10"
  }
}
```

- To enable an existing agent, use the PUT method to set the `disabled` parameter to `false`.

For example:

```
PUT https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_
agents/com.myco.bgp_mon.Agent1
{
  "disabled":false,
  "parameters_values":{
    "threshold":"10"
  }
}
```

Disabling an agent using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

To disable an agent, use the PUT method to set the `disabled` parameter to `true`.

For example:

```
PUT https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_
agents/com.myco.bgp_mon.Agent1
{
    "disabled":true
}
```

Changing the configuration of an agent using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

1. Disable the agent.

For example:

```
PUT https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_
agents/com.myco.bgp_mon.Agent1
{
    "disabled":true
}
```

2. Update and enable the agent by using the PUT method on the URI of the agent.

The PUT method replaces the agent specified by the URI. If the agent has multiple parameters in `parameters_values`, any parameters that you do not specify in the PUT method are reset to their default values. If you do not specify a value for a required parameter, and that parameter does not have a default value, the agent configuration is changed, but the agent has the following error message:

The NAE Agent has Python errors. Please check hpe-policyd logs for Python errors.

This example changes the value of the short-term high threshold for `com.myco.bgp_mon.Agent1` to 22.

```
PUT https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_
agents/com.myco.bgp_mon.Agent1
{
  "disabled":false,
  "parameters_values":{
    "short_term_high_threshold":"22"
  }
}
```

Deleting an agent using the REST API

Instructions and examples in this document use an IP address that is reserved for documentation, 192.0.2.5, as an example of the IP address for the switch. To access your switch, you must use the IP address or hostname of that switch.

Prerequisites

- You must be logged in to the switch REST API with a user name that has administrator rights.
- The switch REST API access mode must be `read-write`.

Procedure

To delete an agent, use the DELETE method on the URI of the agent.

You are not required to disable the agent before deleting it.

The following example deletes `Agent1` of the script: `com.myco.bgp_mon`.

```
DELETE https://192.0.2.5/rest/v1/system/nae_scripts/com.myco.bgp_mon/nae_
agents/com.myco.bgp_mon.Agent1
```



Deleting a script also deletes all agents associated with that script.

Agent status

Agents are either enabled or disabled.

An agent that is enabled can have the following status values:

CRITICAL

The agent has encountered a critical error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

MAJOR

The agent has encountered a major error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

MINOR

The agent has encountered a minor error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

NORMAL

Indicates that the agent is actively monitoring network conditions and handling events.

Behaviors when multiple agents monitor the same resource

A **time series** is a sequence of data points taken in successive equally spaced points in time.

A time series is associated with the monitored resource attribute, not the agent:

- Only one time series is created for that resource URI (including the attribute), regardless of how many agents are monitoring the same resource.
- The time series remains in the time series database as long as there is at least one agent monitoring that URI. The entire time series in the time series database is deleted only when **all** the agents monitoring that URI are deleted.

If the agents have different conditions or threshold values, actions (such as alert generation) are taken for each condition met, but the time series being used is the same for all agents. As a result, an agent might evaluate conditions at the time it is enabled based on data that existed prior to that time, such as if the condition aggregates data over time.

For example, consider the following scenario:

1. You create two agents, both of which are disabled:
 - Agent A1 monitors CPU utilization.
 - Agent A2 monitors CPU utilization and has an action to create an alert when the average over the last 1 minute of CPU utilization is greater than 90%.
2. You enable agent A1, which starts monitoring CPU utilization.
3. The CPU utilization is above 90% for 5 minutes.
4. You enable agent A2.
5. Agent A2 creates an alert immediately—instead of after one minute—because the time series already shows that the average of the previous minute of CPU utilization is greater than 90%.

Showing the current and maximum number of agents, monitors, and scripts

Procedure

- To use the CLI, enter the `show capacities-status nae` command. For example:

```
switch# show capacities-status nae

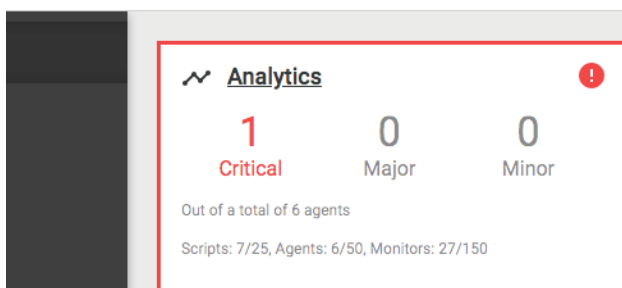
System Capacities Status: Filter NAE
Capacities Status Name                                     Value Maximum
-----
Number of configured NAE agents currently active in the system      1      25
Number of configured NAE monitors currently active in the system     7      50
Number of configured NAE scripts currently active in the system      1      12
```

For detailed information about the `show capacities-status` command, see the *Command-Line Interface Guide*.

- To use the Web UI, on the Overview page, look at the **Analytics** panel to see the total number of scripts, agents, and monitors compared to the total number supported on the switch.

For example, `Agents: 6/50` indicates that there are total of six enabled and disabled agents out of a maximum of 50 agents supported on this switch.

Overview



- To use the REST API, do the following:
 1. To get information about maximum number of scripts, agents, and monitors, send a GET request to the `/system` resource, specifying the `attributes=capacities` query parameter. For example:

```
GET /rest/v1/system/?attributes=capacities
```

The response body contains capacity information about multiple switch features. The information about the Network Analytics Engine starts with the string: `nae_`

For example:

```
{
  "capacities": {
    ...
    ...
    "nae_agents": 50,
    "nae_monitors": 150,
    "nae_notification_handlers": 8,
    "nae_notification_queue_size": 25000,
    "nae_notifications_rate": 25000,
    "nae_notifications_rate_per_monoitor": 500,
    "nae_scripts": 25,
    "nae_tsdb_disk_quota": 9,
    ...
    ...
  }
}
```

```
{
  "capacities": {
    ...
    ...
    "nae_agents": 50,
    "nae_monitors": 150,
    "nae_notification_handlers": 8,
    "nae_notification_queue_size": 25000,
    "nae_notifications_rate": 25000,
    "nae_notifications_rate_per_monoitor": 500,
    "nae_scripts": 25,
    "nae_tsdb_disk_quota": 9,
    ...
    ...
  }
}
```

2. To get information about the current number of scripts, agents, and monitors, send a GET request to the `/system` resource, specifying the `attributes=capacities_status` query parameter. For example:

```
GET /rest/v1/system/?attributes=capacities_status
```

The response body contains information about multiple switch features. The information about the Aruba Network Analytics Engine starts with the string: `nae_`

For example:

```
{
  "capacities_status": {
    ...
    ...
    "nae_agents": 6,
    "nae_monitors": 27,
    "nae_scripts": 7,
    ...
    ...
  }
}
```

```
}  
}
```

High switch CPU and memory usage are affecting switch performance

Symptom

A switch that has Network Analytics Engine (NAE) agents installed is experiencing high CPU usage, high memory usage, or both, and overall performance is affected.

Cause

The NAE is attempting to monitor too many switch resources. For example, a script uses wildcard characters in a URI to monitor all interfaces, ACLs, or VLANs, and the switch has hundreds or thousands of those items.

Action

1. Verify that the resources associated with the NAE are consuming the memory and CPU resources by entering the following commands in the switch CLI:
 - `show system resource-utilization daemon hpe-tsdbd`
 - `show system resource-utilization daemon hpe-policyd`
 - `show system resource-utilization daemon prometheus`

The response to the `show system resource-utilization daemon <daemon>` command shows the CPU and memory usage the specified daemon.

2. Identify installed scripts that include the wildcard character (*) in the URIs of monitors.

For example, the following scripts written by Aruba contain wildcard characters to specify resources that can exist in large numbers on a switch:

- `tx_rx_stats_monitor`
- `fault_finder`

In the Web UI, the **Script Details** page shows the contents of the script.

Look for monitor URIs that specify the wildcard character for resources that your switch has in large numbers. For example:

- `interfaces/*`
- `acls/*`
- `vlangs/*`

3. Delete agents associated with the scripts you identified.

Disabling the agent is not sufficient because the switch does not delete time series data when an agent is disabled.

Downloading NAE support files

You can download support information specific to the Aruba Network Analytics Engine (NAE) as a file in `tar.gz` format using TFTP, SFTP, or USB.

Procedure

From the CLI, use the `copy support-files` command and specify the NAE feature.

Example of copying NAE support files to a remote server using TFTP:

```
switch# copy support-files feature nae tftp://10.100.0.12/file.tar.gz vrf mgmt
```

Example of copying NAE support files to a remote server using SFTP:

```
switch# copy support-files feature nae sftp://root@10.0.14.206/file.tar.gz vrf mgmt
```

Example of copying the NAE support files to a storage drive connected to the USB port of the switch:

```
switch# copy support-files feature nae usb:/file.tar.gz
```

For detailed information about the `copy support-files` command, see the *Command-Line Interface Guide*.

NAE support file content

The Aruba Network Analytics Engine (NAE) support files include the following information:

- Configuration information about all the NAE agents and scripts
- All NAE scripts in the same format as when they are returned by the `show running-config` command
- The `nae_alert.journal` file, which contains the NAE alerts
- If your support specialist has executed commands to collect memory and CPU profile information before the `copy support-files` command is executed, the following information is included:
 - The memory and CPU profiles of the `hpe-policyd` daemon
 - The memory and CPU profiles of the `hpe-tsdbd` daemon

Error: "Switch time and browser time are not in sync"

Symptom

The Web UI displays a yellow caution triangle in the top banner and an error dialog box with the following title:

```
Switch time and browser time are not in sync
```

The content of the dialog box indicates how many seconds the switch time is ahead of or behind the browser time, and states that the information displayed in the Web UI might not be accurate.

In addition, time series graphs on the Analytics page might be missing expected data, might have data shown with inaccurate times, or might display incorrect data.

Cause

The switch and the client from which you are viewing the UI are not set to the same time. For example:

- The switch has been rebooted and the time has not been set correctly. For example, the switch could not reach its configured NTP server.
- Instead of using Coordinated Universal Time (UTC) with a time zone offset or getting the time from an NTP server, either the switch or the client is manually set to a different time.

Action

- Try clearing or resetting the web client browser cache.
- Ensure that the web client from which you are viewing the Web UI is set to a time zone based on UTC.

For example, if your workstation is set to Eastern Standard Time (EST), and you want to use Pacific Standard Time (PST), change the time by setting the time zone instead of by manually resetting the time.

- Ensure that the switch is set to use NTP or to a time zone based on UTC time.
 - NTP synchronizes the time of day among a set of distributed time servers and clients to correlate events when receiving system logs and other time-specific events from multiple network devices. All NTP communications use Coordinated Universal Time (UTC). To show the NTP status, use the `show ntp status` command.
 - For information about configuring the switch to use NTP, see the *Fundamentals Guide*.

After you configure the switch, clear the NAE data by entering the `clear nae-data` command from the manager context.

For example:

```
switch# clear nae-data
```

- If the switch is set to use NTP and there has been a significant clock change, clear the NAE data by using the `clear nae-data` command.

For example:

```
switch# clear nae-data
```

Analytics time series graph displays message instead of data: "Agent data not found, please verify..."

Symptom

The time series graph of a monitor displays the following message instead of graphed data:

```
Agent data not found, please verify that the agent has not be deleted
```

Cause

The switch operating system software was updated to a different software release, and the name of the agent after the update is not an exact match to the name of the agent before the update.

For example:

- Before the software update, there was an Analytics time series graph for an agent with the following name:

```
system_resource_monitor.1.1.default
```

- The same script for the new software has the name `system_resource_monitor` instead of the name `system_resource_monitor.1.1`.
- After the software update:
 - The name of the new agent is based on the new script name. The new agent name is the following: `system_resource_monitor.default`
 - The Web UI continues to display the Analytics graph of the older agent, `system_resource_monitor.1.1.default`, which has no data and displays the error message.

Action

1. Close the panel that is displaying the error message.
2. If there is no Analytics time series graph displayed with the new agent name, add the panel by clicking the + plus sign next to the agent in the Agents panel.

Inaccurate or no data displayed in analytics time series graph

Symptom

The time series graph of a monitor is missing expected data, has data shown with inaccurate times, or incorrect data.

For example, a chart that graphs when a port goes down shows a port as up when it is down, does not show any state for the port, or shows the event as happening at a time that does not match the actual event-- such as in future.

Solution 1

Cause

The switch and the client from which you are viewing the UI are not set to the same time. For example, instead of using Coordinated Universal Time (UTC) with a time zone offset or getting the time from an NTP server, either the switch or the client is manually set to a different time.

Action

- Try clearing or resetting the web client browser cache.
- Ensure that the web client from which you are viewing Web UI is set to a time zone based on UTC. For example, if your workstation is set to Eastern Standard Time (EST), and you want to use Pacific Standard Time (PST), change the time by setting the time zone instead of by manually resetting the time.
- Ensure that the switch is set to use NTP or to a time zone based on UTC time.
 - NTP synchronizes the time of day among a set of distributed time servers and clients to correlate events when receiving system logs and other time-specific events from multiple network devices. All NTP communications use Coordinated Universal Time (UTC).
To show the NTP status, use the `show ntp status` command.
 - For information about configuring the switch to use NTP, see the *Fundamentals Guide*.

After you configure the switch, clear the NAE data by entering the `clear nae-data` command from the manager context.

For example:

```
switch# clear nae-data
```

- If the switch is set to use NTP and there has been a significant clock change, clear the NAE data by using the `clear nae-data` command.

For example:

```
switch# clear nae-data
```

Solution 2

Cause

The switch operating system software was updated to a different software release.

Action

1. Clear the NAE data by entering the `clear nae-data` command from the manager context.

For example:

```
switch# clear nae-data
```

2. Examine the alerts for agent or script errors.
Ensure that the scripts loaded on the switch support the software release installed on the switch.
3. If a script does not support the current software release, delete the script and upload the replacement script.
4. After you upload replacement scripts, clear the NAE data again by entering the `clear nae-data` command from the manager context.

URI errors

Agents can encounter errors not related to script syntax. For example:

- The agent attempts to monitor a port that exists on the switch but is not up.
- The agent attempts to monitor a port that does not exist on this switch.
- The agent attempts to monitor a LAG that has not been configured.
- The specified attribute is not correct for the resource. For example, the query string of the resource URI specifies `nx_packets` instead of `rx_packets`.

LAGs and ports are examples of switch resources. Every switch resource is identified by a URI (Universal Resource Identifier), so the kinds of errors listed previously are a category of errors called URI errors.

Error: "The NAE Agent is not created...DB constraint violation errors"

Symptom

When you attempt to create an agent, the agent appears in the **Agents** panel with a red triangle error symbol and status of `Unknown`. The error message is the following:

```
The NAE Agent is not created. Please check hpe-policd logs for DB constraint violation errors.
```

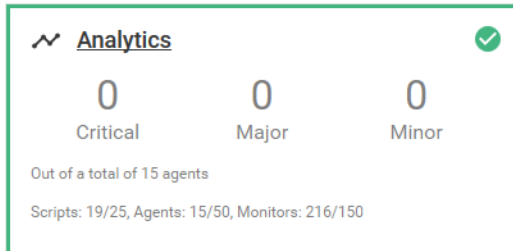
Cause

Attempting to create the agent resulted in creating more monitors than the NAE supports on the switch.

Action

1. Delete the agent that has the error.
2. To view the maximum number of monitors supported on the switch, see the **Analytics** panel on the Overview page of the Web UI.

In the following example, the **Analytics** panel does not display an agent error because the agent was not created. However, the **Analytics** panel does show that the number of monitors exceeds the maximum number of 150.



3. Reduce the number of monitors by identifying and deleting other agents.
Disabling an agent is not sufficient. The switch does not reduce the count of monitors used when agents are disabled.
4. Create the agents you want to use.

Error: "The NAE Agent has Python errors."

Symptom

An agent has the following error:

The NAE Agent has Python errors. Please check hpe-policyd logs for Python errors.

Solution 1

Cause

When the agent was created, the user supplied an invalid value for a required parameter.

Action

Change the configuration of the agent to include a valid value for the parameter.

Solution 2

Cause

The agent has an error that was not caused by an invalid required parameter.

For example, the value provided for an optional parameter might have caused a processing error.

Action

Examine the log files for the `hpe-policyd` daemon to identify which Python errors occurred, and take corrective action.

Error: "Timeseries data cannot be generated...The URI is invalid or not configured"

Symptom

An agent is in an error state and is not collecting data for one of its monitored resources.

The Agent Error message is the following:

Timeseries data cannot be generated due to the following agent error: The URI is invalid or not configured. Either please upload a new script with a valid URI, or configure the resource, disable the instance, and enable again: `<uri>`

In the message, `<uri>` is the resource that is invalid or not configured. For example:

```
/rest/v1/system/bridge/vlans/*/macs?count
```

Solution 1

Cause

If the resource URI is supported on this switch and operating system version, the most likely reason for the error is that the switch resource—such as a port—is not in an UP state.

Action

1. Resolve the problem that is causing the resource to be unavailable.
For example, if a port is administratively down, bring the port to an UP state.
2. Disable the agent.
3. Enable the agent.

Solution 2

Cause

The agent uses a resource ID from a user-supplied parameter, and the user supplied a resource ID that is invalid (such as a port that does not exist).

Action

1. Delete the agent.
2. Create and enable an agent that specifies a valid resource ID (such as port number).

Solution 3

Cause

The resource URI is not a user-supplied parameter, and is not supported on this switch and software version.

For example, a URI that includes `/bridge` in the path is not valid on 10.03 and later versions.

Action

- If the script was downloaded from the Aruba Solutions Exchange (ASE), replace or update the script with a script that supports the software version running on the switch.
 1. If the script was loaded on a switch running version 10.03 or later and the script name matches the new script name, you can use the update process to replace the script.
 2. If the script was loaded while the switch was running a version earlier than 10.03, you must delete the existing script, upload the new script, and create new agents.
- If the script not a script downloaded from the ASE, do the following:
 1. Delete all agents associated with the script.
Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.
 2. Modify the script to specify only URIs that are valid for this switch and operating system version.
 3. Replace the script on the switch with the modified script.

4. Create agents from the modified script.
5. Enable the new agents.

Error: "The script syntax is invalid"

Symptom

Script validation fails with the error: `The script syntax is invalid`

Cause

During the validation process, a syntax error was detected in the script Python code.

Action

1. Modify the script to correct the syntax errors.
2. Replace the script on the switch with the modified script.

Error: "The script agent syntax is invalid"

Symptom

Agent creation fails with the error: `The script agent syntax is invalid`

Cause

In the agent creation request, one the following occurred:

- The creation request specified a script name that does not match any script installed on the switch.
- A parameter value was used that does not match the syntax or type of the parameter defined by the script.

Action

Modify the agent creation request.

Error: "Sandbox timed out while running script"

Symptom

An agent is in an error state and is not collecting data. In the Web UI, the **Agent Details** page displays the error:

`Sandbox timed out while running script`

Solution 1

Cause

The switch is operating under a heavy load.

Action

1. If the agent is not already disabled, disable the agent.
2. When the operating load for the switch returns to normal, enable the agent.

Solution 2

Cause

The script that has many monitors and functions or executes loops that take a long time.

Action

1. Modify the script to correct the design issues, such as replacing a larger script with several smaller scripts.
 - a. Delete all agents associated with the script. Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.
 - b. Modify the script.
 - c. Replace the script on the switch with the modified script.
 - d. Create agents from the modified script.
 - e. Enable the new agents.

Error: "The agent instantiated sandbox has timed out"

Symptom

An agent is in an error state and is not collecting data. In the Web UI, the **Agent Details** page displays the error:

```
The agent instantiated sandbox has timed out
```

Solution 1

Cause

The switch is operating under a heavy load.

Action

1. If the agent is not already disabled, disable the agent.
2. When the operating load for the switch returns to normal, enable the agent.

Solution 2

Cause

The script has many monitors and functions or executes loops that take a long time.

Action

1. Modify the script to correct the design issues, such as replacing a larger script with several smaller scripts.
 - a. Delete all agents associated with the script.
Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.
 - b. Modify the script.
 - c. Replace the script on the switch with the modified script.
 - d. Create agents from the modified script.
 - e. Enable the new agents.

Error: "Unable to parse condition expression..."

Symptom

Time-series data for the monitored resource is being collected, but no alerts are triggered for the condition. In the Web UI, the **Agent Details** page displays the following error:

```
Unable to parse condition expression, invalid condition syntax <condition>  
<condition> is the condition expression that has the error.
```

For example:

```
Unable to parse condition expression, invalid condition syntax: rate  
/rest/v1/system/interfaces/*?attributes=link_resets per and 10 seconds > 1
```

In addition, the switch event logs might contain errors related to the rule that contains the condition. You can enter the `show event -d hpe-policyd` command to display events related to the Aruba Network Analytics Engine scripts.

For example:

```
switch# show event -d hpe-policyd  
2018-0one-09:09:21:49.906768|hpe-policyd|5504|LOG_ERR|AMM|-|Error executing NAE  
action CLI belonging to condition system_resource_monitor.1.0.default.condition_7  
and agent system_resource_monitor.default due to Command failed: non-zero  
exit status.
```

Cause

The cause is one of the following:

- There is a syntax error in the condition expression.

Condition expressions can include operators, functions, and keywords—all of which must be in the required sequence. For example:

- Correct syntax: `r.condition('{} > {}'.format(value, threshold))`
- Incorrect syntax: `r.condition('> {} {}'.format(value, threshold))`

- There is a mismatch between the number of parameters passed to the condition expression and the number of parameters expected.

For example, the following expression expects two inputs but is only passed one input:

```
r.condition('{} > {}'.format(value))
```

When the inputs to a function are a string and an array, the Python interpreter does not detect mismatches in parameters and values. For example, according to the Python interpreter, the expression `r.condition('>', [])` could be valid, so it does not return a syntax error during script validation. However when the script is run, the expression has no meaning because the comparator has no values to compare, so the "Unable to parse condition expression..." error is returned.

Action

1. Modify the script to correct the condition expression:
 - a. Delete all agents associated with the script.
Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.
 - b. Modify the script.
 - c. Replace the script on the switch with the modified script.
 - d. Create agents from the modified script.
 - e. Enable the new agents.

Error: "The CLI command is invalid"

Symptom

An agent is in an error state and is not successfully executing a CLI command specified in the script. The script has the following error:

```
The CLI command is invalid
```

Cause

A script action specifies a CLI command that is not valid on this switch and software version.

Action

1. Determine which CLI command is invalid. To see which command failed, in the Web UI, you can view the **Action Result** section of the **Alert Details** screen of the alert that generated this error.

You can search for the following error in the event log:

```
NAE Action CLI command <command> is not supported
```

<command> is the name of the unsupported command.

For example:

```
NAE Action CLI command datetime is not supported
```

2. Modify the script to change the CLI command or remove the script action that contains the command.
 - a. Delete all agents associated with the script.

Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.
 - b. Modify the script.
 - c. Replace the script on the switch with the modified script.
 - d. Create agents from the modified script.
 - e. Enable the new agents.

Error: "Command failed: non-zero exit status"

Symptom

An agent is in an error state and is not successfully executing a CLI command specified in the script, and the following conditions are true:

- The agent has the following error:

```
Command failed: non-zero exit status
```
- In the **Action Result Output** dialog box, the output of the CLI command is the following:

```
Cannot execute command. Command not allowed.
```
- The switch is configured to use a remote authentication and authorization service, such as TACACS+.

Cause

Network Analytics Engine (NAE) scripts execute CLI commands as the `admin` user, and do not attempt to authenticate before executing the CLI command. The remote authentication and authorization service is not allowing the `admin` user to execute commands without prior authentication.

Action

Configure the remote authentication and authorization service to allow authorization without authentication for the `admin` user.

Error: "The action is invalid"

Symptom

An agent is in an error state and is not successfully executing an action specified in the script. The script has the following error:

```
The action is invalid
```

Cause

A script action specifies an action that is not a CLI command and is not valid on this switch and software version.

Action

1. Determine which script action is invalid.

To see which command or action failed, in the Web UI, view the **Action Result** section of the **Alert Details** screen of the alert that generated this error.

2. Modify the script to change or remove the script action that contains the command.

- a. Delete all agents associated with the script.

Consider recording information about the agent names and parameter values so that you can create equivalent agents from the modified script.

- b. Modify the script.
- c. Upload the modified script.
- d. Create agents from the modified script.
- e. Enable the new agents.

ActionShell output error: "not available in enhanced secure mode"

Symptom

In the Web UI, an alert has an **Alert Details** dialog box that shows an ActionShell command that has a result of `ERROR`. The **Action Result Output** dialog box for that command includes the following error:

```
not available in enhanced secure mode
```

The script and the agent are not in an error state.

Cause

The switch is configured in enhanced secure mode.

Action

Do one of the following:

- Ignore the error. The enhanced secure mode of the switch denies access to shell commands by design.
- Change the configuration of the switch from enhanced secure mode to standard mode. For more information about enhanced secure mode, see the *Security Guide*.

Finding NAE scripts on the ASE website

No login is required to browse the website or view information about a script. To download a script or to view the source code from the website, you must be logged in. Alternatively, you can download scripts from the Web UI without logging in to the ASE.

Prerequisites

You must have access to the Aruba Solutions Exchange (ASE) website at:

<https://ase.arubanetworks.com/>

Procedure

1. On the top toolbar, select **SOLUTIONS**.
2. In the navigation pane, under **PRODUCTS**, select **NAE**.
3. To filter the list by additional criteria, select one or more tags under **TAGS**. For example:
 - To view Aruba certified NAE solutions, select the tag: **nae-aruba-certified**
 - To view solutions that are related to ports, select the tag: **port**
 - To view solutions that apply to your switch series, select the tag that contains the product number, for example: **8400x**


Filtering on multiple tags is treated as an OR operation. Solutions are displayed if they have any one of the selected tags.

Finding NAE scripts on the ASE using the Web UI


Prerequisites

You must be logged in to the AOS-CX Web UI.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the Analytics Dashboard, in the Scripts panel, click the  ASE download button to view and download scripts from the Aruba Solution Exchange (ASE). From the Web UI, only Aruba-certified scripts are listed. For other scripts, go to the ASE directly.

Scripts
clearcondition.action
com.arubanetworks.cpu_poll_interval
fault_finder_monitor
interface_link_state_monitor
interface_tx_rx_stats_monitor
system_resource_monitor

You can also access the Aruba Solution Exchange from the Script Management page where you select the  ASE button.

3. The Aruba Solution Exchange page is displayed, listing the available scripts. You can select a script and click **View Script** to see the programmatic script contents.

Installed	Name	Tags	Last Modified
<input checked="" type="checkbox"/>	daemon_resource_monitor.1.0	nae-aruba-certified, 8400x, nae, daemon, system resource	10/17/17 08:57:21
<input checked="" type="checkbox"/>	interface_link_flap_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, link-flaps, port	10/17/17 08:56:46
<input checked="" type="checkbox"/>	interface_link_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:57:00
<input checked="" type="checkbox"/>	interface_state_stats_monitor.1.0	interface, nae-aruba-certified, nae, 8400x, link, tx, rx, port	10/17/17 08:51:44
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_impact_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:57:53
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:54:27
<input checked="" type="checkbox"/>	port_admin_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:56:21
<input type="checkbox"/>	stp_bpdu_tcn_rate_monitor.1.0	8400x, nae, nae-aruba-certified, stp, spanning-tree, tcn, topology-cha	10/17/17 08:56:03

Including 8 of 8 - 8400X

Viewing recent changes to existing NAE solutions

No login is required to browse the website or view information about a script. To download a script or to view the source code from the website, you must be logged in. Alternatively, you can download scripts from the Web UI without logging in to the ASE.

Prerequisites

You must have access to the Aruba Solutions Exchange (ASE) website at:

<https://ase.arubanetworks.com/>

Procedure

1. Select the solution.
2. On the solution details page, view the recent changes in the **Recent Changes** box. You can view additional change information by clicking **More Details**.



Downloading or installing a script from the ASE using the Web UI

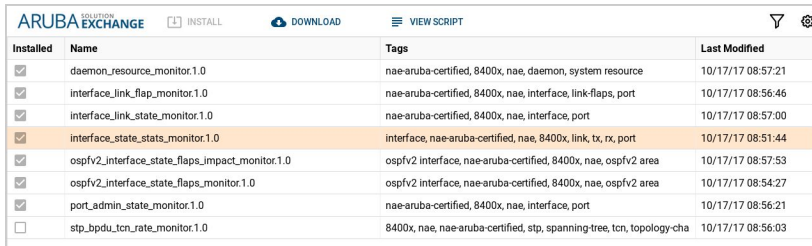
This procedure downloads an NAE script from the ASE to the specified location.

Prerequisites

You must be logged in to the AOS-CX Web UI. Administrator rights are required to install a script, but not required to download a script to your local system.

Procedure

1. Select **Analytics** from the navigation pane.
2. In the **Analytics Dashboard**, in the **Scripts** panel, click the  ASE download button to view and download scripts from the Aruba Solution Exchange (ASE). You can also access the Aruba Solution Exchange from the **Script Management** page where you select the  ASE button.
3. The Aruba Solution Exchange page is displayed, listing the available scripts.



Installed	Name	Tags	Last Modified
<input checked="" type="checkbox"/>	daemon_resource_monitor.1.0	nae-aruba-certified, 8400x, nae, daemon, system resource	10/17/17 08:57:21
<input checked="" type="checkbox"/>	interface_link_flap_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, link-flaps, port	10/17/17 08:56:46
<input checked="" type="checkbox"/>	interface_link_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:57:00
<input checked="" type="checkbox"/>	interface_state_stats_monitor.1.0	interface, nae-aruba-certified, nae, 8400x, link, tx, rx, port	10/17/17 08:51:44
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_impact_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:57:53
<input checked="" type="checkbox"/>	ospfv2_interface_state_flaps_monitor.1.0	ospfv2 interface, nae-aruba-certified, 8400x, nae, ospfv2 area	10/17/17 08:54:27
<input checked="" type="checkbox"/>	port_admin_state_monitor.1.0	nae-aruba-certified, 8400x, nae, interface, port	10/17/17 08:56:21
<input type="checkbox"/>	stp_bpdu_tcn_rate_monitor.1.0	8400x, nae, nae-aruba-certified, stp, spanning-tree, tcn, topology-cha	10/17/17 08:56:03

4. Select an installed script you want to download and click **Download**. If the script has not yet been installed, see the next step to install it.

You are prompted to either open the file or save the file to the location you select.



Saving will overwrite any existing file in the specified location with the same name.

Click **OK** or **Cancel**.

5. If the script has not yet been installed on the switch, you can select **Install** instead of Download. In the message dialog box, click **Confirm** to install the script or **Cancel**.

Downloading a solution from the ASE website to your workstation

Use this procedure when you want to download a solution to your workstation from the ASE website before you upload the script to a switch. Alternatively you can download a script directly to your switch from the Web UI.

Prerequisites

You must be logged in to the Aruba Solutions Exchange (ASE).

Procedure

1. On the ASE, search or browse for the solution you want to use.
2. In the **DETAILS** column, click the title of the solution.
The detailed information about the solution is displayed.
3. Verify that the displayed solution is the solution you want to download.

The **Description** shows information about the solution, including software required and platforms supported.

4. Click **Finish**.
5. In the dialog box, select **CONFIG**.
6. Do one of the following:
 - To download the solution, click **Download**.
 - To receive the solution as an email attachment, click **Email Me**.

The email is sent to the email address associated with the account you used to log in to the ASE.

- To send the solution to as an attachment to an email address other than the email address associated with the account you used to log in to the ASE, enter the email address in the **Email address** text box and click **Forward**.
7. Click **Close**.

Aruba publishes Aruba Network Analytics Engine scripts and examples to the following GitHub repository:

<https://github.com/aruba/nae-scripts>

You can use this repository like any other GitHub repository:

- You can fork and clone the repository into your own account profile.
- You can explore the scripts (in the `agents` folder) and examples and use them to create your own customized scripts.
- You can use the pull-request process to propose any additions or changes to the scripts.

The scripts and examples are organized by feature or protocol and then by supported platform. For more information about the structure and use of this repository, see the `README.md` file in the repository.

For more information about using GitHub, see:

<https://help.github.com/>

Aruba-certified scripts have been written and tested by Aruba. However, it is a good practice to examine all scripts before you upload them to understand what actions agents created from them will take on your switch.

It is also a good practice to save a switch checkpoint before you create and enable an agent for first time so that you can recover the switch configuration if an agent performs an undesirable action.

Reading the description of the script in the ASE or readme file on the GitHub repository can give you a good idea about what the script is intended to do, but to know what actions a script will take requires examining the Python code. Look for text phrases such as the following (the lack of closing parentheses in the search string is intentional):

- `action("CLI"`
- `ActionCLI`
- `action("SHELL"`
- `ActionShell`
- `requests.`

Action CLI functions that execute a switch CLI command. The command is provided as an argument to that function. Commands that include the text `config` or `configure` can be an indicator that the function is changing the configuration of the switch.

Action SHELL functions execute a command in the underlying operating system of the switch. The command is provided as an argument to that function.

Functions that begin with the text `requests.` can be REST API requests that use the requests library to make POST, PUT, or DELETE messages to the switch or to other network locations.

Python version and library support

Python version

The Aruba Network Analytics Engine supports scripts written in Python 3.5.X.

Python modules available

In addition to the standard Python libraries, the following Python modules are provided with the Aruba Network Analytics Engine:

python3-misc

Usage example:

```
import os
```

For more information, see:

<https://docs.python.org/3.5/library/os.html#miscellaneous-system-information>

python3-pkgutil

Usage example:

```
from pkgutil import extend_path
...
__path__ = extend_path(__path__, __name__)
...
```

For more information, see:

<https://docs.python.org/3.5/library/pkgutil.html>

python3-importlib

Usage example:

```
import importlib
```

For more information, see:

<https://docs.python.org/3.5/library/importlib.html>

python3-datetime

Usage example:

```
import datetime
```

For more information, see:

<https://docs.python.org/3.5/library/datetime.html>

python3-enum

Usage example:


```
from enum import Enum
...
class Color(Enum):
...     red = 1
...     green = 2
...     blue = 3
...
```

For more information, see:

<https://docs.python.org/3.5/library/enum.html>

python3-compression

Usage examples:

```
import zlib
import gzip
import bz2
import lzma
import zipfile
import tarfile
```

For more information, see:

<https://docs.python.org/3.5/library/archiving.html>

python3-numbers

Usage example:

```
import numbers
```

For more information, see:

<https://docs.python.org/3.5/library/numbers.html>

python3-selectors

Usage example:

```
import selectors
```

For more information, see:

<https://docs.python.org/3.5/library/selectors.html>

python3-signal

Usage example:

```
import signal
...
# Set the signal handler and a 5-second alarm
signal.signal(signal.SIGALRM, handler)
signal.alarm(5)
...
```

For more information, see:

<https://docs.python.org/3.5/library/signal.html>

tar

Usage example:

```
import tarfile
...
tar = tarfile.open("sample.tar.gz")
tar.extractall()
tar.close()
...
```

For more information, see:

<https://docs.python.org/3/library/tarfile.html>

Third-party Python libraries available

The following third-party Python libraries are provided with the Aruba Network Analytics Engine framework. To use the library in a script, import the library.

requests

The `requests` library is an HTTP library for Python.

Usage example:

```
import requests
...
r = requests.get('https://api.github.com/events')
r.json()
...
```

In a Python NAE script, when you specify the URI of a local switch resource, do not specify the external IP address or host name. Instead, use the `HTTP_ADDRESS` global constant.

For example, instead of the following:

```
LOCAL_SYSTEM = '127.0.0.1:5577'
uri = 'http://' + LOCAL_SYSTEM + '/rest/v1/system/ports/1%2F1%2F5?attributes=link_state'
```

Specify the following:

```
uri = HTTP_ADDRESS + '/rest/v1/system/ports/1%2F1%2F5?attributes=link_state'
```

For more information about the `requests` library, see:

<http://docs.python-requests.org/en/master/>

jsdiff

The `jsdiff` library is an MIT-licensed Python library used for comparing and patching JSON and JSON-like structures in Python.

Usage example:

```
import jsdiff
...
diff({'a': 1, 'b': 2}, {'b': 3, 'c': 4})
```

For more information, see:

<https://github.com/ZoomerAnalytics/jsdiff>

REST API version support

Network Analytics Engine (NAE) scripts can monitor REST v1 resources only.

Rules for script files

- Format: If you upload script using the REST API, you must convert the file to base64-encoding. If you upload the script using the Web UI, the conversion is done automatically.
- For information about coding style, naming conventions, and other best practices, see the style guide for Python code ([PEP 8](#)).

Script example

Script file `port_admin_state_monitor.1.0.py`

```

#-*- coding: utf-8 -*-
#
#Copyright (c) 2017 Hewlett Packard Enterprise Development LP
#
#Licensed under the Apache License, Version 2.0 (the "License");
#you may not use this file except in compliance with the License.
#You may obtain a copy of the License at
#
#http://www.apache.org/licenses/LICENSE-2.0
#
#Unless required by applicable law or agreed to in writing,
#software distributed under the License is distributed on an
#"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
#KIND, either express or implied. See the License for the
#specific language governing permissions and limitations
#under the License.

Manifest = {
    'Name': 'port_admin_state_monitor',
    'Description': 'Port Admin Status Monitoring Agent',
    'Version': '1.0',
    'Author': 'Aruba Networks'
}

ParameterDefinitions = {
    'port_id': {
        'Name': 'Port Id',
        'Description': 'Port to be monitored',
        'Type': 'string',
        'Default': '1/1/1'
    }
}

class Policy(NAE):

    def __init__(self):

#Port status
        url1 = '/rest/v1/system/ports/{}?attributes=admin'
        self.m1 = Monitor(
            url1,
            'Port admin status',
            [self.params['port_id']])
        self.r1 = Rule('Port disabled administratively')
        self.r1.condition('transition {} from "up" to "down"', [self.m1])
        self.r1.action(self.action_port_down)

#Reset policy status when port is up
        self.r2 = Rule('Port enabled administratively')
        self.r2.condition('transition {} from "down" to "up"', [self.m1])
        self.r2.action(self.action_port_up)

    def action_port_down(self, event):
        ActionSyslog(
            'Port {} is disabled administratively',
            [self.params['port_id']])
        ActionCLI("show lldp configuration {}", [self.params['port_id']])
        ActionCLI("show interface {} extended", [self.params['port_id']])
        if self.get_alert_level() != AlertLevel.CRITICAL:
            self.set_alert_level(AlertLevel.CRITICAL)
        self.logger.debug("### Critical Callback executed")

    def action_port_up(self, event):
        self.logger.info("Current alert level: " + str(self.get_alert_level()))
        if self.get_alert_level() is not None:
            ActionSyslog(

```


- Agent class constructor, which can contain the following:

Agent functions

The agent constructor can contain functions that are restricted to a specific monitor, rule, or action.

Examples of agent functions include the following:

- The `Graph`, `Title`, and `Baseline` functions.
- Functions related to agent events, such as `on_agent_restart` and `on_parameter_change`.
- Functions related to Aruba Analytics Data Collections (ADCs).

Monitors

A monitor uses the REST URI of a resource to define the resource on the switch to be monitored by the agent.

- All monitors generate time-series data about resource they monitor. The data can be viewed in time-series graphs in the Web UI.
- Optionally, a monitor can be associated with rules that execute actions when certain network conditions are true.

Rules

Rules define under which conditions to execute actions.

- Rules are optional.
- A rule can be defined with conditions that reference multiple monitors.
- A rule must contain a condition.
- A rule is not required to contain actions. A rule that does not contain actions does not generate an alert when an active condition transitions to `true`.

Conditions

The condition of a rule defines the circumstance under which actions, if any, are executed. The clear condition is an optional part of the rule that determines when a network condition or event is no longer occurring.

Actions

Actions are the tasks done by the agent in response to defined conditions. Actions can automate many of the things an administrator might do to troubleshoot network issues. Actions are part of a rule.

Actions are optional.

Header

The script header includes legal, copyright, and license information.

The following is an example of a script header for an Aruba-certified script:

```
#!/** coding: utf-8 **  
#  
#Copyright (C) 2017 Hewlett Packard Enterprise Development LP  
#  
#Licensed under the Apache License, Version 2.0 (the "License");  
#you may not use this file except in compliance with the License.  
#You may obtain a copy of the License at  
#  
#http://www.apache.org/licenses/LICENSE-2.0  
#  
#Unless required by applicable law or agreed to in writing,
```

```
#software distributed under the License is distributed on an
#"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
#KIND, either express or implied. See the License for the
#specific language governing permissions and limitations
#under the License.
```

Import statements

The import statements specify the Python modules that define functions used elsewhere in the script. If you are using several functions from the same module, consider importing the whole module.

The following example imports the `Enum` functions from the `enum` library and imports the `datetime` module.

```
from enum import Enum
import datetime
```

Manifest

The manifest introduces the script to the Python compiler. The manifest is required.

Example of a manifest:

```
Manifest = {
    'Name': 'com.arubanetworks.cpu_monitor_threshold',
    'Description': 'System CPU Monitoring Policy on configured threshold values',
    'Version': '0.1',
    'Author': 'Aruba Networks'
    'APIVersion': '1.0.0',
    'TargetPlatform': '8400',
    'TargetSoftwareVersion': '10.03'
    'Tags': ['application']
}
```

Required items

Name

Specifies a unique name for the script. The name must be a text string that starts with a letter or number and can contain alphanumeric characters and the special characters `.` (dot), `-` (hyphen), and `_` (underscore). The maximum length of the name plus the version must not exceed 80 characters.

Hewlett Packard Enterprise recommends that the name be a canonical name (CNAME). The canonical name follows the reverse domain name notation. For example: `com.myco.bgp_mon1.0`

Description

Describes what the script monitors. Must be a text string.

Version

Specifies version information about the script. The version is appended to the script name. The version must be a text string that starts with a letter or number and can contain alphanumeric characters and the special characters `.` (dot), `-` (hyphen), and `_` (underscore). The maximum length of the name plus the version must not exceed 80 characters. Version must not be empty.

Author

Specifies the author of the script. Must be a text string.

Optional items

APIVersion

Specifies which NAE script API version is required to run this script. This item is informational only. No validation or action is performed for this item.

Default: `1.0.0`

TargetPlatform

Specifies the switch platform on which the script is designed to be executed. If this item is not specified, the script is considered to be platform independent. This item is informational only. No validation or action is performed for this item.

Example: 8325

TargetSoftwareVersion

Specifies the switch firmware version or versions on which the script is designed to be executed. If `TargetPlatform` is not specified, the script is expected to work with any software version. This item is informational only. No validation or action is performed for this item.

Example: 10.03

Tags

Specifies one or more tags to provide information about the script.

The following tags are used by NetEdit to categorize and group the script and related alerts, and to automatically incorporate new NAE scripts into the health ranking for each of the following categories:

- application
- segmentation
- service (includes Client Services)
- routing
- bridging
- device
- other

Each tag must be a text string. This item is informational only. No validation or action by the NAE is performed for this item.

Examples:

- `'Tags': ['application']`
- `'Tags': ['segmentation','vlan']`

The manifest tags are not shown in the Web UI or the CLI, and they are not the same as the script tags in the ASE. To view the tags associated with a script installed on a switch, you can do one of the following:

- Look at the `Manifest` in the script by viewing the script contents on the Script Details page of the Web UI.
- Use the REST API to get information about the script.

Parameter definitions

The `ParameterDefinitions` statement defines the parameters that are used in the script.

When a user creates an agent, the user can specify values for the parameters. The user must specify values for parameters that the script identifies as required parameters.

When you write a script, Hewlett Packard Enterprise recommends that you create parameters for any user-configurable option in the script.

If the script has no parameters, omit the `ParameterDefinitions` statement.

The following example of a `ParameterDefinitions` statement defines the following parameters:

threshold

The `threshold` parameter is an integer and has a default value 90.

password

The `password` parameter is a string. The parameter is optional. If the user supplies a value, the user-supplied text is encrypted when the text is stored.

```
ParameterDefinitions = {
  'threshold': {
    'Name': 'Critical CPU Threshold value in percentage',
    'Description': ('When System CPU utilization exceeds this value, '
                  'set the policy status to Critical and the policy will log '
                  'system daemon CPU utilization details and CPU queue
statistics in syslog.'),
    'Type': 'integer',
    'Default': 90
  }
  'password': {
    'Name': 'Password',
    'Description': ' Service Password',
    'Type': 'string',
    'Encrypted': True,
    'Required': False
  }
}
```

ParameterDefinitions description

Structure and syntax

The `ParameterDefinitions` is a Python dictionary that contains one or more parameters. Each parameter is defined by a Python dictionary containing information specific to the parameter. At the time the agent is created, the default value is assigned to the parameter unless the user specifies a different value.

This part of the script is omitted if the script does not include user-defined parameters.

```
ParameterDefinitions = {
  '<param_name>': {
    'Name': '<descriptive_parameter_name>',
    'Description': '<description>',
    'Type': '<datatype>',
    'Default': <default_value>,
    'Encrypted': '<True_or_False>',
    'Required': '<True_or_False>'
  }
}
```

Components

Each parameter definition must contain the following:

The parameter declaration

Specifies a unique name for parameter. This name must be unique within the script. The name is a text string that can contain alphanumeric characters, - (hyphen), and _ (underscore). Example: `threshold`

Name

The name of the parameter. This name is displayed in the user interfaces.

Description

Describes the parameter. This description is displayed to users in the **Create Agent** screen of the Web UI.

Type

Specifies the datatype of parameter. Parameter type is one of the following values:

- integer
- string

Default

Specifies the default value of the parameter. At the time the agent is created, the default value is assigned to the parameter unless the user specifies a different value. You must specify a default if the parameter is part of a monitor or a condition. Otherwise, providing a default value is optional.

Encrypted

Specifies whether the user-supplied text is to be encrypted when the text is stored.

When `Encrypted` is `True`, the parameter is stored in a secure way. The parameter value will be available in plain text only when the Network Analytics Engine script is executed.

Default: `False`

Required

Specifies whether the user is required to supply a value. Use this attribute to force the user to provide a value for this parameter.

When `Required` is `True`, the parameter is required. If the user does not provide a value for this parameter, an error is returned and the agent is not created.

When `Required` is `False`, the parameter is optional. If the user does not provide a value for this parameter, the default value of the parameter is used.

Default: `False`

Agent class constructor

Syntax

The agent class constructor must begin with the following declaration:

```
class Agent (NAE):  
    def __init__(self):
```

The name of the constructor, `Agent`, is recommended, but using a different name does not result in an error.

Description

The main body of the script is the agent class constructor. The agent class constructor can contain:

- Monitors
- Conditions
- Rules
- Actions
- Agent functions such as `Graph`, `on_parameter_change`, and `Baseline`.
- Analytics Data Collections (ADCs)

Example

Example of an agent class constructor:

```
class Agent (NAE):  
    def __init__(self):  
        url = '/rest/v1/system/subsystems/*/*/power_supplies/*?attributes=status'  
        self.m1 = Monitor(url, name='System CPU utilization')  
  
        self.r1 = Rule('High CPU utilization')  
        self.r1.condition('{} > {}'.format, [self.m1, self.params['threshold']])  
        self.r1.action(self.action_high_cpu)
```

```

self.r1.action("CLI","top cpu")
self.r1.action("SHELL","ps -ef")

def action_high_cpu(self, event):
    self.set_alert_level(AlertLevel.CRITICAL)

        # CPU value when the Condition met
    cpu = event["value"]
    ActionSyslog("CPU Utilization at %s%." % cpu)
        ActionCLI("top cpu")
        ActionShell("ps -ef")

```

Graph

Syntax

```
Graph(<monitor-set>[, title=<title>][, dashboard_display=True])
```

Description

The `Graph` function enables you to select a monitor or group of monitors to display together in a graph on the Agent Details page in the Web UI.

Parameters

<monitor-set>

Specifies a comma-separated list of monitors to be included in this graph. The list must be enclosed in brackets (`[]`), which define a list in Python. The list must include at least one monitor.

Examples:

- `[self.monitor1]`
- `[self.m1, self.m2, self.m3]`

<title>

Specifies the title to be displayed for this graph. The definition of <title> must use the `Title` function.

For example: `title=Title("My graph title")`

`dashboard_display=True`

Specifies that this graph represent this agent on the Analytics Dashboard of the Web UI. You must specify this parameter for exactly one graph per agent.

If either of the following is true, a script error is generated:

- The script has multiple `Graph` functions defined, but none of those `Graph` functions set the parameter `dashboard_display=True`.
- The script has multiple `Graph` functions defined, and more than one of those `Graph` functions sets the parameter `dashboard_display=True`.

Usage

The Aruba Network Analytics Engine (NAE) provides the `Graph` function to enable you to select a monitor or group of monitors to display together in a graph on the Agent Details page in the Web UI.

The Agent Details page can include up to nine graphs. If the script does not include a `Graph` function, a default graph is created automatically. The default graph includes all the monitors in the script. You use the `dashboard_display=True` parameter to specify which graph is to represent the agent in on the Analytics Dashboard of the Web UI.

The graph displays alerts for all metrics being monitored. However, the graph can show graphed data for a maximum of eight metrics at a time. The metrics that are being shown on the graph are listed at the bottom of the graph.

When the agent is created, the NAE selects the most appropriate metrics to graph, up to a maximum of eight. After the agent is created, a Web UI user can customize the graph and choose which metrics to display.

Examples

In the following example, graph `g1` includes data from monitors `m1` and `m2`. The graph does not have a custom title and is not the graph that represents the agent in the Analytics Dashboard.

```
self.g1 = Graph([self.m1, self.m2])
```

In the following example, graph `g2` includes data from monitor `m1`. The graph has a custom title, but it is not the graph that represents the agent in the Analytics Dashboard.

```
title = Title("CPU utilisation by {} daemon ", [self.params["daemon_name"]])
self.g2 = Graph([self.m1], title=title)
```

In the following example, graph `g2` includes data from monitor `m1`. The graph has a custom title and it is the graph that represents the agent in the Analytics Dashboard.

```
title = Title("CPU utilisation by hpe-policyd daemon")
self.g2 = Graph([self.m1], title=title, dashboard_display=True)
```

Title

Syntax

```
Title("<title-string>"[, <params>])
```

Description

Creates a text string to be used as a custom title for a graph or for a predefined action.

Parameters

`<title-string>`

Specifies the python string to be used for the title. The string can contain parameters. To specify that a parameter is expected, use the following characters:

```
{ }
```

For example: "Print interface { } details"

`<params>`

Optional. Contains the parameters to be used in the title string. If multiple parameters are used, they must be passed to the function in the order they are required by the title string.

Usage

The Aruba Network Analytics Engine provides the `Title` function to enable you to specify custom titles when defining multiple graphs, or to show specific information about an action being performed by an agent. For example, if you define a title for each CLI action in a script, a user can see which action is being performed by looking at the Alert Details page in the Web UI.

Examples

The following example defines a title that is used for a graph. The title string includes a parameter.

```
title = Title("CPU utilisation by {} daemon ", [self.params["daemon_name"]])
self.g1 = Graph([self.m1], title=title)
```

The following example defines a title that is used for a CLI action. The title string does not include a parameter.

```
title = Title("Show switch configuration")
self.r.action("CLI","show running-config ", title=title)
```

on_agent_re_enable

Syntax

```
on_agent_re_enable(<agent>, <event>)
```

Description

Performs the specified actions when the NAE agent is re-enabled.

Parameters

<agent>

Specifies the agent that has the changed parameters.

Typically, this parameter is defined as the variable: `self`

<event>

Specifies a Python dictionary that contains the event information.

Typically, this parameter is defined as the variable: `event`

Usage

When an agent is disabled, it does not perform monitoring tasks. When an agent is re-enabled, it might not be in the correct state to perform its monitoring tasks. The Aruba Network Analytics Engine provides the `on_agent_re_enable` function to enable you to specify actions to perform to update an agent after it is re-enabled.

Example

```
def on_agent_re_enable(self, event):
    self.remove_alert_level()
    ActionCustomReport("Agent re-enabled")
```

on_agent_restart

Syntax

```
on_agent_restart(<agent>, <event>)
```

Description

Performs the specified actions when the NAE agent is restarted.

Parameters

<agent>

Specifies the agent that has the changed parameters.

Typically, this parameter is defined as the variable: `self`

<event>

Specifies a Python dictionary that contains the event information.

Typically, this parameter is defined as the variable: `event`

Usage

The NAE daemon might be stopped and restarted because of a management module failover operation or because a core dump operation has been performed.

When the NAE daemon restarts, it restarts the agents. When an agent is restarted, it might not be in the correct state to perform its monitoring tasks. The Aruba Network Analytics Engine provides the `on_agent_restart` function to enable you to specify actions to perform to update an agent after it is restarted.

Example

```
def on_agent_restart(self, event):
    ActionCLI("show core-dumps")
    self.remove_alert_level()
```

on_parameter_change

Syntax

```
on_parameter_change(<agent>, <params>)
```

Description

Performs the specified actions when a user changes the value of an agent parameter.

Parameters

<agent>

Specifies the agent that has the changed parameters.

Typically, this parameter is defined as the variable: `self`

<params>

Specifies a Python dictionary that contains the name of the changed parameter, the old value, and the new value.

Usage

Parameters that are defined in the `ParameterDefinitions` of a script can be changed by a user after the user creates the agent.

The Aruba Network Analytics Engine provides the `on_parameter_change` function to enable you to specify actions the agent is to perform when the user changes the value of one or more parameters.

Example

```
ParameterDefinitions = {
    'interface_id': {
        'Name': 'Interface Id',
        'Description': 'Interface to be monitored',
        'Type': 'string',
        'Default': '1/1/1'
    }
}

url1 = '/rest/v1/system/interfaces/{}?attributes=link_state'
self.m1 = Monitor(url1, 'Interface Link State')

def on_parameter_change(self, params):
    interface_id = params['interface_id']
    if interface_id["old"] != interface_id["new"]:
        self.remove_alert_level()
```

Baselines for dynamic thresholds for monitors

Many agents monitor things like network traffic, which can vary throughout the day.

Monitors usually contain rules to generate an alert—and possibly execute other actions—when something like the rate of incoming packets exceeds a certain value. This value is commonly referred to as a threshold.

It is difficult for a script writer to choose a threshold value that is appropriate for all switches in all networks. For example:

- The rate of incoming traffic for one switch might be relatively high, even under normal circumstances.
- On a different switch on a different network, that same incoming traffic rate might be considered abnormally high, and the network admin would want an alert to be generated in those circumstances.
- If the script writer chooses a threshold based on the lower incoming traffic rate, the agent monitoring the high-traffic switch will either generate numerous alerts or will remain in an alert state for most of the time. Because the threshold is lower than a traffic rate that is considered normal for that switch, the alerts generated because of the lower thresholds are considered "false positives" by the network administrator.

The `Baseline` function provides a way to specify thresholds that are appropriate to the network conditions on the switch. When an agent is created and enabled, it spends a specified amount of time "learning" about the data it is monitoring before it sets thresholds that are calculated based on what it learned.

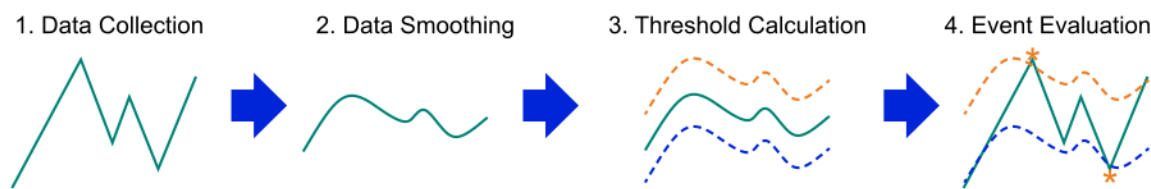
In addition, these thresholds are dynamic. The agent continues to learn about the data it monitors and the `Baseline` function adjusts the thresholds accordingly. For example, if the lower-traffic switch starts to get consistently higher incoming traffic rates, the `Baseline` function adjusts the thresholds to reflect the newly learned rates.

If desired, the script writer can specify default thresholds that can be used to determine when to set and clear alerts while the agent in a learning state. Otherwise the agent does not generate alerts while it learning about the data.

The methods used to determine the baseline from which to calculate the thresholds—both during the initial learning state and over time—depend on the algorithm selected in the `Baseline` function.

Baseline workflow and considerations

The following diagram shows a summary of the workflow of a `Baseline` function that uses `MaxAlgorithm` in its threshold calculations:



Choosing threshold multipliers

The high threshold is used in the determination of the condition which, when true, triggers the generation of an alert and, optionally, the execution of additional actions.

The low threshold is used in the rule to determine the clear condition, which—when true—triggers actions such as resetting the alert level.

At the end of the initial learning period and at the end of the continuous learning window, the `MaxAlgorithm` function calculates a single baseline value based on the smoothed data. In the `Baseline` function, you specify a high-threshold multiplier and a low-threshold multiplier to apply to this baseline value, resulting in the high threshold and the low threshold, against which datapoints are evaluated.

In effect, this strategy creates a "corridor" in which data can fluctuate without triggering alerts.

- If you choose a low number for the high-threshold multiplier, smaller variations from the baseline trigger alerts, which can result in alerts being triggered for what might be normal fluctuations in data.
- If you choose a high number for the high-threshold multiplier, the threshold might be exceeded less often, resulting in fewer alerts.

Effect of learning periods

Both the continuous learning window and the initial learning period are part of the look-back mechanism used by the `Baseline` function. These learning durations are used to determine how many datapoints to consider when calculating the baseline.

Using a period of time instead of specifying a number of datapoints is useful for situations in which knowing what a representative number of datapoints might be is difficult, but a representative amount of time is easier to estimate. However, getting enough data during the learning period to make a good calculation can depend on the length of the learning period and how typical the network conditions are when the agent is enabled.

Choosing a longer learning period enables the `Baseline` algorithms to distinguish important trends while ignoring temporary large fluctuations in data. Choose a learning period that is significantly longer than a situation that you would consider to be temporary for that kind of data.

For example:

- If the agent is enabled at a time when network traffic is low and the initial learning period is 10 minutes, the thresholds that are calculated are based on that low traffic. When more users arrive two hours later and network traffic increases, the measured traffic quickly exceeds the threshold.
- However, if you choose a learning period of one day, the "normal" fluctuations in traffic throughout the day are included in the baseline, resulting in thresholds that are appropriate to the situation.

Anomalies and baseline recalculations

Data that exceeds the high threshold is considered an anomaly.

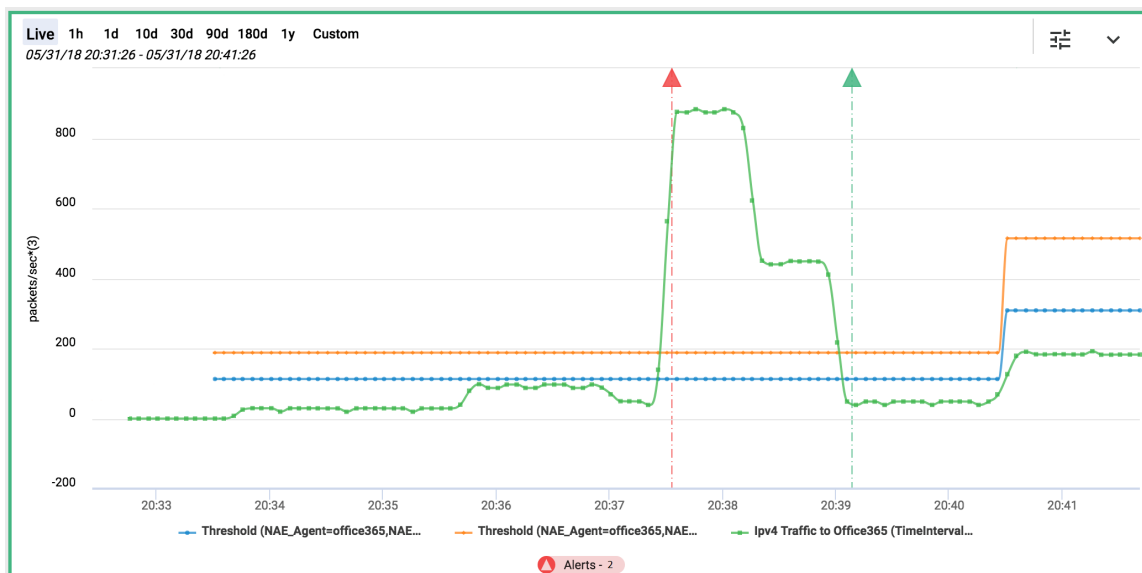
If an anomaly occurred during the continuous learning window, all data points that occurred during the continuous learning window are ignored and thresholds are not recalculated. This design prevents the thresholds from being reset as a result of a temporary "spike" in data.

If no anomalies occurred during the continuous learning window, the `Baseline` function updates the thresholds based on the latest result provided by the `MaxAlgorithm` function.

Example of baselines in a time series graph

The following is an example the time series graph for a monitor that includes baselines:

- The initial learning time is one minute.
- There are no default thresholds.



In this graph:

- The green line is the raw data.
- The orange line is the high threshold as calculated by the baseline.
- The blue line is the low threshold as calculated by the baseline.

The events in the timeline are as follows:

1. At 20:32:30, an agent is created and enabled. The baseline enters the learning state. Because the script did not specify default thresholds, there are no thresholds defined. In the graph, just the green line for the raw data is displayed.
2. At 20:33:30, the baseline exits the learning state and enters the active state:
 - The high threshold and the low threshold calculations are completed.
 - The graph begins the display of the orange line for the high threshold and the blue line for the low threshold.
 - The agent will generate an alert when the monitored traffic rate (in packets per second) exceeds the high threshold.
 - The agent will clear the alert when the monitored traffic rate (in packets per second) drops below the low threshold.
3. At 20:37:33, an alert is triggered because the monitored traffic rate exceeds the high threshold.
4. At 20:39:15, the alert is cleared because the monitored traffic rate (in packets per second) is lower than the low threshold.
5. At 20:40:30, the thresholds are updated.

In this case, the thresholds are set significantly higher because the algorithm includes all the data in its continuous learning window, which included the time in which the traffic rate was much higher than the previous threshold.

If the script specified a longer initial learning time, such as one day, the calculations used to create the thresholds can include the typical fluctuations in data that can occur, resulting in more appropriate thresholds and alerts that trigger only for significant anomalies.

Example of a script that uses baselines

The following is an example of a script that includes multiple monitors and baselines:

```
Manifest = {
    'Name': 'single_interface_tx_rx_stats_monitor',
    'Description': 'Policy to monitor tx/rx packets stats of a interface',
    'Version': '2.0',
    'Author': 'Aruba Networks'
}

ParameterDefinitions = {
    'interface_id': {
        'Name': 'Interface Id',
        'Description': 'Interface to be monitored',
        'Type': 'string',
        'Default': '1/1/1'
    }
}

class Agent (NAE):
    def __init__(self):
        # algorithm for dynamic Threshold calculation
        self.alg = MaxAlgorithm(continuous_learning_window="10m")
```



```

# rx packets
uri1 = '/rest/v1/system/interfaces/{}?attributes=statistics.rx_packets'
rate_m1 = Rate(uri1, "10 seconds", [self.params['interface_id']])
self.m1 = Monitor(
    rate_m1,
    'Rx Packets (packets per second)')
self.r1 = Rule('Rule for Monitor Interface rx Packets')
title1 = Title("Baseline for Interface rx Packets")
self.baseline1 = Baseline(self.m1, algorithm=self.alg, title=title1,
                           high_threshold_factor=2,
                           low_threshold_factor=1.2,
                           initial_learning_time='1d')
self.r1.condition('{} > {}'.format(self.m1, self.baseline1))
self.r1.clear_condition('{} < {}'.format(self.m1, self.baseline1))
self.r1.action("ALERT_LEVEL", AlertLevel.CRITICAL)
self.r1.clear_action("ALERT_LEVEL", AlertLevel.NONE)

# rx packets dropped
uri2 = '/rest/v1/system/interfaces/{}?attributes=statistics.rx_dropped'
self.m2 = Monitor(
    uri2,
    'Rx Packets Dropped (packets)',
    [self.params['interface_id']])

# tx packets
uri3 = '/rest/v1/system/interfaces/{}?attributes=statistics.tx_packets'
rate_m3 = Rate(uri3, "10 seconds", [self.params['interface_id']])
self.m3 = Monitor(
    rate_m3,
    'Tx Packets (packets per second)')
self.r3 = Rule('Rule for Monitor Interface tx Packets')
title3 = Title("Baseline for Interface tx Packets")
self.baseline3 = Baseline(self.m3, algorithm=self.alg, title=title3,
                           high_threshold_factor=2,
                           low_threshold_factor=1.2,
                           initial_learning_time='1d')
self.r3.condition('{} > {}'.format(self.m3, self.baseline3))
self.r3.clear_condition('{} < {}'.format(self.m3, self.baseline3))
self.r3.action("ALERT_LEVEL", AlertLevel.CRITICAL)
self.r3.clear_action("ALERT_LEVEL", AlertLevel.NONE)

# tx packets dropped
uri4 = '/rest/v1/system/interfaces/{}?attributes=statistics.tx_dropped'
self.m4 = Monitor(
    uri4,
    'Tx Packets Dropped (packets)',
    [self.params['interface_id']])

```

Baseline

Syntax

```

Baseline(<monitor>
[, title=<title>]
[, algorithm=<algorithm>]
[, high_threshold_factor=<high-multiplier>]
[, low_threshold_factor=<low-multiplier>]
[, initial_learning_time="<duration>"]
[, default_thresholds=(<lthresh>,<hthresh>)]
)

```

Description

The `Baseline` function calculates and sets low and high thresholds for a monitor based on data gathered during a learning period after an agent is enabled or restarted.

The calculated thresholds account for typical fluctuations in the data, enabling alerts to be triggered only for conditions exceeding the learned "normal" range. The `Baseline` function also calculates new thresholds and updates baselines at specified intervals, enabling the monitor to adjust as network conditions change.

Parameters

`<monitor>`

Specifies the monitor to which the baseline applies.

If the monitored URI contains wildcards, the baseline is calculated against all resources pointed to by expanding the wildcard. The same baseline thresholds apply to all resources.

`<title>`

Specifies the title to be displayed for this baseline. If specified, the title is displayed in the Web UI as the name of the baseline.

The definition of `title` must use the `Title` function.

For example: `title=Title("Baseline for CPU utilization")`

`<algorithm>`

Specifies the algorithm to use to calculate the baseline thresholds.

The supported and default algorithm is: `MaxAlgorithm`

`<high-multiplier>`

Specifies the high-threshold multiplier to apply to the value calculated by the algorithm.

The algorithm result multiplied by the high-multiplier determines the current high threshold for the monitor.

Typically, the high threshold is used to determine when the rule condition is `true` and therefore to trigger an alert and execute actions.

Default: 2

`<low-multiplier>`

Specifies the low-threshold multiplier to apply to the value calculated by the algorithm.

The algorithm result multiplied by the low-multiplier determines the current low threshold for the monitor.

Typically, the low threshold is used to determine when the clear condition is `true` and therefore to execute clear actions, such as clearing the alert.

Default: 1

`<duration>`

Specifies amount of time series data required for learning.

The initial learning time is used to determine how long to stay in the initial learning state after an agent is enabled. If there is already a time series in the database for the monitored resource, the `Baseline` function compares the amount of data to the initial learning time and includes that data.

For example, if the initial learning time is one hour and one or more hours of time series data exists in the database, the thresholds are set immediately. If there is 55 minutes of data available, the agent stays in the initial learning state for five minutes. If there is no data available, the baseline stays in the initial learning state for one hour.

During the initial learning state, the `baseline` function looks at the monitored data and uses its algorithm to determine normal patterns versus anomalies. Unless you specify values for the `default_thresholds` parameter, the agent does not generate alerts during the initial learning state.

Ensure that the initial learning time is long enough to gather enough data to determine normal versus abnormal patterns.

For example:

- If you are monitoring something that is updated infrequently, a longer initial learning time might be required to obtain enough data.
- If an agent is monitoring network traffic and that agent is enabled during a time that has unusually low traffic, the baseline calculates a low value, and "normal" traffic levels exceed the threshold, creating an alert. By setting a longer duration for the initial learning period, you maximize the ability of the algorithm to make calculations based on typical conditions.

The format for `<duration>` is `<number><unit>`, where `<unit>` is one of the following:

Value	Meaning
s	seconds
m	minutes
h	hours
d	days
w	weeks

Default: 1h

`<lthresh>` and `<hthresh>`

Specifies the default low and default high thresholds. These thresholds are used while the baseline is in the initial learning state. Both thresholds must be specified.

If default thresholds are not specified, alerts are not triggered for the monitor while the baseline is in the learning state.

Example

The following example contains a monitor that includes a baseline:

```
# algorithm for dynamic Threshold calculation
self.alg = MaxAlgorithm(continuous_learning_window="10m")

# rx packets
url = '/rest/v1/system/interfaces/{}?attributes=statistics.rx_packets'
rate_m1 = Rate(url, "10 seconds", [self.params['interface_id']])
self.m1 = Monitor(
    rate_m1,
    'Rx Packets (packets per second)')
self.r1 = Rule('Rule for Monitor Interface rx Packets')
title1 = Title("Baseline for Interface rx Packets")
self.baseline1 = Baseline(self.m1, algorithm=self.alg, title=title1,
    high_threshold_factor=2,
    low_threshold_factor=1.2,
    initial_learning_time='1d')
self.r1.condition('{} > {}'.format, [self.m1, self.baseline1])
self.r1.clear_condition('{} < {}'.format, [self.m1, self.baseline1])
self.r1.action("ALERT_LEVEL", AlertLevel.CRITICAL)
self.r1.clear_action("ALERT_LEVEL", AlertLevel.NONE)
```

MaxAlgorithm

Syntax

```
MaxAlgorithm(continuous_learning_window="<duration>")
```

Description

The `MaxAlgorithm` function smooths raw data within the initial learning time or the continuous learning window by calculating an average over time to find the greatest normal data during the past learning window.

Parameters

`continuous-learning-window="<learning-window>"`

Specifies how frequently to update baseline calculations when the baseline is in the active state, and how far to look back at data when recalculating the baseline threshold after an agent is restarted.

If an agent is disabled and then re-enabled, the calculation is based on the initial learning time defined by the `Baseline` function instead.

Default: 1h

For example, if the continuous learning window is one hour, and there is 55 minutes of stored data, the `MaxAlgorithm` function recalculates the baseline used for thresholds after five minutes.

The format for `<learning-window>` is `<number><unit>`, where `<unit>` is one of the following:

Value	Meaning
s	seconds
m	minutes
h	hours
d	days
w	weeks

Example

```
# algorithm for dynamic Threshold calculation
self.alg = MaxAlgorithm(continuous_learning_window="2h")
```

How the MaxAlgorithm function works

The `MaxAlgorithm` function uses smoothed data over time to find the maximum value and calculate a baseline of "typical" data. By setting a high threshold based on this calculation, the number of false positives can be reduced. The agent generates alerts for extreme anomalies only.

The `MaxAlgorithm` function is best for data that has no limit on its maximum value, such as incoming network traffic.

Smoothing data

The `MaxAlgorithm` function uses the "simple moving average" algorithm to smooth data.

In general, data smoothing is the process of detecting and removing "noise" data, allowing important patterns to remain.

In its formula, the simple moving average algorithm reduces the influence of datapoints that are not similar to the others because it includes multiple datapoints in the calculation. In addition, at each subsequent measurement point, the new data point is added and the oldest data point is removed, so short-term anomalies are eventually removed from the calculation.

For example, consider a simple moving average that is calculated based on five datapoints. If incoming traffic rates—measured in packets per second at regular intervals—are 10, 11, 100, 4 and 5:

- The highest datapoint, 100, is very different from the other datapoints. However the influence of that datapoint on the result is limited because the formula takes the average of five datapoints. In this case, the average is 26.
- In addition, over time, the result will be a lower number if the additional datapoints are also lower.

Choosing a continuous learning window

The longer the learning window, the more datapoints that are collected, and the less influence a short-term fluctuation in data has on the result.

Consider the network traffic example. If the continuous learning window is 10 minutes, the new thresholds will be calculated based on traffic during that time. If, during that time, network traffic is low, the `Baseline` function calculates the new thresholds based on that value. When the traffic increases, the traffic rate quickly exceeds the threshold and the agent generates an alert. If the network traffic was unusually low during the learning period, the typical network traffic can result in an undesired alert (false positive).

ADCs

Aruba Analytics Data Collections (ADCs) are script-defined rules to provide statistics about different types of network traffic passing through a switch. Network traffic statistics can be gathered based on many different frame or packet characteristics, including—but not limited to—the following:

- Source IP address (IPv4 or IPv6)
- Destination IP address (IPv4 or IPv6)
- Layer 3 (IP) protocol
- Layer 4 application ports
- Physical switch port

You can create ADCs and view ADC information through NAE scripts only. Multiple agents can use the same ADC.

ADCs are one of two types: IPv4 and IPv6. There can be multiple ADCs of each type.

ADCs are similar to ACLs in the following ways:

- ADCs and ACLs are installed in the switch hardware ASIC and share underlying mechanisms and limitations, including using TCAM entries.
- ADCs and ACLs are defined in switch configuration database in similar ways.

An ADC is composed of one or more ADC entries ordered and prioritized by sequence numbers. The lowest sequence number is the highest prioritized ADC entry. The ADC processes a packet sequentially against entries in the list until either the packet matches an ADC entry or the last ADC entry in the list has been evaluated.

Notes:

- When the agent instantiated from the script that defines the ADC is disabled or deleted, the NAE deletes the ADC. Disabling or deleting the agent is the only method to delete the ADC.
- If the switch is configured to create an automatic checkpoint (using the `checkpoint auto <time>` command), the NAE does not create the ADC.
- The `ADCList` class includes methods to add entries and to get entries. After an entry is added, it cannot be modified or deleted.

The following is an example of a script that defines an ADC list and entries:

```

Manifest = {
    'Name': 'adc_hit_counters_monitor',
    'Description': 'Network Analytics Agent Script to monitor'
                  'ADC hit counters',
    'Version': '1.0',
    'Author': 'Aruba Networks'
}

ParameterDefinitions = {
    'office365_traffic_bound': {
        'Name': 'office365_traffic_bound',
        'Description': 'Traffic flow to/from Office 365, '
                      'parameter options: \'to\' or \'from\', default is \'to\'',
        'Type': 'String',
        'Default': 'to'
    }
}

# IPv4Addresses contains a large list of IP addresses. For readability,
# a small sample of addresses is shown in this example.
IPv4Addresses = ["13.65.240.22/255.255.255.255", "13.66.58.59/255.255.255.255",
"13.70.156.206/255.255.255.255",
                  "13.71.145.114/255.255.255.255", "13.71.145.122/255.255.255.255",
"13.71.151.88/255.255.255.255",
                  "191.237.218.239/255.255.255.255",
"207.46.134.255/255.255.255.255",
                  "207.46.153.155/255.255.255.255"]

class Agent (NAE):

    def __init__(self):
        if str(self.params['office365_traffic_bound']) == 'to':
            self.adc_outgress = ADCList("office365_outgress", ADCList.Type.IPV4,
            "Traffic from Office365")
            for i in range(0, len(IPv4Addresses)):
                entry = ADCEntree(ADCEntree.Type.MATCH).dst_ip(IPv4Addresses[i])
                self.adc_outgress.add_entry(i, entry)
            ipv4_outgress_traffic = Rate(
                "/rest/v1/system/adc_lists/office365_
            outgress/ipv4?attributes=statistics.*", "15s")
            ipv4_outgress_sum = Sum(ipv4_outgress_traffic)
            self.ipv4_outgress_monitor = Monitor(
                ipv4_outgress_sum, "Ipv4 Traffic to Office365")
        elif str(self.params['office365_traffic_bound']) == 'from':
            self.adc_ingress = ADCList("office365_ingress", ADCList.Type.IPV4,
            "Traffic to Office365")
            for i in range(0, len(IPv4Addresses)):
                entry = ADCEntree(ADCEntree.Type.MATCH).src_ip(IPv4Addresses[i])
                self.adc_ingress.add_entry(i, entry)
            ipv4_ingress_traffic = Rate(
                "/rest/v1/system/adc_lists/office365_
            ingress/ipv4?attributes=statistics.*", "15s")
            ipv4_ingress_sum = Sum(ipv4_ingress_traffic)
            self.ipv4_ingress_monitor = Monitor(
                ipv4_ingress_sum, "Ipv4 Traffic from Office365")
        else:
            raise Exception("Invalid Parameters, "
                "please create agent with parameter 'to' or 'from',
            default: \'to\'")

```

ADCList class

Syntax

```
ADCList("<name>", <type>[, "<description>"])
```

Description

Python class for an Analytics Data Collections (ADC) list. Creates and returns the ADC list.

Parameters

<name>

Specifies the name of the list. The format of *<name>* is a Python text string.

<type>

Specifies the type of ADC.

Valid values are the following:

- `ADCList.Type.IPV4`
- `ADCList.Type.IPV6`

<description>

Description of the ADC list. The format of *<description>* is a Python text string.

Methods

```
add_entry(<seq>, <adc_entry>)
```

Adds the ADC entry object *<adc_entry>* to the ADC list at sequence number *<seq>*.

The sequence number is a positive integer that is unique within the ADC list. If the sequence number exists in the ADC list, an agent error is generated. Range: 0 through 4294967295

Example:

```
self.adc.add_entry(10, entry1)
```

```
get_entries()
```

Returns a Python list of all ADC entries in the ADC list.

Example:

```
self.adc.get_entries()
```

Example

```
self.adc_outgress = ADCList("office365_outgress", ADCList.Type.IPV4, "Traffic from Office365")
```

ADCEntry class

Syntax

```
ADCEntry(<type>)
```

Description

Python class that represents an entry in an Analytics Data Collections (ADC) list. Creates the ADC entry. Returns the ADC entry object.

Parameters

<type>

Specifies the action to be taken by the ADC entry.

Valid values are the following:

```
ADCEntry.Type.MATCH
```

Update statistics for packets that match this ADC entry.

ADCEntry.Type.IGNORE

Take no action for packets that match this ADC entry.

Methods

```
src_ip("<IP-ADDR>[/<MASK>]")
```

Adds the source IP address information to the entry.

Returns the ADC entry object.

<IP-ADDR>

Specifies the IP address.

/<MASK>

Specifies the subnet mask.

Example:

```
src_ip("10.0.0.1/255.255.255.255")
```

```
dst_ip("<IP-ADDR>[/<MASK>]")
```

Adds the destination IP address information to the entry.

Returns the ADC entry object.

<IP-ADDR>

Specifies the IP address.

/<MASK>

Specifies the subnet mask.

Example:

```
dst_ip("10.0.0.2/255.255.255.255")
```

```
src_port("<PORT>")
```

Adds the source port information to the entry.

Default: If no source port is specified, traffic ingressing any source port is matched.

Returns the ADC entry object.

<PORT>

Specifies the port number.

Example:

```
src_port("1/1/2")
```

```
src_l4_port(<L4-PORT>)
```

Adds the layer 4 source port information to the entry.

Returns the ADC entry object.

<L4-PORT>

Specifies the layer 4 port number.

Example:

```
src_l4_port(6640)
```

```
dst_l4_port(<L4-PORT>)
```

Adds the layer 4 destination port information to the entry.

Returns the ADC entry object.

<L4-PORT>

Specifies the layer 4 port number.

Example:

```
dst_l4_port(8080)
```

```
protocol(<protocol>[, flags={<flags>}])
```

Adds the network protocol information to the entry, which enables you to match or ignore packets based on the network protocol used or based on specific protocol flags.

Returns the ADC entry object.

`<protocol>`

Specifies the network protocol.

Use the following value to specify the TCP protocol:

```
ADCEntry.Protocol.TCP
```

```
flags={<flags>}
```

Sets protocol flags for this ADC entry. Contains a comma-separated list of flag names and values in the following format:

```
"<flag-name>": True
```

If an entry sets multiple flags, the packet matches the entry if all the flag conditions match.

The following flag names are supported for the TCP protocol:

```
tcp_urg
```

Urgent.

```
tcp_ack
```

Acknowledgment.

```
tcp_psh
```

Push buffered data to receiving application.

```
tcp_rst
```

Reset the connection.

```
tcp_syn
```

Synchronize sequence numbers.

```
tcp_fin
```

Finish connection.

```
tcp_established
```

Established connection.

Default: No flags are set.

Example:

```
protocol(ADCEntry.Protocol.TCP, flags={"tcp_ack": True, "tcp_psh": True, "tcp_rst": True, "tcp_syn": True, "tcp_fin": True})
```

Example

```
entry1 = ADCEntry(ADCEntry.Type.MATCH).src_ip("10.0.0.1/255.255.255.255").dst_ip("10.0.0.2/255.255.255.255").protocol(ADCEntry.Protocol.TCP)
```

Monitors

In a script, a monitor defines what resource the agent will monitor when the agent is enabled on a switch.

There is no rule that limits how many monitors can be specified in a single script. However the total number of scripts, agents, and monitors supported depends on the computing capacity of the switch.

A resource is any concept (received packets on a particular interface, user, CPU utilization, and so forth) that can be addressed and referenced using a URI.

Monitor function

The monitor is defined by the `Monitor` function, which has the following arguments:

- The item to be monitored—expressed as the REST URI of a system resource. This URI must include a query string that specifies the attribute or attributes to be monitored.
- If the URI includes a parameter, the name of the parameter.

- The name of the monitor, which is a string that is unique among the monitors defined in that script.

In the example, the name of the monitor is: `Interface Received Bytes`.

The following is an example of a monitor that uses a parameter for the resource ID. The monitor, named `Interface Received Bytes`, records the received bytes of the interface that will be specified by the user when the agent is created.

```
self.monitor = Monitor('/rest/v1/system/interfaces/{}?attributes=rx_bytes',
    [self.params['iface_id'] ],name='Interface Received Bytes')
```

The following is an example of the same monitor except that it does not use a parameter for the resource ID of the interface. Instead, the resource ID is defined as `1%2f1%2f7`, which is the percent-encoded representation of the switch member/slot/port notation: `1/1/7`.

```
self.monitor = Monitor('/rest/v1/system/interfaces/1%2f1%2f7?attributes=rx_bytes',
    name='Interface Received Bytes')
```

In the previous example:

- The URI path is: `/rest/v1/system/interfaces/1%2f1%2f7`
- The resource ID portion of the path is: `1%2f1%2f7`
- The URI query string is: `?attributes=rx_bytes`

To get the URI of a resource to monitor, you can use the AOS-CX REST API Reference browser interface to access the REST API. For more information about the REST API and the Swagger UI, see the *REST API Guide*.

When you specify the monitored URI, you can also use variables to represent the URI. This strategy can simplify updating a script with the REST API version changes or to make code more readable when there are long URI paths.

For example:

```
#Received bytes on a user-specified interface
url = '/rest/v1/system/interfaces/{}?attributes=rx_bytes'
self.m1 = Monitor(
    url,
    'Interface Received Bytes',
    [self.params['iface_id']])
```

URIs for monitors

The URI of a monitored resource is composed of several components, including the host name or host IP address, the path, and the query string that specifies the attribute to monitor.

In a Python NAE script, when you specify the URI of a local switch resource to monitor, omit the server URL portion of the URI. Specify only the path and query string portion of the URI.

For example:

```
cpu_uri = '/rest/v1/system/daemons/{}?' \
    'attributes=resource_utilization.cpu'
```

If the switch is a VSX switch, you can specify the peer switch by prepending `/vsx-peer` to the path portion of the URI.

For example:

```
peer_cpu_uri = '/vsx-peer/rest/v1/system/daemons/{}?' \
    'attributes=resource_utilization.cpu'
```

Path component of the URI

The path component of the URI is everything before the question mark (?) character. The path is a hierarchy. The forward slash (/) character indicates the hierarchical relationship between resources.

Because the forward slash character has special meaning, forward slash characters that are part of the URL path must be percent-encoded, with the code `%2F` representing the forward slash.

For example, the following URI specifies interface 1/1/5:

```
/rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.rx_packets
```

The path portion of a monitored resource URI can contain the following:

- [The wildcard character](#)
- [User-defined parameters](#)

Query component of the URI

The query component is sometimes called the query string. In a URI, the question mark (?) character indicates the beginning of the query component.

The query component of a monitored URI must contain at least one *key=value* pair. Multiple pairs are separated by the ampersand (&) character.

The following keys are supported:

attributes

When the key is *attributes*, *value* is the name of an attribute. The attribute must be an attribute of the resource specified by the path component of the URI. Multiple values are supported. Commas separate the values.

For example:

```
/rest/v1/system/vlans?depth=1&attributes=id,name,type
```

When a URI defines a monitor in an Aruba Network Analytics Engine script, attribute values in the query string support an additional dot notation that the Network Analytics Engine uses to access additional information. For example:

```
/rest/v1/system/subsystems/management_module/1%2F5?attributes=resource_utilization.cpu
```

This dot notation is supported for certain URIs that define monitors in Network Analytics Engine scripts only.

filter

When the key is *filter*, *value* is an *attribute-name:attribute-value* pair that specifies the name and value of an attribute. The attribute must be an attribute of the resource specified by the path component of the URI. Multiple attribute and value pairs are supported. Commas separate the pairs.

For example:

```
...&filter=type:vlan,admin_state:up
```

Examples of URIs for monitors

CPU utilization

```
/rest/v1/system/subsystems/system/base?attributes=resource_utilization.cpu
```

Memory utilization

```
/rest/v1/system/subsystems/system/base?attributes=resource_utilization.memory
```

Packets received on interface 1/1/5

```
/rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.rx_packets
```

Packets transmitted from a user-specified interface

```
/rest/v1/system/interfaces/{ }?attributes=tx_bytes
```

Link states of all physical interfaces

```
/rest/v1/system/interfaces/*?attributes=link_state&filter=type:system
```

Link states of all physical interfaces in an administrative down state

```
/rest/v1/system/interfaces/*?attributes=link_state&filter=type:system,admin_state:down
```

Wildcard characters in monitored URIs

The URI passed to the `Monitor` function can contain the asterisk (*) wildcard character instead of a component in the URI path.

For example:

- The following URI specifies the configuration attributes of all interfaces:
`"https://192.0.2.5/rest/v1/system/interfaces/*?selector=configuration"`
- The following URI specifies all routes regardless of VRF:
`"https://192.0.2.5/rest/v1/system/vrfs/*/routes"`

You can use wildcard characters in multiple places in the path. For example, the following `Monitor` function monitors the connection state of all BGP neighbors belonging to all BGP routers in the "red" VRF:

```
self.monitor = Monitor("/system/vrfs/red/bgp_routers/*/  
  bgp_neighbors/*?attributes=conn_state",  
  name="BGP Neighbor Connection State")
```

You cannot use a wildcard character as part of the query string. For example, you cannot use the wildcard character to specify all attributes of a BGP neighbor.

The wildcard character must replace the entire component in the path. The wildcard character is not part of a regular expression. For example:

- You can use a wildcard to specify all VRFs, but you cannot use a wildcard character to specify all VRFs that begin with the letter `r`.
- You can use a wildcard to specify all interfaces, but you cannot use the wildcard character to specify the interfaces in member 1, slot 2.

When designing scripts that use wildcard characters in monitored URIs, be aware that using a wildcard character for certain resources can result in high switch CPU and memory utilization, which can in turn affect the performance of the switch.

Using a wildcard for resources such as ACLs, interfaces, VLANs, or VRFs, might not result in performance issues in a customer environment that includes a small number of those resources. However, when the same script is installed in a customer environment that has a large number of those resources—such as 500 interfaces—thousands of time series instances are created.

Each monitored resource creates one time series. Each time series requires switch CPU, memory, and storage space in the time series database. Switch performance can be affected when the resident memory resources used by the NAE exceeds approximately 500 MB.

Monitoring the count of resources by including the count parameter in the URI—even on thousands of resources—does not cause performance issues.

Parameters in monitored URIs

You can use user-defined parameters to define the resource ID in the URI passed to the `Monitor` function. For example, instead of creating a monitor for a specific interface on a switch, you can define a parameter for that interface. Then, when the agent is created, the user supplies the identifier of interface to be monitored.

Parameters in monitored URIs can only be used to specify resource identifiers, which are the last part of the path before the query string. Using parameters to specify attributes or other URI components is not supported.

Parameters are indicated by braces `{}`.

The following example shows the use of a parameter for the ID of an interface:

```
self.monitor = Monitor("/system/interfaces/{}?attributes=rx_bytes",  
  [self.params["iface_id"] ],name="Interface Received Bytes")
```

In the example:

- In the URI, instead of specifying a specific interface, the URI contains the following characters to indicate that a parameter is expected:

```
{}
```

- The following argument indicates that the name of the user-defined parameter is `iface_id`:

```
[self.params["iface_id"] ],
```

- The `iface_id` parameter is defined in the `ParameterDefinitions` portion of the script:

```
ParameterDefinitions = {
  'iface_id': {
    'Name': 'Interface Id',
    'Description': 'Interface to be monitored',
    'Type': 'string',
    'Default': '1/1/1'
  }
}
```

Examples of invalid parameter definitions

The following parameter definition is invalid because it attempts to create a parameter for an attribute, such as received bytes or transmitted bytes:

```
# Example of invalid parameter usage: attribute
self.monitor = Monitor("/system/interfaces/1%2F1%2F5?attributes={}",
[ self.params["attr"] ], name="Interface Received Bytes")
```

The following parameter definition is invalid because it attempts to create a parameter for a component of the URI that is not the resource ID:

```
# Example of invalid parameter: URI component not resource ID
self.monitor = Monitor("/system/{}/1%2F1%2F5?attributes=rx_bytes",
[ self.params["resource"] ], name="Interface Received Bytes")
```

Slash (/) characters in monitored URIs

To avoid mistakenly interpreting slashes in resource ID as part of the URI path, slash (/) characters in a resource ID that is not a user-definable parameter must be encoded as: `%2f` or `%2F`.

In the path component of a URI, the slash (/) character is a reserved character that is a delimiter between path segments. For example:

```
path/to/resourceID
```

Resources on a switch might also use the slash character in their identifications. For example:

- An interface might be identified by a member/slot/port notation, such as: `1/1/7`
- A power supply unit might be identified by a member/PSU notation, such as: `1/3`

To correctly specify a resource ID in a monitored URI that does not include parameters, you must substitute `%2f` or `%2F` for each slash character in the resource ID.

The following example shows the definition of a monitor for interface `1/1/7` and does not use a parameter for the resource ID:

```
self.monitor = Monitor('/rest/v1/system/interfaces/1%2f1%2f7?attributes=tx_bytes',
'Monitored interface')
```

If the script defines a user-defined parameter to specify the resource ID, URL encoding for the slash character is not used when specifying the default value. Likewise, when a user enters a value for that parameter, they do not have to use URL encoding for the slash character because the Web UI automatically uses URL encoding when storing the value.

The following example shows parts of a script that defines `interface_id` as a user-defined parameter and then passes that parameter to a `Monitor` function and to an `ActionCLI` function.

```
ParameterDefinitions = {
  'interface_id': {
    'Name': 'Monitored Interface',
    'Description': 'The id of the monitored interface',
    'Type': 'string',
```

```

        'Default': '1/1/1'
    }
}
...
self.monitor = Monitor(
    '/rest/v1/system/interfaces/{}?attributes=tx_bytes',
    'Monitored interface',
    [ self.params['interface_id']]
...
ActionCLI('show interface {}', [self.params['interface_id']]
...

```

Attribute filters in monitored URIs

Some types of attributes can be filtered further by specifying an attribute name and value pair instead of just an attribute name.

For example, you might want to specify that only physical interfaces be monitored. There is no way to specify only the physical interfaces in the URI path. However, physical interfaces have an attribute called `type`, which has a value of `system`.

The `filter` keyword indicates that the value that follows the equal sign (=) is an attribute and value pair.

The filtered attribute must be the name of an attribute of the resource specified by the path component of the URI.

The attribute must be one of the following data types:

- integer
- string
- boolean

For information about the names, types, and values of attributes of a resource, see the AOS-CX REST API Reference.

The syntax of the attribute and value pair is the following:

```
attribute-name:attribute-value
```

Multiple attribute and value pairs are supported. Commas separate the pairs. For example:

```
...&filter=type:vlan,admin_state:up
```

Examples

The following example shows a monitored URI that specifies the received packets of all physical interfaces.

```
uri1 = '/rest/v1/system/interfaces/*?attributes=statistics.rx_packets&filter=type:system'
self.m1 = Monitor(uri1, 'rx on all physical interfaces')
```

The following example shows a monitored URI that specifies the received packets of VLAN interfaces that are in an `up` state.

```
uri2 = '/rest/v1/system/interfaces/*?attributes=statistics.rx_
packets&filter=type:vlan,admin_state:up'
self.m2 = Monitor(uri2, 'rx on UP VLAN interfaces')
```

Constructing a URI using the AOS-CX REST API Reference

To construct the URI of a resource to monitor, you can use the GET method in the AOS-CX REST API Reference to generate most of the URI. However, to do the following, you must edit the URL you copy and paste.

- Use the dot notation to specify a component of an attribute.
- Use a filter to specify an attribute name and value pair.

Procedure

1. Log into the switch from the AOS-CX REST API Reference:
 - a. Use a supported browser to access the switch at: `https://<IP-ADDR>/api/<IP-ADDR>` is the IP address or hostname of your switch.
 - b. Expand the **Login** resource by clicking the resource name, `/login`, or by clicking **Expand Operations**.
 - c. Enter your user name in the value of the **username** parameter.
 - d. Enter your password in the value of the **password** parameter.
 - e. Click **Submit**.

2. To expand the possible endpoints of a resource, click the resource name.

For example, CPU utilization is a kind of resource utilization, which is part of the subsystems resource collection:

Subsystem

GET	/system/subsystems
GET	/system/subsystems/{id1}/{id2}
PUT	/system/subsystems/{id1}/{id2}

3. Click the endpoint in the GET method to expand the collection.

Subsystem

GET	/system/subsystems
Implementation Notes	
Get a list of resources	
Parameter of the filter parameter type	
Response Class (Status 200)	
OK	

4. Navigate to the Parameters section and select the attribute or attributes you want to display.

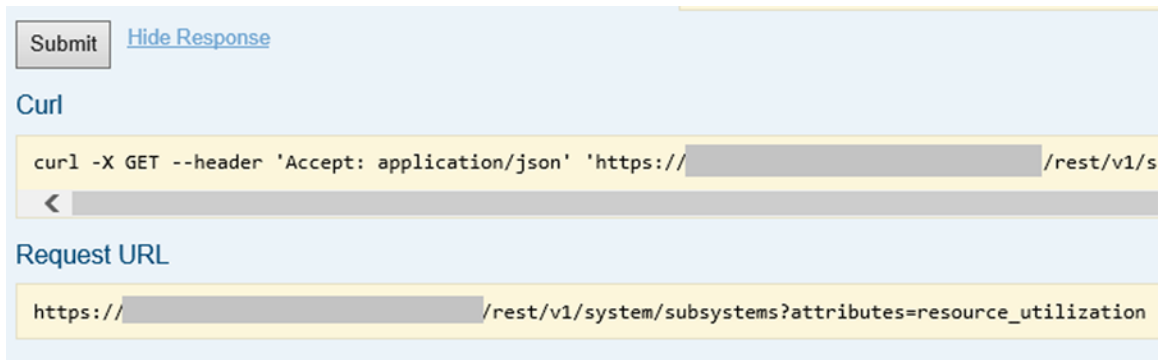
Parameters	
Parameter	Value
attributes	<div style="border: 1px solid black; padding: 2px;"> resets_requested resource_utilization selftest selftest_disable </div>

You can select multiple attributes:

- To select a range of attributes, click the first attribute, then press **Shift**, and then click the last attribute in the range you want to select.
 - To select attributes that are not adjacent in the list, press **Ctrl**, then click each attribute you want to select.
5. Click **Submit**.

The displayed information expands to show the curl command equivalent, the request URL, and the response body.

In the following example, the IP address of the switch is hidden by the gray box, and the response body is not shown.



6. Construct the monitor URI from the **Request URL** and the **Response Body** (if needed).
 - a. Copy and paste the **Request URL** into your script.
 - b. Remove the server information and any other information you specify elsewhere (such as the REST API version).

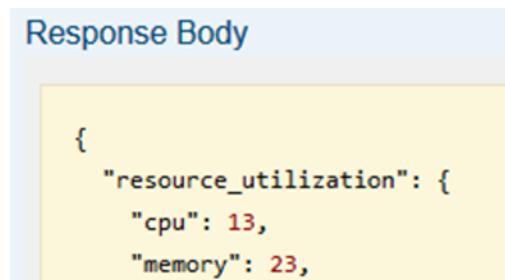
For example, the resulting URI is:

```
/rest/v1/system/subsystems?attributes=resource_utilization
```

- c. If you are specifying subcomponent of the attribute, view the **Response Body** and record the exact name of the attribute component to add to the query portion of the URI.

For URIs specifying monitors only, you can specify components within attributes by using a "dot" notation after the attribute name.

For example, In the **Response Body**, the CPU utilization is part of the `resource_utilization` resource. You can specify that only the CPU resources be monitored by adding `.cpu` after the attribute name. For example: `resource_utilization.cpu`.



The resulting example URI to monitoring CPU utilization is the following:

```
/rest/v1/system/subsystems?attributes=resource_utilization.cpu
```

- d. If you want to use the filter to specify both an attribute and a value, you must edit the URI to use the `filter` keyword and specify the attribute name and the attribute value.

Aggregate operators

Aggregate operators are aggregate operations that can be computed independently of the passage of time. Transition and ratio calculations are not considered aggregate operators.

If the monitored URI includes parameters and the monitor is an aggregate operator, the parameter must be passed to the aggregate operator instead of to the `Monitor` function.

In the examples of the functions, `uri2` is the following:

```
uri2 = '/rest/v1/system?attributes=resource_utilization_poll_interval'
```

The aggregate operators are the following:

Count

Counts the number of distinct time-series data points that have been generated and implicitly reflects the count of data points collected at every data collection interval.

Example:


```
count_m = Count(uri2)
self.m11 = Monitor(count_m, name='Resource Utilization Poll Interval')
```

Sum

Calculates the sum of the monitored resource values. When the monitored URI contains wildcard characters, this function sums over the current values of all resources pointed to by expanding the wildcards.

Example:

```
sum_m = Sum(uri2)
self.m9 = Monitor(sum_m, name='Resource Utilization Poll Interval')
```

Min or Max

Calculates the minimum or maximum of the monitored resource values. When the monitored URI contains wildcard characters, this function returns the minimum or maximum value over the current values of all resources pointed to by expanding the wildcards.

Example `Max`:

```
max_m = Max(uri2)
self.m15 = Monitor(max_m, name='Resource Utilization Poll Interval')
```

Average

Calculates the average of the monitored resource values. When the monitored URI contains wildcard characters, this function calculates and returns the average value over the current values of all resources pointed to by expanding the wildcards.

Example:

```
avg_m = Average(uri2)
self.m17 = Monitor(avg_m, name='Resource Utilization Poll Interval')
```

Aggregate functions

Aggregate functions are time-dependent and are computed over a time interval. Aggregate functions include the `rate` function and functions with names that include the keyword `OverTime`. Aggregate functions must specify a time interval.

Transition and ratio calculations are not considered aggregate functions.

If the monitored URI includes parameters and the monitor is an aggregate function, the parameter must be passed to the aggregate function instead of to the `Monitor` function.

For example:

```
uri2 = '/rest/v1/system/interfaces/{}?attributes=statistics.rx_packets'
avg_m = AverageOverTime(uri2, "1 hour", [self.params['interface_id']])
self.m2 = Monitor(avg_m, 'Rx Packets')
```

In the examples of the functions, `uri2` is the following:

```
uri2 = '/rest/v1/system?attributes=resource_utilization_poll_interval'
```

The aggregate functions are the following:

CountOverTime

Counts the number of distinct time-series data points that have been generated during the specified period of time.

Example `CountOverTime`:

```
count_over_m = CountOverTime(uri2, "5 minutes")
self.m2 = Monitor(
    count_over_m, name='Resource Utilization Poll Interval')
```

SumOverTime

Calculates the sum of the monitored resource values that occurred during the specified period of time. When the monitored URI contains wildcard characters, this function sums over the current values of all resources pointed to by expanding the wildcards.

Example `SumOverTime`:

```
sum_over_m = SumOverTime(uri2, "5 minutes")
    self.m10 = Monitor(
        sum_over_m, name='Resource Utilization Poll Interval')
```

`MinOverTime`

Calculates the minimum of the monitored resource values. When the monitored URI contains wildcard characters, this function returns the minimum value over the current values of all resources pointed to by expanding the wildcards.

Example `MinOverTime`:

```
min_over_m = MinOverTime(uri2, "5 minutes")
    self.m2 = Monitor(
        min_over_m, name='Resource Utilization Poll Interval')
```

`MaxOverTime`

Calculates the maximum of the monitored resource values. When the monitored URI contains wildcard characters, this function returns the maximum value over the current values of all resources pointed to by expanding the wildcards.

Example: `MaxOverTime`:

```
max_over_m = MaxOverTime(uri2, "5 minutes")
    self.m16 = Monitor(
        max_over_m, name='Resource Utilization Poll Interval')
```

`AverageOverTime`

Calculates the average of the monitored resource values over the specified period of time. When the monitored URI contains wildcard characters, this function calculates and returns the average value over the current values of all resources pointed to by expanding the wildcards.

Example `AverageOverTime`:

```
avg_over_m = AverageOverTime(uri2, "5 minutes")
    self.m18 = Monitor(
        avg_over_m, name='Resource Utilization Poll Interval')
```

`Rate`

Calculates the per-second average rate of increase for the monitored resource value over the specified period of time. When the monitored URI contains wildcard characters, this function calculates the per-second average rate of increase for each of the monitored resources pointed to by expanding the wildcards.

Example:

```
rate_m = Rate(uri2, "5 minutes")
    self.m8 = Monitor(rate_m, name='Resource Utilization Poll Interval')
```

Time interval

The format of the time interval is the following:

"<number> <unit>"

The value for <unit> is one of the following:

- second
- seconds

- minute
- minutes
- day
- days
- hour
- hours

For example: "12 seconds"

Nested aggregate functions and operators

Nesting an aggregate function or operator inside of an aggregate function enables you to combine the capabilities of both to create one result. The nested function or operator is executed first, and then the enclosing operator is executed.

For example, the following nested function calculates the rate of the switch resource and sums the calculated rates across all the URIs pointed to by the expanded wildcard:

```
Sum(Rate(<uri-with-wildcard>))
```

Rules

- You can nest up to one aggregate function or aggregate operator inside an aggregate operator.
For example: `Min(CountOverTime(<uri>))` and `Min(Sum(<uri>))` are supported.
- You cannot nest an aggregate operator or aggregate function inside an aggregate function.
For example, `Rate(Sum(<uri>))` and `CountOverTime(Min(<uri>))` are not supported.
- You cannot nest two of the same aggregate operators.
For example, `Sum(Sum(<uri>))` is not supported.

Aggregate functions in monitors and conditions

If an aggregate function or operator is used in the definition of the monitor, an aggregate function or operator can not be used in a condition expression for that monitor.

If the aggregate function is defined at monitor level, the time series graph for the monitored resource is a graph of the results of that aggregate function. In contrast, if the aggregate function is used in the condition expression, the time series graph for the agent in the Web UI is based on raw data.

The following example shows an aggregate function, `Average(uri)`, that is defined in the monitor. The function uses the wildcard to average the received packets from all the interfaces, resulting in a graph with a single curve, and triggering an alert when that average is over 50.

```
uri = '/rest/v1/system/interfaces/*?attributes=statistics.rx_packets'
avg = Average(uri)
self.monitor1 = Monitor(avg, name='Average Rx Packets on All Interfaces')
self.rule1 = Rule('Rx Packets')
self.rule1.condition('{} > 50', [self.monitor1])
```

The following example shows the `avg` aggregate function used in a condition expression. The monitor results in a graph that has multiple curves, with one curve corresponding to each interface received packet value. In the example, the condition `self.rule2.condition` triggers an alert when the average of received packets on an interface becomes greater than 50 within the past five minutes.

```
uri = '/rest/v1/system/interfaces/*?attributes=statistics.rx_packets'
avg = AverageOverTime(uri, "5 minutes")
self.monitor2 = Monitor(uri, name='Rx Packets on All Interfaces')
self.rule2 = Rule('Rx Packets')
self.rule2.condition('avg {} > 50', [self.monitor2])
```

Rules

Rules are optional components of the agent constructor, which is the main body of the Aruba Network Analytics Engine script.

When you monitor something on a switch, you can define rules to execute actions when certain conditions are true. For example, you might define a rule to send a message to the system log when CPU utilization exceeds 90% of capacity.

A script can have multiple rules using any combination of monitors previously specified on the script. A rule must include a condition, but actions are not required. A rule that does not include actions do not generate an alert when a condition transitions to `true` or to `false`.

A rule consists of the following components:

- The `Rule` function, to which a text string identifying the rule is passed as an argument. For example:
`Rule('High CPU utilization by ' + daemon_name)`
- A condition, defined by the `condition` function of the rule.
- Zero or more actions, each of which is defined by the `action` function of the rule. Actions are executed if the condition transitions from `false` to `true`.
- Zero or more clear actions, each of which is defined by the `clear_action` function of the rule. Clear actions are executed if the condition transitions from `true` to `false`.

```
self.r1 = Rule('Minor CPU Utilization')
self.r1.condition(
    '{} >= {}'.format(
        self.m1,
        self.params['minor_threshold']))
self.r1.action(self.action_minor_avg_cpu)
self.r1.clear_action(self.action_clear_minor_avg_cpu)
```

You can also create time-based rules that execute actions at a regular interval instead of in response to the conditions of a monitored URI.

For example, you can use the `every` keyword to create a rule that executes an action every 60 seconds:

```
self.r1 = Rule("Periodic callback")
self.r1.condition("every 60 seconds")
self.r1.action(self.my_periodic_callback)
```

You can use parameters to make the interval configurable. However time-based rules do not support clear conditions, conjunction operators, or disjunction operators.

Conditions

The condition and the clear condition are the part of the rule that determines which actions are executed.

When an agent is created, the condition associated with the rule is active and the clear condition is inactive. Time series data is collected every 5 seconds. Conditions and clear conditions—if any—are evaluated at each time series data collection point.

When the condition is active and transitions from `false` to `true`:

- The rule actions, if any, are executed.
- The condition becomes inactive. The condition does not become active again until the clear condition is `true` and becomes inactive.
- The clear condition becomes active.

When the clear condition is active and transitions from `false` to `true`:

- The clear actions, if any, are executed.
- The clear condition becomes inactive.
- The condition becomes active.

A condition can contain any number of monitors in its expression.

The following is an example of a condition in a rule:

```
# Monitoring average cpu value crossing 'minor_threshold' value that is set
# over a given time interval
self.rl.condition(
    '{} >= {}'.format(self.m1,
        self.params['minor_threshold'])
    self.rl.action(self.action_minor_avg_cpu)
    self.rl.clear_action(self.action_clear_minor_avg_cpu)
```

The `condition` function contains the condition expression (in quotes) and, optionally, any parameters used in the condition expression.

In the example, the condition expression is relatively simple. The condition expression uses the `>=` operator to compare the current CPU value to the value of the `minor_threshold` parameter.

Condition and clear condition expressions can be complex and include aggregation functions, pauses and durations in monitoring, and support for time-series data.

Clear conditions

The clear condition is an optional part of the rule that determines when a network condition or event is no longer occurring.

When an agent is created, the condition associated with the rule is active and the clear condition is inactive.

When the condition becomes `true`:

- The condition becomes inactive and the clear condition becomes active.
- The condition does not become active again until the clear condition is `true`.

Using a clear condition can help to prevent the numerous alerts that can occur when monitored data fluctuates narrowly above and below the threshold.

For example, consider a rule that sets the alert level to critical when CPU utilization exceeds 90%, and removes the alert level when the condition is no longer true.

- If you do not define a clear condition and the CPU utilization frequently fluctuates between 85% and 91%, the monitor generates alerts and then removes the alert level every time the fluctuation occurs.
- Instead, if you define a clear condition to be when the CPU utilization drops below 70%, the following behavior occurs:
 - The first time the CPU utilization exceeds 90%, the alert level is set to critical. The condition becomes inactive and it is no longer evaluated. Instead, the clear condition becomes active, and is evaluated at regular intervals.
 - The CPU utilization can fluctuate from above 90% to as low as 70%, an unlimited number of times. The alert level remains critical and additional alerts are not generated.
 - The clear action—removing the alert level—is executed only when the clear condition becomes true—the CPU utilization drops below 70%. The clear condition becomes inactive and is no longer evaluated. The condition becomes active again and is evaluated at regular intervals.

In effect, one of the ways you can use a clear condition is to use it in combination with the condition to set both a high threshold and a low threshold.

You define a clear condition using the `clear_condition` function. The `clear_condition` function contains the condition expression (in quotes) and, optionally, any parameters used in the condition expression.

Example of a clear condition in a monitor:

```
self.monitor = Monitor(AverageOverTime("/rest/v1/system/subsystems/management_
module/1%2F5?attributes=resource_utilization.cpu", "5 minutes")
self.rule = Rule("")
self.rule.condition("{} > 90", [self.monitor])
self.rule.action("ALERT_LEVEL", AlertLevel.CRITICAL)
self.rule.clear_condition("{} < 70", [self.monitor])
self.rule.clear_action(self.set_normal)
```

Condition expression syntax

The condition of a rule is a one-line expression that uses the following syntax, expressed in EBNF (Extended Backus-Naur) notation:

```
uri          = ? https://tools.ietf.org/html/rfc3986 ?
enum         = ? string ?
integer      = ? signed int64 ?
list_element = "'" , ( enum | integer ) , "'"
list        = list_element , { "," , list_element }
operator     = "<" | ">" | "==" | "!=" | "<=" | ">="
over        = "over" , integer , time
for         = "for" , integer , time
time        = "second" | "seconds" | "minute" | "minutes" | "day"
            | "days" | "hour" | "hours" | "day" | "days"
aggregator  = "count" | "sum" | "min" | "max" | "avg"
pause       = "pause" , integer , time
operation1  = uri , operator , list , [ for ] , [ pause ]
operation2  = aggregator , [ over ] , uri , operator , integer ,
            [ for ] , [ pause ]
operation3  = "rate" , uri , "per" , integer , time , operator , float ,
            [ for ] , [ pause ]
operation4  = "transition" , uri , "from" , list , "to" , list
operation5  = "ratio" , "of" , uri , "and" , uri | integer , operator , integer ,
            [ for ] , [ pause ]
operation6  = "every" , integer , time
```

A condition expression can be only in the format described by the operations `operation1` through `operation6`.

In `operation1`, the `>`, `<`, `<=`, and `>=` operators cannot be applied if list token is of type `enum`.

The behaviors of the supported functions depend on whether the monitored URI contains wildcards.

If an aggregation function is used in the definition of the monitor, the equivalent aggregation function cannot be used in a condition expression for that monitor.

Durations, evaluation delays, and pauses in condition expressions

The following are the behaviors when condition expression includes durations and either evaluation delays or evaluation pauses.

A duration is an number followed by a space followed by one of the values for the `time` keyword.

for followed by a duration

If a condition expression has the `for` keyword followed by a duration, the expression will be evaluated and monitored for that particular duration **before** the condition is considered `true` and any rule actions are executed.

For example, the following condition expression is `true` only if the count of the `conn_state` has been less than 5 for three minutes:

```
count /v1/system/vrfs/red/bgp_routers/bgp_neighbors/?attributes=conn_state < 5 for 3
minutes
```

pause followed by a duration

If a condition expression has the `pause` keyword followed by a duration, the condition will not be monitored for the specified duration after the condition is met. However, because the condition is true, any rule actions are executed before monitoring is paused.

For example, the following condition expression specifies that the spanning tree `tcn_count` will not be monitored for one minute after the `tcn_count` is greater than 10.

```
/v1/system/vlan/1/spanning_tree/tcn_count > 10 pause 1 minute
```

Using these kinds of expressions can reduce the number of actions taken—such as log entries generated—when a network condition is temporary, but still ensure that actions are taken at intervals the administrator considers to be reasonable.

Examples

```
rate /v1/system?attributes=resource_utilization_poll_interval > 300 per 1 hour for 5 hours
```

```
rate /v1/system?attributes=resource_utilization_poll_interval > 300 per 2 hours for 5 days
```

```
avg over 1 minute /v1/system/interface/1?attributes=statistics.tx_dropped < 5 for 2 minutes
```

```
rate /v1/system?attributes=resource_utilization_poll_interval > 300 per 10 seconds for 5 minutes
```

```
avg over 1 minute /v1/system/interface/1?attributes=statistics.tx_dropped < 5 for 2 minutes  
pause 10 seconds
```

Conjunction (AND), disjunction (OR), and multiple subconditions

When you define a condition, you can create an expression that combines multiple subcondition expressions using the operators AND, OR, and parenthesis.

The `condition` function returns `true` if the combination of the subconditions is true.

Operator precedence:

1. Parenthesis operator: `()`
2. AND operator
3. OR operator

For example, consider the following expression:

```
SubCondition1 AND SubCondition2 OR SubCondition3
```

Because there are no parentheses in this expression, the AND operation is evaluated first. The result of that evaluation is then evaluated against `SubCondition3` using the OR operator. The `condition` function returns the result of the second evaluation. The following table shows the results according to the evaluation of each subcondition:

Subcondition1	Subcondition2	Subcondition3	Result
true	true	false	true
true	false	false	false
false	true	false	false
false	false	false	false
true	true	true	true

Subcondition1	Subcondition2	Subcondition3	Result
true	false	true	true
false	true	true	true
false	false	true	true

The following example shows a condition with three different subconditions:

1. Resource Utilization Poll Interval > 50
2. Spanning tree hello time > 5
3. VLAN admin status = down

Example:

```
uri1 = '/rest/v1/system?attributes=resource_utilization_poll_interval'
self.m1 = Monitor(uri1, name='Resource Utilization Poll Interval')

uri2 = '/rest/v1/system?attributes=stp_config.hello_time'
self.m2 = Monitor(uri2, name='Spanning tree hello time')

uri3 = '/rest/v1/system/vlans/{}?attributes=admin'
self.m3 = Monitor(uri3, 'Vlan admin status', [self.params['vlan_id']])

self.r1 = Rule("Rule 1")
self.r1.condition('{{} > 50 AND {} > 5) OR {} == "down"', [self.m1, self.m2, self.m3])
self.r1.action(self.action_callback1)
```

Function behavior when monitored URI does not contain wildcards

The following are the behaviors of the supported condition expression functions when the monitored URI does not contain wildcards—and thus points to one specific resource:

count

This function counts the number of distinct time-series data points that have been generated and implicitly reflects the "count" of data points collected at each data collection interval.

For example, applying the `count` function to the following URI results in a value of 1.

```
/rest/v1/system/subsystems/chassis/base?attributes=resource_utilization.cpu
```

sum, min, max, and avg

Because there is just one resource, these aggregators return the current value of the resource. These aggregators only apply to resources represented by integers, such as number of packets received or CPU utilization.

Aggregator over time

When the keyword `over` follows one of the aggregators, the function behaves in a similar way but performs its aggregation "over" the specified time in the past instead of over the current value only.

rate

The `rate` function calculates the per-second average rate of increase of the monitored resource.

For example, applying the `rate` function to the following URI calculates the per-second average of the received packets on interface 1/1/5 for the last one minute:

```
/rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.rx_packets per 1 minute
```

transition

The `transition` evaluates to `true` if the specified resource changes state as defined.

The `transition` function only applies to columns of type `boolean`, `enum`, and `list`.

For example, the following expression is `true` when port 1/1/5 goes down:

```
transition /rest/v1/system/ports/1%2F1%2F5?attributes=admin from "up" to "down"
```

ratio

The `ratio` function calculates the ratio between numerator and denominator of the specified resource.

The `ratio` function only applies to columns of type `integer` and has the restriction that the part of URI preceding the token '?' be the same. That is, the `ratio` function can calculate the ratio between two attributes of the same resource, but not between attributes of different resources.

For example, to determine the ratio of the received packets to transmitted packets for interface 1/1/5, use the following expression:

```
ratio of /rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.rx_packets and  
/rest/v1/system/interfaces/1%2F1%2F5?attributes=statistics.tx_packets
```

Function behavior when monitored URI has wildcards

The following are the behaviors of the supported condition expression functions when the monitored URI contains wildcards and thus points more than one resource:

count

This function counts the number of distinct time-series data points that have been generated and implicitly reflects the "count" of data points collected at each data collection interval.

For example, the `count` function applied to the following URI results in the number of subsystems that are present:

```
/rest/v1/system/subsystems/*/base?attributes=resource_utilization.cpu
```

sum

When the monitored URI contains wildcards, this function sums over the current values of all resources pointed to by expanding the wildcard. This function only applies to columns of type `integer`.

For example, the `sum` function applied to the following URI is the sum total of the value of the CPU utilization of all subsystems configured in the system.

```
/rest/v1/system/subsystems/*/base?attributes=resource_utilization.cpu
```

min and max

When the monitored URI contains wildcards, these functions return the minimum or maximum of the current value of all currently present resources pointed to by expanding the wildcard. This function only applies to columns of type `integer`.

For example, the `min` function applied to the following URI returns the current minimum number of received packets across all configured interfaces. In other words, you can use this function to determine which interface received the fewest number of packets at this evaluation point.

```
/rest/v1/system/interfaces/*?attributes=statistics.rx_packets
```

avg

This function performs an average over the current values of all currently present resources pointed to by expanding the wildcard. This function only applies to columns of type `integer`.

For example, the `avg` function applied to the following URI returns the average number of transmitted packets across all configured interfaces:

```
/rest/v1/system/interfaces/*?attributes=statistics.tx_packets
```

Aggregator over time

When the keyword `over` follows one of the aggregators, the function behaves in a similar way but performs its aggregation "over" the specified time in the past instead of over the current value only. When the URI contains wildcards, the computation happens over each resource represented by the wildcard.

rate

The `rate` function calculates the per-second average rate of increase for each of the monitored resources.

For example, if interfaces 1/1/5 and 1/1/6 have been configured, using the following expression returns the ratio of received packets to transmitted packets for each of the two interfaces:

```
/rest/v1/system/interfaces/*?attributes=statistics.rx_packets per 1 minute
```

transition

Similar to `rate`, the `transition` function finds the transitions for each of the resources pointed to by expanding the wildcard in the monitored URI.

The `transition` function only applies to columns of type `boolean`, `enum`, and `list`.

For example, the following expression is `true` for every configured port that goes down:

```
transition(/rest/v1/system/ports/*?attributes=admin) from "up" to "down"
```

ratio

The `ratio` function finds the ratio between numerator and denominator for each of the resources pointed to by expanding the wildcard.

For example, if interfaces 1/1/5 and 1/1/6 have been configured, using the following expression returns the ratio of received packets to transmitted packets for each of the two interfaces:

```
ratio of /rest/v1/system/interfaces/*?attributes=statistics.rx_packets and  
/rest/v1/system/interfaces/*?attributes=statistics.tx_packets
```

Actions

Actions are executed depending on the truth value of the condition in a rule.

Actions might include things like setting the agent status, executing certain CLI commands, or generating system log messages.

Types of actions include the following:

Predefined action

Predefined actions are actions that are built in to the Aruba Network Analytics Engine framework.

Callback actions

Callback actions are callback functions defined in the script to perform additional processing.

Clear actions

Clear actions are predefined actions or callback actions that are executed only when the clear condition of a rule is active and transitions from `false` to `true`.

Predefined actions

Predefined actions are action functions that are built in to the Aruba Network Analytics Engine framework. These functions enable the agents of a script to do the following:

- Execute CLI commands in the AOS-CX network operating system.
- Execute shell commands in the underlying operating system of the switch.
- Send messages to the system log.
- Generate custom reports that communicate information collected by the agent.

ActionCLI, CLI action

Syntax

Inside a callback action:

```
ActionCLI("<command>"[, title=<title>])
```

Outside of a callback action:

```
self.r.action("CLI", "<command>"[, title=<title>])
```

Description

The CLI action executes the specified command from the switch CLI.

Parameters

<command>

A text string of the CLI command as it would be entered at a switch prompt. Examples:

- show vlan 100
- show running-config
- top memory

The string can be either a single line or a multiline string with line breaks using any Python line break technique.

<title>

Specifies the title to be displayed for this action. The definition of <title> must use the `Title` function.

For example: `title=Title("interface 1/1/1 no shutdown")`

Usage

- All CLI commands are supported as input to CLI actions.
- Command strings are passed without validation.
- The first CLI action executed in a script starts from the manager (#) command context. To execute commands in another context, you must include the commands to enter that context.

For example, to execute a configuration command, you must enter the correct configuration context, as in the following action:

```
ActionCLI("config\ninterface 1/1/1\nno shutdown")
```

Similar to a single CLI session, subsequent CLI actions in the script execute from the command context that is in place after the previous command is executed. In the previous example, the command context is the `config` context, and the next CLI action starts in the `config` context. Commands that cannot be executed in the `config` context fail.

- As a best practice, include commands in each CLI command to return to a standard command context appropriate for the script—such as the manager (#) context. For example:

```
ActionCLI("config\ninterface 1/1/1\nno shutdown\nexit\nexit")
```

- You can create a custom title for this action. The title you create can help the user of the Web UI to determine what action was executed. The title is displayed in the **Action Results** section of the **Alert Details** dialog box.

Examples

Examples of a multiline CLI actions:

```
ActionCLI("config\ninterface 1/1/1\nno shutdown\nexit\nexit")
```

Example of a CLI action that uses a parameter:

```
ActionCLI("show interface {}", [self.params["iface_id"]])
```

Example of a CLI action that uses a parameter and a custom title:

```
title = Title("Print interface {} details", [self.params["iface_id"]])  
self.ActionCLI("show interface {}", [self.params["iface_id"]], title=title)
```

ActionCustomReport

Syntax

Inside a callback action:

```
ActionCustomReport(<template>[, <params>][, title=<title>])
```

Description

Generates a multiple line report with customized content. The report content can be viewed in the Web UI from the alert details for the agent.

Parameters

<template>

Specifies the report template to use when generating this report. This template must be defined elsewhere in the script.

<title>

Specifies the title to be displayed for this action. The definition of *<title>* must use the `Title` function.

For example: `title=Title("Generate final MAC report")`

Usage

This function can only be used inside a callback action.

The script must contain the template (code) that generates and formats the report.

You can create a custom title for this action. The title you create can help the user of the Web UI to determine what action was executed. The title is displayed in the **Action Results** section of the **Alert Details** dialog box.

Examples

Example of calling the custom report action with a parameter:

```
ActionCustomReport(my_template, [self.params["author"]])
```

Example of generating a custom report in HTML format:

```
url = '/rest/v1/system/vlans/{}/macs/?count'
self.m1 = Monitor(url, 'Mac address count', [self.params['vlan_id']])
self.r1 = Rule('MAC Address Learnt')
self.r1.condition(
    '{} > 0 pause {} minutes',
    [self.m1, self.params['alert_pause']])
self.r1.action(self.gen_custom_report)

def gen_custom_report(self, event):
    vlan_id = self.params['vlan_id'].value
    type_s = ["dynamic", "static", "mclag", "vrrp"]
    mac_report = []
    for m in type_s:
        mac_all_dict = self.get_mac_result(vlan_id, m)
        mac_report.append(mac_all_dict)
    final_report = self.generate_htmlreport(mac_report)
    ActionCustomReport(final_report, title=Title("Generate final MAC report"))

def get_mac_result(self, vlan_id, mac_type):
    url = 'http://127.0.0.1:5577/rest/v1/system/vlans/' \
        + str(vlan_id) + '/macs?count=true&filter=from:' + \
        mac_type
    r1 = self.rest_get(url)
    result = str(r1.json()["count"])
    return result
```

```

def generate_htmlreport(self, mac_report):
    html_prefix = '''<html>
<head>
    <title>HPE-Aruba</title>
</head>
<body>
    <table border="1">
        <tr align="center">
            <th><td colspan="4"><b>Summary</b></td></th>
        </tr>
        <tr align="center">
            <th>
                <td colspan="4">
                    <b>Number of MAC learnt on various types</b>
                </td>
            </th>
        </tr>
        <tr>
            <th> DYNAMIC </th>
            <th> STATIC </th>
            <th> MCLAG </th>
            <th> VRRP </th>
        </tr>
        ''' + self.get_mac_htmlcontent(mac_report) + '''
    </table>
</body>
    </html>'''
    return html_prefix

```

For another example of a custom report, see the `ospfv2_interface_state_flaps_impact_monitor` solution on the Aruba Solutions Exchange.

ActionShell, SHELL action

Syntax

Inside a callback action:

```
ActionShell("<command>"[, title=<title>])
```

Outside of a callback action:

```
self.r.action("SHELL", "<command>"[, title=<title>])
```

Description

The SHELL action executes the specified command in the Bash shell of the underlying Linux operating system of the switch.

Parameters

<command>

A text string of the shell command as it would be entered at a Linux prompt.

The string can be either a single line or a multiline string with line breaks using any Python line break technique.

<title>

Specifies the title to be displayed for this action. The definition of *<title>* must use the `Title` function.

For example: `title=Title("ps -aux")`

Usage

The SHELL action takes a string as input and pipes it to the switch Bash shell.

For example, the following `ActionShell` request lists the current processes running on the switch:

```
ActionShell("ps -aux")
```

You can create a custom title for this action. The title you create can help the user of the Web UI to determine what action was executed. The title is displayed in the **Action Results** section of the **Alert Details** dialog box.



Shell commands provide advanced and powerful access to the switch. Improper changes can make the switch inoperable, potentially requiring the switch to be zeroized.

The shell is enabled on the switch by default. However, if the switch is configured to use enhanced secure mode, access to the shell is denied. Agent attempts to execute a shell action are denied, and the **Action Result** dialog box for that shell action indicates an error. However, the agent and the script are not set to an error state, so the error can be ignored. For more information about enhanced secure mode, see the *Security Guide*.

When shell actions enabled on a switch:

- SHELL actions execute with root privileges.
- The first SHELL action in a script executes from the root (/) directory.
- All Bash shell commands are supported as SHELL actions.
- Command strings are passed without validation.
- SHELL actions can be used with parameters.

Examples

Example of a SHELL action inside a callback action:

```
ActionShell("cd {}", [ self.params["path"]])
```

Example of a SHELL action outside of a callback action:

```
self.r.action("SHELL", "cd {}", [ self.params["path"]])
```

Example of a SHELL action that includes a custom title:

```
title=Title("cd {}", [ self.params["path"]])
self.r.action("SHELL", "cd {}", [ self.params["path"], title=title)
```

ActionSyslog, Syslog action

Syntax

Inside a callback action:

```
ActionSyslog(
"<log_message>"
[, severity=<severity>]
[, title=<title>])
```

Outside of a callback action:

```
self.r.action("Syslog",
"<log_message>"
[, severity=<severity>]
[, title=<title>])
```

Description

The `ActionSyslog` action enters the specified message in the event log. Users can access event log messages from the CLI, REST API, or Web UI of the switch.

Parameters

`<log_message>`

A text string of the message to be entered into the log.

The size and content of this message is subject to the same requirements as any other event log message on this system.

`<severity>`

Specifies the severity of the log message.

The Aruba Network Analytics Engine supports the following values:

- SYSLOG_ALERT
- SYSLOG_CRIT
- SYSLOG_ERR
- SYSLOG_WARNING
- SYSLOG_INFO

Default: SYSLOG_INFO

`<title>`

Specifies the title to be displayed for this action. The definition of `<title>` must use the `Title` function.

For example: `title=Title("Report CPU utilization")`

Usage

Use this action to record event log messages that can be accessed from the CLI, REST API, or Web UI of the switch.

This action sends a message to the event log. Log messages are labeled with a severity level. Only the specified message is sent to the log—additional information such as the monitor or agent name is not included.

You can create a custom title for this action. The title you create can help the user of the Web UI to determine what action was executed. The title is displayed in the **Action Results** section of the **Alert Details** dialog box.

Examples

The following example sends a message to the event log, using the parameter `message`, inside a callback action:

```
ActionSyslog("Sample message: {}", [self.params["message"]], severity=SYSLOG_WARNING, facility=SYSLOG_DAEMON)
```

The following example sends a message to the event log, using the parameter `message`, outside of a callback action:

```
self.r.action("Syslog", "Sample message: {}", [self.params["message"]])
```

Callback actions

Callback actions are callback functions that call other Python methods in the script to perform further processing. Callback actions can also call predefined actions and functions that have been imported from the supported Python modules or third-party libraries.

Callback actions are executed when the condition of a rule is true. If a callback action calls a predefined action, that predefined action is executed asynchronously. Callback actions time out after five minutes. This timeout interval is not editable.

This example contains a callback action, `action_high_poll_interval`, that calls the predefined actions

`ActionSyslog` and `ActionCLI`:

```
uri = '/rest/v1/system?attributes=resource_utilization_poll_interval'
self.m = Monitor(uri, name='My Monitor')
self.r = Rule('Resource Utilization Poll Interval')
self.r.condition('{} > 100 for 30 seconds', [self.m])
self.r.action(self.action_high_poll_interval)
```

```
def action_high_poll_interval(self, event):
    self.set_alert_level(AlertLevel.CRITICAL)
    ActionSyslog('High Poll Interval')
    ActionCLI("show running-config")
```

Aruba Network Analytics Engine agent sandboxes run in the default VRF. If a script has a function that requires access to the Internet, this function fails unless the switch administrator configures a DNS name server on the default VRF.

When a condition is met, an event is generated and the callback action is executed using the event details. The `event` parameter passed to the callback action contains those event details.

The `event` parameter contains a dictionary of key-value pairs. The dictionary includes the following keys:

`event['labels']`

The labels created by the time-series database.

`event['condition_name']`

The name of the condition that triggers the event.

`event['monitor_name']`

The name of the monitor.

`event['time_series']`

The name of the time series metric.

`event['value']`

The value of the time series metric.

`event['timestamp']`

The timestamp of the generated event.

`event['condition_description']`

The description of the condition that triggered the event.

Clear actions

You can define actions to execute when a clear condition is active and becomes `true`. These actions are called "clear actions" because they often do things like set an alert level back to a normal value or reset status.

For example, if you define an action to send a message to the log if the CPU utilization gets above 90%, you can also define an action to send a different message to the log when a CPU utilization that was above 90% drops to a lower utilization.

The clear action is executed only once—when the condition becomes invalid and the clear condition becomes valid:

- If the rule does not have a clear condition defined, the default behavior is that the clear condition becomes valid when the condition transitions from `true` to `false`.

For example, if the CPU utilization is greater than 90% (the condition is valid or `true`), the clear action is executed when the CPU utilization is equal to or less than 90% (the condition becomes invalid), but not if it continues to drop or if it remains less than 90% the next time the condition is evaluated. However, if the CPU utilization becomes greater than 90% again, the clear action will be executed the next time the CPU utilization drops to less than the threshold of 90%.

- If the rule has a clear condition defined, the clear condition becomes active the first time the condition becomes `true`. However, the clear condition does not become valid until it is both active and it becomes `true`.

For example, consider a clear condition that becomes true when the CPU utilization drops below 70%. After the CPU utilization is becomes greater than 90% the clear condition becomes active, but it does not become

true until the CPU utilization drops below 70%. Therefore the clear action is not executed until a CPU utilization that has been above 90% drops below 70%.

Both predefined and callback actions can be used in clear actions.

The following is an example of a rule that includes an action and two clear actions. In the example:

- One of the clear actions calls the CLI action to execute the `show running-config` CLI command:

```
self.rule.clear_action("CLI", "show running-config")
```

- The other clear action calls the `set_normal` function, which removes the alert level:

```
self.rule.clear_action(self.set_normal)
```

The following example shows a monitor that has a rule that defines a clear actions and a clear condition:

```
self.monitor = Monitor(AverageOverTime("/rest/v1/system/subsystems/management_
module/1%2F5?attributes=resource_utilization.cpu", "5 minutes"))
self.rule = Rule("")
self.rule.condition("{} > 90", [self.monitor])
self.rule.action("ALERT_LEVEL", AlertLevel.CRITICAL)
self.rule.clear_condition("{} < 70", [self.monitor])
self.rule.clear_action(self.set_normal)
```

```
def set_normal(self, event):
    self.remove_alert_level()
```

The following example shows a monitor with a rule that defines clear actions but does not define a clear condition:

```
self.monitor = Monitor(AverageOverTime("/rest/v1/system/subsystems/management_
module/1%2F5?attributes=resource_utilization.cpu", "5 minutes"))
self.rule = Rule("High CPU Utilization")
self.rule.condition("{} > 90" [self.monitor])
self.rule.action(self.action_high_cpu)
self.rule.clear_action("CLI", "show running-config")
self.rule.clear_action(self.set_normal)
```

```
def action_high_cpu(self, event):
    # CPU value when the Condition met
    cpu = event["/v1/system/subsystems/system/base?attributes=resource_utilization.cpu"]
    ActionSyslog("CPU Utilization at %d" % cpu)
    ActionCLI("top cpu")
```

```
def set_normal(self, event):
    self.remove_alert_level()
```

Alert level functions

The Aruba Network Analytics Engine provides several predefined functions related to alerts.

The alert level functions get, set, or remove the alert level of the agent:

```
get_alert_level()
```

Gets the current alert level of the agent.

Example:

```
status = self.get_alert_level()
```

```
set_alert_level(<Alert_Level>)
```

Sets the alert level of the agent.

Example:

```
self.set_alert_level(AlertLevel.CRITICAL)
```

The `<Alert_Level>` must be one of the following values:

```
AlertLevel.CRITICAL
```

The agent has detected a critical issue.

AlertLevel.MAJOR

The agent has detected a major issue.

AlertLevel.MINOR

The agent has detected a minor issue.

remove_alert_level()

Removes the alert level of the agent.

Example:

```
self.remove_alert_level()
```

Logging functions

The Aruba Network Analytics Engine provides the predefined `logger` functions for sending messages to the log and to set logging levels.

logger.setLevel(<level>)

Sets the default logging level specified by `<level>`. Valid values for `<level>` are the following:

- EMERG
- ALERT
- CRITICAL
- ERROR
- WARNING
- NOTICE
- INFO
- DEBUG

logger.log("<message>")

Sends the specified `<message>` at the default log level to the system log.

To send a message at a log level that is not the default log level, you can use one of the following functions:

- `logger.emerg("<message>")`
- `logger.alert("<message>")`
- `logger.critical("<message>")`
- `logger.error("<message>")`
- `logger.warning("<message>")`
- `logger.notice("<message>")`
- `logger.info("<message>")`
- `logger.debug("<message>")`

An Aruba Network Analytics Engine agent is a specifically-configured executable instance of an Aruba Network Analytics Engine script on a switch. When the agent is enabled, it performs the tasks defined by the script.

Aruba Network Analytics Engine agent sandboxes run in the default VRF. If a script has a function that requires access to the Internet, this function fails unless the switch administrator configures a DNS name server on the default VRF.

Agents and user authority

Agents have administrator authority to perform tasks such as executing CLI commands.

You must have administrator authority to create, delete, enable, disable, or change the configuration of agents. You cannot delete built-in agents.

Rules for agent names

The `<agent_name>` must have the following characteristics:

- The name must be unique among the agents created for the script.
- The name must be a text string that starts with a letter or number and can contain alphanumeric characters and the special characters . (dot), - (hyphen), and _ (underscore). Minimum length: three characters. Maximum length: 80 characters.

show nae-agent

show nae-agent [vsx-peer]

Description

Shows information about the Aruba Network Analytics Engine agents that are configured and enabled on the switch.

Parameter	Description
vsx-peer	Shows the output from the VSX peer switch. If the switches do not have the VSX configuration or the ISL is down, the output from the VSX peer switch is not displayed. This parameter is available on switches that support VSX.

Usage

This command shows the following information about the Aruba Network Analytics Engine agents that are configured and enabled on the switch:

Agent Name

The name of the agent. Length: 3 through 80 characters.

Script Name

The name of the script. Length: 3 through 80 characters.

Example: `system_resource_monitor_mm1.default`

Version

The version number of the script.

Origin

The origin of the script:

`system`

Indicates that the script is provided as part of the system software.

`user`

Indicates that a user loaded the script.

Disabled

Indicates whether the agent is disabled or enabled on the switch:

`true`

Indicates that the agent is disabled.

`false`

Indicates that the agent is enabled on the switch.

Status

The current state of the agent. Status values are the following:

CRITICAL

The agent has encountered a critical error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

MAJOR

The agent has encountered a major error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

MINOR

The agent has encountered a minor error during execution. For information about the error, see the Analytics Dashboard of the Web UI.

NORMAL

Indicates that the agent is actively monitoring network conditions and handling events.

UNKNOWN

Indicates that the status is unknown.

Example

```
switch# show nae-agent
-----
--
Agent Name                               Script Name                               Version
Origin   Disabled   Status   Time Series Count   Error
-----
--
com.arubanetworks.monitor.agent          com.arubanetworks.monitor                1.0
user     true        Unkown   0                      NONE
interface_monitor.agent                  interface_tx_rx_stats_monitor            2.3
user     true        Unknown  168                    NONE
com.arubanetworks.wildcard.vlan.agent     com.arubanetworks.wildcard.vlan          1.0
user     false       Unknown  0                      ERROR
system_resource_monitor.default          system_resource_monitor                  1.3
system   false       Normal   6                      None
```

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
6200 6300 6400 8320 8325 8360 8400	Manager (#)	Administrators or local user group members with execution rights for this command.

show nae-script

```
show nae-script [vsx-peer]
```

Description

Shows information about the Aruba Network Analytics Engine scripts that are available on the switch.

Parameter	Description
<code>vsx-peer</code>	Shows the output from the VSX peer switch. If the switches do not have the VSX configuration or the ISL is down, the output from the VSX peer switch is not displayed. This parameter is available on switches that support VSX.

Usage

This command shows the following information about the Aruba Network Analytics Engine scripts that are available on the switch:

Script Name

The name of the script. Length: 3 through 80 characters.

Example: `system_resource_monitor_mml.default`

Version

The version number of the script.

Origin

The origin of the script:

`system`

Indicates that the script is provided as part of the system software.

`user`

Indicates that a user loaded the script.

Status

The current state of the script. Status values are the following:

`CREATED`

The script has been uploaded to the switch, but script validation has not begun.

`ERROR`

The script validation process detected an error that would result in execution errors if an agent runs the script. Resolve the error by modifying the script. For information about the error, see the **Analytics Dashboard** of the Web UI.

`VALIDATING`

The script syntax and components (manifest, parameters, monitor, condition, and action) are in the process of being validated.

`VALIDATED`

The script syntax and components (manifest, parameters, monitor, condition, and action) have been validated and no errors have been found.

Example

```
switch# show nae-script
```

```
-----  
Script Name                               Version   Origin    Status  
-----  
fan_monitor                               1.0      system    VALIDATED  
interface_link_flap_monitor               1.0      system    VALIDATED  
interface_link_state_monitor              1.0      system    VALIDATED  
interface_tx_rx_stats_monitor             1.0      system    VALIDATED  
lag_imbalance_monitor                     1.0      system    VALIDATED
```

lag_status_monitor	1.0	system	VALIDATED
power_supply_monitor	1.0	system	VALIDATED
stp_bpdu_tcn_rate_monitor	1.0	system	VALIDATED
system_resource_monitor_mm1.default	1.0	system	VALIDATED
system_resource_monitor_mm2.default	1.0	system	VALIDATED
temp_sensor_monitor	1.0	system	VALIDATED

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
6200 6300 6400 8320 8325 8360 8400	Manager (#)	Administrators or local user group members with execution rights for this command.

show running-config nae

show running-config nae

Description

Shows the nae-specific running configurations, including the full base64 nae-script content.

Usage

This command can be used in place of `show running-config all` to include the full base64 nae-script content.

Example

```
switch# show running-config nae
nae-script hardware_device_health_monitor false
Iy0qLSBjb2Rpbmc6IHV0Zi04IC0qLQ0KIw0KIyhjKSBD3B5cmln...
nae-agent hardware_device_health_monitor hw_device_health false
```

Command History

Release	Modification
10.08	Command introduced

Command Information

Platforms	Command context	Authority
6200 6300 6400 8320 8325 8360 8400	Manager (#)	Administrators or local user group members with execution rights for this command.

https-server rest access-mode

```
https-server rest access-mode {read-only | read-write}
```

Description

Changes the REST API access mode. The default mode is read-write.

Parameter	Description
read-write	Selects the read/write mode. Allows POST, PUT, and DELETE methods to be called on all configurable elements in the switch database.
read-only	Selects the read-only mode. Write access to most switch resources through the REST API is disabled.

Usage

Setting the mode to `read-write` on the REST API allows POST, PUT, and DELETE methods to be called on all configurable elements in the switch database.

By default, REST APIs in the device are in the read-write mode. Some switch resources allow POST, PUT, and DELETE regardless of REST API mode. REST APIs that are required to support the Web UI or the Network Analytics Engine expose POST, PUT, or DELETE operations, even if the REST API access mode is set to read-only.

The REST API in read/write mode is intended for use by advanced programmers who have a good understanding of the system schema and data relationships in the switch database.



Because the REST API in read/write mode can access every configurable element in the database, it is powerful but must be used with extreme caution: No semantic validation is performed on the data you write to the database, and configuration errors can destabilize the switch.

On 6300 switches or 6400 switches, by default, the HTTPS server is enabled in `read-write` mode on the `mgmt` VRF. If you enable the HTTPS server on a different VRF, the HTTPS server is enabled in `read-only` mode.

Example

```
switch(config)# https-server rest access-mode read-only
```

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
All platforms	config	Administrators or local user group members with execution rights for this command.

https-server session close all

```
https-server session close all
```

Description

Invalidates and closes all HTTPS sessions. All existing Web UI and REST sessions are logged out and all real-time notification feature WebSocket connections are closed.

Usage

Typically, a user that has consumed the allowed concurrent HTTPS sessions and is unable to access the session cookie to log out manually must wait for the session idle timeout to start another session. This command is intended as a workaround to waiting for the idle timeout to close an HTTPS session. This command stops and starts the `hpe-restd` service, so using this command affects all existing REST sessions, Web UI sessions, and real-time notification subscriptions.

Example

```
switch# https-server session close all
```

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
All platforms	Manager (#)	Administrators or local user group members with execution rights for this command.

https-server vrf

```
https-server vrf <VRF-NAME>
no https-server vrf <VRF-NAME>
```

Description

Configures and starts the HTTPS server on the specified VRF. HTTPS server features include the REST API and the web user interfaces.

The `no` form of the command stops any HTTPS servers running on the specified VRF and removes the HTTPS server configuration.

Parameter	Description
<code><VRF-NAME></code>	Specifies the VRF name. Required. Length: Up to 32 alpha numeric characters.

Usage

By using this command, you enable access to both the Web UI and to the REST API on the specified VRF. You can enable access on multiple VRFs.

By default, 8320, 8325, 8360, and 8400 Switch Series have an HTTPS server enabled on the `mgmt` VRF.

By default, the 6200, 6300, and 6400 Switch Series have an HTTPS server enabled on the `mgmt` VRF and on the `default` VRF.

When the HTTPS server is not configured and running, attempts to access the Web UI or REST API result in 404 Not Found errors.

The VRF you select determines from which network the Web UI and REST API can be accessed.

For example:

- If you want to enable access to the REST API and Web UI through the OOBM port (management IP address), specify the built-in management VRF (`mgmt`).
- If you want to enable access to the REST API and Web UI through the data ports (for "inband management"), specify the built-in default VRF (`default`).
- If you want to enable access to the REST API and Web UI through only a subset of data ports on the switch, specify other VRFs you have created.

Aruba Network Analytics Engine scripts run in the default VRF, but you do not have to enable HTTPS server access on the default VRF for the scripts to run. If the switch has custom Aruba Network Analytics Engine scripts that require access to the Internet, then for those scripts to perform their functions, you must configure a DNS name server on the default VRF.

Examples

Enabling access on all ports on the switch, specify the default VRF:

```
switch(config)# https-server vrf default
```

Enabling access on the OOBM port (management interface IP address), specify the management VRF:

```
switch(config)# https-server vrf mgmt
```

Enabling access on ports that are members of the VRF named `vrfprogs`, specify `vrfprogs`:

```
switch(config)# https-server vrf vrfprogs
```

Enabling access on the management port and ports that are members of the VRF named `vrfprogs`, enter two commands:

```
switch(config)# https-server vrf mgmt
switch(config)# https-server vrf vrfprogs
```



The 6200 switches support only two VRFs. One management VRF and one default VRF. You cannot add another VRF.

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
All platforms	config	Administrators or local user group members with execution rights for this command.

show https-server

```
show https-server [vsx-peer]
```

Description

Shows the status and configuration of the HTTPS server. The REST API and web user interface are accessible only on VRFs that have the HTTPS server features configured.

Parameter	Description
vsx-peer	Shows the output from the VSX peer switch. If the switches do not have the VSX configuration or the ISL is down, the output from the VSX peer switch is not displayed. This parameter is available on switches that support VSX.

Usage

Shows the configuration of the HTTPS server features.

VRF

Shows the VRFs, if any, for which HTTPS server features are configured.

REST Access Mode

Shows the configuration of the REST access mode:

read-write

POST, PUT, and DELETE methods can be called on all configurable elements in the switch database. This is the default value.

read-only

Write access to most switch resources through the REST API is disabled.

Examples

```

switch# show https-server

HTTPS Server Configuration
-----

VRF                : default, mgmt
REST Access Mode   : read-write

Max sessions per user : 6

Session timeout     : 20

```

Command History

Release	Modification
10.07 or earlier	--

Command Information

Platforms	Command context	Authority
All platforms	Manager (#)	Administrators or local user group members with execution rights for this command.

Accessing Aruba Support

Aruba Support Services	https://www.arubanetworks.com/support-services/
Aruba Support Portal	https://asp.arubanetworks.com/
North America telephone	1-800-943-4526 (US & Canada Toll-Free Number) +1-408-754-1200 (Primary - Toll Number) +1-650-385-6582 (Backup - Toll Number - Use only when all other numbers are not working)
International telephone	https://www.arubanetworks.com/support-services/contact-support/

Be sure to collect the following information before contacting Support:

- Technical support registration number (if applicable)
- Product name, model or version, and serial number
- Operating system name and version
- Firmware version
- Error messages
- Product-specific reports and logs
- Add-on products or components
- Third-party products or components

Other useful sites

Other websites that can be used to find information:

Airheads social forums and Knowledge Base	https://community.arubanetworks.com/
Software licensing	https://lms.arubanetworks.com/
End-of-Life information	https://www.arubanetworks.com/support-services/end-of-life/
Aruba software and documentation	https://asp.arubanetworks.com/downloads

Accessing Updates

You can access updates from the Aruba Support Portal or the HPE My Networking Website.

Aruba Support Portal

<https://asp.arubanetworks.com/downloads>

If you are unable to find your product in the Aruba Support Portal, you may need to search My Networking, where older networking products can be found:

My Networking

<https://www.hpe.com/networking/support>

To view and update your entitlements, and to link your contracts and warranties with your profile, go to the Hewlett Packard Enterprise Support Center **More Information on Access to Support Materials** page:

<https://support.hpe.com/portal/site/hpsc/aae/home/>

Access to some updates might require product entitlement when accessed through the Hewlett Packard Enterprise Support Center. You must have an HP Passport set up with relevant entitlements.

Some software products provide a mechanism for accessing software updates through the product interface. Review your product documentation to identify the recommended software update method.

To subscribe to eNewsletters and alerts:

<https://asp.arubanetworks.com/notifications/subscriptions> (requires an active Aruba Support Portal (ASP) account to manage subscriptions). Security notices are viewable without an ASP account.

Warranty Information

To view warranty information for your product, go to <https://www.arubanetworks.com/support-services/product-warranties/>.

Regulatory Information

To view the regulatory information for your product, view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products*, available at <https://www.hpe.com/support/Safety-Compliance-EnterpriseProducts>

Additional regulatory information

Aruba is committed to providing our customers with information about the chemical substances in our products as needed to comply with legal requirements, environmental data (company programs, product recycling, energy efficiency), and safety information and compliance data, (RoHS and WEEE). For more information, see <https://www.arubanetworks.com/company/about-us/environmental-citizenship/>.

Documentation Feedback

Aruba is committed to providing documentation that meets your needs. To help us improve the documentation, send any errors, suggestions, or comments to Documentation Feedback (docsfeedback-switching@hpe.com). When submitting your feedback, include the document title, part number, edition, and publication date located on the front cover of the document. For online help content, include the product name, product version, help edition, and publication date located on the legal notices page.