

Apstra4.0 モニタリングガイド

June. 2021

JUNIPER | Engineering
NETWORKS | Simplicity

CONFIDENTIALITY AND LEGAL NOTICE

This material contains information that is confidential and proprietary to Juniper Networks, Inc. Recipient may not distribute, copy, or repeat information in the document.

This statement of product direction sets forth Juniper Networks' current intention and is subject to change at any time without notice. No purchases are contingent upon Juniper Networks delivering any feature or functionality depicted in this presentation.

subject to a license agreement that describes program terms and conditions.

本資料は融資でベストエフォートで記載している資料となります。
内容に不備がある場合はご了承ください。
最新の状況などは公式のマニュアルをご確認ください。
また、内容は予告なしに変更になる場合があります。

はじめに

本資料はApstraのインストールからネットワーク構築までの手順をまとめたものです。主にJunosを管理する手法を記載しており、他のネットワークOSを考慮していません。不明点はJuniper Networks、またはパートナー様にご連絡いただくか、Apstraのマニュアルを参照下さい。

※Apstraの構築、管理、運用に関わる設定は別紙参照。

Apstraマニュアル

<https://www.juniper.net/documentation/product/us/en/apstra>

目次

ネットワーク監視概要

- 監視イメージ
- 監視範囲

Default Telemetry

- ダッシュボード
- シスログ転送
- テレメトリデータ

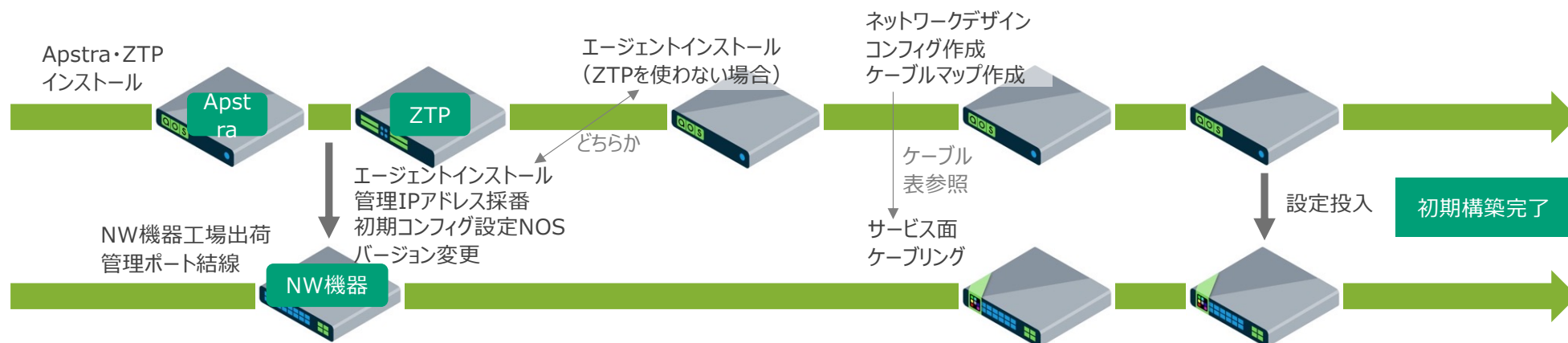
Analytics

- 概要
- アーキテクチャ
- Probe
- Widgets
- Dashboard

その他の監視機能

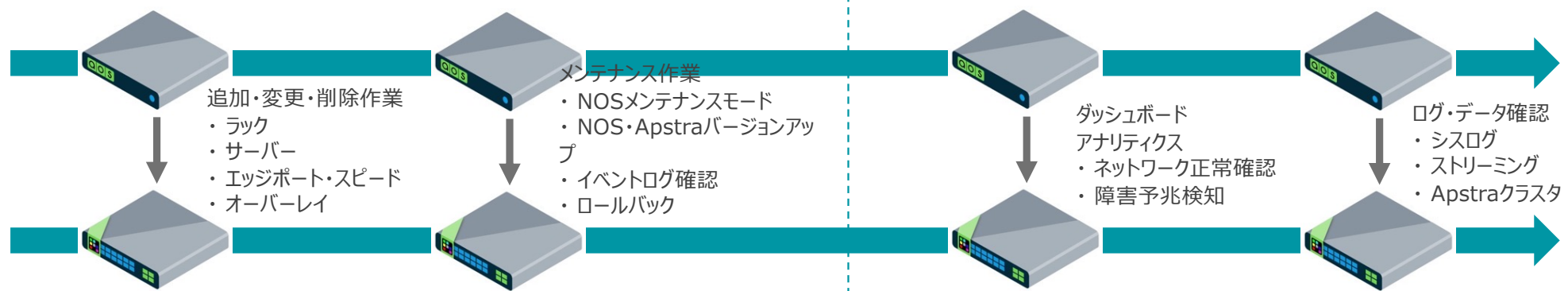
- トラフィック量カラーリング
- 外部モニタリングサーバ
- ネットワーク障害解析

ネットワーク構築・運用の全体イメージ



※Apstra導入前にケーブルリングしてもよい。

本資料の範囲



ネットワーク監視概要

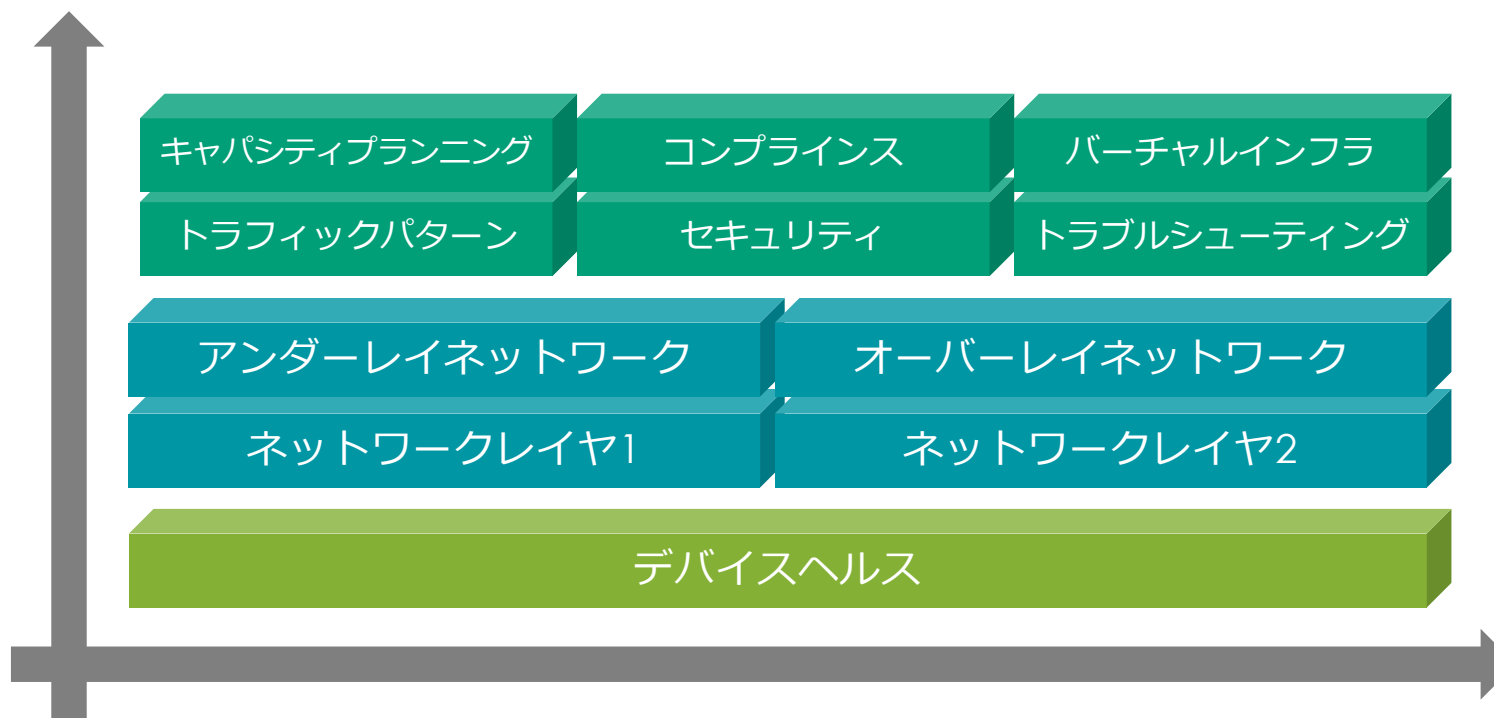
- ・ 監視イメージ
- ・ 監視範囲



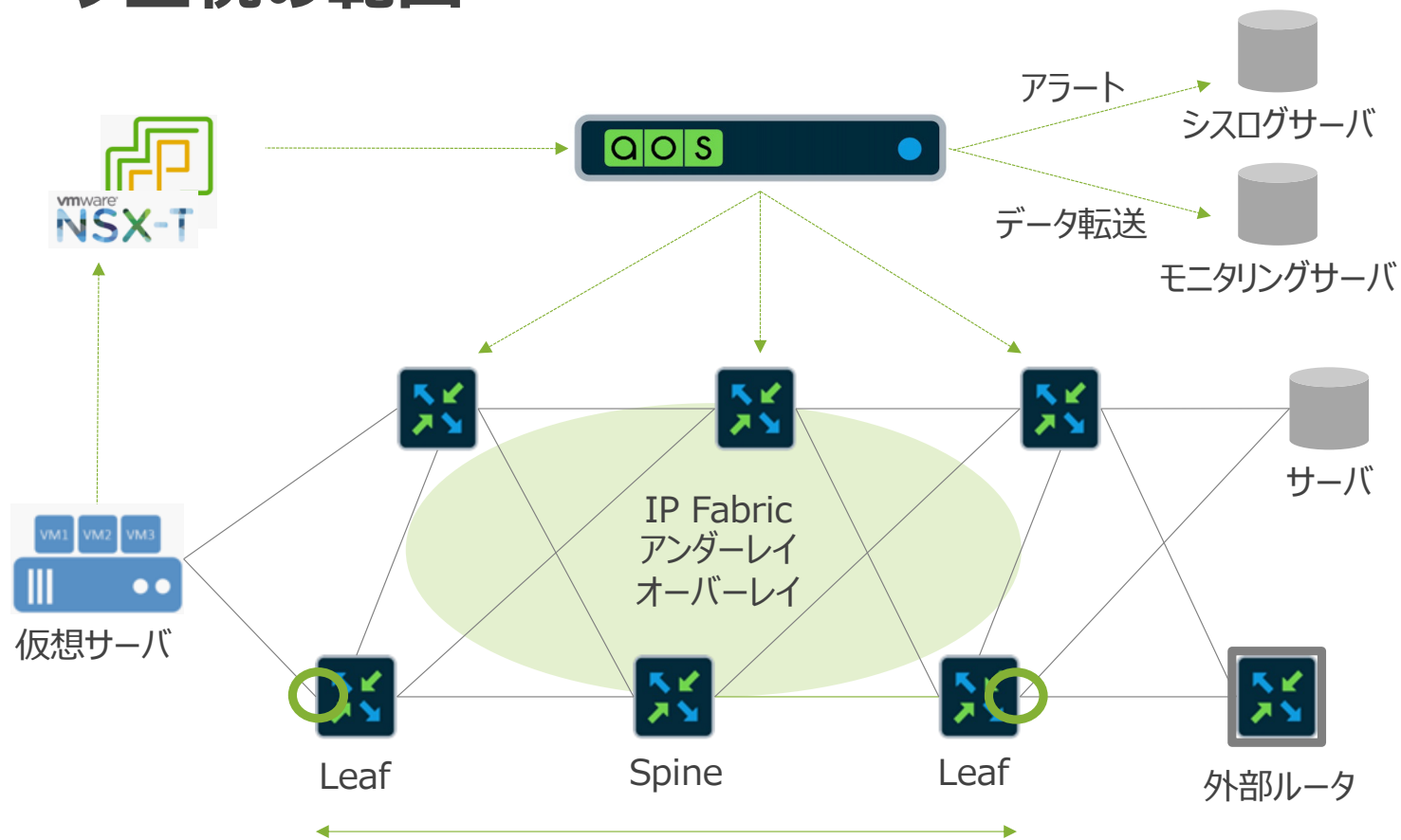
ネットワーク監視のイメージ

Apstraのネットワーク監視は2つに分類される。

- デフォルトで監視するもの (Default Telemetry)
- 手動で監視を追加するもの (Analytics、またはIntent Based Analytics)



ネットワーク監視の範囲



Leafのエッジポートまでを監視。外部ルータやサーバは対象外。
仮想サーバから特定の情報を入手することは対応。

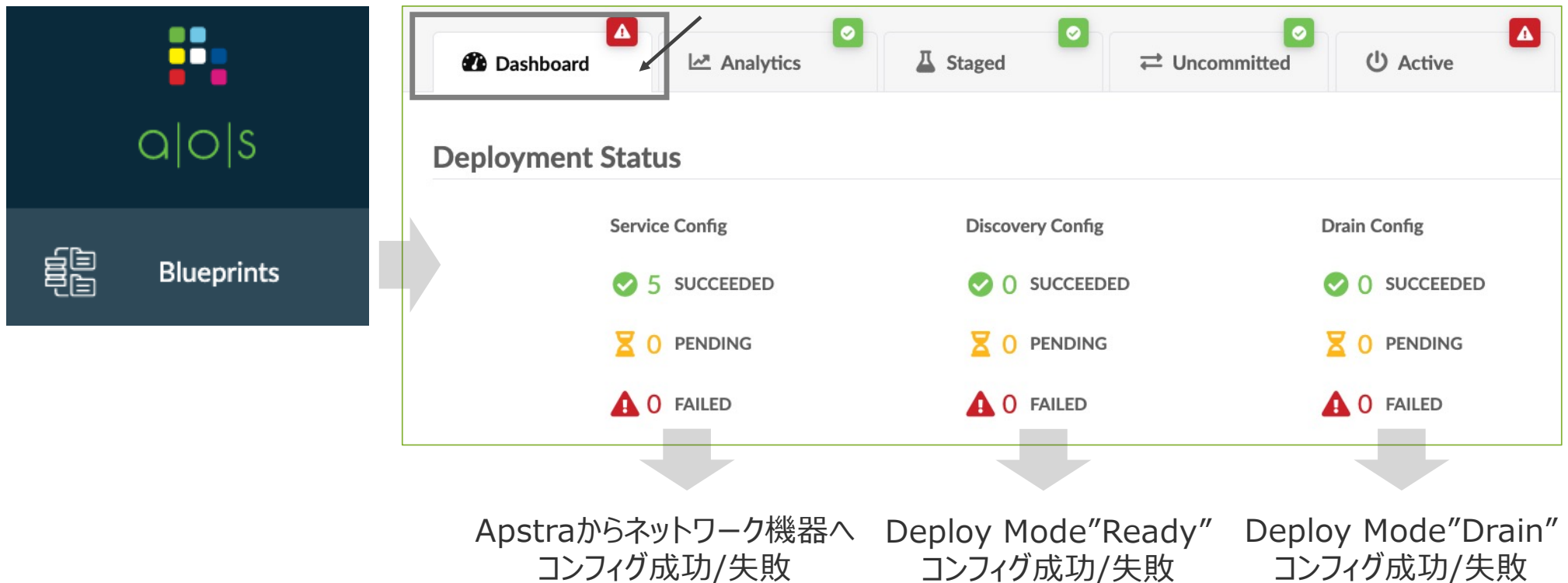
Default Telemetry

- ・ ダッシュボード
- ・ シスログ転送
- ・ テレメトリデータ



ダッシュボード

ネットワーク構築後に、Apstraが自動で監視を開始。



ダッシュボード

ネットワークが正常な場合は緑色、不具合が発生している場合は赤色で表示。



ダッシュボードの監視項目

検証項目	内容	監視対象
All Probes	IBA Anomaly (IBAは後述)	SS, Spine, Leaf
IP Fabric		
BGP	BGP隣接関係(ネイバー)がIntentのとおりに確立されているか	SS, Spine, Leaf
Cabling	デバイス間の物理配線がIntentのとおりに行われているか	SS, Spine, Leaf
Interface	インタフェースのUp/DownステータスがIntentと一致しているか	SS, Spine, Leaf
Hostname	デバイスのホスト名がIntent通りか	SS, Spine, Leaf
Generic System Connectivity		
BGP	BGP隣接関係(ネイバー)がIntentのとおりに確立されているか	Leaf
Interface	インタフェースのUp/DownステータスがIntentと一致しているか	Leaf
MLAG/LAG	リンクアグリゲーションがIntentのとおりに確立できているか	Leaf
Liveness (Spine, Leaf, Generics)	各デバイスがApstraによる制御通信に応答しているか	SS, Spine, Leaf
Deployment Status		SS, Spine, Leaf
Deployment	各デバイスがApstraによるコンフィグレーションを正常に展開することができるか	SS, Spine, Leaf
Config Dev.	各デバイスのコンフィグが、Apstraにより作成されたものと一致しているか	SS, Spine, Leaf
Route Verification - Route Table	各デバイスが各BGPネイバーから想定どおりのプレフィックスを受信し、ルーティングテーブルに反映できているか	SS, Spine, Leaf

取得しているコマンド

Junosの監視用に取得しているshowコマンドは以下の通り。

Interface counters & Interface error counters

```
show interfaces extensive
```

LAG & Interface status

```
show interfaces terse
```

LLDP neighbors

```
show lldp neighbors
```

BGP Sessions

```
show bgp neighbors
```

Hostname

```
show system information
```

ARP

```
show arp no-resolve
```

MAC Table

```
show ethernet-switching table extensive
```

Routing table

```
Underlay - show route table inet.0
```

```
Overlay - show route table bgp.evpn.0
```

取得インターバル

監視データの取得間隔はプロトコルにより異なる。

プロトコル	間隔 (秒)	プロトコル	間隔 (秒)
Interface Status	120	Interface Counter	5
LLDP	10	BGP	120
LAG	120	Route	120
ARP	120	MAC	120
Hostname	120	EVPN Type5	600

その他プロトコルの取得間隔は以下のAPIから確認できる。

GET /api/systems/{system_id}/services List telemetry services

デフォルトの取得間隔は次のAPIから変更できるが、変更内容をサポートできるか確認するため、事前にJuniperへ連絡すること。

PUT /api/systems/{system_id}/services/{service_name} Update telemetry service

ダッシュボードのアノマリ内容

赤いグラフをクリックすると、アノマリの内容を確認できる。

IP Fabric

クリック

BGP
24 anomalies

Cabling
12 anomalies

Interface
0 anomalies

Hostname
0 anomalies

監視内容

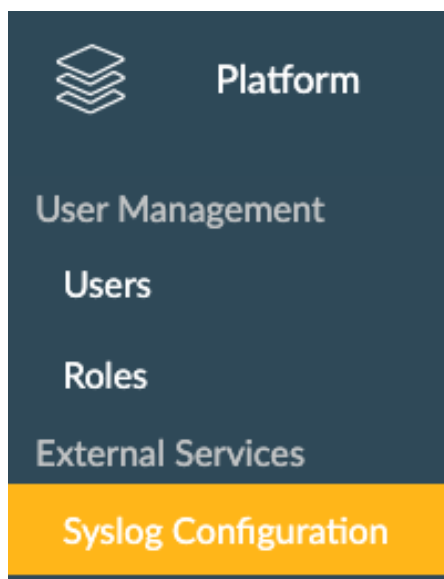
あるべき状態

実際のステータス

Node	Hostname	Service	Anomaly Type	Role	Anomaly Extra Details	Expected	Actual																								
対象ノード evpn_esi_001_leaf1	evpn-esi-001-leaf1	IP Fabric	bgp	Spine to Leaf	<table border="1"><thead><tr><th>Property</th><th>Value</th></tr></thead><tbody><tr><td>Address Family</td><td>"evpn"</td></tr><tr><td>Destination ASN</td><td>"64512"</td></tr><tr><td>Destination IP</td><td>"10.0.0.0"</td></tr><tr><td>Destination Name</td><td>"spine1"</td></tr><tr><td>Source ASN</td><td>"64514"</td></tr><tr><td>Source IP</td><td>"10.0.0.2"</td></tr><tr><td>VRF Name</td><td>"default"</td></tr></tbody></table>	Property	Value	Address Family	"evpn"	Destination ASN	"64512"	Destination IP	"10.0.0.0"	Destination Name	"spine1"	Source ASN	"64514"	Source IP	"10.0.0.2"	VRF Name	"default"	<table border="1"><thead><tr><th>Property</th><th>Value</th></tr></thead><tbody><tr><td>value</td><td>"up"</td></tr></tbody></table>	Property	Value	value	"up"	<table border="1"><thead><tr><th>Property</th><th>Value</th></tr></thead><tbody><tr><td>value</td><td>"down"</td></tr></tbody></table>	Property	Value	value	"down"
Property	Value																														
Address Family	"evpn"																														
Destination ASN	"64512"																														
Destination IP	"10.0.0.0"																														
Destination Name	"spine1"																														
Source ASN	"64514"																														
Source IP	"10.0.0.2"																														
VRF Name	"default"																														
Property	Value																														
value	"up"																														
Property	Value																														
value	"down"																														

アノマリの外部サーバ通知

アノマリはApstraのGUIだけでなく、Syslogサーバへ通知できる。



IP Address *
1.2.3.4 ← Syslogサーバ

Port *
514 ← Port番号

Protocol *
udp ← UDP or TCP

Facility *
syslog ← ファシリティ選択

Create



IP Address and Port ▲	Protocol ◆	Facility ◆	Use for Audit ◆	Forward Anomalies ◆
1.2.3.4:514	UDP	syslog	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> ← アノマリのSyslogを有効

アノマリの外部サーバ通知

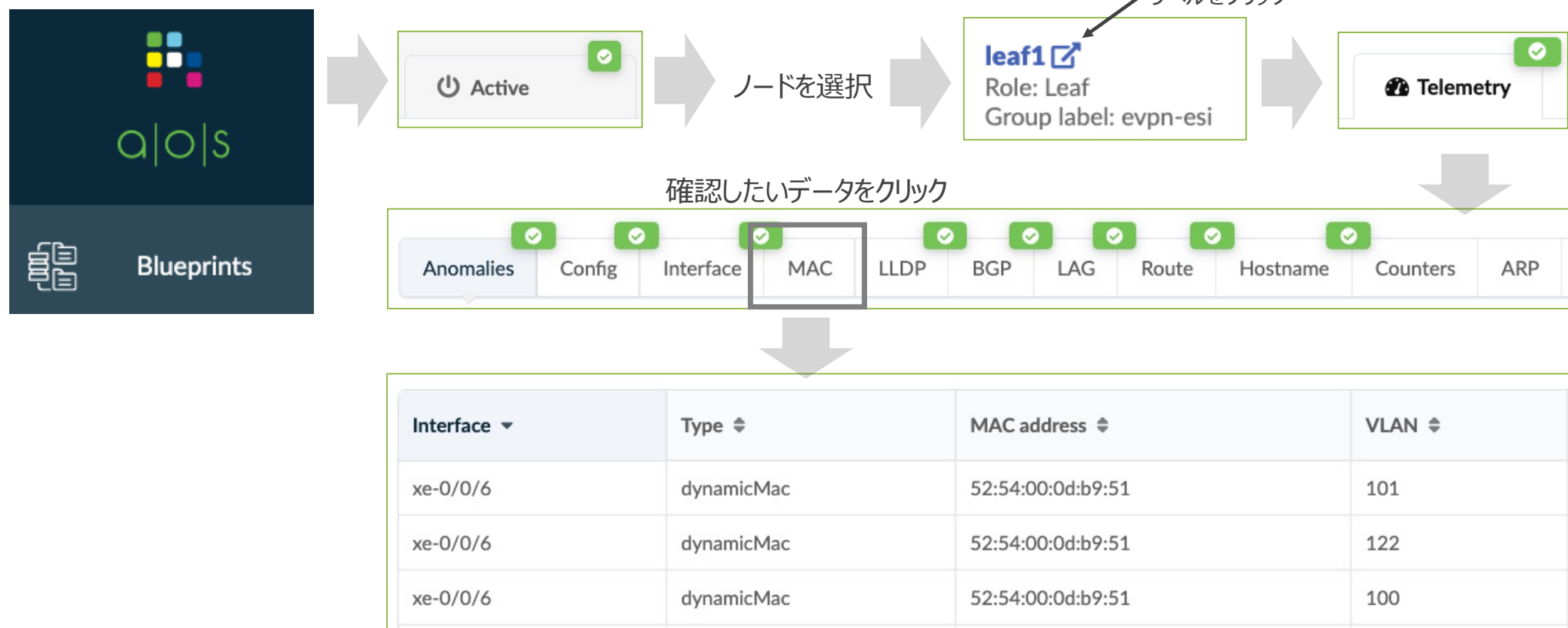
Syslogサーバへ通知するフォーマットはドキュメントに記載あり。

アノマリ通知サンプル。

```
Nov 29 13:55:12 172.27.113.223 2021-11-29T13:55:12.564418JST aos-server CEF:0|Apstra|AOS|4.0.1-1045|101|Alert|10|msg={u'blueprint_label': u'4.0.1-Hatsudai', u'timestamp': 1638161712564418, u'origin_name': u'DD272', u>alert': {u'probe_alert': {u'stage_name': u'ingress_has_sustained_pkt_discards', u'probe_label': u'Sustained packet discards', u'expected_float_max': 1.7976931348623157e+308, u'probe_id': u'd3c62fd7-e477-4b12-be00-c5f56eec26a4', u'expected_float': 21.0, u'key_value_pairs': [{u'value': u'"facing_1leaf-5110-32-001-leaf1:et-0/0/1"', u'key': u'description'}, {u'value': u'"et-0/0/0"', u'key': u'interface'}, {u'value': u'"DD272"', u'key': u'system_id'}]}, u'item_id': u'803e000b-6a3d-481f-9f82-21c02f7533b4', u'actual_float': 22.181704}, u'first_seen': 1638161712564403, u'raised': True, u'severity': 3, u'id': u'ff4da097-7357-4e00-a465-9f8826567eaa'}, u'origin_hostname': u'spine1', 'device_hostname': 'spine1', u'origin_role': u'spine'}
```

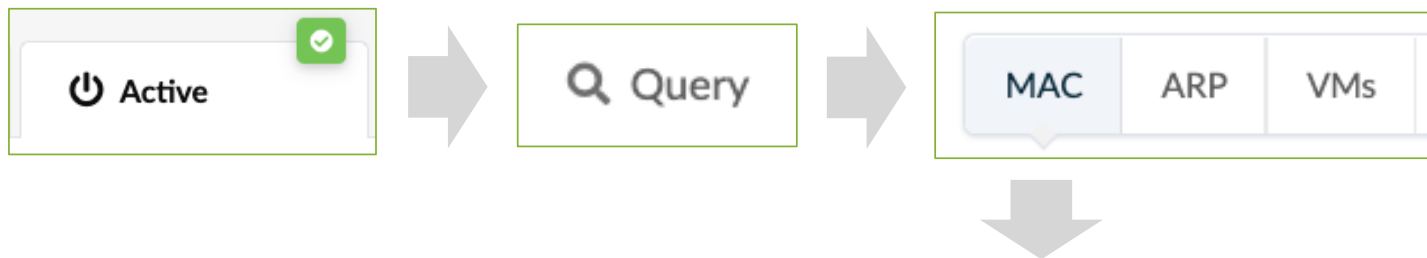
テレメトリデータ

ダッシュボードのデータに加え、ネットワーク機器が学習しているMACアドレス、ARPテーブル、バーチャルマシン一覧（サードパーティコントローラと連携している場合）などを表示。



テレメトリデータ

MACアドレス、ARPテーブル、バーチャルマシン一覧は別の確認方法あり。



Node Name	S/N	Hostname	Type	VLAN	VXLAN	MAC	Interface
leaf2	525400523B46	leaf2	Dynamic	120	25000	52:54:00:66:be:8b	xe-0/0/2
leaf2	525400523B46	leaf2	Dynamic	120	25000	52:54:00:0d:b9:51	vtep.32770
leaf2	525400523B46	leaf2	Dynamic	120	25000	52:54:00:1f:7b:e2	ae1
leaf2	525400523B46	leaf2	Dynamic	100	10000	52:54:00:66:be:8b	xe-0/0/2
leaf2	525400523B46	leaf2	Dynamic	100	10000	52:54:00:fa:6c:9c	vtep.32769
leaf2	525400523B46	leaf2	Dynamic	100	10000	52:54:00:0d:b9:51	vtep.32770
leaf2	525400523B46	leaf2	Dynamic	100	10000	52:54:00:1f:7b:e2	ae1
leaf2	525400523B46	leaf2	Dynamic	122	25002	52:54:00:66:be:8b	xe-0/0/2

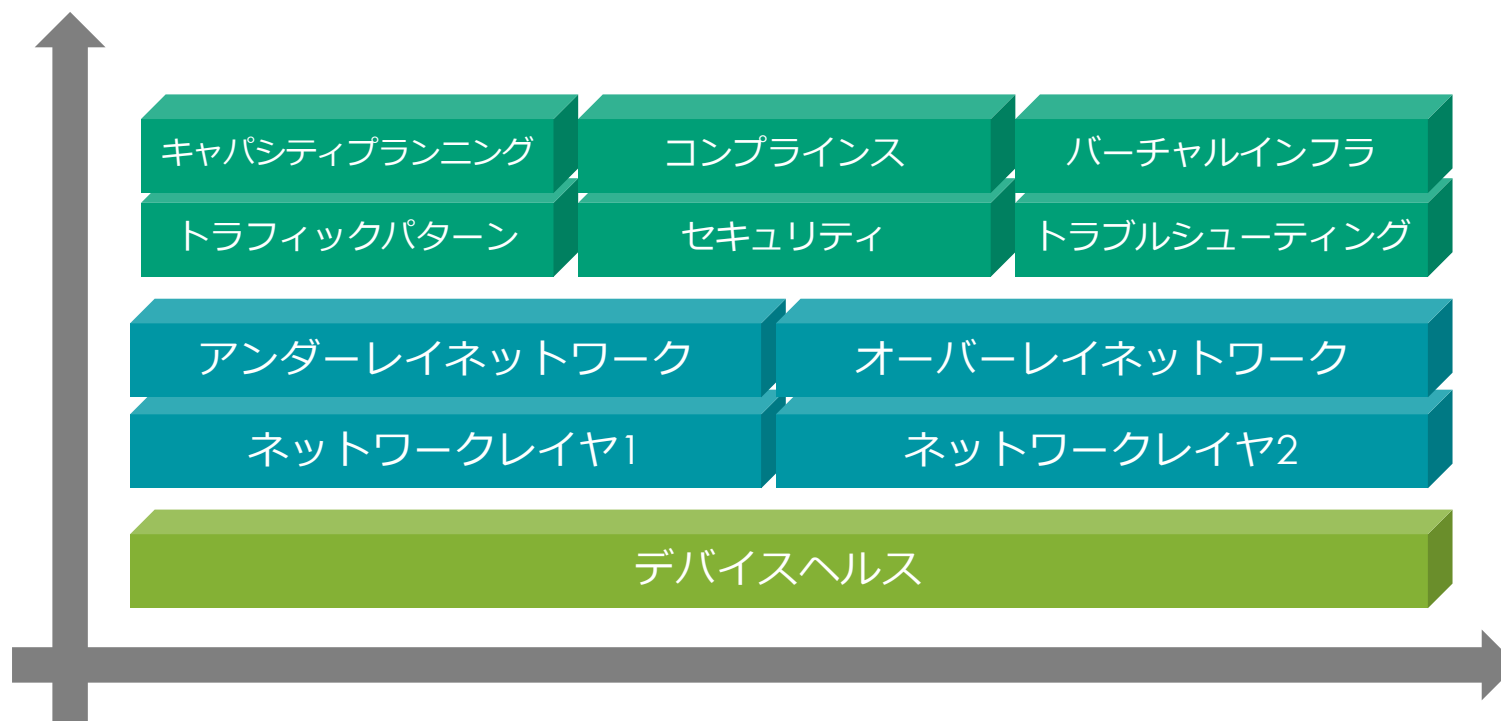
Analytics

- ・ 概要
- ・ アーキテクチャ
- ・ Probe
- ・ Widgets
- ・ Dashboards



Analytics概要

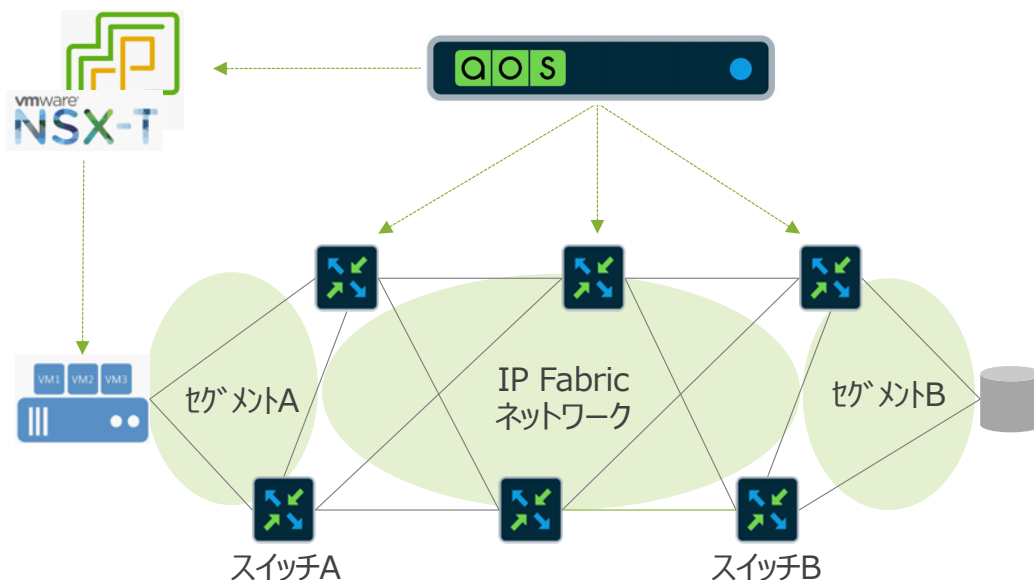
Default Telemetryでモニタリングできない要件を、新規作成する監視。
監視内容、アラートの閾値、モニタ間隔などを個々に設定。



Analytics概要

使用例：ネットワーク障害箇所の特定

ネットワークを各レイヤ毎に監視して障害を早期に発見。
(例)スイッチAはオーバーレイでセグメントBを学習しているか？



最新情報はこちら <https://github.com/Apstra/iba>

ネットワークレイヤ	(ネットワーク監視例)
仮想インフラ	Vmware側の設定との整合性
オーバーレイ	EVPN Type3,5等 特定NWセグメントの学習状況
ネットワークレイヤ3	ルーティングテーブルの整合性 BGPステータス
データプレーン	インターフェースエラーカウンタ インターフェースキュードロップ
ネットワークレイヤ1	トランシーバ光レベル インターフェースフラップ
デバイスヘルス	メモリーク、CPU使用率 電源・ファン

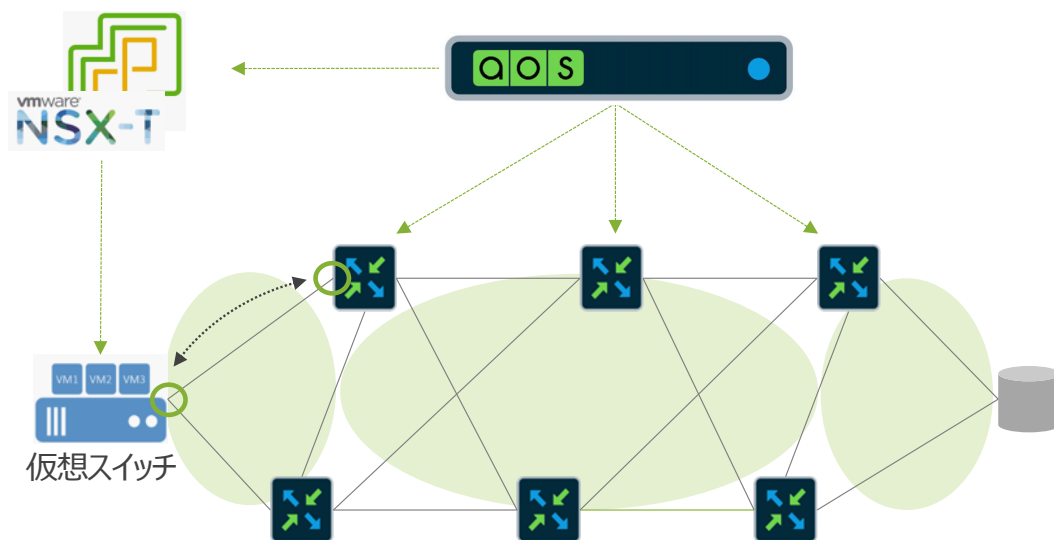
定期取得する間隔は最短5秒。

Analytics概要

使用例：バーチャルインフラ連携。

VMWare vCenter, NSX-Tと連携。

仮想スイッチとApstra管理スイッチのポート設定を比較、不一致を検知するとアラート。

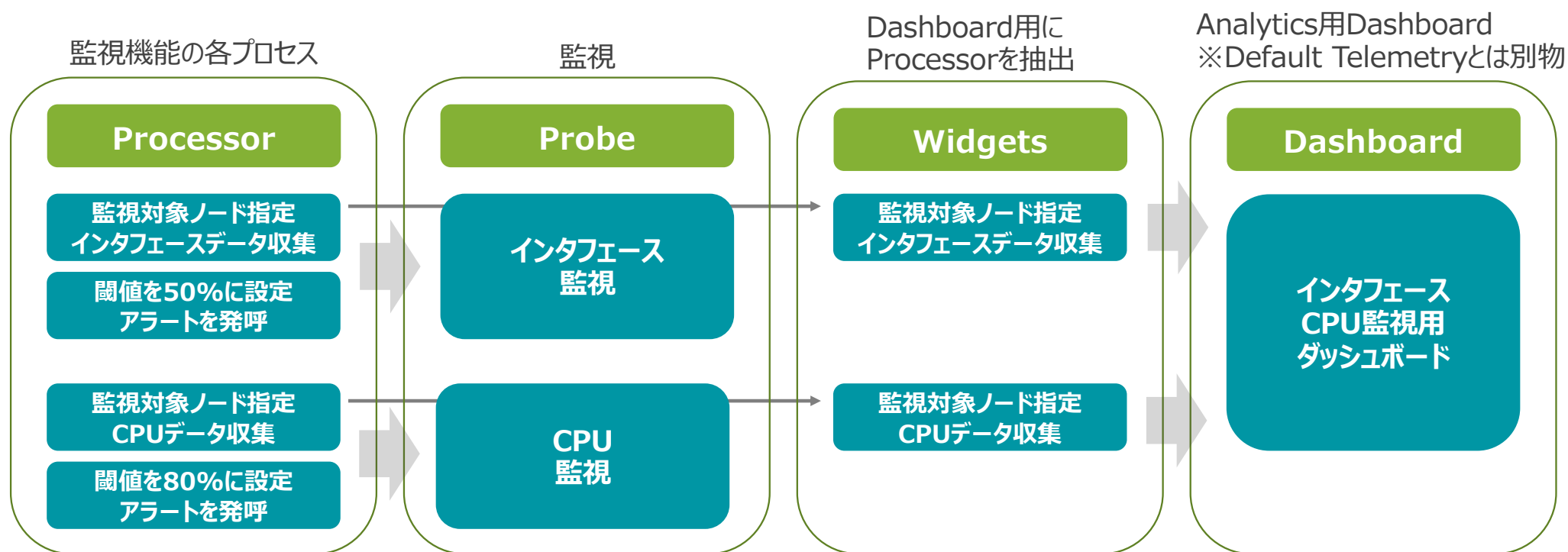


- VLAN Tag Mismatch
検知後、Apstra管理スイッチにVLANを自動設定。
- MTU Mismatch
- LAG Config Mismatch
- No LLDP Settings

VM Name	Hosted On	Hypervisor Hostname	Hypervisor Version
Nutanix%2fNSX lab Ext Router VM	nsxtegedgehost01 (nsxtegedgehost01.dc1.apstra.com)	nsxtegedgehost01	ESXI:6.7.0
PoSBareMetalApp30	nsxtegedgehost01 (nsxtegedgehost01.dc1.apstra.com)	nsxtegedgehost01	ESXI:6.7.0
PoSBareMetalDB20	nsxtegedgehost01 (nsxtegedgehost01.dc1.apstra.com)	nsxtegedgehost01	ESXI:6.7.0
PoSVMApp60	nsxtcomputehost01 (srv104-26a)	srv104-26a	ESXI:6.7.0

Analyticsのアーキテクチャ

例：SpineとLeafスイッチの全インタフェースをモニターし、送信トラフィック量が50%を超えたらアラートを出す。
LeafスイッチのCPU使用率をモニターし、80%を超えたらアラートを出す。



Probeは複数のProcessorから成る。Processorの組み合わせによりProbeを自由に作成できる。監視自体はProbeの作成までOK。Widgets/Dashboardは可視化のオプション。

Probeの作成

Probeの作成手法は3つ。

ビルトインのProbeを使用し、BP生成時に自動で稼働するもの

Default Probe

<input type="checkbox"/>	Name ▲
<input type="checkbox"/>	Device system health
<input type="checkbox"/>	Device Traffic
<input type="checkbox"/>	ECMP Imbalance (External Interfaces)
<input type="checkbox"/>	ECMP Imbalance (Fabric Interfaces)
<input type="checkbox"/>	ESI Imbalance
<input type="checkbox"/>	Hot/Cold Interface Counters (Fabric Interfaces)
<input type="checkbox"/>	LAG Imbalance

ビルトインのProbeを使用し、BP生成時に手動で追加するもの

Predefined Probe

Predefined Probe *

- Bandwidth utilization
- Bandwidth utilization**
- Device Traffic
- Device system health
- Drain traffic anomaly
- ECMP Imbalance (External Interfaces)
- ECMP Imbalance (Fabric Interfaces)

Processorを組み合わせ、手動でProbeを作成するもの

New Probe

Probes ▶ New Probe

Name *

DIY_Probe

Description

All_Monitor

+ Add Processor

何を監視できるのか？

データ収集ProcessorはApstraエージェントのスクリプトファイルを使い、ネットワーク機器からデータを取得している。つまり、デフォルトで対応しない監視データは、手動でApstraエージェントへスクリプトファイルを追加する。

Processor (一例)		Probe作成手法			
		Default Probe	Predefined Probe	New Probe	AOS_CLI
デフォルト	インタフェースUP/DOWN	可	可	可	可
	BGPステータス				
	EVPN Type-5 ステータス				
AOS_CLI	インタフェース queue drop	N/A	N/A	N/A	可
	SFP TX/RX Power				
	VXLAN ステータス				
スクリプト作成	オーバレイPing	N/A	N/A	N/A	可
	RPM				
	ASICステータス				

Probeの種類

デフォルト及びAOS CLIを使用した公開済みのProbe一覧。

※注意:Juniper社非公式サポート外のprobeも含まれるため取扱は注意。
今後predefined probeなどは整理される予定のためJuniperにコンタクト。

IBA Probes

The devices managed by Apstra AOS generate large amounts of data over time. On its own, this data is voluminous and unhelpful. Through Intent-Based Analytics (IBA), AOS allows the operator to combine intent from the AOS graph database with current and historic data from devices to reason about the network at-large.

For a detailed explanation of AOS IBA, please watch our recent webinar "[Intent-Based Analytics: Prevent Network Outages and Gray Failures](#)"!

Probes are the basic unit of abstraction in IBA. Operators can configure, create, and delete probes. Generally, a given probe consumes some set of data from the network, does various successive aggregations and calculations on it, and optionally specifies some conditions of said aggregations and calculations on which anomalies are raised.



<https://github.com/Apstra/iba>

※Junosで使えるProbeの一覧は、別途追記の予定。

Default Probe

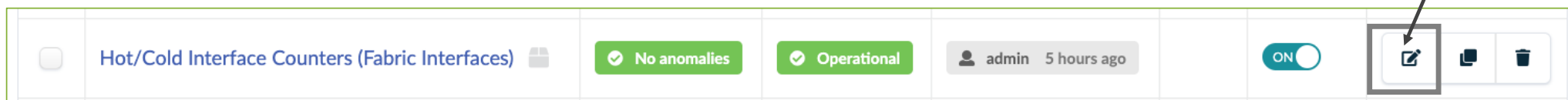
ブループリント作成後に自動生成されるProbe。後から追加した全てのProbeもここで管理。



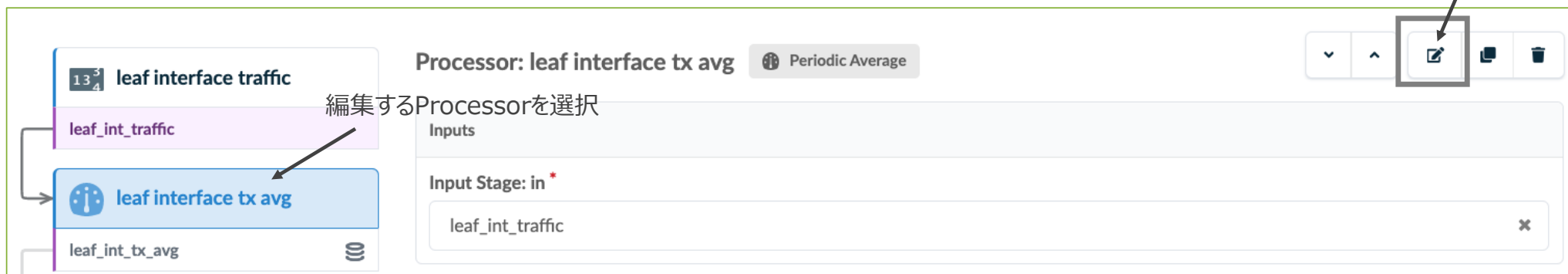
<input type="checkbox"/> 0 selected	Name ▲	Anomalies ▾	State ▾	Updated By ▾	Tags ▾	Enabled ▾ Off/On	Actions 編集 削除
<input type="checkbox"/>	Device system health 📄	✔ No anomalies	✔ Operational	⚙ System a day ago		ON	✎ 📄 🗑
<input type="checkbox"/>	Device Traffic 📄	✔ No anomalies	✔ Operational	👤 admin a day ago		ON	✎ 📄 🗑
<input type="checkbox"/>	ECMP Imbalance (External Interfaces) 📄	✔ No anomalies	✔ Operational	⚙ System a day ago		ON	✎ 📄 🗑
<input type="checkbox"/>	ECMP Imbalance (Fabric Interfaces) 📄	✔ No anomalies	✔ Operational	⚙ System a day ago		ON	✎ 📄 🗑
<input type="checkbox"/>	ESI Imbalance 📄	✔ No anomalies	✔ Operational	⚙ System a day ago		ON	✎ 📄 🗑

Default Probe

Probe内容の変更は編集ボタンから行う。



主にProcessorの編集を行う。以下、“Hot/Cold Interface Counters”の例。



Processor編集の詳細は、“Processor”の項で別途解説。

Predefined Probe

Predefined Probeを新規作成。

Blueprints → Analytics → Probes → **Create Probe**

- New Probe
- Instantiate Predefined Probe** (選択)
- Import Probes

Predefined Probe *

- Device system health
- Bandwidth utilization
- Device Traffic
- Device system health** (Probeを選択)
- Drain traffic anomaly
- ECMP Imbalance (External Interfaces)
- ECMP Imbalance (Fabric Interfaces)

Configuration for Device system health:

- CPU utilization threshold:** 80 (閾値を定義)
If percentage CPU utilization exceeds the threshold, an anomaly is raised
- Memory utilization threshold:** 80
If percentage memory utilization exceeds the threshold, an anomaly is raised
- Disk utilization threshold:** 80
If percentage disk utilization exceeds the threshold, an anomaly is raised

閾値以外にも、平均値算出の間隔、より詳細なパラメータの選択など、Probeにより様々

Create

Predefined Probe

<https://www.juniper.net/documentation/us/en/software/apstra4.1/apstra-user-guide/topics/task/probe-predefined-instantiate.html>

Predefined Probes (Partial List)

- ECMP Imbalance (External Interfaces) Probe
- ECMP Imbalance (Fabric Interfaces) Probe
- EVPN Probes
- External Routes Probe
- Device Traffic Probe (aka Headroom probe)
- Hot/Cold Interface Counters Probe
- Hypervisor MTU Mismatch Probe (Virtual Infra)
- Hypervisor MTU Threshold Check Probe (Virtual Infra)
- Hypervisor and Fabric LAG Config Mismatch Probe (Virtual Infra)
- Hypervisor and Fabric VLAN Config Mismatch Probe (Virtual Infra)
- Hypervisor Missing LLDP Config Probe (Virtual Infra)
- Hypervisor Redundancy Checks Probe (Virtual Infra)
- Interface Flapping Probe
- Interface Policy 802.1x Probe
- MLAG Imbalance Probe
- OSPF Sessions Probe
- Total East/West Traffic Probe
- VMs without Fabric Configured VLANs Probe (Virtual Infra)

Probeの追加 – AOS CLI/SDK

ApstraにビルトインされたProbe以外を使う場合、AOS CLIでインストールする。Predefined Probeには登録されていないが、Apstra SDKからいくつかのprobeを利用することが可能。但、どのようなprobeが用意されていてどのデバイスで利用できるかは、現在非公開。確認が必要。

インストール手順

事前準備

- [AOS-CLIインストール](#)

※AOS-CLIにアクセスするadminのパスワードはAPI/GUIのパスワード。

- [AOS SDKをDL](#)し、Apstraにアップロードし、解凍 ※要アカウント

1. Apstra GUIより、Custom Collector PackageをApstra Agentにインストール。
2. Apstra GUI/CLIより、サービスレジストリのアップデート。(option)
3. Apstra CLIより、Probeを1つずつ選択肢、Apstraにインストール。
4. Apstra GUIより、インストール確認

インストール後のProbeの編集は、ビルトインのProbeと同様。

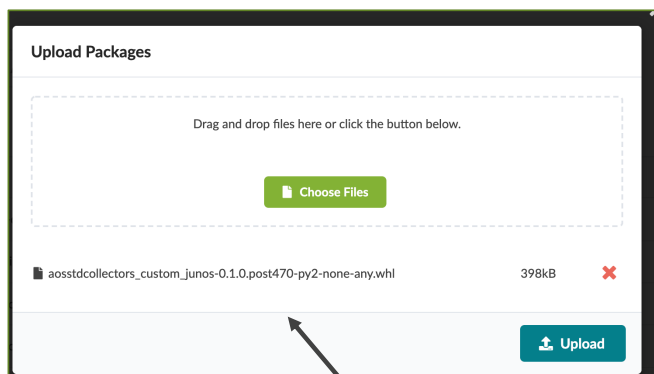
Probeの追加 – AOS CLI/SDK

1. Apstra GUIより、Custom Collector Packageアップロードし、Apstra Agentにインストール。

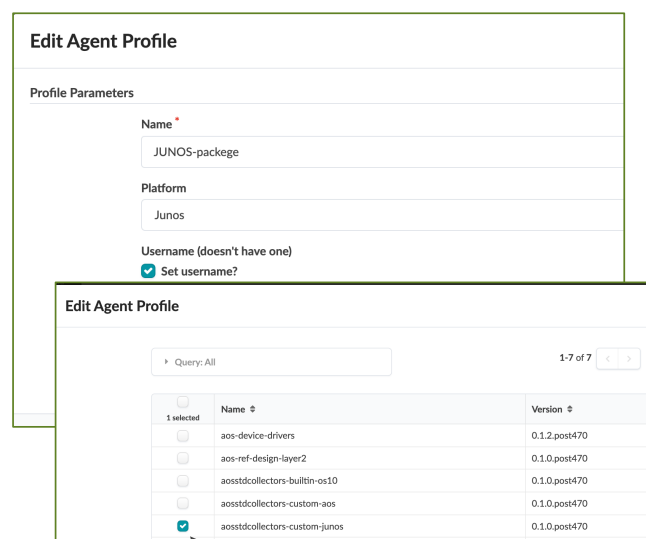
ローカルPCでSDKを解凍し、
Devices > Packages 必要Packageをアップロード。
※junosの場合、aosstdcollectors-custom-junos

Devices > Agent Profilesにて
Profilesを作成

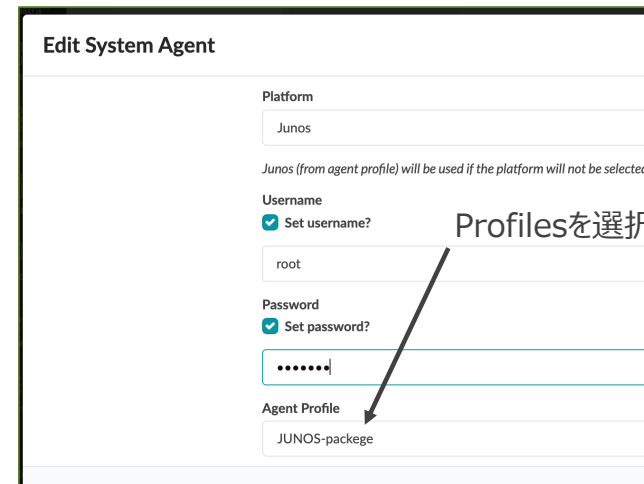
Devices > Agent >各Agent
該当Profilesを選択



Packageを選択



該当Packageを選択



Profilesを選択

Probeの追加 – AOS CLI/SDK

3. AOS CLIより、Probeを選択し、1つずつApstraにインストール

APS CLIログイン

```
$ docker image load -i aoscli-release_4_0_1_12.tar.gz
Loaded image: aoscli:release_4.0.1.12
$ docker run --rm -ti -v $HOME:/mytmp aoscli:release_4.0.1.12 -s 172.27.113.223
```

AOS CLIよりProbeインストール

```
aos> probe create --blueprint 7d47be2e-4ef7-4a8b-a42a-35bfc9a8b6d9 --file
/usr/local/lib/python2.7/site-packages/aos_cli/resources/probes/sfp.j2
```

※AOS CLIでは、以下のように保管され、選択可能

```
aos> probe create --blueprint 7d47be2e-4ef7-4a8b-a42a-35bfc9a8b6d9 --file /usr/local/lib/python2.7/site-packages/aos_cli/resources/probes/sfp.j2
bum_to_total_traffic_anomalies.j2
static_vxlan_vtep_anomalies.j2
counters_error_anomalies.j2
interface_status_anomalies.j2
sfp.j2
arp_usage_anomalies.j2
hardware_vtep_counters_enabled.j2
```

Probeの追加 – AOS CLI/SDK

4.Apstra GUIより、インストール確認

Devices > Services
インストール状況を確認。
エラーが有る場合、ここでエラー確認も可能

Device	Service Started?	Interval	Input	Run Count	Success Count	Failure Count	Max Run Count	Execution Time, ms	Waiting Time, ms	Last Run Timestamp	Last Error Timestamp	Error message
DD272 (spine1, 172.27.115.195)	yes	120		816	816	0		1890.70	0.23	2021-11-30, 18:14:34		N/A
WT3717370014 (1leaf-5110-32-002-leaf1, 172.27.114.93)	yes	120		816	816	0		2214.29	0.27	2021-11-30, 18:14:35		N/A
WT3717370020 (1leaf-5110-32-001-leaf1, 172.27.114.92)	yes	120		816	816	0		2145.13	1.15	2021-11-30, 18:14:34		N/A

エラーがないことを確認

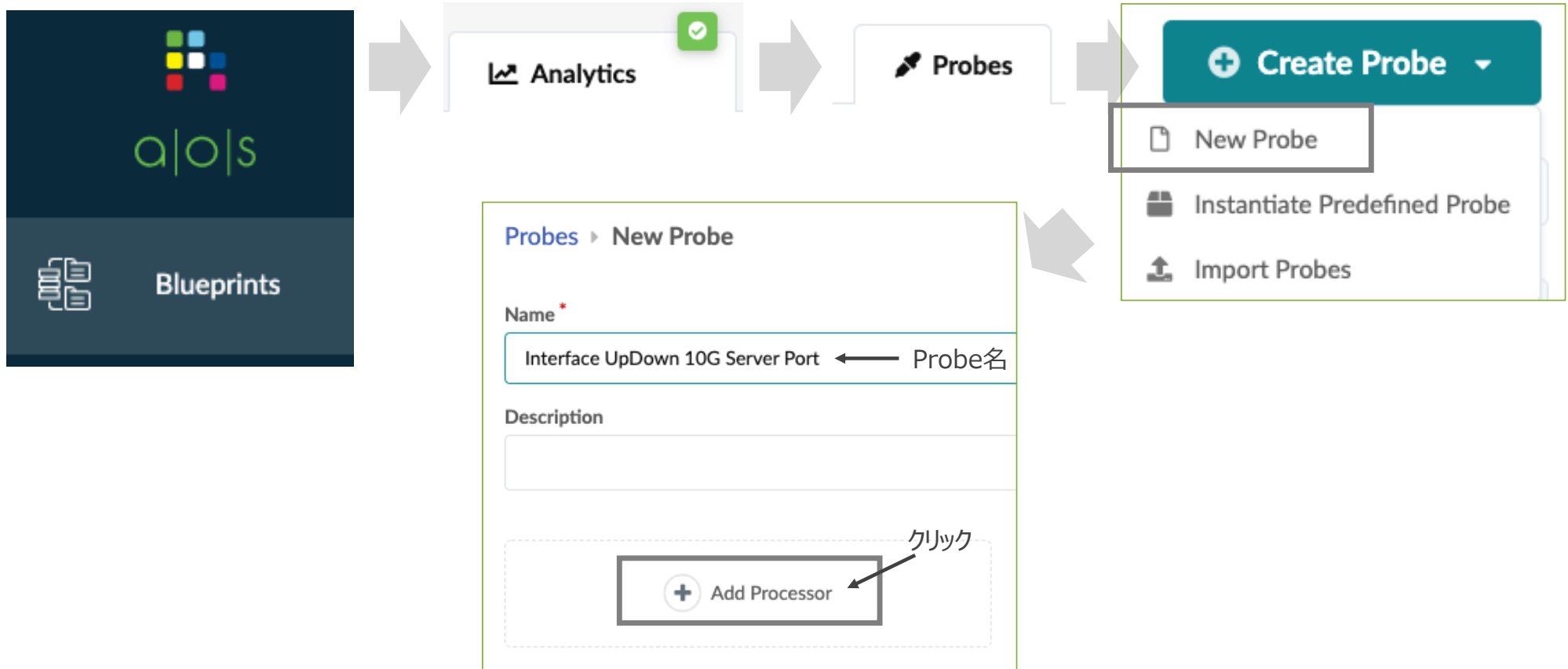
Blueprints > xxx > Analytics > Probes
該当プローブを選択し、状態を確認

各パラメーターの取得状況を確認

System ID	Interface	Interface Desc	Key	Value	Updated
DD272 spine1 Spine	et-0/0/0	facing_1leaf-5110-32-001-leaf1:et-0/0/1	et-0/0/0 media_type	40GBASE CU 1M	a day ago
DD272 spine1 Spine	et-0/0/1	facing_1leaf-5110-32-002-leaf1:et-0/0/1	et-0/0/1 media_type	40GBASE CU 3M	a day ago
WT3717370014 1leaf-5110-32-002-leaf1 Leaf	et-0/0/1	facing_spine1:et-0/0/1	et-0/0/1 media_type	40GBASE CU 3M	a day ago


New Probe

ビルトインで実装されているProcessorを使い、Probeを新規作成。



New Probe - Processor Type

Processorはネットワーク機器から監視情報を取得する「データ取得型」、そのデータをどのように処理するか定義する「データ処理型」に分類される。はじめにデータ取得型を決め、その後データ処理型を紐付ける。

データ取得型Processor	概要
 Interface Counters	インタフェースカウンターを取得
 Service Data Collector	BGP、LLDP、インタフェースのステータス、Hostname
 Generic Service Data Collector	上記以外のデータを取得（出力フォーマット：String）
 Extensible Service Data Collector	上記以外のデータを取得（出力フォーマット：Dictionary）
 Generic Graph Collector	グラフデータベースのクエリに対する応答を出力
 EVPN Type 5	EVPN Type5 Routeを取得
 EVPN Type 3	EVPN Type3 Routeを取得

New Probe - Processor

データ処理型のProcessor一覧。

- Processor: Accumulate
- Processor: Average
- Processor: Comparison
- Processor: Detailed Interface Counters
- Processor: EVPN Type 3
- Processor: EVPN Type 5
- Processor: Extensible Service Data Collector
- Processor: Generic Graph Collector
- Processor: Generic Service Data Collector
- Processor: Interface Counters
- Processor: Match Count
- Processor: Match Percentage
- Processor: Match String
- Processor: Max
- Processor: Min
- Processor: Periodic Average
- Processor: Range
- Processor: Ratio
- Processor: Service Data Collector
- Processor: Set Comparison
- Processor: Set Count
- Processor: Standard Deviation
- Processor: State
- Processor: Subtract
- Processor: Sum
- Processor: System Utilization
- Processor: Time in State
- Processor: Union

データ処理型Processorの詳細はこちら。

<https://portal.apstra.com/docs/probes.html#creating-probe>

New Probe – Processor Data Type

データ処理型のProcessorはその内容により、インプット/アウトプットできるデータタイプが定義されている。

Add Processor (例)

Processor Type * Min Processor Type.

Min Groups as described by group_by, then finds minimum value and outputs it for each group.

Processor Name * Possible input types: NS, NSTS. Possible output types: NS.

データタイプ	概要
Number Set	数値
Text Set	テキスト
Discrete State Set	['up','down']のように複数の中から一つ出力されるもの
Number Set Time Series	時系列のデータ（主にHistoryで使用）
Discreate State Set Time Series	時系列で複数の値を扱うもの（主にHistoryで使用）

New Probe

Probeの新規作成の続き。

(例) サーバーが接続され、10G SpeedのインタフェースUp/Downをモニタ。
インタフェースがダウンしたらアラートを発呼。

Add Processor

Processor Type *
Service Data Collector ← データ取得Processorを選択

Processor Name *
Input UpDown Status ← 任意のProcessor名

Output Stage Name: out *
Output UpDown Status ← データ処理Processorに引き渡す任意の名前

Service Data Collector Processor.
Collects data from the specified service. For example, for 'bgp' service that would be status of BGP sessions. Objects to be monitored are configured via the graph query and key. In our example with BGP, key should evaluate to (localIp, localAs, remoteIp, remoteAs). For interface-based services such as 'interface' and 'lldp', key is an interface name.

This processor has no inputs.

Add

New Probe

Input UpDown Status

Output UpDown Status

+ Add Processor

Processor: Input UpDown Status

Service Data Collector

Properties

Graph Query *

```
node('system', name='sys_one', role='leaf')
  .out('hosted_interfaces')
  .node('interface', name='int_one')
  .out('link')
  .node('link', name='link', speed='10G')
  .in('link')
  .node('interface', name='int_two')
  .in('hosted_interfaces')
  .node('system', name='sys_two', role='l2_server')
  .ensure_different('int_one', 'int_two')
```

Graphクエリで以下の条件を定義。

- RoleがLeafであること。
- リンクスピードが10Gであること。
- 接続先がL2サーバであること。

+ Add Graph Query

One or more queries on the graph to get nodes to be monitored. Results from all queries are concatenated and they must have the same named nodes as names used in properties.

Service Name *

interface ← サービス名をプルダウンから選択

Keys *

interface × ← 監視するグラフクエリのプロパティ

List of property names which values will be used as a key parameters for the service.

System ID *

sys_one.system_id ← グラフクエリの'name'.プロパティ'system_id'

Expression mapping from graph query to a system_id, e.g. "system.system_id" if "system" is a name in the graph query.

Additional keys ← 監視項目に加えたキー

interface ← str(int_one.if_name) ← グラフクエリの'name'.プロパティ'if_name'

+ Add Key

Enable Streaming

Makes samples of output stages streamed if enabled.

New Probe – Graph Database

ApstraはGraph Databaseでネットワーク全体を管理している。
IBAで監視データを取得する際、ターゲットのノードをクエリ構文で定義する。

Graph Query *

```
node('system', name='sys_one', role='leaf') ← 'system'ノード
.out('hosted_interfaces') ← 'system'と'interface'のリレーション名
.node('interface', name='int_one') ← 'interface'ノード
.out('link') ← 'interface'と'link'のリレーション名
.node('link', name='link', speed='10G') ← 'link'ノード
.in_('link')
.node('interface', name='int_two')
.in_('hosted_interfaces')
.node('system', name='sys_two', role='l2_server')
.ensure_different('int_one', 'int_two')
```

※ノード内の'system'や'interface'といったラベルや、リレーション名は予めDB内で定義されている。

Graph Databaseの詳細はこちら。

https://portal.apstra.com/docs/database_concepts.html

New Probe

Probeの新規作成の続き。

データ処理型Processorを追加

Add Processor

Processor Type *
State ← Processorタイプを選択

Processor Name *
Interface UpDown Status Check ← 任意のProcessor名

Output Stage Name: out *
Interface UpDown Status Check ← 任意のStage名

Check that a value in states.
Accepts DSS or DSTS as types for the 'in' input and checks that a value is one of the specified anomalous states. Outputs DSS with anomaly values, such as 'true' if the value is in the specified states and 'false' otherwise.

Add

New Probe

The screenshot shows the configuration page for a probe named "Interface UpDown Status Check". On the left, a flow diagram shows the "Input UpDown Status" stage connected to the "Interface UpDown Status Check" processor. Below this is an "Add Processor" button.

The main configuration area is titled "Processor: Interface UpDown Status Check" and includes a "State" toggle. It contains several sections:

- Inputs:** "Input Stage: in" is set to "Output UpDown Status". A Japanese annotation "作成済みのデータ取得Processorを選択" (Select the existing data acquisition processor) points to this field.
- Properties:**
 - Graph Query:** "No items." with an "Add Graph Query" button.
 - Anomalous States:** Set to "'down'". A Japanese annotation "Anomalyと判断するステータス" (Status used to judge anomaly) points to this field.
 - Anomaly MetricLog Retention Duration:** Set to "86400". A Japanese annotation "ログを保持する時間" (Time to retain logs) points to this field.
 - Anomaly MetricLog Retention Size:** Set to "1073741824". A Japanese annotation "ログの保管サイズ" (Log storage size) points to this field.
 - Anomaly Metric Logging:** A checkbox that is currently unchecked.
 - Enable Streaming:** A checked checkbox. A Japanese annotation "Anomalyの外部モニタリングサーバへの通知有無" (Whether to notify external monitoring server of anomaly) points to this field.
 - Raise Anomaly:** A checked checkbox. A Japanese annotation "Anomalyと判断した時のアラート有無" (Whether to alert when anomaly is judged) points to this field.

At the bottom right, there is a large blue "Create Probe" button.

New Probe

Probes ▶ Interface UpDown 10G Server Port Operational No anomalies admin in a few seconds Enabled ON

Search stages...

Input UpDown Status

Output UpDown Status

Interface UpDown Status Check

Interface UpDown Status Check

Stage: Interface UpDown Status Check Discrete State Set

Anomalies Only

Query: All

false true

System ID	Interface	Anomaly	Value
5254005CFCBF evpn-esi-001-leaf2 Leaf	xe-0/0/1	No anomaly	false
5254005CFCBF evpn-esi-001-leaf2 Leaf	xe-0/0/2	No anomaly	false
52540081FC6E evpn-single-001-leaf1 Leaf	xe-0/0/2	No anomaly	false

ProcessorのStageからステータスを確認できる。
ProbeのAnomalyはDefault Telemetryの
Dashboardでも確認可。

All Probes

All Probes
0 anomalies

Widgets

Analytics用のDashboardを作成するため、widgetsで監視するProbeのProcessor Stageを選択。

Type *
Stage ← Stageを選択

Name * ← 任意の名前
Interface UpDown Status Check

Stage *

	Visible
System ID	<input checked="" type="checkbox"/>
Interface	<input checked="" type="checkbox"/>
Anomaly	<input checked="" type="checkbox"/>
Value	<input checked="" type="checkbox"/>
Updated	<input checked="" type="checkbox"/>

Widgets

Show Contextual Information チェックするとメーター形式で表示

Show information from related stages.

▶ Query: All

Spotlight View

Show an extended view of a single stage item.

↓ チェックすると複数ノード分を1つにまとめる。

System ID	Value
5254003804CB spine2 <small>Spine</small>	
525400523B46 leaf2 <small>Leaf</small>	
5254009B6BB0 leaf1 <small>Leaf</small>	

System ID	Value
525400523B46 leaf2 <small>Leaf</small>	1
5254009B6BB0 leaf1 <small>Leaf</small>	1
525400CA6856 leaf3 <small>Leaf</small>	2

チェックしないとグラフで表示。

System ID: 5254003804CB
spine2

Updated

System ID	Value
5254003804CB spine2 <small>Spine</small>	
525400523B46 leaf2 <small>Leaf</small>	
5254009B6BB0 leaf1 <small>Leaf</small>	

< 1 of 5 >

チェックしないとノード分を表示。

Dashboard

WidgetsからAnalytics用のDashboardを作成。

The diagram illustrates the workflow for creating a dashboard. It begins with the 'a|o|s' logo, followed by the 'Analytics' section (marked with a green checkmark), then the 'Dashboards' section, and finally the 'Create Dashboard' button. A dropdown menu is shown with 'New Dashboard' selected.

The 'New Dashboard' configuration page includes the following fields and options:



- Name:** Network - Layer 1 (Annotated as 'Dashboard名')
- Description:** (Empty text field)
- Layout:** One-column, **Two-column** (Annotated as 'カラム数'), Three-column. Note: Dashboards may have up to three columns for widgets.
- Default:** OFF (Annotated as 'Default TelemetryのDashboardに写す場合はON.'). Note: Default analytics dashboard will be shown on the blueprint's dashboard.
- Widget Selection:** Two buttons: '+ Add Existing Widget' and '+ Create New Widget' (Annotated as 'クリックしてwidgetsを選択').


Dashboard

Add Widget

Interface Status ← Widgets選択

Add Widget

Interface Status  



System ID	5254003804CB spine2 Spine
Updated	18 days ago

[View stage](#)

クリックしてwidgetsを選択

+ Add Existing Widget

OR

+ Create New Widget

Add Widget

Interface Flapping (Fabric) ← Widgets選択

Add Widget

Create Dashboard

その他の監視機能

- ・ トラフィック量カラーリング
- ・ 外部モニタリングサーバ
- ・ ネットワーク障害解析

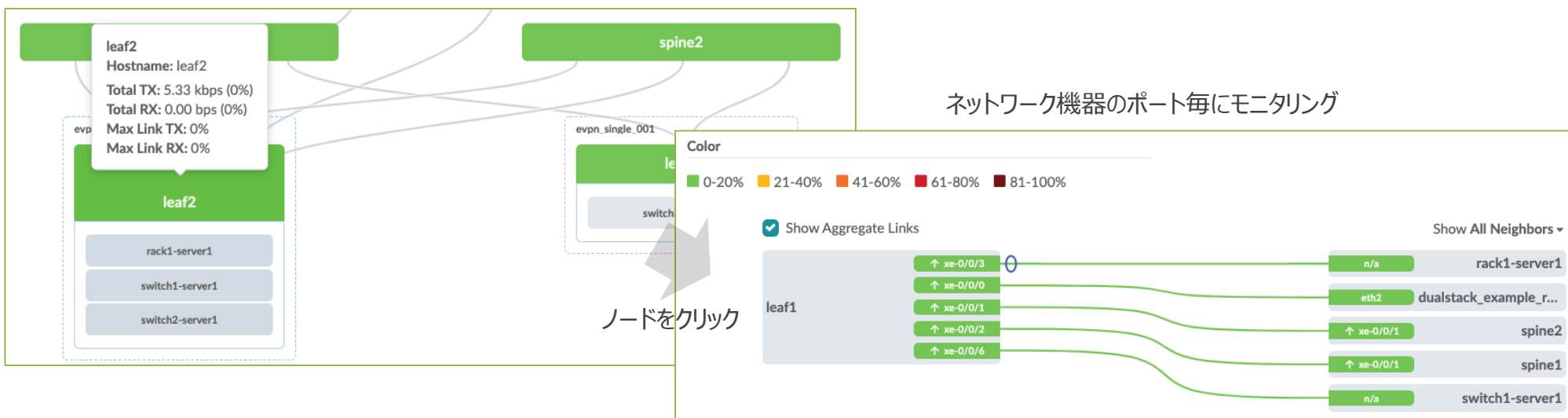


トラフィック量カラーリング

ネットワークのトラフィック量をApstraのGUIでカラーリング。
※AnalyticsのProbe “Device Traffic”が有効であること。

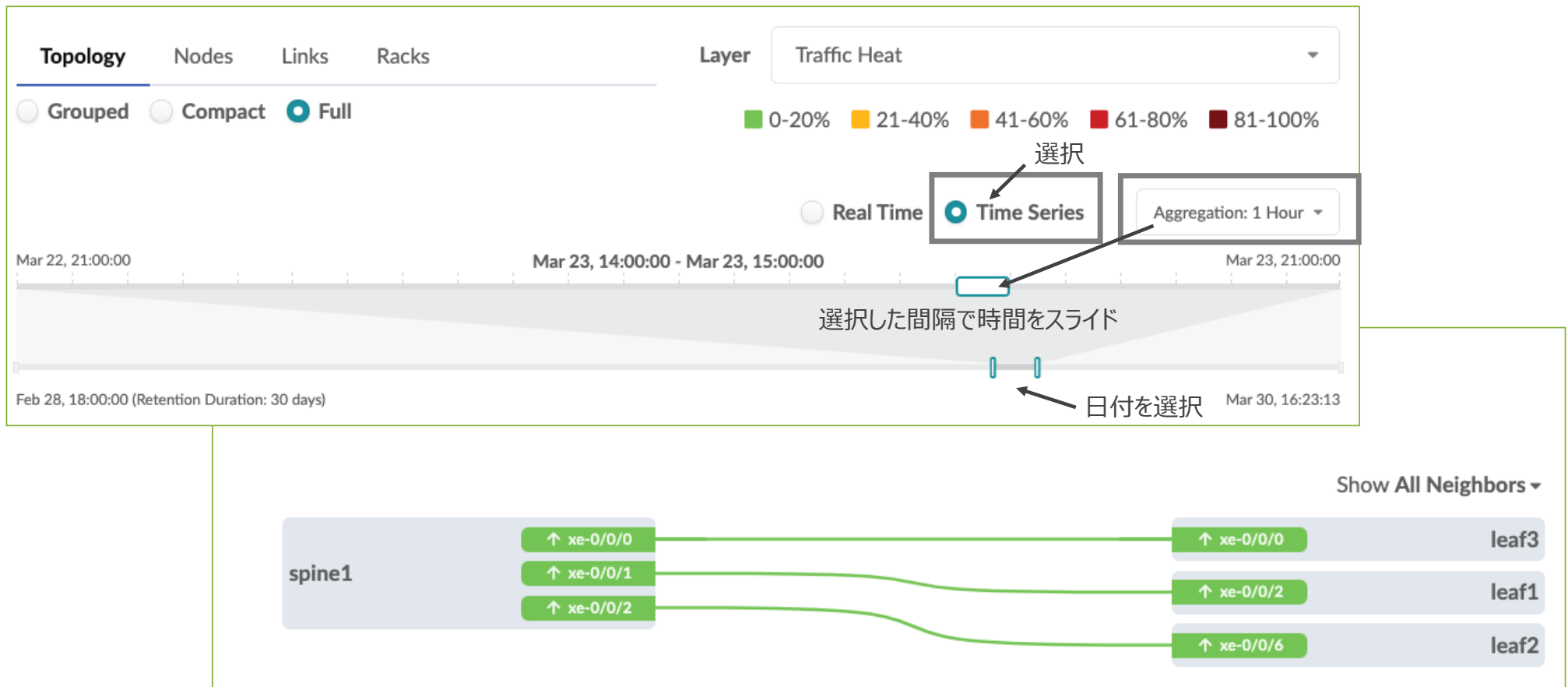


ネットワーク全体をモニタリング



トラフィック量カラーリング

Time Seriesを選択すると、過去に遡りトラフィック量を可視化できる。



外部モニタリングサーバ

Apstraが保持する監視データやイベントを外部サーバへGPBで転送できる。
Juniperがプラグインを開発しサポートするメトリクスコレクタ。

- Telegraf
- Elastic Stack (次期メジャーバージョンでサポート予定)

Telegraf設定方法

Receiver Configuration

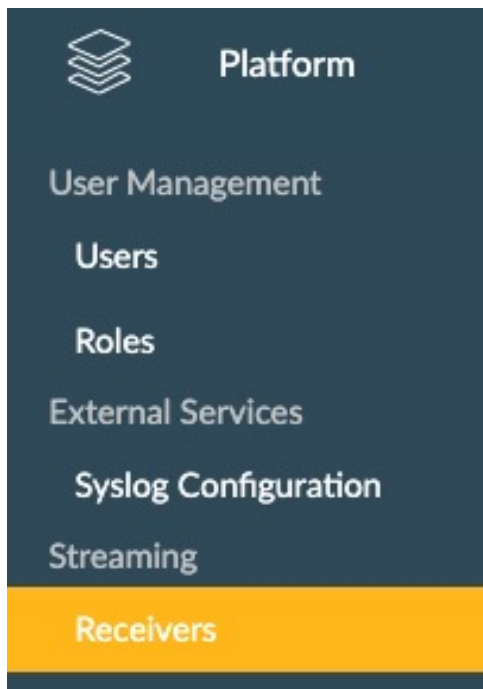
https://portal.apstra.com/docs/receiver_configuration.html

Apstra github aosom-streaming

<https://github.com/Apstra/aosom-streaming>

外部モニタリングサーバ

外部モニタリングサーバの登録。



+ Create Receiver

Hostname *
1.2.3.4 ← 受信サーバIPアドレス

Port *
4444 ← 受信サーバポート番号

Message Type *
Alerts ← 送信するメッセージタイプ

Sequencing Mode *
Sequenced ← (※1)

タイプ	内容
Alerts	イベント内容の通知 (Prometheus用)
Events	イベント内容の通知 (InfluxDB用)
Perfmon	Stats等のカウンターデータ

Create

(※1)

https://portal.apstra.com/docs/aosom_streaming.html?highlight=sequenced#aosom-streaming-reconfiguration-after-aos-server-upgrade

ネットワーク障害解析

Apstraは管理しているネットワークのデータベースから、ネットワーク障害の原因を報告。

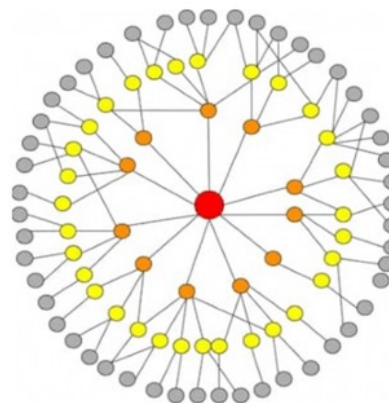
- ネットワーク障害の早期復旧は運用上の最重要課題。
- 障害原因の確認、復旧作業にかかる時間の短縮が求められている。

(例)

インタフェースダウン

LLDPエラー

BGPネイバーダウン



真の原因を解析・報告

ネットワーク障害解析

Root Cause Identificationの設定。



Description	Timestamp	Symptoms
障害の真の原因を報告 Disconnected between spine2 and evpn-esi-001-leaf2	a few seconds ago	障害の事象を報告 <ul style="list-style-type: none">BGP session down from spine2 (ASN: 64513 IP: 172.16.0.8) to evpn-esi-001-leaf2 (ASN: 64515 IP: 172.16.0.9) (link)BGP session down from evpn-esi-001-leaf2 (ASN: 64515 IP: 172.16.0.9) to spine2 (ASN: 64513 IP: 172.16.0.8) (link)Interface xe-0/0/2 is miscabled on host evpn-esi-001-leaf2 (link)Interface xe-0/0/1 is miscabled on host spine2 (link)



Thank you

JUNIPER
NETWORKS

Engineering
Simplicity