NetFlow データ収集

ここでは、次の内容について説明します。

NetFlow データ収集

WAE は、エクスポートされた NetFlow および関連するフロー測定値を収集して集約できます。これらの測定値を使用して、WAE Design の正確なデマンドトラフィックデータを構築できます。フロー収集は、デマンド推論を使用したインターフェイス、LSP、およびその他の統計からのデマンドトラフィックの推定に代わる手段を提供します。NetFlow は、トラフィックフローに関する情報を収集し、トラフィックとデマンドのマトリックスを構築するのに役立ちます。フロー測定値のインポートは、ネットワークのエッジルータのフローカバレッジが完全またはほぼ完全な場合に特に役立ちます。さらに、外部の自律システム（AS）間の個々のデマンドの精度が重要な場合にも役立ちます。

トポロジ、BGP ネイバー、インターフェイス統計など、NIMO によって個別に収集されたネットワークデータは、フロー測定値と組み合わされてフローをスケーリングし、外部の自律システムと内部のノードの両方の間で完全なデマンドメッシュを提供します。

WAE は、次のタイプのデータを収集して、フローとそのトラフィック測定値を時間の経過とともに集約したネットワークモデルを構築します。

- NetFlow、JFlow、CFlowd、IPFIX、および Netstream フローを使用したフロートラフィック
- SNMP 経由のインターフェイス トラフィックと BGP ピア
- ピアリングセッション上の BGP パス属性

NetFlow 収集アーキテクチャ

フロー収集アーキテクチャには 2 つのタイプがあります。

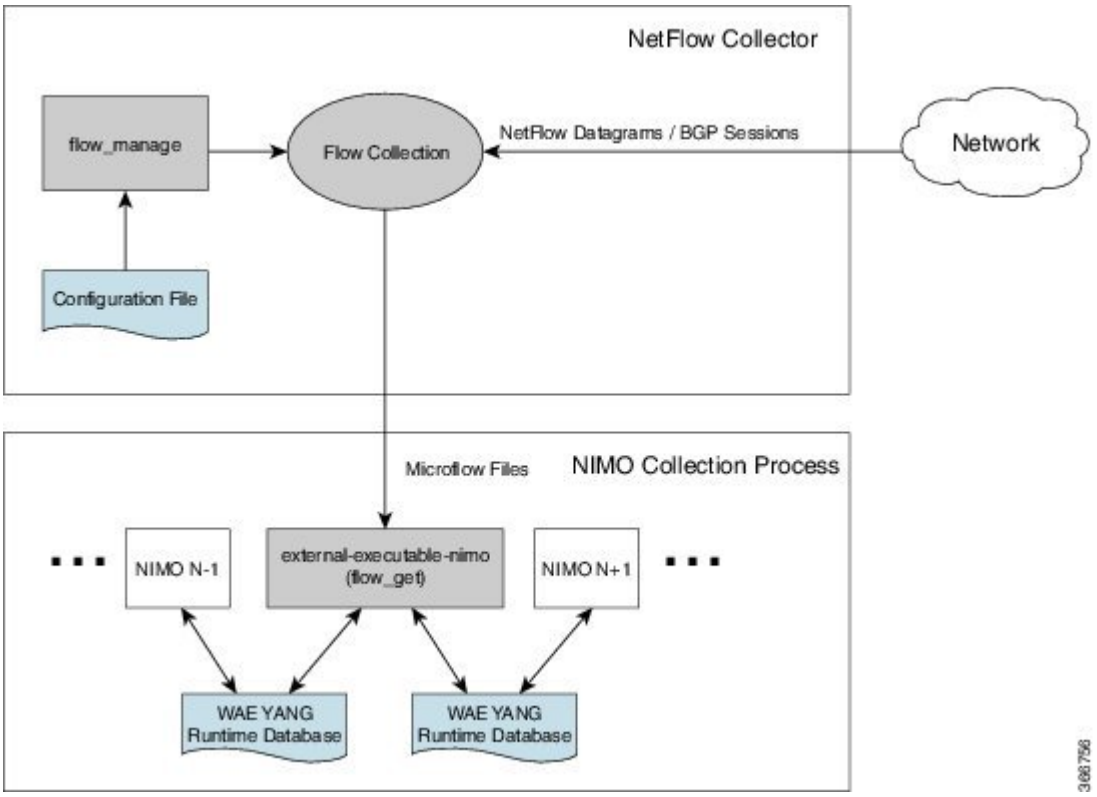
（注） 展開する収集アーキテクチャは、ネットワークからの NetFlow トラフィックエクスポートの測定レートまたは推定レート（Mbps または fps 単位）によって異なります。

- 集中型 NetFlow（CNF）：通常、小規模から中規模のネットワークに使用されます。これは単一サーバーアーキテクチャです。
- 分散 NetFlow（DNF）：通常、大規模なネットワークに使用されます。このアーキテクチャは、JMS ブローカ、マスター、およびエージェントで構成されています。

CNF 収集

次の図は、CNF でフローデータを収集および計算するためのワークフローを示しています。WAE Collector CLI ツールである `flow_manage` および `flow_get` は、それぞれ外部構成ファイルおよびNIMO収集プロセスと統合されています。フローベースのデマンドおよびデマンドトラフィックは、WAE YANG ランタイムシステムに渡されます。

図 *1*: 一元的な収集とデマンドの作成



- **flow_manage**：この CLI ツールは、ネットワーク接続を設定し、フロー収集プロセスの開始、停止、構成など収集サーバーを管理します。構成ファイルの<NodeFlowConfigs>テーブルからの入力を使用して、構成情報を生成し、フロー収集サーバーに送信します。
- **フロー収集サーバー**：このバックグラウンドプロセスは、`flow_manage` から構成情報を受け取り、それを使用して収集サーバーを構成し、フローデータと BGP 属性を受け取ります。次に、収集サーバーはこのデータを集約し、マイクロフローファイルを `flow_get` ツールに転送します。
- **flow_get**：この CLI ツールは、`nimo_flow_get.sh` スクリプト内で構成され、external-executable-nimo 内で実行されます。収集サーバーからフローデータ（マイクロフローファイル）を読み取り、NetFlowデマンドとデマンドトラフィックデータを生成して、このデータを WAE YANG ランタイムデータベースに挿入します。デマンドデータとトラフィックデータの生成に加えて、`flow_get` は Inter-AS（IAS）フローファイルも生成します。

（注） 実稼働ネットワークでは、`flow_get` に -log-level=INFO | DEBUG | TRACE を使用しないでください。

DNF 収集

次の図は、DNF アーキテクチャと DNF ワークフローを示しています。このアーキテクチャでは、ネットワークデバイスの各セットがフローデータを対応する収集サーバーにエクスポートします。DNF クラスタはフロー計算を実行するため、各エージェントは、フローコレクタを実行する対応するフロー収集サーバーのフロー計算を担当します。マスターノードはこの情報を集約し、`flow_collector_ias` に返します。

図 *2 : DNF* アーキテクチャ

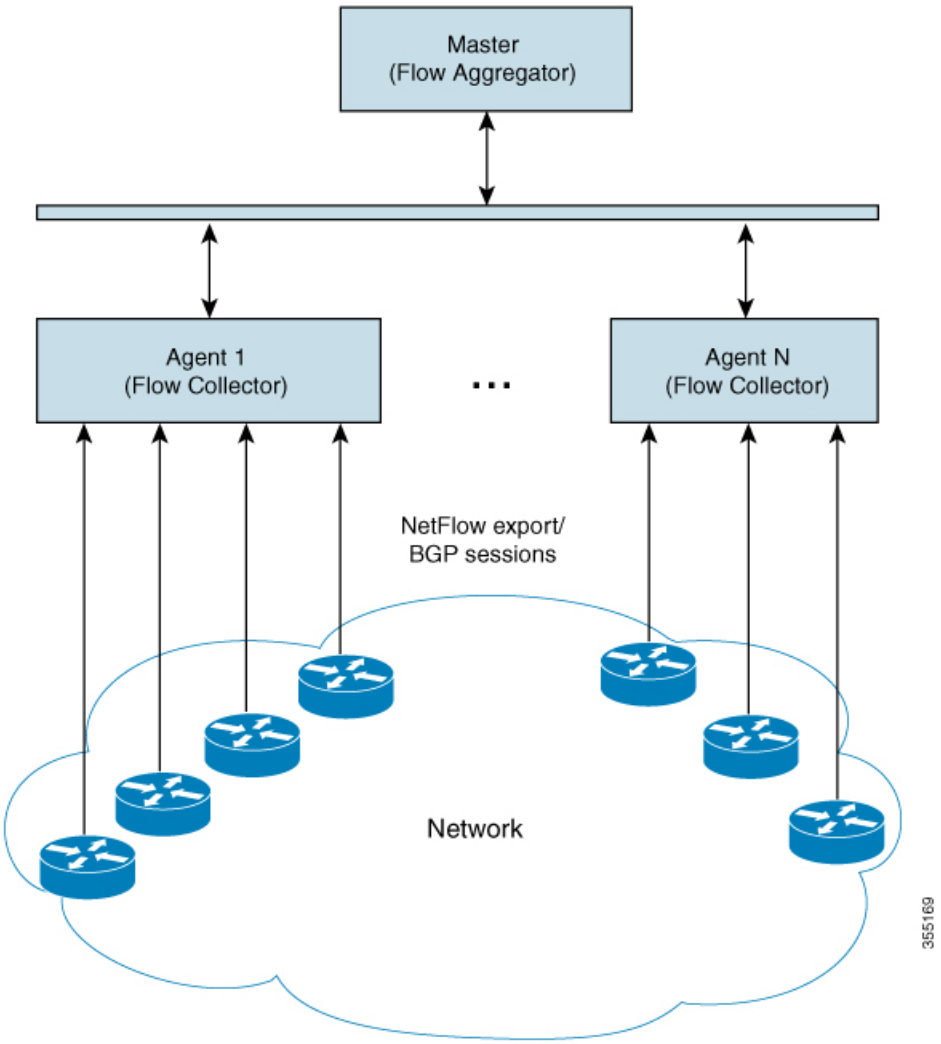
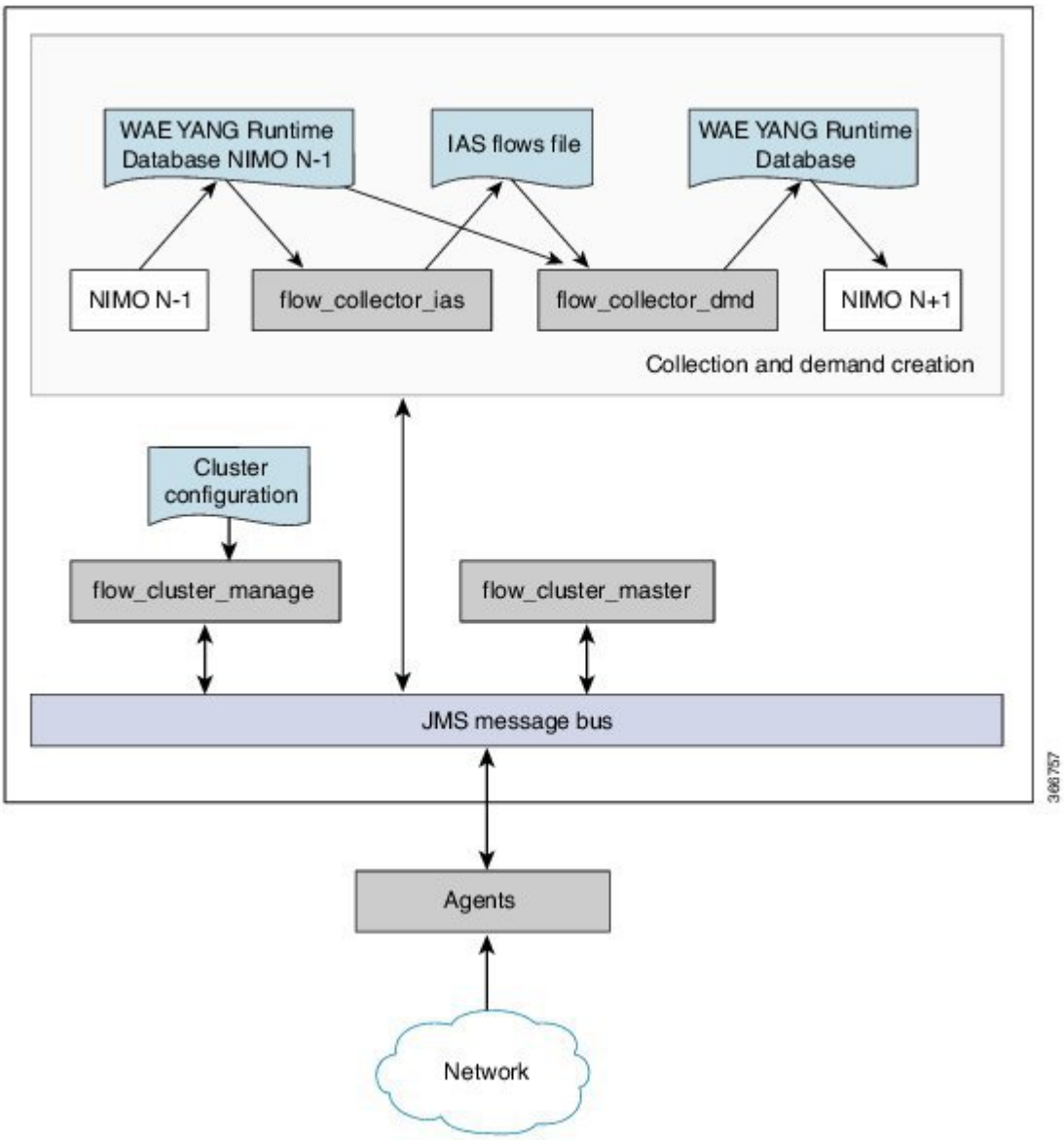


図 *3 : DNF* 収集ワークフロー



- **flow_cluster_manage**：このCLIツールは、クラスタの構成とステータスの取得に使用されます。クラスタ構成ファイルを受け取り、構成をクラスタに送信します。詳細については、「DNF 構成ファイルの使用（flow_cluster_manage の実行）（22 ページ）」を参照してください。

  `flow_cluster_manage` を使用する代わりに、REST API を使用して、クラスタのステータスを構成およびリクエストすることもできます。詳細については、次のいずれかの場所にある API ドキュメントを参照してください。

  - `<wae-installation-directory>docs/api/netflow/distributed-netflow-rest-api.html`
  - http://<*master-IP-address*>:9090/api-doc たとえば、クラスタ構成を取得するには：

たとえば、クラスタ構成を取得するには：

```
curl -X GET http://localhost:9090/cluster-config > config-file-1
```

たとえば、クラスタ構成を設定するには：

```
curl -X PUT http://localhost:9090/cluster-config @config-file-2
```

たとえば、クラスタのステータスを取得するには：

```
curl -X GET http://localhost:9090/cluster-status > config-file-1
```

- **flow_cluster_master**：マスターサービスは、すべてのエージェントからのすべてのフローデータ結果を収集し、データを集約して、flow_collector_ias に返します。詳細については、「マスターとエージェント （13 ページ）」を参照してください。
- **flow_cluster_agent**：エージェントサービスは、関連付けられたフローコレクタのステータスを管理および追跡します。各エージェントは、対応する収集サーバーからフローデータを受信して計算します。
- **flow_cluster_broker**：（図には示されていない）JMS ブローカサービスは、マスターとエージェントを含むアーキテクチャ内のすべてのコンポーネント間の通信を可能にします。詳細については、「Java メッセージサーバー（JMS）ブローカ （13 ページ）」を参照してください。
- **flow_collector_ias**：この CLI ツールは、nimo_flow_collector_ias_and_dmd.sh ファイル内で構成され、external-executable-nimo 内で実行され、マスターからフローデータを受信し、IAS フローファイルを生成します。詳細については、「flow_collector_ias および flow_collector_dmd の構成 （24 ページ）」を参照してください。
- **flow_collector_dmd**：この CLI ツールは、NetFlow デマンドとデマンドトラフィックを WAE YANG ランタイムデータベースに送信します。nimo_flow_collector_ias_and_dmd.sh ファイル内で構成され、external-executable-nimo 内で実行されます。

（注） 実稼働ネットワークでは、flow_collector_ias または flow_collector_dmd に -log-level=INFO | DEBUG | TRACE を使用しないでください。

集中型 NetFlow 構成ワークフロー

CNF を構成して収集を開始するには、次の手順を実行します。

（注） 特に明記されていない限り、WAE のインストール中に展開されたファイルの権限を変更しないでください。

ステップ 1 CNF NetFlow の要件 （7 ページ）が満たされていることを確認します。
ステップ 2 CNF 用のオペレーティングシステムの準備 （7 ページ）
ステップ 3 CNF 構成ファイルの作成 （8 ページ）
ステップ 4 CNF 構成ファイルの使用（flow_manage の実行） （9 ページ）
ステップ 5 CNF 収集の構成 （10 ページ）
a) flow_get の構成 （10 ページ）
b) CNF 用の external-executable-nimo の構成 （11 ページ）

CNF NetFlow の要件

システム要件については、『Cisco WAE System Requirements』ドキュメントを参照してください。

ライセンシング

`flow_manage` および `flow_get` ツールを使用するときに、フローおよびフローデマンドを取得するための正しいライセンスがあることを Cisco WAE の担当者に確認してください。

CNF 用のオペレーティングシステムの準備

OS を CNF 用に準備するには、WAE CLI から次の `flow_manage` コマンドを実行します。

```
sudo -E ./flow_manage -action prepare-os-for-netflow
```

`prepare-os-for-netflow` オプションは、次の処理を実行します。

- `setcap` コマンドを使用して、非ルートユーザーに特権ポート（0 ～ 1023）への制限付きアクセスを許可します。これは、フローコレクタが 1024 未満のポートを使用して BGP メッセージをリッスンするように構成する場合に必要です。
- CNF アーキテクチャで `flow_get` によって生成される可能性のある大量の一時ファイルを考慮して、最大 15,000 のファイル記述子を予約するように OS インスタンスを構成します。

（注） このコマンドの実行後、サーバーを再起動する必要があります。

NetFlow 収集の構成

フロー収集プロセスは、入力方向のルータによってキャプチャおよびエクスポートされる IPv4 および IPv6 フローをサポートしています。また、IPv4 および IPv6 iBGP ピアリングもサポートしています。

ルータは、フローをフロー収集サーバーにエクスポートし、フロー収集サーバーとの BGP ピアリングを確立するように構成する必要があります。次の推奨事項に留意してください。

- NetFlow v5、v9、および IPFIX データグラムは、フロー収集サーバーの UDP ポート番号にエクスポートされます。デフォルト設定は 2100 です。IPv6 フローのエクスポートには、NetFlow v9 または IPFIX が必要です。
- ルータでフロー収集サーバーを iBGP ルート リフレクタ クライアントとして構成し、BGP ルートをエッジルータまたは境界ルータに送信できるようにします。これが不可能な場合は、関連するすべてのルーティングテーブルの完全なビューを持つルータまたはルート サーバーを構成します。
- フロー エクスポート データグラムの送信元 IPv4 アドレスが iBGP メッセージの送信元 IPv4 アドレスと同じネットワークアドレス空間にある場合は、同じアドレスになるように構成します。
- BGP ルータ ID を明示的に構成します。
- 静的ルーティングを構成します。
- BGP ルートを受信する場合、BGP **AS_path** 属性の最大長は 3 ホップに制限されます。その理由は、単一の IP プレフィックスに付加された BGP 属性（**AS_path** を含む）の合計長が非常に大きくなる（最大 64 KB）可能性があることを考慮して、過度のサーバーメモリ消費を防ぐためです。

CNF 構成ファイルの作成

<NodeFlowConfigs> テーブルには、フロー収集サーバーに渡す構成情報を生成するときに、`flow_manage` ツールによって使用される基本的なノード構成情報が含まれています。したがって、`flow_manage` を実行する前に、このテーブルを次のように作成する必要があります。

- タブまたはカンマ区切り形式を使用します。
- フローデータを収集するノード（ルータ）ごとに 1 行を含めます。
- これらのノードごとに、次の表に記載されている内容を入力します。BGP の列は、BGP 情報を収集する場合にのみ必要です。

表 1 : <NodeFlowConfigs> テーブルの列

カラム	説明
名前	ノード名

カラム	説明
SamplingRate	ノードからエクスポートされたフローのパケットのサンプリングレート。たとえば、値が 1,024 の場合、1,024 あるパケットから 1 つが決定論的またはランダムな方法で選択されます。
FlowSourceIP	フローエクスポートパケットの IPv4 送信元アドレス。
BGPSourceIP	iBGP 更新メッセージの IPv4 または IPv6 送信元アドレス。 この列は、`flow_manage -bgp` オプションが true の場合に必要です。
BGPPassword	MD5 認証の BGP ピアリングパスワード。 この列は、`flow_manage -bgp` オプションが true で、BGPSourceIP に値がある場合に使用します。

以下は、<NodeFlowConfigs> テーブルの例です。

名前	SamplingRate	FlowSourceIP	BGPSourceIP	BGPPassword
paris-er1-fr	1024	192.168.75.10	69.127.75.10	ag5Xh0tGbd7
chicago-cr2-us	1024	192.168.75.15	69.127.75.15	ag5Xh0tGbd7
chicago-cr2-us	1024	192.168.75.15	2001:db9:8:4::2	ag5Xh0tGbd7
tokyo-br1-jp	1024	192.168.75.25	69.127.75.25	ag5Xh0tGbd7
brazilia-er1-bra	1024	192.168.75.30	2001:db8:8:4::2	ag5Xh0tGbd7

CNF 構成ファイルの使用（flow_manage の実行）

`flow_manage` ツールは、フロー収集プロセス（pmacct）を開始および停止したり、<NodeFlowConfigs> テーブルを変更するときに保存された構成情報をリロードしたりします。そのため、CNF 収集プロセスを実行する前に実行する必要があります。

```
flow_manage -server-ip 198.51.100.1 -action start -node-flow-configs-table flowconfigs.txt
```

システムの起動時またはシャットダウン時に、`flow_manage` を自動的に開始および停止するようにオペレーティングシステムを構成することをお勧めします。

次のコマンドは、flowconfigs.txt ファイル内の <NodeFlowConfigs> テーブルを、192.168.1.3 の IP アドレスを持つフロー収集サーバーにリロードします。

```
flow_manage -server-ip 198.51.100.1 -action reload -node-flow-configs-table flowconfigs.txt
```

サンプル構成ファイル

```
<NodeFlowConfigs>
Name,BGPSourceIP,FlowSourceIP,BGPPassword,SamplingRate
ar1.dus.lab.test.com,1.2.3.4,1.2.3.5,bgp-secret,666
ar1.ham.lab.test.com,1.2.3.41,1.2.3.52,bgp-secret-2,667
cr1.ams.lab.test.com,1.2.3.51,1.2.3.53,bgp-secret-3,8000
<IPPrefixFiltering>
NetworkAddress
198.51.100.1/24
198.51.100.1/23
198.51.100.1/22
198.51.100.1/21
```

`flow_manage` オプションの詳細については、*`wae-installation-directory`*`/bin` に移動し、**`flow_manage -help`** と入力してください。

CNF 収集の構成

flow_get の構成

この CLI ツールは、`<WAE_installation_directory>/etc/netflow/ansible/bash/nimo_flow_get.sh` スクリプト内で構成され、external-executable-nimo 内で実行されます。このツールは、トポロジ NIMO ネットワークモデルとフロー収集サーバーからのデータを結合します。

編集する前に、このファイルの権限を変更します。

```
chmod +x nimo_flow_get.sh
```

`nimo_flow_get.sh` を次のように編集します。

- **CUSTOMER_ASN**：ASN を入力します。
- **SPLIT_AS_FLOWS_ON_INGRESS**：複数の外部 ASN が IXP スイッチに接続されている場合、すべての ASN からのトラフィックを集約するのか、MAC アカウンティング入力トラフィックに比例して分散するのかを決定します。デフォルト値は aggregate です。もう 1 つの値は mac-distribute です。
- **ADDRESS_FAMILY**：含めるプロトコルバージョンのリストを入力します（カンマ区切りのエントリ）。デフォルトは ipv4,ipv6 です。

`nimo_flow_get.sh` の例：

```
#!/bin/bash
# modify as needed - BEGIN
CUSTOMER_ASN=4103291
SPLIT_AS_FLOWS_ON_INGRESS=aggregate
ADDRESS_FAMILY=ipv4,ipv6
# modify as needed - END
```

flow_get オプションの詳細については、https://www.cisco.com/c/en/us/td/docs/net_mgmt/wae/6-4/platform/configuration/guide/WAE_Platform_Configuration_Guide/wp_netflow.html#pgfId-1082437を参照するか、または *wae-installation-directory*/bin に移動し、**flow_get -help** と入力してください。

CNF 用の external-executable-nimo の構成

external-executable-nimo は、選択したネットワークモデルに対して nimo_flow_get.sh スクリプトを実行します。この場合、WAE で作成された既存のモデルを取得し、nimo_flow_get.sh からの情報を追加して、必要なフローデータを含む最終ネットワークモデルを作成します。

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他のNIMO収集を含む最終ネットワークモデルです。
- 集中型 NetFlow 構成ワークフロー （6 ページ）の準備作業が完了したことを確認します。

ステップ 1 エキスパート モードから、**/wae:networks** に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。簡単に識別できる一意の名前をお勧めします。たとえば、networkABC_CNF_flow_get などです。

ステップ 3 [nimo] タブをクリックします。

ステップ 4 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[external-executable-nimo] を選択します。

ステップ 5 [external-executable-nimo] をクリックし、送信元ネットワークを選択します。

ステップ 6 [advanced] タブで、以下の情報を入力します。

- [input-file-version]：**7.1** と入力します。
- [input-file-format]：送信元ネットワークモデルのプランファイルフォーマットとして [.pln] を選択します。
- [argv]：***<directory_path>*/nimo_flow_get.sh $$input $$output** を入力します。

ステップ 7 構成を確認するには、[external-executable-nimo] タブから [run] をクリックします。

例

WAE CLI を（構成モードで）使用している場合は、次のように入力します。

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network> advanced argv nimo_flow_get.sh $$input $$output ]
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
admin@wae(config-network-<network-model-name>)# exit
admin@wae(config)# exit
```

```
admin@wae# networks network <network-model-name> nimo external-executable-nimo run
```

次のタスク

external-executable-nimo を構成したら、WAE Design からデータの実行またはアクセスをスケジュールできます。

DNF NetFlow 構成ワークフロー

DNF を構成して収集を開始するには、次の手順を実行します。

（注） 特に明記されていない限り、WAE のインストール中に展開されたファイルの権限を変更しないでください。

ステップ 1 分散 NetFlow の要件 （12 ページ）が満たされていることを確認します。

ステップ 2 DNF クラスタのセットアップ （14 ページ）

a) DNF 構成ファイルの変更 （14 ページ）

b) DNF クラスタの展開 （18 ページ）

ステップ 3 DNF クラスタの構成 （19 ページ）

a) DNF クラスタ構成ファイルの作成 （19 ページ）

b) DNF 構成ファイルの使用（flow_cluster_manage の実行） （22 ページ）

ステップ 4 DNF 収集の構成 （24 ページ）

a) flow_collector_ias および flow_collector_dmd の構成 （24 ページ）

b) DNF 用の external-executable-nimo の構成 （25 ページ）

分散 NetFlow の要件

システム要件については、『Cisco WAE System Requirements』ドキュメントを参照してください。

さらに、すべてのクラスタ要素（マスター、エージェント、JMS ブローカ）に以下が必要です。

- Ansible 2.1 以降。
- Java 仮想マシン（JVM）ですべての要素に対して同じインストールパスを使用しています。Java 実行可能ファイルは、すべてのユーザーが読み取り可能なパスにある必要があります。

- クラスタ（ブローカ、マスター、およびすべてのエージェント）専用の各サーバーに同じ名前の sudo SSH ユーザーが存在します。このユーザー名は group_vars/all Ansible ファイル（このセクションで後述）で使用されるため、書き留めておきます。

  WAE Planning ソフトウェアは、適切なライセンスファイルを使用してサーバー（インストールサーバー）にインストールされる必要があります。

- エージェントのシステム要件が、WAE のインストールに必要な要件と同じ要件を満たしています。

- フロー収集プロセスは、入力方向のルータによってキャプチャおよびエクスポートされる IPv4 および IPv6 フローをサポートしています。また、IPv4 および IPv6 iBGP ピアリングもサポートしています。ルータは、フローをフロー収集サーバーにエクスポートし、フロー収集サーバーとの BGP ピアリングを確立するように構成する必要があります。詳細については、「NetFlow 収集の構成 （8 ページ）」を参照してください。

ライセンシング

`flow_cluster_master`、`flow_collector_ias`、および `flow_collector_dmd` ツールを使用するときに、フローおよびフローデマンドを取得するための正しいライセンスがあることを Cisco WAE の担当者に確認してください。

Java メッセージサーバー（JMS）ブローカ

クラスタ内のマスター、エージェント、およびクライアントが情報を交換するには、分散フロー収集のセットアップごとに 1 つの JMS ブローカインスタンスが必要です。すべての情報はブローカを介して交換され、すべてのコンポーネントが相互に通信できます。DNF は、専用の JMS ブローカをサポートします。

すべての JMS クライアント（マスター、エージェント、および `flow_collector_ias` インスタンス）が機能するには、ブローカで次の機能が有効になっている必要があります。

- アウトオブバンド ファイル メッセージング
- 構成ファイルでの難読化されたパスワードのサポート

マスターとエージェント

Ansible ファイルは、JMS ブローカ、マスター、およびエージェントサーバーに DNF 構成をインストールして実行するために使用されます。

Master

マスターノードは、クラスタ内で次のサービスを提供します。

- エージェントのステータスを監視および追跡します。
- 最後に完了した IAS 計算のステータスを監視および追跡します。
- すべてのエージェントからクライアントに返される IAS フローデータを集約します。

- クラスタからの構成およびステータスリクエストを処理します。

エージェント（Agents）

サーバーごとに 1 つのエージェントのみがサポートされます。エージェントは、WAE インストールまたはデータ収集サーバーに配置できません。各エージェントは、対応する収集サーバーからフローデータを受信して計算します。

（注） クラスタにエージェントを1つだけ展開するオプションがあります。これは、サイズの拡大またはトラフィックの増加が予想されるネットワークで、CNF に代わるものです。

DNF クラスタのセットアップ

DNF 構成ファイルの変更

デフォルトの WAE インストールオプションを使用する場合、変更が必要な必須パラメータはわずかです。これらについては、該当する構成トピックで説明します。この項で説明するトピックは、次のことを前提としています。

- マスターサーバー（インストールサーバー）には WAE プランニングソフトウェアがインストールされていて、デフォルトのディレクトリが使用されている。特に、インストールサーバーで DNF に使用される構成ファイルが *<wae_installation_directory>*/etc/netflow/ansible にある。
- DNF 構成で専用の JMS ブローカが使用される。
- 構成例では、次の値が使用されている。
  - マスターおよび JMS ブローカの IP アドレス：198.51.100.10
  - エージェント 1 の IP アドレス：198.51.100.1
  - エージェント 2 の IP アドレス：198.51.100.2
  - エージェント 3 の IP アドレス：198.51.100.3

group_vars/all

ファイルは *<WAE_installation_directory>*/etc/netflow/ansible/group_vars/all にあります。このファイルは、プレイブックファイルで使用される変数定義を含む Ansible ファイルです。

次のオプションを編集します。

オプション	説明
LOCAL_WAE_INSTALLATION_DIR_NAME	WAE インストールファイルを含むローカルパス。
WAE_INSTALLATION_FILE_NAME	WAE インストールファイルのファイル名。

オプション	説明
TARGET_JDK_OR_JRE_HOME	Oracle JRE ファイルのフルパスとファイル名。クラスタ内のすべてのマシン（ブローカ、マスター、およびすべてのエージェント）には、この変数の下に JRE があらかじめインストールされている必要があります。
LOCAL_LICENSE_FILE_PATH	ライセンスファイルのフルパス。
SSH_USER_NAME	各マシンで SSH が有効になっているときに作成または使用された SSH ユーザー名。 この sudo ユーザーは、SSH 経由でクラスタを展開するために Ansible によって使用されます。

例（コメントは削除）：

```
LOCAL_WAE_INSTALLATION_DIR_NAME: "/wae/wae-installation"
WAE_INSTALLATION_FILE_NAME: "wae-linux-v16.4.8-1396-g6114ffa.rpm"
TARGET_JDK_OR_JRE_HOME: "/usr/lib/jvm/java-1.8.0-openjdk-1.8.0_45"
LOCAL_LICENSE_FILE_PATH: "/home/user1/.cariden/etc/MATE_Floating.lic"
TARGET_SSH_USER: ssh_user
```

hosts

ファイルは *<WAE_installation_directory>*/etc/netflow/ansible/hosts にあります。このファイルは Ansible インベントリファイルであり、クラスタ内のすべてのサーバーのリストが含まれています。

ブローカ、マスター、およびすべてのエージェントに対応する IP アドレスのみを編集します。他の変数は編集しないでください。必要に応じて、エージェントをさらに追加します。

次に例を示します。

```
[dnf-broker]
198.51.100.10 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-master]
198.51.100.10 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-1]
198.51.100.1 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-2]
198.51.100.2 ansible_ssh_user={{SSH_USER_NAME}}
[dnf-agent-3]
198.51.100.3 ansible_ssh_user={{SSH_USER_NAME}}
```

prepare-agents.yml

このファイルは、編集する必要はなく、指定されたすべてのエージェントに次の情報を提供します。

- 非ルートユーザーに特権ポート（0 ～ 1023）への制限付きアクセスを許可します。これは、フローコレクタが 1024 未満のポートを使用して BGP メッセージをリッスンするように構成する場合に必要です。

- 生成される可能性のある大量の一時ファイルを考慮して、最大15,000のファイル記述子を予約するようにOSインスタンスを構成します。
- すべてのエージェントを再起動します。

ファイルは *<WAE_installation_directory>*/etc/netflow/ansible/prepare-agents.yml にあります。

startup.yml

ファイルは *<WAE_installation_directory>*/etc/netflow/ansible/startup.yml にあります。

このファイルは、ブローカ、マスター、およびエージェントを自動的に開始するために使用されます。エージェントが 3 つ以上ある場合は、このファイルを編集してさらに追加します。

次に例を示します。

```
- hosts: all
roles:
- check-ansible-version
- hosts: dnf-broker
roles:
- start-broker
- hosts: dnf-master
roles:
- start-master
- hosts: dnf-agent-1
roles:
- {role: start-agent, instance: instance-1}
- hosts: dnf-agent-2
roles:
- {role: start-agent, instance: instance-2}
- hosts: dnf-agent-3
roles:
- {role: start-agent, instance: instance-3}
```

service_conf

ファイルは *<WAE_installation_directory>*/etc/netflow/ansible/bash/service.conf にあります。

このファイルは、ブローカ、マスター、およびエージェントによって使用される共通の構成オプションを提供します。

次のオプションを編集します。

オプション	説明
jms-broker-server-name-or-ip-address	ブローカの IP アドレス。
jms-broker-jms-port	ブローカに使用されている JMS ポート番号。
jms-broker-http-port	ブローカに使用されている HTTP ポート番号。
jms-broker-username	これは内部で使用され、変更する必要はありません。

オプション	説明

jms-broker-password	難読化されたパスワードを生成して使用することをお勧めします。次に例を示します。
	`# ./flow_cluster_manage -action print-obfuscation`
	`type in the clear text > password-0`
	`obfuscated text: ENC(h4rWRpG54WgVZRTE90Zb/JszY4dd4CGc)`

obfuscated text	上記の例から： **ENC(h4rWRpG54WgVZRTE90Zb/JszY4dd4CGc)**
jms-broker-use-tls	DFC クラスタ内のすべてのデータ通信を暗号化するには、true を入力します。true に設定すると、パフォーマンスが低下します。
append-to-log-file	ローカルログファイルに情報を追加する場合は、true を入力します。
use-flume	Flume サーバーを使用する場合は、true を入力します。
flume-server	Flume エージェントを実行しているサーバーの IP アドレスを入力します。WAE サーバーのインストール時に自動的にインストールされる Flume サーバーを使用する場合は、インストールサーバーの IP アドレスを入力します。
log-level	ロギングレベルタイプを入力します。 • off • アクティビティ • fatal • error • warn • notice • 情報 • debug • トレース

次に例を示します。

```
# jms
jms-broker-server-name-or-ip-address=198.51.100.10
jms-broker-jms-port=61616
jms-broker-http-port=8161
jms-broker-username=user-0
jms-broker-password=ENC(ctrG7GGRJm983M0AsPGnabwh)
jms-broker-use-tls=false

# local logging
append-to-log-file=false
```

```
# distributed logging
use-flume=true
flume-server=198.51.100.10

# default for all commands, will be superseded if specified locally in each .sh
log-level=info
```

DNF クラスタの展開

DNF クラスタを展開するには、次の手順を実行します。

ステップ 1 ブローカ、マスター、およびエージェントをインストールします。

```
# ansible-playbook -i hosts install.yml
```

（注） uninstall.yml プレイブックファイルは、ファイルをアンインストールし、all ファイルで定義されている TARGET_WAE_ROOT ディレクトリを削除します。

ステップ 2 DNF のエージェントを準備して再起動します。

```
# ansible-playbook -i hosts prepare-agents
```

ステップ 3 マスター、ブローカ、およびエージェントを開始します。

```
# ansible-playbook -i hosts startup.yml
```

（注） shutdown.yml プレイブックファイルは、マスター、ブローカ、およびエージェントをシャットダウンします。

ステップ 4 マスター、ブローカ、およびエージェントが実行されていることを確認します。

```
# ansible-playbook -i hosts list.yml
```

ステップ 5 マシンの再起動後、次のコマンドを実行して、すべてのエージェントが起動しているかどうかを確認できます。

```
# flow_cluster_manage -active request-cluster-status
```

成功すると、マスターとすべてのエージェントの実行中の詳細がリストされます。結果の最後に、CLUSTER SUMMARY が次のように表示されます。

```
CLUSTER SUMMARY - BEGIN
cluster all OK: false
configured size: 0
agents up: 2
daemons up: 0
agents w/wrong IDs: []
agents w/low ulimit IDs: []
computation mode: ias-in-the-background
last result time: n/a
last no-result time: n/a
max diff time: 2 ms
max diff time OK: true
CLUSTER SUMMARY - END
```

（注） 前の例では、[agents up] に 2 つの実行中のエージェントがあることが示されています。クラスタがまだ構成されていないため、[cluster all OK] フィールドは [false] です。このステータスは、クラスタの構成後に変更されます。

DNF クラスタの構成

DNF クラスタ構成ファイルの作成

`flow_manage_cluster` のクラスタ構成ファイルをより簡単に作成するために、`flow_manage` から生成された CNF 構成ファイルをクラスタ構成ファイルのテンプレートとして使用できます。

次に例を示します。

ステップ 1 テンプレート構成ファイルを生成します。

```
${CARIDEN_HOME}/flow_manage \
-action produce-config-file \
-node-flow-configs-table <input-path> \
-cluster-config-file <output-path> \
-interval 120 \
-bgp true \
-bgp-port 10179 \
-port 12100 \
-flow-size lab \
-server-ip ::
```

ここで、*<input-path>* は、CNF で使用されるノード configuration.txt ファイルのパスです（このファイルの作成の詳細については、「コレクタサーバーの構成と実行」を参照してください）。*<output-path>* は、得られるシードクラスタ構成ファイルを配置するパスです。シードクラスタ構成ファイルの出力が次のようになっていることを確認します。

```
{
    "agentConfigMapInfo": {
        "cluster_1::instance_1":
        {
            "flowManageConfiguration":
            {
                "maxBgpdPeers": 150,
                "bgpTcpPort": 179,
                "flowType": "Netflow",
                "useBgpPeering": true,
                "outfileProductionIntervalInSecs": 900,
                "networkDeploymentSize": "medium",
                "netflowUdpPort": 2100,
                "keepDaemonFilesOnStartStop": true,
                "purgeOutputFilesToKeep": 3,
                "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                "daemonOutputFileMaskPrefix": "out_matrix_",
                "daemonOutputSoftLinkName": "flow_matrix_file-latest",
```

```
                "extraAggregation": [],
                "routerConfigList":
                    [
                        {
                            "name": "ar1.dus.lab.cariden.com",
                            "bGPSourceIP": "1.2.3.4",
                            "flowSourceIP": "1.2.3.5",
                            "bGPPassword": "bgp-secret",
                            "samplingRate": "666"
                        },
                        {
                            "name": "cr1.ams.lab.cariden.com",
                            "bGPSourceIP": "1.2.3.51",
                            "flowSourceIP": "1.2.3.53",
                            "bGPPassword": "bgp-secret-3",
                            "samplingRate": "8000"
                        }
                    ],
                "appendedProperties":
                {
                    "key1": "value1",
                    "key2": "value2"
                }
            }
        },
}
```

ステップ 2 ファイルを編集して、各エージェント構成を組み込みます。クラスタ内の各エージェントに適用されるように、各セクションをコピー、貼り付け、および編集します。この例は、2 つのエージェントを示しています。

```
{
    "agentConfigMapInfo": {
        "cluster_1::instance_1":
        {
            "flowManageConfiguration":
            {
                "maxBgpdPeers": 150,
                "bgpTcpPort": 179,
                "flowType": "Netflow",
                "useBgpPeering": true,
                "outfileProductionIntervalInSecs": 900,
                "networkDeploymentSize": "medium",
                "netflowUdpPort": 2100,
                "keepDaemonFilesOnStartStop": true,
                "purgeOutputFilesToKeep": 3,
                "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                "daemonOutputFileMaskPrefix": "out_matrix_",
                "daemonOutputSoftLinkName": "flow_matrix_file-latest",
                "extraAggregation": [],
                "routerConfigList":
                    [
                        {
                            "name": "ar1.dus.lab.anyname.com",
                            "bGPSourceIP": "1.2.3.4",
                            "flowSourceIP": "1.2.3.5",
                            "bGPPassword": "bgp-secret",
                            "samplingRate": "666"
                        },
                        {
                            "name": "cr1.ams.lab.anyname.com",
```

```
                            "bGPSourceIP": "1.2.3.51",
                            "flowSourceIP": "1.2.3.53",
                            "bGPPassword": "bgp-secret-3",
                            "samplingRate": "8000"
                        }
                    ],
                "appendedProperties":
                {
                    "key1": "value1",
                    "key2": "value2"
            }
        }
},
```

2 番目のエージェントの情報はここから始まります。

```
        "cluster_1::instance_2":
        {
            "flowManageConfiguration":
            {
                "maxBgpdPeers": 150,
                "bgpTcpPort": 179,
                "flowType": "Netflow",
                "useBgpPeering": true,
                "outfileProductionIntervalInSecs": 900,
                "networkDeploymentSize": "medium",
                "netflowUdpPort": 2100,
                "keepDaemonFilesOnStartStop": true,
                "purgeOutputFilesToKeep": 3,
                "daemonOutputFileMaskSuffix": "%Y.%m.%d.%H.%M.%s",
                "daemonOutputDirPath":
"<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
                "daemonOutputFileMaskPrefix": "out_matrix_",
                "daemonOutputSoftLinkName": "flow_matrix_file-latest",
                "extraAggregation": [],
                "routerConfigList":
                    [
                        {
                            "name": "ar1.dus.lab.anyname.com",
                            "bGPSourceIP": "5.6.7.8",
                            "flowSourceIP": "5.6.7.9",
                            "bGPPassword": "bgp-secret-2",
                            "samplingRate": "666"
                        },
                        {
                            "name": "cr1.ams.lab.anyname.com",
                            "bGPSourceIP": "5.6.7.81",
                            "flowSourceIP": "5.6.7.83",
                            "bGPPassword": "bgp-secret-4",
                            "samplingRate": "8000"
                        }
                    ],
                "appendedProperties":
                {
                    "key1": "value1",
                    "key2": "value2"
            }
        }
},
```

DNF 構成ファイルの使用（flow_cluster_manage の実行）

flow_cluster_manage ツールは、分散 NetFlow 収集クラスタを診断および制御します。構成ファイルを作成したら、flow_cluster_manage を使用してクラスタ構成ファイルをクラスタに送信します（**flow_cluster_manage -send-cluster-configuration**）。すべてのエージェントのすべてのフロー収集プロセスが、その構成ファイルに格納されている構成情報をリロードします。

（注） システムの起動時またはシャットダウン時に、flow_cluster_master、flow_cluster_agent、および flow_cluster_broker を自動的に開始および停止するようにシステムを構成することをお勧めします。

また、flow_cluster_manage ツールを使用してクラスタステータスを取得することもできます。次に例を示します。

```
# flow_cluster_manage -action request-cluster-status
```

（注） クラスタの構成には約 1 分かかります。

クラスタステータスの結果例：

```
CLUSTER STATUS - BEGIN

    AGENT NODE - BEGIN
        cluster ID:            cluster_1
        instance ID:           instance_1
        process ID:            15292
        start time:            2017-07-10.09:19:43.000-0700
        up time:               00d 00h 00m 40s 824ms
        unique ID:
   bc.30.5b.df.8e.b5-15292-1729199940-1499703582925-1a23cb00-ed76-4861-94f5-461dcd5b2070

        last HB received:      2017-07-10.09:20:24.004-0700
        last HB age:           00d 00h 00m 04s 779ms
        skew time:             00d 00h 00m 00s 010ms computation sequence   0
        computational model    ias-in-the-background computing IAS:         false
        ip addresses:          [128.107.147.112, 172.17.0.1,
2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
    2001:420:30d:1320:cd72:ec61:aac8:2e72,2001:420:30d:1320:dc55:a772:de80:a73f]
        mac address:           bc.30.5b.df.8e.b5 jvm memory utilization:
4116Mb/4116Mb/3643Mb max opened files:      15000
        processors:            8
        daemon period:         00d 00h 15m 00s 000ms
        daemon out dir:

/media/1TB/user1/sandboxes/git/netflow-flexible/package/linux-release/lib/ext/pmacct/insta

    nces/flow_cluster_agent_cluster_1::instance_1
        daemon process ID: 15344
        daemon is: running
        bgp port: 179
        bgp port status: up
```

```
        netflow port: 2100
        netflow port status: up
    AGENT NODE - END

    AGENT NODE - BEGIN
        cluster ID: cluster_1
        instance ID: instance_2
        process ID: 15352
        start time: 2017-07-10.09:19:49.000-0700
        up time: 00d 00h 00m 30s 748ms
        unique ID:
   bc.30.5b.df.8e.b5-15352-1729199940-1499703589727-12989336-b314-4f85-9978-242882dd16da

        last HB received: 2017-07-10.09:20:20.746-0700
        last HB age: 00d 00h 00m 08s 037ms
        skew time: 00d 00h 00m 00s 014ms
        computation sequence 0
        computational model ias-in-the-background
        computing IAS: false
        ip addresses: [128.107.147.112, 172.17.0.1,
    2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
    2001:420:30d:1320:cd72:ec61:aac8:2e72, 2001:420:30d:1320:dc55:a772:de80:a73f]
        mac address: bc.30.5b.df.8e.b5
        jvm memory utilization: 4116Mb/4116Mb/3643Mb
        max opened files: 15000
        processors: 8
        daemon period: 00d 00h 15m 00s 000ms
        daemon out dir:

  /media/1TB/user1/sandboxes/git/netflow-flexible/package/linux-release/lib/ext/pmacct/insta

    nces/flow_cluster_agent_cluster_1::instance_2
        daemon process ID: 15414
        daemon is: running
        bgp port: 10179
        bgp port status: up
        netflow port: 12100
        netflow port status: up
    AGENT NODE - END

    MASTER NODE - BEGIN
        cluster ID: cluster_1
        instance ID: instance_id_master_unique
        process ID: 15243
        start time: 2017-07-10.09:19:34.000-0700
        up time: 00d 00h 00m 50s 782ms
        unique ID:
   bc.30.5b.df.8e.b5-15243-415138788-1499703574719-cd420a81-f74c-49d4-a216-ffeb7cde31d5

        last HB received: 2017-07-10.09:20:25.563-0700
        last HB age: 00d 00h 00m 03s 220ms
        ip addresses: [128.107.147.112, 172.17.0.1,
    2001:420:30d:1320:24a8:5435:2ed5:29ae, 2001:420:30d:1320:be30:5bff:fedf:8eb5,
    2001:420:30d:1320:cd72:ec61:aac8:2e72, 2001:420:30d:1320:dc55:a772:de80:a73f]
        mac address: bc.30.5b.df.8e.b5
        jvm memory utilization: 2058Mb/2058Mb/1735Mb
        processors: 8
    MASTER NODE - END

    CLUSTER SUMMARY - BEGIN
        cluster all OK: true
        configured size: 2
        agents up: 2
        daemons up: 2
```

```
        agents w/wrong IDs: []
        agents w/low ulimit IDs: []
        computation mode: ias-in-the-background
        last result time: n/a
        last no-result time: n/a
        max diff time: 4 ms
        max diff time OK: true
    CLUSTER SUMMARY - END

CLUSTER STATUS - END
```

結果の最後にある CLUSTER SUMMARY エントリには、クラスタ構成が動作しているかどうかの簡単な要約が示されます。`cluster all OK` が true であること、および `configured size`、`agents up`、および `daemons up` が構成したエージェントの数と一致することを確認する必要があります。`agents w/wrong IDs` および `agents w/low ulimit IDs` には値があってはなりません。`max diff` time OK も true に設定されている必要があります。そうでない場合は、エージェントとマスターの詳細を調べてトラブルシューティング情報を入手してください。

`flow_manage_cluster` オプションの詳細については、*`wae-installation-directory`*`/bin` に移動し、**`flow_manage_cluster -help`** と入力してください。

DNF 収集の構成

flow_collector_ias および flow_collector_dmd の構成

これらの CLI ツールは、
`<WAE_installation_directory>``/etc/netflow/ansible/bash/nimo_flow_collector_ias_dmd.sh` スクリプト内で構成され、external-executable-nimo 内で実行されます。`flow_collector_ias` および `flow_collector_dmd` ツールは、クラスタから受信した NetFlow データを使用してデマンドおよびデマンドトラフィックを生成します。次のように編集します。

編集する前に、このファイルの権限を変更します。

```
chmod +x nimo_flow_collector_ias_dmd.sh
```

- **CUSTOMER_ASN**：ASN を入力します。
- **SPLIT_AS_FLOWS_ON_INGRESS**：複数の外部 ASN が IXP スイッチに接続されている場合、すべての ASN からのトラフィックを集約するのか、MAC アカウンティング入力トラフィックに比例して分散するのかを決定します。デフォルト値は aggregate です。もう 1 つの値は mac-distribute です。
- **ADDRESS_FAMILY**：含めるプロトコルバージョンのリストを入力します（カンマ区切りのエントリ）。デフォルトは ipv4,ipv6 です。
- **WAIT_ON_CLUSTER_TIMEOUT_SEC**：IAS フローの計算を分散クラスタに委任するときにタイムアウトするまで待機する秒数を入力します。デフォルトは 60 秒です。

`nimo_flow_collector_ias_dmd.sh` の例：

```
#!/bin/bash

# this script should be called from NSO's 'external executable NIMO' configuration window
# in this way:
# /path-to/nimo_flow_collector_ias_and_dmd.sh $$input $$output

# modify as needed - BEGIN
CUSTOMER_ASN=142313
SPLIT_AS_FLOWS_ON_INGRESS=aggregate
ADDRESS_FAMILY=ipv4,ipv6
WAIT_ON_CLUSTER_TIMEOUT_SEC=60
# modify as needed - END
```

`flow_collector_ias` または `flow_collector_dmd` オプションの詳細については、*wae-installation-directory*`/bin` に移動し、**flow_collector_ias -help** または **flow_collector_dmd -help** と入力してください。

DNF 用の external-executable-nimo の構成

external-executable-nimo は、選択したネットワークモデルに対して `nimo_flow_collector_ias_dmd.sh` スクリプトを実行します。この場合、WAE で作成された既存のモデルを取得し、`nimo_flow_collector_ias_dmd.sh` からの情報を追加して、必要なフローデータを含む最終ネットワークモデルを作成します。

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他のNIMO収集を含む最終ネットワークモデルです。
- DNF NetFlow 構成ワークフロー （12 ページ）の準備作業が完了したことを確認します。

ステップ 1 エキスパート モードから、**/wae:networks** に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。簡単に識別できる一意の名前をお勧めします。たとえば、networkABC_DNF_flow_ias_dmd などです。

ステップ 3 [nimo] タブをクリックします。

ステップ 4 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[external-executable-nimo] を選択します。

ステップ 5 [external-executable-nimo] をクリックし、送信元ネットワークを選択します。

ステップ 6 [advanced] タブで、以下の情報を入力します。

- [input-file-version]：**7.1** と入力します。
- [input-file-format]：送信元ネットワークモデルのプランファイルフォーマットとして [.pln] を選択します。
- [argv]：***<directory_path>*/nimo_flow_collector_ias_dmd.sh $$input $$output** を入力します。

ステップ 7 構成を確認するには、[external-executable-nimo] タブから [run] をクリックします。

例

WAE CLI を（構成モードで）使用している場合は、次のように入力します。

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network> advanced argv nimo_flow_collector_ias_dmd.sh $$input $$output ]
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
admin@wae(config-network-<network-model-name>)# exit
admin@wae(config)# exit

admin@wae# networks network <network-model-name> nimo external-executable-nimo run
```

次のタスク

external-executable-nimo を構成したら、WAE Design からデータの実行またはアクセスをスケジュールできます。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。