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

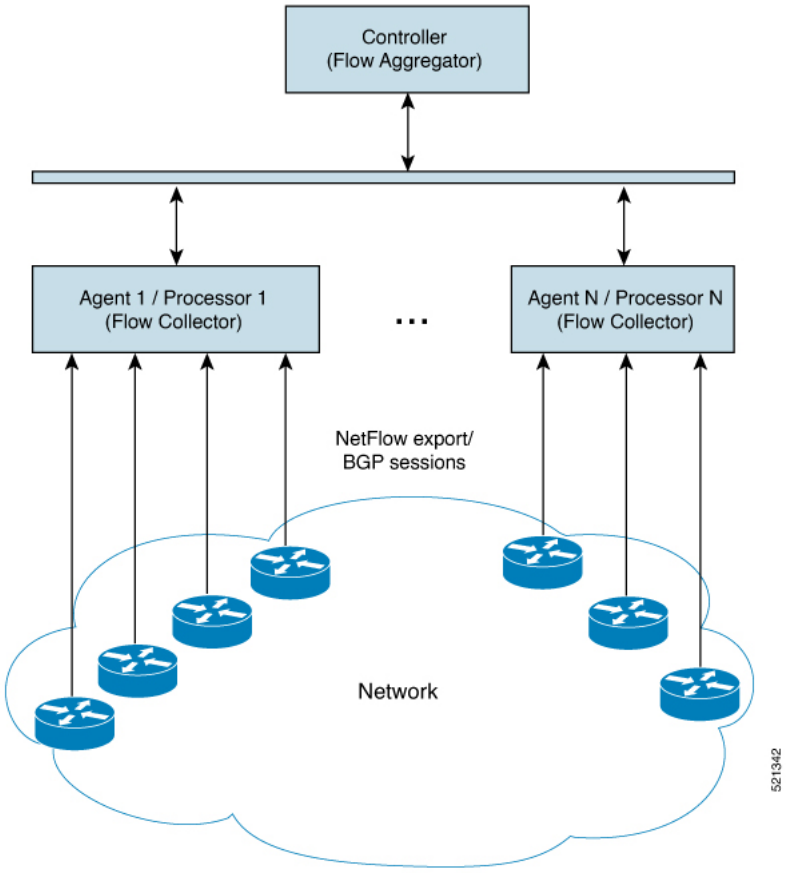
分散型 NetFlow（DNF）収集アーキテクチャは通常、大規模なネットワークに使用されます。このアーキテクチャは、JMS ブローカ、コントローラノード、および 1 つ以上のエージェントで構成されています。ブローカとコントローラノードは、WAE 収集サーバーがインストールされているマシンと同じマシンで実行します。各エージェントは異なるマシンで実行します。DNF 収集では、Ansible を使用してブローカ、コントローラノード、エージェントなどのコンポーネントを個別にインストールする必要があります。

集中型 NetFlow（CNF）収集は通常、小規模のネットワークに使用されます。CNF 収集は DNF 収集アーキテクチャを使用して実装され、JMS ブローカ、コントローラノード、および WAE サーバーがインストールされているマシンと同じマシン上で実行される単一のエージェントで構成されます。コンポーネント（ブローカ、コントローラノード、およびエージェント）は、WAE インストールサーバーに事前にインストールされています。CNF のコンポーネントを個別にインストールする必要はありません。

分散 Netflow（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。コントローラノードはこの情報を集約し、`flow_collector_ias` に返します。

図 *1 : DNF* アーキテクチャ



Java メッセージサーバー（JMS）ブローカ

クラスタ内のコントローラノード、エージェント、およびクライアントが情報を交換できるように、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存在します。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（コントローラノード、エージェント、および`flow_collector_ias`インスタンス）が機能できるように、ブローカで次の機能が有効になっています。

- 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

コントローラノードとエージェント

- コントローラ ノード

  コントローラノードプロセス（`flow_cluster_controller`）は、クラスタで次のサービスを提供します。

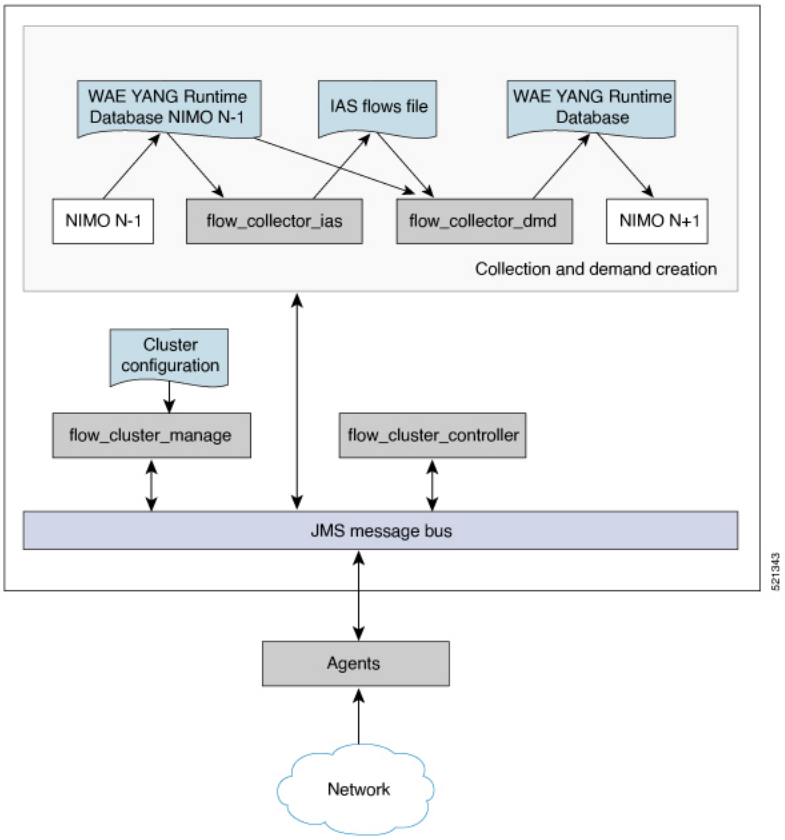
  - すべてのエージェントのステータスを監視および追跡します。

- 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（`flow_collector_ias`）。
- 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（`flow_cluster_manage`）。

- [エージェント（Agents）]

  サーバーごとに 1 つのエージェントプロセス（`flow_cluster_agent`）のみがサポートされます。各エージェントプロセス（`flow_cluster_agent`）は、対応する収集サーバー（`pmacct`）からフローデータを受信して計算します。

図 *2 : 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。

  `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://<*controller-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_controller**：コントローラノードサービスは、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。
- **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、コントローラノード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。
- **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、コントローラノード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。
- **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（CNF）収集

集中型 NetFlow（CNF）収集は通常、小規模のネットワークに使用されます。CNF 収集は DNF 収集アーキテクチャを使用して実装され、JMS ブローカ、コントローラノード、および WAE サーバーがインストールされているマシンと同じマシン上で実行される単一のエージェントで構成されます。

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。
- フローコレクタサーバーの iBGP ルートリフレクタ クライアントとして設定されたルータで BGP セッションを定義します。ルータ自体でこれを設定できない場合は、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備えた BGP ルートリフレクタ サーバーを代わりに使用できます。
- 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。
- 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。

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 node-flow-configs-table ファイルの作成 （7 ページ）
ステップ 4 CNF 構成ファイルの作成 （8 ページ）
ステップ 5 CNF 収集の構成 （11 ページ）

a) CNF 用の netflow-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_cluster_controller`、`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。

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、OS ターミナルから次の `flow_manage` コマンドを実行します。

```
sudo -E ./flow_cluster_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。
- `flow_collector` ツール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。

node-flow-configs-table 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
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 構成ファイルの作成

CNF または DNF のクラスタ構成ファイルを作成するには、`flow_manage` を `-action generate-cluster-config-file` オプションとともに使用し、必要に応じて JSON ファイルを編集します。

次に例を示します。

ステップ 1 次のサンプルファイルを使用して、.json ファイルを作成します。

`waerc` ファイルをソースに設定します。

```
${CARIDEN_HOME}/flow_manage \
-action produce-cluster-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、node-flow-configs-table ファイルの作成 （7 ページ）で作成された `node-flow-configs-table` のパス、*<output-path>*は、生成されるシードクラスタ構成ファイルが配置されるパスです。

<input-path> ファイルの例は次のようになります。

```
<NodeFlowConfigs>
Name    SamplingRate    FlowSourceIP    BGPSourceIP     BGPPassword
node1   1024    192.168.75.10   69.127.75.10    ag5Xh0tGbd7
node2   1024    192.168.75.11   69.127.75.11    ag5Xh0tGbd7
```

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_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
```

```
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    },
    "cluster_1::instance_y": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

ステップ 2 ファイルを編集して、単一のエージェント構成のみを組み込みます。以下は、単一のエージェント構成のみを組み込んだ構成の例です。

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
```

```
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```


CNF 収集の構成

CNF 用の netflow-nimo の構成

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。
- シングルモードで動作するように WAE NetFlow エージェントを設定します。エキスパートモードを使用した NetFlow エージェントの構成を参照してください

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、[netflow-nimo] を選択します。

ステップ 5 [netflow-nimo] をクリックし、[source-network] を選択します。

ステップ 6 [設定（config）] タブをクリックします。

ステップ 7 [共通（common）] をクリックして、次の情報を入力します。

- [split-as-flows-on-ingress]：外部 ASN のトラフィック集約方法を選択します。
- [asn]：ネットワーク内の内部 AS の ASN を入力します。
- [address-family]：IAS のフローおよび需要の計算に含めるプロトコルバージョンを選択します。
- [number-of-threads]：並列計算で使用するスレッドの最大数を入力します。
- [ext-node-tags]：1 つ以上のノードタグのカンマ区切りリストを入力します。
- [extra-aggregation]：集約キーのリストをカンマで区切って入力します。
- [log-level]：ツールのログレベルを選択します。

ステップ 8 [ias-flows] をクリックして、次の情報を入力します。

- [ias-computation-timeout-in-minutes]：IAS フロー計算のタイムアウトを分単位で入力します。
- [trim-inter-as-flows]：トラフィックの inter-as-flow が破棄されない下限値をメガビット/秒単位で入力します。この値を下回ると厳密に破棄されます。
- [match-on-bgp-external-info]：BGP ピア関係の出力 IP アドレスを照合するかどうかを選択します。
- [flow-dir]：インポートするフローマトリックスファイルが含まれているディレクトリを入力します。ファイルはインポート後すぐに削除されます。
- [flow-file]：インポートするフローマトリックスファイルが含まれているファイルパスを入力します。ファイルはインポート後すぐに削除されます。
- [ingress-interface-flow-filter]：ノードとインターフェイスのフィルタを Node:InterfaceName の形式で入力します。これは、フローマトリックスを読み取るときに適用され、対象の入力インターフェイスのみをフィルタリングします。
- [egress-interface-flow-filter]：ノードとインターフェイスのフィルタを Node:InterfaceName の形式で入力します。これは、フローマトリックスを読み取るときに適用され、対象の出力インターフェイスのみをフィルタリングします。
- [backtrack-micro-flows]：入力ファイルからのマイクロフローと、それらの需要またはそれらを集約する inter-as-flow との関係を示すファイルを生成するかどうかを選択します。
- [flow-import-flow-ids]：データのインポート元のフロー ID をカンマで区切って入力します。すべてのフローからインポートするには、" を使用します。

ステップ 9 [需要（demands）] をクリックして、次の情報を入力します。

- [demand-name]：新しい需要の名前を入力します。
- [demand-tag]：新しい需要のタグを入力するか、既存のタグに追加するタグを入力します。
- [trim-demands]：需要が破棄されない下限値をメガビット/秒単位で指定します。この値を下回ると需要が厳密に破棄されます。
- [service-class]：需要のサービスクラスを指定しますの
- [traffic-level]：需要のトラフィックレベルを指定します。
- [missing-flows]：フローを受信していないインターフェイスを含むファイルが生成されるパスを入力します。

ステップ 10 **[run-netflow-collection]** > **[run-netflow-collectionの呼び出し（Invoke run-netflow-collection）]** をクリックします。

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。

ステップ 1 分散 NetFlow の要件 （13 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 クラスタの構成 （14 ページ）

a) Ansible を使用した DNF クラスタの展開 （14 ページ）

ステップ 3 DNF 収集の構成 （16 ページ）

a) flow_collector_ias および flow_collector_dmd の構成 （16 ページ）

b) DNF 用の external-executable-nimo の構成 （17 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（コントローラノード、エージェント、JMS ブローカ）に以下が必要です。

- 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。
- 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。
- ルータは、フローをエクスポートし、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。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。
- Python3 ベースの Ansible 2.9.18 以降。
- Java 仮想マシン（JVM）ですべての要素（コントローラノード、エージェント、JMS ブローカ）に対して同じインストールパスを使用している。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、コントローラノード、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 対応、SSH 対応ユーザーが存在する。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。

ライセンシング

flow_cluster_controller、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。

DNF クラスタの構成

Ansible を使用した DNF クラスタの展開

（注）
- インストールサーバー（WAE がインストールされているインスタンス）からのみ、次の手順を実行します。Ansible は、インストール、エージェントのセットアップ、起動などを処理します。
- WAE実行可能バイナリファイルがインストールサーバーに存在している必要があります。

ステップ 1 python3 ベースの Ansible バージョン 2.9.18 以降をインストールします。次のコマンドを使用します。

```
sudo yum install ansible
```

ステップ 2 WAE ディレクトリ内で、etc/netflow/ansible/hosts ファイルを変更して、エージェント、ブローカ、コントローラノードの IP アドレスを含めます。これを行うには、必要に応じて <element-x> を IP アドレスまたはサーバー名に置き換えます。

次の例を参考にしてください。

3 つのエージェントと 1 つのコントローラノード（ブローカは通常コントローラノードに存在します）の場合、デフォルトの DNF agent-1 の他に 2 つの行を追加する必要があります。

（注） シスコでは、ansible_ssh_pass の使用を推奨していません。代わりに、ansible-playbook コマンドの実行中に --ask-pass を使用してください。

設定例：

```
[dnf-broker]
10.10.10.1 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-controller]
10.10.10.1 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-1]
10.10.10.2 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-2]
10.10.10.3 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-3]
10.10.10.4 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}
```

ステップ 3 package/linux/wae/etc/netflow/ansible/startup.yml ファイルを変更して、必要に応じてエージェントを含める、コメント解除する、または追加し、package/linux/wae/etc/netflow/ansible/hosts のエージェントまたはコントローラノードアドレスの数と一致させます。

ステップ 4 package/linux/wae/etc/netflow/ansible/bash/service.conf ファイルを変更します。<jms-broker-server-name> をコントローラノードの IP アドレスに変更します（上記の設定例では、IP アドレスは 10.10.10.1）。

ステップ 5 package/linux/wae/etc/netflow/ansible/group_vars/all ファイルを変更します。参照および使用されているとおりに、すべての関連変数を変更します。group_vars/all （18 ページ）を参照してください

（注） 必要に応じて次の行を追加できますが、シスコではこのように追加することは推奨していません。

```
SSH_USER_PASS: "ciscowae"
```

ステップ 6 ANSIBLE_HOME=${WAE_ROOT}/etc/netflow/ansible をエクスポートします。

使用目的

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/install.yml
```

または

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/install.yml --ask-pass --ask-become-pass
```

SSH_USER_PASS が設定されているかどうかによって異なります。

ステップ 7 NetFlow 用に OS を準備します。次のコマンドを使用します。

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/prepare-agents.yml --ask-pass
--ask-become-pass
```

ステップ 8 クラスタを起動します。次のコマンドを使用します。

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/startup.yml --ask-pass
```

ステップ 9 次のコマンドを使用して、クラスタ要素を一覧表示します（オプション）。

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/list.yml --ask-pass
```

ステップ 10 cluster-config をクラスタに送信します。次のコマンドを使用します。

```
${WAE_ROOT}/bin/flow_cluster_manage -action send-cluster-configuration \
-options-file ${ANSIBLE_HOME}/bash/service.conf \
-cluster-config-file-path <path-of-config>/flow-config-cluster.json
```

（注） JAVA_HOME を設定する必要があります。

例：

export JAVA_HOME=/usr/java_latest

<path-of-config>/flow-config-cluster.json の作成については、DNF クラスタ構成ファイルの作成 （19 ページ）を参照してください。

ステップ 11 次のコマンドを使用して、クラスタのステータスを確認します（オプション）。

```
${WAE_ROOT}/bin/flow_cluster_manage -action request-cluster-status \
-options-file ${ANSIBLE_HOME}/bash/service.conf
```

DNF クラスタのシャットダウンまたはアンインストール

クラスタをシャットダウンするには、次のコマンドを使用します。

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/shutdown..yml --ask-pass
```

アンインストールするには、次のコマンドを実行します。

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/uninstall.yml --ask-pass
```

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 （13 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。

- [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。

（注） 集約の場合、アグリゲータの設定で、external-executable-nimo nimo nimo ネットワークを netflow-nimo タイプの依存関係として追加します。

Ansible 構成ファイルの作成

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。

- コントローラノードサーバー（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。
  - コントローラノードおよび 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。
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、プライマリノード、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-v7.4.0.bin"
TARGET_JDK_OR_JRE_HOME: "/usr/lib/jvm/jre-11-openjdk-11.0.7"
LOCAL_LICENSE_FILE_PATH: "/home/user1/.cariden/etc/MATE_Floating.lic"
TARGET_SSH_USER: ssh_user
```

DNF クラスタ構成ファイルの作成

CNF または DNF のクラスタ構成ファイルを作成するには、`flow_manage` を `-action generate-cluster-config-file` オプションとともに使用し、必要に応じて JSON ファイルを編集します。

次に例を示します。

ステップ 1 次のサンプルファイルを使用して、.json ファイルを作成します。

`waerc` ファイルをソースに設定します。

```
${CARIDEN_HOME}/flow_manage \
-action produce-cluster-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、node-flow-configs-table ファイルの作成 （7 ページ）で作成された `node-flow-configs-table` のパス、*<output-path>*は、生成されるシードクラスタ構成ファイルが配置されるパスです。

<input-path> ファイルの例は次のようになります。

```
<NodeFlowConfigs>
Name    SamplingRate    FlowSourceIP    BGPSourceIP     BGPPassword
node1   1024    192.168.75.10   69.127.75.10    ag5Xh0tGbd7
node2   1024    192.168.75.11   69.127.75.11    ag5Xh0tGbd7
```

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_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
```

```
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    },
    "cluster_1::instance_y": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

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_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    },
    "cluster_1::instance_y": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
```

```
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

（注） .json ファイルの設定は、コントローラノードにのみ必要であり、プロセッサノードには必要ありません。

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。