ネットワーク インターフェイス モジュール（NIMO）

次のセクションでは、さまざまなタイプのネットワーク収集とその他のNIMO機能を構成する方法について説明します。次のトピックでは、構成にエキスパートモードを使用する方法を示していますが、WAE UIまたはWAE CLIを使用することもできます。これらのトピックでは、任意のインターフェイスを使用して構成できるオプションについて説明します。

NIMO の説明

各NIMOには、NETCONFプロトコル機能から派生した、収集または展開する対象を決定する機能があります。次の表に、各 NIMO の説明を示します。

各 NIMO の機能を一覧表示するには、NIMO の設定後に（エキスパート モードにある）[機能を表示（get-capabilities）] ボタンをクリックします。

（注） 異なるデータ収集（NIMO収集）を単一のネットワークモデルに統合する場合は、収集を実行する前にアグリゲータを設定します。詳細については、NIMO 収集の統合 （9 ページ）を参照してください。

収集または機能	NIMO	説明	前提条件/注意事項
ネットワーク収集 NIMO			
IGP データベースを使用したトポロジ収集（5 ページ）	topo-igp-nimo	ログインと SNMP を使用して IGP トポロジを検出します。	これは、基本的なトポロジ収集（トポロジ NIMO）です。結果として得られるネットワークモデルは、他の NIMO の送信元ネットワークとして使用されます。
XTC を使用したトポロジ収集 （6 ページ）	topo-bgpls-xtc-nimo	XTC 経由で BGP-LS を使用してレイヤ 3 トポロジを検出します。トポロジの送信元として生の XTC データを使用します。ノードおよびインターフェース/ポートのプロパティは、SNMP を使用して検出されます。	• 収集を実行する前に、XTC エージェントを設定する必要があります。「エキスパートモードを使用した XTC エージェントの構成」を参照してください。 • これは、XTC を使用するネットワークの基本的なトポロジ収集です。結果として得られるネットワークモデルは、他の NIMO の送信元ネットワークとして使用されます。
VPN 収集 （14 ページ）	topo-vpn-nimo	レイヤ 2 およびレイヤ 3 VPN トポロジを検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。
BGP ピア収集 （19 ページ）	topo-bgp-nimo	ログインと SNMP を使用して BGP ピアリングを検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。

収集または機能	NIMO	説明	前提条件/注意事項
構成解析を使用したポート、LSP、SRLG、および VPN 収集 （17 ページ）	cfg-parse-nimo	ネットワーク内のルータ設定から情報を検出して解析します。	• 基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。 • 収集を実行する前に、構成解析エージェントを設定する必要があります。エキスパートモードを使用した構成解析エージェントの構成 を参照してください。
マルチレイヤ（L3-L1）収集	光学ニモ	最終的なネットワーク収集は、他の NIMO と連携してレイヤ 1（光）およびレイヤ 3 トポロジを検出します。	optical-nimo を設定する前に実行する必要がある設定があります。エキスパートモード：マルチレイヤ収集を参照してください 。
LSP 構成の収集 （14 ページ）	lsp-config-nimo	NETCONF 経由で NED および LSP バインド SID を使用して LSP を検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。
SNMP を使用した LSP 収集 （21 ページ）	lsp-snmp-nimo	SNMP を使用して LSP を検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。
XTC を使用した LSP 収集 （16 ページ）	lsp-pcep-xtc-nimo	XTC を使用して PCEP LSP を検出します。	収集を実行する前に、XTC を使用したトポロジ収集 （6 ページ） を完了する必要があります。
セグメント ルーティング トラフィック マトリックス収集 （13 ページ）	sr-traffic-matrix-nimo	SR LSP トラフィック情報を検出します。	• 基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。 • テレメトリはルータで設定する必要があります。
NIMO 収集の統合 （9 ページ）	—	さまざまな NIMO 情報を単一の統合ネットワークモデルに集約します。	1 つの最終的なネットワークモデルにマージすべき情報が含まれた設定済みのネットワークモデルです。

収集または機能	NIMO	説明	前提条件/注意事項
AS プランファイルのマージ （39 ページ）	inter-as-nimo	さまざまな自律システム（AS）からの計画ファイルは、inter-as-nimo を使用してマージできます。inter-as-nimo は、計画ファイル間の競合を解決します。ネイティブ形式の計画ファイルもサポートされています。	• マージする個々の AS ネットワークモデルで収集が完了していることを確認してください。 • topo-bgpls-xtc NIMO を使用する AS ネットワークモデルには、それぞれ自律システム番号（ASN）が割り当てられている必要があります。
追加の NIMO			
トラフィック収集（30 ページ）	traffic-poll-nimo	SNMP ポーリングを使用してトラフィック統計（インターフェイス測定）を収集します。	• 基本的なトポロジ収集を備えたネットワークモデルです。 • LSP トラフィックを収集する場合、LSP を備えたネットワークモデルが存在する必要があります。SNMP を使用した LSP 収集（21 ページ） を参照してください。 • VPN トラフィックを収集する場合、VPN を備えたネットワークモデルが存在する必要があります。VPN 収集 （14 ページ） を参照してください。
ネットワークモデルのレイアウト（可視化）（35 ページ）	layout-nimo	送信元モデルにレイアウトプロパティを追加して、視覚化を改善します。	• 統合されたネットワークモデルです。 • layout-nimo が設定されたら、レイアウトのプロパティを含む計画ファイルを layout-nimo モデルにインポートして戻す必要があります。
ネットワークモデルに対する外部スクリプトの実行 （40 ページ）	external-executable-nimo	カスタマイズされたスクリプトを実行して、送信元ネットワークモデルに追加データを付加します。	送信元ネットワークモデルとカスタムスクリプトです。

基本的なトポロジ収集

基本的なトポロジ収集（トポロジNIMO）から得られるネットワークモデルは、追加のデータ収集のソースネットワークとして使用されます。トポロジ収集とその他のデータ収集を統合するには、収集を実行する前に、最初にアグリゲータを設定する必要があります。アグリゲータの詳細については、NIMO 収集の統合 （9 ページ）を参照してください。

IGP データベースを使用したトポロジ収集

IGP トポロジ（topo-igp-nimo）は、ノードプロパティの収集と、SNMP を使用したインターフェイスとポートの検出により、IGP データベースを使用してネットワークトポロジを検出します。これは、必要となる基本的なデータ収集を提供するため、通常、他の NIMO の前に設定される最初の NIMO です。この NIMO は、完全なトポロジ検出を提供します。OSPF および ISIS のマルチインスタンスの収集もサポートされています。ルータから収集されたすべてのリンクには、関連付けられた IGP プロセス ID があります。

このトポロジディスカバリから得られるネットワークモデルは、追加の収集の送信元ネットワークとして使用されます。他の NIMO が使用するコアノード、回路、およびインターフェイス情報を提供します。

（注）

- このトピックで説明されているタスクを実行するときは、ネットワークモデルの作成中であることを前提としています。詳細については、ネットワークモデルの作成を参照してください。
- このトピックでは、設定のためにエキスパートモードを使用する方法を示していますが、WAE UI または WAE CLI を使用してオプションを設定する場合にも参照できます。

始める前に

デバイスおよびネットワーク アクセス プロファイルを設定する必要があります。ネットワーク アクセスの設定 を参照してください。

ステップ 1 NIMO タイプとして [topo-igp-nimo] を選択します。

ステップ 2 ネットワーク アクセス プロファイルを選択します。

ステップ 3 [collect-interfaces] フィールドから [true] を選択して、完全なネットワークトポロジを検出します。

ステップ 4 [igp-config] タブをクリックして、シードルータを設定します。

ステップ 5 プラス（+）記号をクリックして IGP を追加します。

ステップ 6 [追加（Add）] をクリックして、インデックス番号を入力します。

a) シードルータの管理 IP アドレスを入力します。

b) ネットワークで実行されている IGP プロトコルを選択します。

c) （オプション）[詳細（advanced）] タブをクリックして、そのIGP設定の追加パラメータを設定します。説明を表示するには、フィールドの上にマウスポインタを合わせます。

ステップ 7 （オプション）IGP 設定をさらに追加するには、[igp-config] タブに戻り、IGP インデックスごとに前の手順を繰り返します。

ステップ 8 （オプション）収集から個々のノードを除外したり含めたりするには、[node-filter] タブをクリックし、ノードフィルタを選択します。[node-filter] が定義されていない場合は、Cisco WAE UI を使用したノードフィルタの設定を参照してください。

ステップ 9 エキスパートユーザーは、[詳細（advanced）] タブをクリックして、さらに多くのオプションを表示できます。オプションの説明を表示するには、フィールドの上にマウスポインタを合わせます。

ステップ 10 [コミット（Commit）] ボタンをクリックします。

ステップ 11 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

ステップ 12 収集が正常に実行されたことを確認するには、ネットワーク（**/wae:networks/network/<*network-name*>**）に戻り、[model] タブをクリックします。

ステップ 13 [nodes] をクリックします。ノードと詳細のリストが表示され、収集が成功したことが示されます。

次のタスク

このネットワークモデルを送信元ネットワークとして使用して、追加の収集を構成します。NIMO の説明 （1 ページ） を参照してください。

XTC を使用したトポロジ収集

topo-bgpls-xtc-nimo は、XTC 経由で BGP-LS を使用してレイヤ 3 トポロジを検出します。トポロジのソースとして生の XTC データを使用します。ノードおよびインターフェース/ポートのプロパティは、SNMP を使用して検出されます。テスト目的では、SNMP アクセスが利用できない場合、XTC のみを使用して（拡張トポロジディスカバリは無効の状態で）BGP-LS XTC トポロジディスカバリを使用することもできます。トポロジディスカバリの結果得られたネットワークモデルは、他の NIMO が使用するコアノード/回路/インターフェイス情報を提供するため、追加の収集用のソースネットワークとして使用されます。

XTC のみを使用した BGP-LS XTC トポロジディスカバリは、ほとんどの NIMO が必要とする必須情報を収集しないため、一部の NIMO のみのソースとして使用されます。

始める前に

- デバイスアクセスとネットワークアクセスを構成する必要があります。詳細については、エキスパートモードを使用したデバイスアクセスの構成およびネットワーク アクセスの設定を参照してください。
- XTC エージェントが構成され、実行されている必要があります。詳細については、エキスパートモードを使用した XTC 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、[設定エディタ（Configuration editor）] で、[/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。NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_bgpls_xtc などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[topo-bgpls-xtc-nimo] を選択します。

ステップ 6 次の情報を入力します。

- [network-access]：ネットワークアクセスを選択します。
- [xtc-host]：XTC エージェントを選択します。
- [backup-xtc-host]：バックアップ XTC エージェントを選択します。バックアップがない場合は、同じ XTC エージェントに入ることができます。
- [asn]：ネットワーク内のすべての自律システムから情報を収集する場合は 0 を入力し、特定の ASN からのみ情報を収集する場合は自律システム番号（ASN）を入力します。たとえば、XTC エージェントが ASN 64010 および ASN 64020 を認識できる場合、64020 と入力すると ASN 64020 からのみ情報を収集します。as-merger NIMO を使用して異なる AS モデルを 1 つのネットワークモデルに統合する場合は、ASN を入力する必要があります。
- [igp-protocol]：ネットワークで実行されている IGP プロトコルを選択します。
- [extended-topology-discovery]：完全なネットワークトポロジ（ノードおよびインターフェイス）を検出するには、[true] を選択します。

（注） 詳細オプションについては、BGP-LS XTC の詳細オプション （8 ページ）を参照してください。WAE UI から、マウスを各フィールドの上に置いてツールチップを表示することもできます。

ステップ 7 [reactive-network] タブをクリックして XTC からの通知をサブスクライブし、ノードまたはリンクの追加または削除を更新します。次の情報を入力します。

- [有効化（enable）]：[true] を選択して通知を有効にし、ノードまたはリンクの追加または削除を更新します。
- [enable-triggering-collection]：[true] を選択して、新しく追加されたトポロジでトポロジ収集を収集します。
- [trigger-debounce-time]：トポロジ収集をトリガーする前に、最後のトリガー通知を待機する時間を設定します。

ステップ 8 [コミット（Commit）] ボタンをクリックします。

ステップ 9 **[run-xtc-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

ステップ 10 収集が正常に実行されたことを確認するには、ネットワーク（**/wae:networks/network/<*network-name*>**）に戻り、[model] タブをクリックします。

ステップ 11 [nodes] をクリックします。ノードと詳細のリストが表示され、収集が成功したことが示されます。

例

たとえば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 topo-bgpls-xtc-nimo network-access
<network-access-ID>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo xtc-host <XTC-agent>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo backup-xtc-host
<XTC-agent-backup>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo asn <ASN-number>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo igp-protocol
<IGP-protocol-type>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo
extended-topology-discovery <true-or-false>
```

次のタスク

このタスクを実行した後、このネットワークモデルをソースネットワークとして使用して、追加の収集を構成できます。詳細については、NIMO の説明 （1 ページ）を参照してください。

BGP-LS XTC の詳細オプション

このトピックでは、XTCを使用してBGP-LSトポロジ収集を実行するときに使用できる詳細オプションについて説明します。

オプション	説明
ノード	
remove-node-suffix	ノードにこのサフィックスが含まれている場合は、ノード名からノードサフィックスを削除します。たとえば、「company.net」はネットワークのドメイン名を削除します。
nodes | interfaces	
net-recorder	「record」に設定すると、ライブネットワークとの間で送受信されるSNMPメッセージは、検出の実行時にnet-record-fileに記録されます。デバッグに使用されます。
net-record-file	SNMPレコードを保存するディレクトリ。デバッグに使用されます。
インターフェイス	
find-parallel-links	IGPデータベースに存在しないパラレルリンクを検索します（IS-IS TE拡張機能が有効になっていない場合）。
ip-guessing	トポロジデータベースに存在しないインターフェイスに対して実行するIPアドレス推測のレベル。（IS-IS TE拡張機能が有効になっていない場合に使用されます。） • off：推測を実行しません。 • safe：あいまいさのない推測を選択します。 • full：あいまいな場合に最善の判断を下します。

オプション	説明
lag	ポートメンバーの LAG 検出を有効にします。
lag-port-match	ポート回路でローカルポートとリモートポートを照合する方法を決定します。 • exact：LACP に基づいて照合します。 • none：ポート回路を作成しません。 • guess：できるだけ多くのポートに一致するポート回路を作成します。 • complete：最初に LACP に基づいて照合してから、できるだけ多くの照合を試みます。
cleanup-circuits	インターフェイスに関連付けられた IP アドレスを持たない回路を削除します。IS-IS アドバタイジングの不整合を修正するために、IS-IS データベースで回路の削除が必要になる場合があります。
copy-descriptions	論理インターフェイスが 1 つだけで、その説明が空白の場合は、物理インターフェイスの説明を論理インターフェイスにコピーします。
get-physical-ports	Cisco の L3 物理ポートを収集します。下位に L1 接続がある場合は、物理ポートを収集します。
min-prefix-length	パラレルリンクを検索するときに許可する最小プレフィックス長。プレフィックス長がそれ以上（ただし 32 未満）であるすべてのインターフェイスが考慮されます。
min-guess-prefix-length	最小の IP 推測プレフィックス長。プレフィックス長がそれ以上であるすべてのインターフェイスが考慮されます。

NIMO 収集の統合

アグリゲータは、デルタ集約ルールエンジン（DARE）を使用して、ユーザー指定の NIMO を単一の統合ネットワークモデルに結合します。アグリゲータは、ソース NIMO の機能を読み取ります。アグリゲータ機能の詳細については、ネットワークモデルを参照してください。

（注）
- XTC を使用するネットワークの場合、自動化されたネットワーク更新を取得して、自動化アプリケーションに使用できるリアルタイムのネットワークモデルを取得できます。詳細については、自動化アプリケーションを参照してください。
- アグリゲータの下に同じ NIMO タイプのネットワークを複数追加することはサポートされていません。外部実行可能な NIMO が設定されている場合、NIMO タイプは **aggregator/sources/source/nimo** で指定する必要があります。

始める前に

- 最終ネットワークモデルに含める NIMO を構成します。
- 最初の設定が完了するまで、収集を実行したり、これらの NIMO を実行したりしないことが重要です。DARE を設定する前に収集を実行する場合は、DARE ネットワークを最初から再構築する必要があります（[/wae:wae/components/aggregators/aggregator] <*network_name*>、次に [resync-aggregator-net] をクリックします）。
- ネットワーク モデル コンポーザ を使用すると、NIMO の集約の設定が簡素化されます。詳細については、ネットワーク モデル コンポーザ を使用 および ネットワーク モデル コンポーザ を使用した NIMO 収集の統合 のトピックを参照してください。

ステップ 1 空のネットワークを作成します。これが最終的な統合ネットワークモデルになります。エキスパートモードから、[設定エディタ（Configuration editor）] で [/wae:networks] に移動し、プラス（[+]）記号をクリックして、最終ネットワークモデル名を入力します。

ステップ 2 **/wae:wae/components/aggregators** に移動し、[アグリゲータ（aggregator）] タブを選択します。

ステップ 3 プラス（[+]）記号をクリックします。

ステップ 4 ドロップダウンの宛先リストから、最終ネットワークを選択し、[追加（Add）] をクリックします。

ステップ 5 ソースリンクをクリックします。

ステップ 6 プラス（[+]） 記号をクリックして送信元 NIMO を追加し、次の情報を入力します。

- [nimo]：NIMO タイプを入力します。
- [direct-source]：false に設定すると、このモデルへの変更は最終モデルに集約されます。デフォルトでは [true] に設定されています。
- [filter-capabilities]：集約前に機能フィルタを適用するかどうかを設定します。false に設定すると、すべての変更が集約に含まれます。たとえば、external-executable-nimo は機能を公開しないため、このフィールドは false に設定します。

（注） オンデマンドの帯域幅と、帯域幅の最適化のために sr-traffic-matrix-nimo を設定します。セグメント ルーティング トラフィック マトリックス収集 （13 ページ） を参照してください。

ステップ 7 （オプション）最終ネットワークモデルの下で収集を統合するすべての送信元 NIMO が追加されるまで続けます。

ステップ 8 （オプション）エージングパラメータを設定するには、[/wae:wae/components/aggregators] に移動し、[エージング（aging）] タブをクリックします。

- [aging-enabled]：[true] を選択してエージングを有効にします。
- [l3-node-aging-duration]：L3 ノードが非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。
- [l3-port-aging-duration]：L3 ポートが非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。
- [l3-circuit-aging-duration]：L3 回路が非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。

（注） l3-node-aging-duration の期間は l3-port-aging-duration より長くする必要があり、l3-port-aging-duration の期間は l3-circuit-aging-duration より長くする必要があります。

- [l1-node-aging-duration]：L1 ノードが非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。
- [l1-port-aging-duration]：L1 ポートが非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。
- [l1-link-aging-duration]：L1 リンクが非アクティブになった後に、ネットワーク内で保持する必要がある期間を入力します。

（注） l1-node-aging-duration の期間は l1-port-aging-duration より長くする必要があり、l1-port-aging-duration の期間は l1-link-aging-duration より長くする必要があります。

ステップ 9 [コミット（Commit）] ボタンをクリックします。

ステップ 10 ソース NIMO を実行します。最終ネットワークモデルがソースネットワークモデルからの最新情報で更新されます。アグリゲータとマルチレイヤ収集の構成例 （11 ページ）も参照してください。

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。

```
# wae components aggregators aggregator <final-network-model>
# sources source <nimo_1>
# sources source <nimo_2>
# dependencies dependency <demands-network>
# dependencies dependency <inventory-network>
# dependencies dependency <layout-network>
# dependencies dependency <traffic-network>
# final-network <sage-network>
# commit
```

アグリゲータが構成されたら、ソース NIMO を実行します。

アグリゲータとマルチレイヤ収集の構成例

この例は、CLI を使用して、レイヤ 3 とレイヤ 1 のネットワークモデル情報を結合するようにアグリゲータを構成する方法を示しています。

以下は、L1（optical）および L3（topo-igp-nimo）ネットワークモデルがネットワーク上に構成されていることを示しています。オプティカル NIMO および topo-igp-nimo を設定する方法の詳細については、IGP データベースを使用したトポロジ収集 （5 ページ）および EPN-M エージェントを使用したマルチレイヤ収集の設定を参照してください。

レイヤ 1 ネットワークモデル：

```
networks network l1-network
```

```
    nimo optical-nimo source-network l3-network
    nimo optical-nimo network-access cisco:access
    nimo optical-nimo optical-agents cisco:network
        advanced use-configure-l3-l1-mapping true
        advanced l3-l1-mapping    bg1_mapping
    !
```

レイヤ 3 ネットワークモデル：

```
networks network l3-network
    nimo topo-igp-nimo network-access bgl-lab-access
    nimo topo-igp-nimo igp-config 1
    seed-router 10.225.120.61
    igp-protocol isis
    !
    nimo topo-igp-nimo collect-interfaces true
    nimo topo-igp-nimo advanced interfaces lag true
    !
```

（注） 構成された L1 および L3 ネットワークモデルでは、収集はまだ実行されていません。

アグリゲータを構成します。

```
# config
# wae components aggregators aggregator l1-l3-final-model
# sources source l1-network
# sources source l3-network
# dependencies dependency dmd-network
# dependencies dependency inv-network
# dependencies dependency lyt-network
# dependencies dependency traffic-network
# final-network sage-1
# commit
```

アグリゲータが構成されたら、L3 および L1 収集を実行します。

```
# networks network l3-network nimo topo-igp-nimo run-collection
```

L1 ネットワーク収集を実行します。

```
# networks network l1-network nimo optical-nimo build-optical-topology
```

WAE Design を開いて、最終ネットワークモデルを表示します（**[ファイル（File）]** > **[開く場所（Open from）]** > **[WAE Automation Server]** から最終ネットワークモデルを選択し、データ収集を確認します）。

セグメント ルーティング トラフィック マトリックス収集

セグメントルーティング（SR）トラフィック収集（sr-traffic-matrix-nimo）は、SR トラフィックを検出します。このNIMOにより、収集されたテレメトリデータからネットワークの外部インターフェイス間でデマンドを生成できます。

始める前に

- 基本的なトポロジネットワークモデルが存在する必要があります。IGP データベースを使用したトポロジ収集 （5 ページ）またはXTC を使用したトポロジ収集 （6 ページ）を参照してください。
- テレメトリを設定する必要があります。WAE でのテレメトリの設定 トピック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、[設定エディタ（Configuration editor）] で、[/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。ソースネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_sr_traffic_matrix などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[sr-traffic-matrix-nimo] を選択します。

ステップ 6 [sr-traffic-matrix-nimo] をクリックして、送信元ネットワークの収集期間に入ります。

（注） LSP デマンドが生成される場合、送信元ネットワークは、すべての LSP データが含まれる集約ネットワークである必要があります。

ステップ 7 [コミット（Commit）] ボタンをクリックします。

ステップ 8 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** 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 sr-traffic-matrix-nimo source-network
<source-network>

# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection
# commit
```

VPN 収集

VPN 収集（topo-vpn-nimo）は、レイヤ 2 およびレイヤ 3 VPN トポロジを検出します。

始める前に

ネットワークトポロジ収集が完了している必要があります。詳細については、ネットワークモデルの作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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_vpn などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[topo-vpn-nimo] を選択します。

ステップ 6 [topo-vpn-nimo] をクリックして、次のように入力します。

- [source-network]：基本的なトポロジ情報を含む、該当するネットワークモデルを選択します。
- [network-access]：ネットワークアクセスを選択します。

ステップ 7 [vpn-types] タブをクリックします。

ステップ 8 プラス（[+]）アイコンをクリックして、少なくとも 1 つの VPN タイプを追加します。

- [VPWS]：ネットワークで Virtual Private Wire Service が使用されている場合は、このタイプを追加します。
- [L3VPN]：ネットワークでレイヤ 3 VPN が使用されている場合は、このタイプを追加します。

ステップ 9 [コミット（Commit）] ボタンをクリックします。

ステップ 10 [topo-vpn-nimo] タブに戻り、**[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

LSP 構成の収集

ネットワーク内の LSP 設定情報は、LSP 設定 NIMO（lsp-config-nimo）および LSP NSO エージェント（cisco-wae-nso-agent）を使用して収集されます。LSP NSO エージェントは、NSO インスタンス間の相互作用を管理し、LSP 情報を取得して、それを WAE ネットワークモデル表現に変換します。

LSP NSO エージェントは、<wae_run_directory> /packages/cisco-wae-nso-agent/res/files: でネットワークモデルの履歴と診断ファイルも保持します。

- Merge-full.xml：最新のネットワークモデルで収集されたデータが含まれています。

- Merge-full-last.xml：以前のネットワークモデルで収集されたデータが含まれています。
- patch.xml：以前のネットワークモデルと最新のネットワークモデルとのコンテンツの相違点（デルタ）のみが含まれます。
- filtered_<*network_name*>.xml ：指定されたネットワーク <network_name> からのノードのみを使用した、merged-full.xml のフィルタ処理されたバージョンが含まれます。
- 最後の NETCONF クエリからの一時的な診断ファイル：
  - nso_config_address_last.xml
  - nso_config_explicit_path_last.xml
  - nso_config_segment-list.last.xml
  - nso_config_tunnels_last.xml

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。詳細については、基本的なトポロジ収集 （5 ページ）を参照してください。

ステップ 1 WAE CLI で、LSP NSO エージェントが wae/agents/nso-agent/nso-servers の下から収集する場所を認識できるように、NSO インスタンスを設定します。

```
admin@wae# config
Entering configuration mode terminal
admin@wae(config)# wae agents nso-agent nso-servers <NSO_instance_name> host <host_ip_address>
port <netconf_ssh_port> user <user> password <password>
admin@wae(config)# commit
```

次のフィールドを設定する必要があります。

- NSO インスタンス名：NSO インスタンスで任意の一意の文字列識別子。
- ホスト：NSO インスタンスの IP アドレスまたはホスト名。
- ポート：NSO インスタンスで使用される NETCONF SSH ポート。これは netconf-north-bound/transport/ssh/port/ncs.conf にあります。デフォルト値は 2022 です。
- ユーザー：NETCONF API へのアクセスを許可されている NSO ユーザー。デフォルトのユーザーは「admin」です。
- パスワード：NSO ユーザーのパスワード。

ステップ 2 NSO インスタンスから LSP 情報を収集します。

```
admin@wae# wae agents nso-agent get-config-netconf
```

（注） 通常、定期的にLSP情報を収集するためにスケジュールされたタスクである必要があります。

ステップ 3 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 4 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。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_lsp_config などです。

ステップ 5 [Add] をクリックします。

ステップ 6 [nimo] タブをクリックします。

ステップ 7 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[lsp-config-nimo] を選択します。

ステップ 8 [lsp-config-nimo] をクリックして、送信元トポロジネットワークを入力します。

（注） lsp-config-nimo は、トポロジ NIMO に従う必要があります。たとえば、レイアウトネットワーク（layout-nimo）を送信元ネットワークとして選択することはできません。

ステップ 9 [コミット（Commit）] ボタンをクリックします。

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

XTC を使用した LSP 収集

XTC（lsp-pcep-xtc-nimo）を使用した LSP 検出は、bgpls-xtc-nimo から収集されたデータを使用し、LSP 情報を追加して、新しいネットワークモデルを作成します。

始める前に

XTC（bgpls-xtc-nimo）を使用した BGP-LS トポロジ収集がネットワークで完了していることを確認します。LSP を収集するための送信元ネットワークとしてこのモデルを使用する必要があります。詳細については、XTC を使用したトポロジ収集 （6 ページ）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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、**`networkABC_lsp_pcep_xtc`** などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[lsp-pcep-xtc-nimo] を選択します。

ステップ 6 [lsp-pcep-xtc-nimo] をクリックして、送信元ネットワークを入力します。これは、bgpls-xtc-nimo を使用して収集されたトポロジ情報を含むネットワークモデルです。

ステップ 7 [xtc-hosts] タブをクリックします。

ステップ 8 プラス（[+]）アイコンをクリックして、次のように入力します。

- [name]：XTC ホスト名を入力します。これは任意の名前にできます。
- [xtc-host]：ドロップダウンリストから、以前に設定された XTC ホストの 1 つを選択します。詳細については、エキスパートモードを使用した XTC エージェントの構成を参照してください。

ステップ 9 [reactive-network] タブをクリックして XTC からの通知をサブスクライブし、追加または削除に基づき LSP を更新します。次の情報を入力します。

- [enable] ：ネットワークトポロジを変更するための通知を有効にするには、[true] を選択します。
- [enable-triggering-index-collection]：SNMP を介して新しいトンネルでトンネルインデックスの収集をトリガーするには、[true] を選択します。
- [trigger-debounce-time]：トンネルインデックス収集をトリガーする前に、最後のトリガー通知を待機する時間を設定します。
- [network-access]：ネットワークのネットワーク アクセス プロファイルを入力します。
- [connect-timeout]：タイムアウトを分単位で入力します。
- [verbosity]：ログの詳細レベルを入力します。
- [net-recorder]：SNMP 記録アクションを選択します。デフォルトはオフです。
- [net-record-file]：SNMP 記録ファイル名を入力します。

ステップ 10 [コミット（Commit）] ボタンをクリックします。

ステップ 11 **[run-xtc-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

ステップ 12 収集が正常に実行されたことを確認するには、ネットワーク（**/wae:networks/network/<*network-name*>**）に戻り、[model] タブをクリックします。

ステップ 13 [nodes] をクリックします。ノードと詳細のリストが表示され、収集が成功したことが示されます。

ステップ 14 LSP があることがわかっているノードの 1 つを選択し、[lsps] タブをクリックします。

ステップ 15 [lsp] リンクをクリックします。検出された LSP のリストを含むテーブルが表示されます。

構成解析を使用したポート、LSP、SRLG、および VPN 収集

このトピックでは、cfg-parse-nimo NIMO について説明します。

（注） cfg-parse-nimo NIMO は、基本のトポロジ収集方法ではありません。cfg-parse-nimo NIMO は、SNMP や XTC など、他の収集方法に入っていない詳細を補う目的のみで使用する必要があります。

始める前に

- トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作成を参照してください。
- 構成解析エージェントが設定され、実行されている必要があります。詳細については、エキスパートモードを使用した構成解析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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_port-cfg-parse などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックし、NIMO タイプとして [cfg-parse-nimo] を選択します。

ステップ 5 [NIMO] リンクをクリックして、次の情報を入力します。

- [source-network]：トポロジ情報を含む、該当するネットワークモデルを選択します。
- [source]：cfg-parse-agent またはディレクトリのいずれかを選択します。cfg-parse-agent を使用して設定を取得した場合は、cfg-parse-agentオプションを選択してから、構成解析エージェントを選択します。または、ディレクトリに設定がある場合は、ディレクトリオプションを選択し、設定が保存されているディレクトリを入力します。

ステップ 6 [parse] タブをクリックします。

ステップ 7 構成解析値を入力します。フィールドの説明を表示するには、マウスポインタをフィールド名の上に置きます。一部のフィールドの詳細については、以下で説明します。

- [igp-protocol]：トポロジの一部であるインターフェイスとして、IS-IS および/またはOSPF 対応インターフェイスを選択します。デフォルトは [ISIS] です。
- [ospf-area]：エージェントは、単一または複数のエリアの情報を読み取ることができます。ospf-area オプションは、エリア ID または all を指定します。デフォルトは area 0 です。
- [isis level]：エージェントは、IS-IS レベル 1、レベル 2、またはレベル 1 とレベル 2 の両方のメトリックを読み取ることができます。両方を選択した場合、エージェントは両方のレベルを1つのネットワークに結合します。レベル 2 のメトリックが優先されます。
- [asn]：ASN はデフォルトで無視されます。ただし、複数の BGP ASN にまたがるネットワークでは、このオプションを使用して、ASN 内の複数の IGP プロセス ID またはインスタンス ID から情報を読み取ります。

[include-object] をクリックして、収集タイプ（lag、srlg、rsvp、vpn、frr、sr_lsps、lmp、sr_policies）を追加します。

（注）
- `l2vpn` 構成解析はサポートされていません。
- cfg-parse NIMO で `l3vpn` 情報を収集する場合、すべての VPN が相互に接続されていると見なされます。
- cfg-parse NIMO が VPN 情報を収集しており、topo-vpn NIMO も実行されている場合は、topo-vpn NIMO が NIMO チェーン内の cfg-parse NIMO の前であることを確認してください。
- 片方の端が欠落しているシングルエンドの SRLG は、SR-PCE を介して収集されます。ただし、SRLGSCircuits テーブルは更新されません。

ステップ 8 [コミット（Commit）] ボタンをクリックします。

ステップ 9 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

次のタスク

このタスクを実行した後、このネットワークモデルを送信元ネットワークとして使用して、追加の収集を設定できます。詳細については、NIMO の説明 （1 ページ）を参照してください。

BGP ピア収集

topo-bgp-nimo は、SNMP とログインを介して BGP トポロジを検出します。トポロジネットワーク（通常は IGP トポロジ収集モデル）をソースネットワークとして使用し、BGP リンクを外部 ASN ノードに追加します。

始める前に

トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作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、[設定エディタ（Configuration editor）] で、[/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。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_topo_bgp などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[topo-bgp-nimo] を選択します。

ステップ 6 [topo-bgp-nimo] をクリックして、次の情報を入力します。

- [source-network]：基本的なトポロジ情報を含む、該当するネットワークモデルを選択します。
- [network-access]：以前に設定されたネットワーク アクセス プロファイルを選択します。
- [min-prefix-length]：（オプション）[min-prefix-length] を入力して、BGP リンクとしてインターフェイスを検出する際の IPv4 サブネット照合の制限を制御します。
- [min-IPv6-prefix-length]：（オプション）[min-IPv6-prefix-length] を入力して、BGP リンクとしてインターフェイスを検出する際の IPv6 サブネット照合の制限を制御します。
- [login-multi-hop]：（オプション）マルチホップピアを含む可能性のあるルータにログインしない場合は、ログインマルチホップを無効にするかどうかを選択します。

詳細オプションについては、BGP トポロジの詳細オプション （20 ページ）を参照してください。

ステップ 7 [peer-protocol] タブをクリックし、使用するピア検出のタイプを選択します。IPv4、IPv6、またはその両方から選択します。

ステップ 8 [コミット（Commit）] ボタンをクリックします。

ステップ 9 **[run-collection] > [run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

BGP トポロジの詳細オプション

このトピックでは、BGP トポロジ収集を実行するときに使用できる詳細オプションについて説明します。

オプション	説明
force-login-platform	プラットフォーム検出をオーバーライドして、指定されたプラットフォームを使用します。有効な値：cisco、juniper、alu、huawei。
fallback-login-platform	プラットフォームの検出が失敗した場合のフォールバックベンダー。有効な値：cisco、juniper、alu、huawei。
try-send-enable	ルータにログインするときに、プラットフォームタイプが検出されない場合は、イネーブルパスワードを送信します。このアクションは、「-fallback-login-platform cisco」と同じ動作です。
internal-asns	内部 ASN を指定します。使用した場合、指定された ASN は内部に設定されます。その他はすべて外部に設定されます。デフォルトでは、検出されたものを使用します。
asn-include	対象となる ASN を指定します。使用した場合、ピア検出はこのリストに制限されます。デフォルトでは、検出されたすべての外部 ASN とピアリングします。
find-internal-asn-links	2 つ以上の内部 ASN 間のリンクを検索します。通常、IGP がこれらのリンクを検出するため、このアクションは必要ありません。
find-non-ip-exit-interface	ネクストホップ IP アドレスとしてではなく、インターフェイスとして表現される出口インターフェイスを検索します（これはまれです）。 （注） このアクションにより、BGP 検出に対する SNMP リクエストの量が増加し、パフォーマンスに影響します。
find-internal-exit-interfaces	内部 ASN への出口インターフェイスを収集します。
get-mac-address	Internet Exchange パブリック ピアリング スイッチに接続されている BGP ピアの送信元 MAC アドレスを収集します。このアクションは、MAC アカウンティングの場合にのみ必要です。
use-dns	DNS を使用して BGP IP アドレスを解決するかどうか。
force-check-all	マルチホップピアの可能性が示されていない場合でも、すべてのルータを確認します。このアクションは遅い可能性があります。
net-recorder	「record」に設定すると、ライブネットワークとの間で送受信される SNMP メッセージは、検出の実行時に net-record-file に記録されます。デバッグに使用されます。
net-record-file	SNMP レコードを保存するディレクトリ。デバッグに使用されます。

オプション	説明
login-record-mode	検出プロセスを記録します。 「record」に設定すると、ライブネットワークとの間で送受信されるメッセージは、ツールの実行時に login-record-dir に記録されます。デバッグに使用されます。
login-record-dir	ログインレコードを保存するディレクトリ。デバッグに使用されます。

SNMP を使用した LSP 収集

lsp-snmp-nimo は、SNMP を使用して LSP 情報を検出します。

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集 （5 ページ） を参照してください。

（注） デフォルトでは、Nokia デバイスは SNMP を使用した LSP 統計収集に対して有効になっていません。収集を成功させるには Nokia デバイスを有効にする必要があります。このオプションを有効にするには、Nokia の担当者にご相談ください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/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。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_lsp_config などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[lsp-snmp-nimo] を選択します。

ステップ 6 [lsp-snmp-nimo] をクリックして、次のように入力します。

- [source-network]：基本的なトポロジ情報を含む、該当するネットワークモデルを選択します。
- [network-access]：以前に設定されたネットワーク アクセス プロファイルを選択します。
- [get-frr-lsps]：マルチプロトコル ラベル スイッチング（MPLS）高速再ルーティング（FRR）LSP（バックアップおよびバイパス）情報を検出する場合は [true] を選択します。

ステップ 7 [コミット（Commit）] ボタンをクリックします。

ステップ 8 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

インベントリ収集

インベントリ収集（inventory-nimo）は、ハードウェアインベントリ情報を収集します。

収集されるハードウェア

`get_inventory` ツールは、ハードウェアタイプに基づいて収集されたハードウェア情報を格納する一連の NetIntHardware* テーブルを作成します。これらのテーブルは WAE Live で直接使用できませんが、そのうちの 4 つは WAE Live で使用するために `build_inventory` によって処理されます。次の各オブジェクトは、ノード IP アドレスと SNMP ID によって定義されます。

- NetIntHardwareChassis：ノード IP アドレスと SNMP ID によって識別されるルータシャーシオブジェクト。
- NetIntHardwareContainer：各エントリは、ルータ内のスロット（現場交換可能ユニット（FRU）タイプのデバイスを取り付けられる任意の機構）を表します。たとえば、シャーシスロット、モジュールスロット、ポートスロットなどです。
- NetIntHardwareModule：他のハードウェアデバイスに取り付けられるハードウェアデバイス通常、これらのデバイスは、ラインカード、モジュール、ルートプロセッサなど、トラフィックを直接サポートするものであり、他の機能固有のハードウェアテーブルのいずれにも分類されません。
- NetIntHardwarePort：ルータ上の物理ポート。

ハードウェア階層

ハードウェアには、オブジェクトがルータ内で存在する場所に基づいて親子関係があります。シャーシには親がなく、ルートオブジェクトと見なされます。シャーシ以外の各オブジェクトには 1 つの親があり、1 つ以上の子オブジェクトを持つことができます。子のないオブジェクトは、リーフオブジェクトと呼ばれます（ポートや空のコンテナなど）。この階層は、通常、ハードウェアオブジェクトが他のオブジェクト内にどのように取り付けられているかを反映しています。たとえば、ラインカードを表すモジュールには、スロットを表すコンテナである親オブジェクトがある場合があります。

親は、NetIntHardware* テーブル内で、ParentTable 列と ParentId 列によって識別できます。これらの 2 つの列を Node（ノード IP アドレス）列とともに使用すると、任意のハードウェアオブジェクトの親オブジェクトを見つけることができます。

例：次の NetIntHardwareContainer エントリは、コンテナ 172.23.123.456 に親としてのシャーシがあることを識別しています。NetIntHardwareChassis には、コンテナの ParentId である 2512347 に一致する SnmpID エントリがあります。

NetIntHardwareContainer							
ノード	SnmpID	ParentID	モデル	名前	NumChildren	ParentTable	SlotNumber
172.23.123.456	2503733	2512347		slot mau 0/0/0/5	0	NetIntHardwareChassis	0

親子関係に基づいて各リーフオブジェクトから対応するルートオブジェクトまで階層をトレースすると、一連のオブジェクトタイプでハードウェア階層が形成されます。このトレースは、`build_inventory` ツールがハードウェアデバイスの処理方法を決定するために使用します。これは、HWInventoryTemplatesテーブルにエントリを追加する場合にも使用する必要があるプロセスです。

例：Chassis-Container-Module-Module-Container-Port

インベントリを処理するためのテーブル

`build_inventory` ツールは、NetIntHardware* テーブルを処理して NetIntNodeInventory テーブルを作成します。このツールには2つの構成ファイルが必要で、オプションの構成ファイルを追加で使用することもできます。指定しない場合、`<run_directory>/packages/cisco-wae-inventory-nimo/priv/etc/inventory` に含まれているファイルが使用されます。

- master_inventory_templates.txt（必須）：このファイルには、次のテーブルが含まれています。
  - HWInventoryTemplates エントリは、最終的な NetIntNodeInventory テーブル内のデバイスを分類します。また、含まれた状態からプルーニングします。
  - HWNameFormatRules エントリは、ハードウェアオブジェクト名の書式を統一して使いやすくします。また、予期しない SNMP 結果を修正します。
- master_exclude_list.txt（必須）：ハードウェアオブジェクトが最終的な NetIntNodeInventory テーブルに含まれないようにするExcludeHWListテーブル（ブロックリスト）が含まれます。これは、トラフィックを転送または伝送しないハードウェアを除外する場合に役立ちます。
- master_hw_spec.txt（オプション）：SNMP によって返されたスロットが不正確な場合に、指定されたデバイスのスロット数に関して収集されたデータを調整するために使用できる HardwareSpec テーブルが含まれています。

テンプレートを変更するか、ファイルを除外することを選択した場合は、これらの変更をソフトウェアのアップグレード後に維持する必要があります。

ハードウェア テンプレートの設定

`build_inventory -template-file` オプションは、HWInventoryTemplates テーブルと HWNameFormatRules テーブルの両方を含むファイルを呼び出します。これらのテーブルは、デフォルトで `<run_directory>/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_inventory_templates.txt` ファイルに含まれます。

HWInventoryTemplates テーブル

HWInventoryTemplates テーブルは、NetIntHardware* テーブルによって参照されるハードウェアを解釈する方法を `build_inventory` ツールに指示します。これにより、`build_inventory` は、オブジェクトをシャーシ、ラインカード、スロットなどの一般的なベンダーに依存しないハードウェアタイプに分類し、関心のないハードウェアタイプを削除できるようになります。

インベントリハードウェアは、シャーシ、スロット、ラインカード、モジュールスロット、モジュール、ポートスロット、ポート、またはトランシーバとして分類されます。コンテナは、スロット、モジュールスロット、またはポートスロットのいずれかに分類されます。モジュールは、モジュールまたはラインカードのいずれかに分類されます。他のすべてのハードウェアオブジェクトは、その名前で分類されます。たとえば、シャーシはシャーシとして分類されます。これらの分類されたハードウェアオブジェクトは、WAE Live アプリケーションを介してインベントリレポートで使用できます。

build_inventory ツールは、HWInventoryTemplates テーブルの次の列で、NetIntHardware* テーブルとの一致をこの順序で調べます。

- DiscoveredHWHierarchy、Vendor、Model
- DiscoveredHWHierarchy、Vendor、*（*は Model 列のすべてのエントリを意味します）

`-guess-template-if-nomatch true` オプションを使用して、検索をさらに強化できます。この場合、最初の 2 つの基準を使用して一致が見つからなかった場合、WAE Collector は DiscoveredHWHierarchy と Vendor の一致のみを検索し、Model を考慮しません。

一致が見つかった場合、DiscoveredHWHierarchy 以降の列により、`build_inventory` によるハードウェアの分類方法が決まります。以降の列により、ハードウェアオブジェクトタイプ（シャーシ、スロット、ラインカード、モジュールスロット、モジュール、ポートスロット、ポート、またはトランシーバ）が識別されます。各列のエントリの形式は次のとおりです。

Type,Identifier,Name

- Type は、検出されたハードウェアタイプ（「コンテナ」など）です。
- Identifier は、（1 つ以上の同じタイプの）どのオブジェクトが参照されているのかを指定します（0、1、...）。
- Name は、NetIntHardware* テーブルの列見出しを指定します。これは、NetIntNodeInventory テーブル（すなわち WAE Live インベントリレポート）で、そのオブジェクトに対して表示される名前です。

例：Module,0,Model

（Model は、NetIntHardwareModule テーブルの列見出しです）

複数の名前ソース列をコロンで指定できます。

例：Container,0,Model:Name

ハードウェアカテゴリが存在しない場合、または空の場合、`build_inventory` は最終的な NetIntNodeInventory テーブルにカテゴリを含めません。

例

デフォルトの master_inventory_templates.txt ファイルの最初の行を使用して、WAE Collector は、次の Vendor、Model、および DiscoveredHWHierarchy 列に一致するエントリを持つ NetIntHardware* テーブルを検索します。

Cisco ASR9K Chassis-Container-Module-Port-Container-Module

その後、WAE Collector はハードウェア階層（DiscoveredHWHierarchy 列）の各エントリを分類し、ハードウェアタイプ列でその位置を定義します。

最初の Module エントリはラインカードとして定義され、#0 として識別されます。NetIntNodeInventory テーブルに表示される名前は、NetIntHardwareModule テーブルの Model 列に表示される名前です。2 番目のモジュールはトランシーバオブジェクトとして定義され、#1 として識別されます。同じ名前形式を使用します。

階層には 2 つのコンテナがありますが、Type として定義されるのは 1 つだけです。これは、2 番目のコンテナが NetIntNodeInventory テーブルに表示されないことを意味します。

HWInventoryTemplates エントリの追加

WAE Collector は、HWInventoryTemplates テーブルにないインベントリデバイスを検出した場合、リーフオブジェクトの SNMP ID やルータの IP アドレスなど、ハードウェア階層の一部を指定して警告を生成します。この情報を使用して、リーフからルートまでオブジェクトを手動でトレースし、HWInventoryTemplates テーブル内の適切なエントリを取得できます。ハードウェア階層のトレースについては、「ハードウェア階層」を参照してください。

ステップ 1 参照用に警告メッセージをコピーし、ステップ 2 で使用します。

ステップ 2 ルータの IP アドレス、およびリーフオブジェクトの SNMP ID、名前、およびモデルを使用して、NetIntHardwarePort または NetIntHardwareContainer テーブルのいずれかの警告で参照されているリーフオブジェクトを見つけます。

ステップ 3 リーフオブジェクトの ParentTable 列と ParentId 列を使用して、リーフを親までトレースします。連続する各親について、NetIntHardwareChassis テーブルのルートオブジェクト（シャーシ）に到達するまで、それぞれの ParentTable 列と ParentId 列を使用します。

ステップ 4 ハードウェア階層内の各オブジェクトが見つかったら、HWInventoryTemplates テーブルの DiscoveredHWHierarchy 列に追加します。また、Vendor 列と Model 列にも入力します。

ステップ 5 ハードウェア階層内の各オブジェクト（DiscoveredHWHierarchy 列）について、標準ハードウェアタイプのいずれかに分類します。これは、DiscoveredHWHierarchy 列の後に表示される列です。

HWNameFormatRules テーブル

HWNameFormatRules テーブルは、NetIntNodeInventory テーブルの名前の形式を指定する方法を指定します。これは、長い名前や意味のない名前を、ユーザーにとって読みやすく明確な名前に変換するのに役立ちます。

HWInventoryTemplates テーブルのエントリごとに、一致するベンダー、ハードウェアタイプ（HWType）、名前（PatternMatchExpression）が HWNameFormatRules テーブルで検索されます。次に、HWInventoryTemplates テーブルで指定された名前を使用するのではなく、ReplacementExpression 列で識別された名前で NetIntNodeInventory テーブルが更新されます。

複数の一致が適用される場合は、最初に見つかった一致が使用されます。PatternMatchExpression と ReplacementExpression はどちらも、一重引用符で囲んだリテラル文字列または正規表現として定義できます。

例：テーブルのエントリは次のように機能します。

- 名前が 4 文字で、A が文字列の先頭、Z が文字列の末尾であるすべての Cisco シャーシ名を 7507 に置き換えます。
- 800-20017-.* に一致するすべての Cisco ラインカード名を 1X10GE-LR-SC に置き換えます。
- 「Juniper (MX960) Internet Backbone Router」という名前のすべての Juniper シャーシを MX960 に置き換えます。

HWNameFormatRules			
ベンダー	HWType	PatternMatchExpression	ReplacementExpression
シスコ	シャーシ	\A4\Z	‘7507’
シスコ	ラインカード	800-20017-.*	‘1X10GE-LR-SC’
Juniper	シャーシ	Juniper (MX960) Internet Backbone Router	$1

（注） SNMP は、多くのスロット名を整数ではなくテキストとして返します。WAE Live インベントリレポートで最適に使用するには、スロット番号からすべてのテキストを削除することをお勧めします。

モデルまたは名前によるハードウェアの除外

`build_inventory -exclude-file` オプションは、ExcludeHWList テーブルが含まれているファイルを呼び出します。デフォルトでは、このテーブルは `<run_directory>/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_exclude_list.txt` ファイルに含まれています。このテーブルを使用すると、モデル、名前、またはその両方に基づいて、NetIntNodeInventory テーブルから除外するハードウェアオブジェクトを特定できます。これは、たとえば、管理ポートとルートプロセッサを除外する場合に役立ちます。モデルと名前は、正規表現またはリテラルを使用して指定できます。

例：テーブルのエントリは次のように機能します。

- ベンダーが Cisco で、名前が CPU0/129 で終わる NetIntHardwarePort テーブル内のすべてのオブジェクトを除外します。
- ベンダーが Cisco、モデルが 800-12308-02 である NetIntHardwareModule テーブル内のすべてのオブジェクトを除外します。
- ベンダーが Cisco、名前が Mgmt である NetIntHardwarePort テーブル内のすべてのオブジェクトを除外します。

ExcludeHWList			
HWTable	Vendor	モデル	名前

NetIntHardwarePort	シスコ		\/CPU0\/129$
NetIntHardwareModule	シスコ	800-12308-02	
NetIntHardwarePort	シスコ		管理

HardwareSpec

`build_inventory -hardware-spec-file` オプションは、HardwareSpec テーブルが含まれているファイルを呼び出します。デフォルトでは、このテーブルは *<run_directory>*`/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_hw_spec.txt` f ファイルに含まれています。このテーブルを使用すると、SNMP から返されるデータを調整できます。スロットの総数（TotSlot）とスロット番号の範囲（SlotNum）の両方を調整できます。たとえば、実際にはルートプロセッサを含めて9個のスロットがあるのに、SNMP はシャーシに 7 個のスロットを返すことがあります。

このテーブルでは、スロット、モジュールスロット、またはポートスロットを含むハードウェアのみを検索します。したがって、ハードウェアタイプ（HWType 列）は、シャーシ、ラインカード、またはモジュールである必要があります。SlotNum はスロット番号の範囲を示します。たとえば、スロット 0 から始まるルータもあれば、スロット 1 から始まるルータもあります。

例：次のテーブルエントリは、Cisco 7609 シャーシに合計 9 個のスロットを設定し、スロット番号付けを 9 から始めます。

HardwareSpec				
ベンダー	HWType	モデル	TotSlot	SlotNum
シスコ	シャーシ	7609	9	1-9

インベントリ収集の設定

始める前に

ネットワークトポロジ収集が完了している必要があります。詳細については、ネットワークモデルの作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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_inventory などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[inventory-nimo] を選択します。

ステップ 6 [inventory-nimo] をクリックして、次のように入力します。

- [source-network]：基本的なトポロジ情報を含む、該当するネットワークモデルを選択します。
- [network-access]：ネットワークアクセスを選択します。

ステップ 7 （オプション）**[詳細（advanced）]** > **[get-inventory-options]** をクリックします。`get-inventory` ツールは、ネットワークハードウェアを収集し、オブジェクトタイプ別に分離された MIB ウォークから収集されたすべてのデバイスを含むNetIntHardwareテーブルを作成します。このツールは、SSHおよびNETCONFを使用して、MIB では利用できないデータを収集します。

- [login-allowed]：[true] に設定すると、ルータにログインしてインベントリデータを収集できるようになります。
- [net-recorder]：このオプションは、通常はデバッグに使用されます。検出の実行時に、net-record-file 内のライブネットワークとの間でやり取りされる SNMP メッセージを記録するには、[record] に設定します。
- [net-record-file]：記録された SNMP メッセージが保存されるファイル名を入力します。

ステップ 8 （オプション）**[詳細設定（advanced）]** > **[build-inventoryオプション（build-inventory-options）]** をクリックします。`build-inventory` ツールは、NetIntHardware* テーブル内の未加工のハードウェアデータ情報を処理し、最終的な NetIntNodeInventory テーブル内の不要なオブジェクトを分類し、削除します。

- [login-allowed]：[true] に設定すると、NETCONF を使用してルータにログインし、インベントリデータを収集できるようになります。Juniper トランシーバの収集にのみ必要です。
- [guess-template-if-nomatch]：[true] に設定すると、未処理のインベントリデータを処理する際に検索範囲が広がります。
- [template-file]：[HWInvetory] テンプレートおよび [HWNameFormatRules] テーブルを含むハードウェア テンプレート ファイルです。
- [hardware-spec-file]：外部から返された SNMP データを確認するため、特定のタイプのハードウェアのスロット数を定義する [HardwareSpec] テーブルです。

ステップ 9 [コミット（Commit）] ボタンをクリックします。

ステップ 10 [inventory-nimo] タブに戻り、**[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

例

WAE CLI（設定モード）：

```
# networks network <network-model-name> nimo inventory-nimo source-network
<source-network> network-access <network_access_name>
# networks network <network-model-name> nimo inventory-nimo advanced get-inventory-options
 login-allowed <false_or_true>
# networks network <network-model-name> nimo inventory-nimo run-collection
# commit
```

auth.enc の作成

auth.encには、SNMPv2c、SNMPv3、またはその両方のログイン情報が含まれています。SNMPv2cは、安全性の低いセキュリティモデルを使用し、コミュニティストリングをクリアテキストで渡します。SNMPv3は、認証、整合性、および機密性をサポートする強力なセキュリティモデルを提供します。

このファイルを生成するには、WAE CLI（設定モード）で次の手順を実行します。

```
# wae nimos network-access generate-auth-file network-access <network_access_name>
file-path <directory>/auth.enc
```

ここで、<*directory*> は、auth.enc ファイルを保存する場所です。次に例を示します。

```
# wae nimos network-access generate-auth-file network-access test_lab file-path
/home/wae/auth.enc
message Successfully generated authfile at /home/wae/auth.enc
```

認証ファイルのパスワードとシードルータのデフォルトのログイン情報は、次の要素で構成されます。

- primary password：ファイルの内容を表示するためのパスワード
- login username：ルータへのログインアクセス用のデフォルトのユーザー名
- login password：ルータへのログインアクセス用のデフォルトのパスワード
- login enable password：ログインアクセス用のデフォルトのイネーブルパスワード

SNMPv2c 情報は、単一の値を使用して定義されます。

- community：デフォルトのコミュニティストリング

SNMPv3 情報は、認証と暗号化の詳細を定義します。

- セキュリティ レベル（Security level）
  - noAuthNoPriv：ユーザー名で認証しますが、ユーザーを検証せず、データを暗号化しません。
  - authNoPriv：ユーザー名で認証し、MD5 または SHA を使用してユーザーを検証しますが、データは暗号化しません。
  - authPriv：ユーザー名で認証し、MD5 または SHA を使用してユーザーを検証し、DES または AES を使用してデータを暗号化します。
- SNMPv3 username：認証用のユーザー名
- Authentication Protocol：MD5 または SHA
- Authentication password：認証用のパスワード
- Encryption protocol type：DES または AES
- Encryption password：暗号化用のパスワード

- Context name：SNMP エンティティによってアクセス可能な管理情報のコレクションの名前

最初の暗号化認証ファイルを作成した後、その内容を手動で編集して複数のプロファイルまたはコミュニティを追加し、ルータをそれらにマッピングできます。各プロファイルには、SNMPv3 認証および暗号化情報の完全なセットが含まれています。ルータの異なるグループが異なる認証情報を使用する場合は、複数のプロファイルまたはコミュニティが必要です。

トラフィック収集

traffic-poll-nimo は、SNMP ポーリングを使用してトラフィック統計（インターフェイス測定）を収集します。

始める前に

この NIMO には、次のものが必要です。

- 基本的なトポロジネットワークモデル。
- VPN トラフィックを収集する場合、VPN ネットワークモデルが存在する必要があります。VPN 収集 （14 ページ） を参照してください。
- LSP トラフィックを収集する場合、LSP ネットワークモデルが存在する必要があります。SNMP を使用した LSP 収集 （21 ページ） を参照してください。
- 開いているファイルの最大数（ulimit -n）：1,000,000

制限事項

- 外部インターフェイスからのノードトラフィック情報は収集されません。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/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。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_traffic_polling などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[traffic-poll-nimo] を選択します。

ステップ 6 [traffic-poll-nimo] をクリックして、次のように入力します。

- [source-network]：該当するネットワークモデルを選択します。
- [network-access]：ネットワークアクセスを選択します。

（注） トラフィックポーラーの開始後に選択されたネットワークアクセスを変更した場合、ネットワークアクセスの変更を有効にするためにトラフィックポーラーを再起動する必要があります。

ステップ 7 インターフェイスの継続的なトラフィック収集を実行するには、[iface-traffic-poller] タブをクリックして、次のように入力します。

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （32 ページ）を参照してください。
- [qos-enabled]：キュートラフィック収集を有効にする場合は、[true] に設定します。
- [vpn-enabled]：VPN トラフィック収集を有効にする場合は、[true] に設定します。[true] に設定する場合は、ソースネットワークモデルで VPN が有効になっていることを確認します。

ステップ 8 LSP の継続的なトラフィック収集を実行するには、[lsp-traffic-poller] タブをクリックして、次のように入力します。

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （32 ページ）を参照してください。

ステップ 9 MAC アカウンティングの継続的なトラフィック収集を実行するには、[mac-traffic-poller] タブをクリックして、次のように入力します。

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （32 ページ）を参照してください。

（注） [mac-traffic-poller] が有効になっている場合は、送信元ネットワークモデルに MAC アドレスがあることを確認してください。

ステップ 10 [コミット（Commit）] ボタンをクリックします。

ステップ 11 [traffic-poll-nimo] タブに戻り、**[run-snmp-traffic-poller]** > **[run-snmp-pollerの呼び出し（Invoke run-snmp-poller）]** をクリックします。今後、継続的な収集を停止するには、[stop-snmp-traffic-poller] をクリックします。

トラフィックポーラーの詳細オプション

このトピックでは、トラフィック収集（traffic-poll-nimo）を構成するときに使用できる詳細オプションについて説明します。

オプション	説明
snmp-traffic-poller	
stats-computing-minimum-window-length	トラフィック計算の最小ウィンドウ長を秒単位で入力します。デフォルトは 300 秒です。
stats-computing-maximum-window-length	トラフィック計算の最大ウィンドウ長を秒単位で入力します。デフォルトは 450 秒です。

オプション	説明
raw-counter-ttl	raw カウンタを保持する期間を分単位で入力します。デフォルトは 15 分です。
net-recorder	このオプションは、通常はデバッグに使用されます。検出の実行時に、net-record-file 内のライブネットワークとの間でやり取りされる SNMP メッセージを記録するには、[record] に設定します。
log-file	トラフィックポーラーのログファイル。
net-record-file	記録された SNMP メッセージが保存されるファイル名を入力します。
verbosity	ポーラーのロギングレベルを設定します。デフォルトは 30 です。 • 40：情報 • 50：デバッグ • 60：トレース
snmp-traffic-population	
scheduler-interval	トラフィックの投入を実行する間隔を秒単位で入力します。デフォルトは 300 秒です。トラフィック統計を構成データベース（CDB）に送信します。 0 に設定すると（通常、オンデマンド帯域幅アプリケーションを使用するときに設定さします）、継続的なポーラーはトラフィックを自動的に計算して入力しません。`nimo traffic-poll-nimo advanced snmp-traffic-population` が実行された場合にのみ、モデルを計算して入力します。WMD は RPC API からトラフィック統計情報をプルします。トラフィック統計は CDB に送信されません。
connect-timeout	トラフィック投入の最大実行時間を分単位で入力します。
kafka-config	
broker-url	Kafka ブローカの URL。
zookeeper-url	Kafka zookeper の URL。

トラフィックポーリングの調整

トラフィックポーラーは、ネットワークから未処理のトラフィックカウンタを収集します。収集時間は、ネットワークサイズ、ネットワーク遅延、および個々のノードからの応答時間によって異なります。

トラフィックポーリングを効率的に実行するには、次の手順を実行します。

1. トラフィックポーラーのロギングレベルを 40 に設定します。

2. デフォルトのオプションで開始し、数時間連続収集を実行します。デフォルトの値は次のとおりです。

```
iface-traffic-poller/period = 60
lsp-traffic-poller/period = 60
advanced/snmp-traffic-poller/stats-computing-minimum-window-length = 300
advanced/snmp-traffic-poller/stats-computing-maximum-window-length = 450
advanced/snmp-traffic-poller/raw-counter-ttl = 15
advanced/snmp-traffic-population/scheduler-interval = 300
```

3. poller.log ファイルを表示します。デフォルトでは、ファイルは *<wae_run_time_directory>*/logs/*<network_name>*-poller.log にあります。

4. 実際の収集時間を検索します。次に例を示します。

```
Info [40]: LSP Traffic Poller: Collection complete. Duration: 43.3 sec
Info [40]: Interface Traffic Poller: Collection complete. Duration: 42.7 sec
```

上記の例で、ポーラーがネットワークをポーリングできる最速のペースは約 40 ～ 50 秒です。この秒数は、`iface-traffic-poller->period` および `lsp-traffic-poller->period` の最小値です。トラフィックポーラーはインターフェイスと lsps の両方のトラフィックを同時に入力するため、両方の値を同じ値に設定することをお勧めします。

5. トラフィックポーラーは、未処理のトラフィック カウンタ c1、c2、～ cn を収集してトラフィックを計算します。トラフィックを計算するには、少なくとも 2 つのカウンタが必要です。

```
(c2.counter - c1.counter)/(c2.timestamp - c1.timestamp)
```

6. スライディングウィンドウ `stats-computing-minimum-window-length` は、2 つのカウンタのサンプリングに使用されます。最も遠い 2 つのカウンタ、すなわち指定された期間の最新のカウンタと最古のカウンタを探します。この期間の平均トラフィックが計算されます。ポーラーには少なくとも 2 つのカウンタが必要であるため、`stats-computing-minimum-window-length` の最小値は `polling period` X 2 になります。バリエーションに対応するには、25% 以上を追加します。

ネットワーク遅延またはノードの応答時間の増加により、`stats-computing-minimum-window-length` で指定された期間のカウンタが見つからなかった場合、トラフィックは `N/A` として報告されます。トラフィックが空になるのを避けるために、`stats-computing-maximum-window-length` という保険ウィンドウがあり、この最小値は `polling period` X 2 に等しくなります。より長いポーリング期間に対応するには、50% 以上を追加します。応答しないノードの場合は、100% 以上を追加します。

7. トラフィックポーラーは、トラフィック計算のために未処理のカウンタをメモリに保存します。これは RAM スペースで場所を取ります。トラフィックポーラーは、メモリに保存されている古いカウンタを時折クリーンアップします。`raw-counter-ttl`（分）より古いものはすべてクリーンアップされます。つまり、上記の制約を考慮すると、`raw-counter-ttl` の最小値は `stats-computing-maximum-window-length` より大きくなります。

8. トラフィックポーラーへのトラフィック投入とは、ネットワーク内のトラフィックを計算し、プランファイル /CDB/WMD に投入するプロセスです。所要時間は、ネットワー

クサイズとターゲット（プランファイル/CDB/WMD）によって異なります。最速のターゲットはプランファイル（ネイティブモード）です。トラフィックの投入にかかる時間は、wae-java-vm ログファイルで確認できます。次に例を示します。

```
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 379976
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 391953
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 388853
```

上記の例では、トラフィックを投入できる（他のツールによって消費される）最速のレートは約 400 秒です。

9. wae-java-vm ログファイルに、カウンタ値が意味をなさないことを示す Invalid counter の警告が表示されることがあります。たとえば、c1.counter が c2.counter よりも大きい（これはネガティブなトラフィックを招く原因になります）場合などです。これは、カウンタがリセットまたはオーバーフローしたときに発生します。32 ビットのカウンタでよく起こります。この現象が多数見られる場合は、スライディングウィンドウのサイズを大きくしてより多くのカウンタを処理し、失敗の可能性を減らします。

10. ただし、トラフィックを投入するよりも速いレートでネットワークをポーリングすることはお勧めしません。上記の例では、トラフィックポーリングの最も積極的な設定は50秒ですが、投入には約 400 秒かかります。ネットワークポーリングを 8 つ無駄にしていることになります。そのため、トラフィックポーリングの期間を（スライディングウィンドウのサイズと raw-counter-ttl とともに）増やすことができます。上記のネットワークに推奨される設定は次のとおりです。

```
nimo traffic-poll-nimo iface-traffic-poller period 180
nimo traffic-poll-nimo lsp-traffic-poller enabled
nimo traffic-poll-nimo lsp-traffic-poller period 180
nimo traffic-poll-nimo advanced snmp-traffic-poller
stats-computing-minimum-window-length 400
nimo traffic-poll-nimo advanced snmp-traffic-poller
stats-computing-maximum-window-length 800
nimo traffic-poll-nimo advanced snmp-traffic-poller raw-counter-ttl 15
nimo traffic-poll-nimo advanced snmp-traffic-population scheduler-interval 400
nimo traffic-poll-nimo advanced snmp-traffic-population connect-timeout 60
```

（注） snmp-traffic-population connect-timeout は、トラフィック投入が 60 分に調整されます。このタイムアウトは通常は使用されないため、十分な長さにする必要があります。

上記のサンプル設定は、トラフィックのポーリングと投入に関しては最も積極的な値です。これらの数値は、CPU リソースとネットワーク帯域幅を節約するために、あまり積極的にならないように調整できます。次に例を示します。

```
nimo traffic-poll-nimo iface-traffic-poller period 240
nimo traffic-poll-nimo lsp-traffic-poller enabled
nimo traffic-poll-nimo lsp-traffic-poller period 240
nimo traffic-poll-nimo advanced snmp-traffic-poller
stats-computing-minimum-window-length 600
nimo traffic-poll-nimo advanced snmp-traffic-poller
stats-computing-maximum-window-length 1200
nimo traffic-poll-nimo advanced snmp-traffic-poller raw-counter-ttl 20
nimo traffic-poll-nimo advanced snmp-traffic-population scheduler-interval 600
nimo traffic-poll-nimo advanced snmp-traffic-population connect-timeout 60
```

ネットワークモデルのレイアウト（可視化）

layout-nimo は、送信元ネットワークモデルにレイアウトプロパティを追加して、プランファイルを Cisco WAE Design にインポートするときの視覚化を改善します。NIMO は、レイアウトプロパティへの変更を自動的に記録します。送信元ネットワークモデルが変更されると、接続先モデルのレイアウトが更新されます。

接続先ネットワークのレイアウトは、送信元ネットワークに適用されるテンプレートとして機能します。得られるネットワークは、新しい接続先ネットワークとして保存されます。送信元レイアウトにレイアウト情報が含まれていない場合、接続先ネットワークのレイアウトが送信元ネットワークに追加されます。送信元ネットワークにレイアウト情報が含まれている場合、そのレイアウトは、接続先ネットワークのレイアウトと競合がない限り維持されます。競合が存在する場合、接続先ネットワークのレイアウト情報が送信元ネットワークの情報よりも優先されます。

たとえば、新しい L1 ノードが送信元ネットワークに追加され、対応するサイト割り当てがあるとします。この L1 ノードは、サイト割り当てとともに接続先ネットワークに追加されます。ここで、既存の L1 ノードでは送信元ネットワークと接続先ネットワークでサイト割り当てが異なると仮定します。この場合、接続先ネットワークのサイト割り当てが保持されます。

次の 2 つの手順があります。

1. layout-nimo を使用して新しいネットワークモデルを作成します。
2. WAE Design を使用して新しいネットワークモデルにレイアウトテンプレートを追加し、パッチを送信します。詳細については、『Cisco WAE ネットワーク可視化ガイド』を参照してください。

始める前に

- サーバーの Cisco WAE Design バージョンが Cisco WAE バージョンと同じかそれ以上であることを確認してください。
- 基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集（5 ページ） 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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。この手順では、例として **networkABC_layout** を使用します。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[layout-nimo] を選択します。

ステップ 6 [layout-nimo] をクリックして、次のように入力します。

- [source-network]：使用するネットワークの送信元ネットワークを入力します。

- [template-plan-file-path]：レイアウトの詳細のコピー元となるテンプレートプランファイルのパスを入力します。

ステップ 7 [コミット（Commit）] ボタンをクリックします。

ステップ 8 WAE Design を起動し、**[ファイル（File）]** > **[開く場所（Open From）]** > **[WAE Automation Server]** を選択します。

ステップ 9 適切な詳細を入力し、作成したばかりのネットワークモデル（networkABC_layout）のプランファイルを選択して、[OK] をクリックします。

ステップ 10 レイアウトを編集します。Cisco WAE ネットワーク可視化ガイドの章「レイアウトの使用」を参照してください。

ステップ 11 **[ファイル（File）]** > **[保存先（Save To）]** > **[WAE Automation Sever]** オプションを使用して、変更したプランファイルをコレクタサーバーの **template-plan-file-path**（ステップ 6 を参照）に保存します。

ステップ 12 **[run-layout]** > **[run-layoutの呼び出し（Invoke run-layout）]** をクリックします。

ステップ 13 Cisco WAE Design のレイアウトネットワークからプランを開き、視覚的なレイアウトが期待どおりであるかを確認します。

例

external-executable-nimo を使用する WAE CLI（設定モード）：

1. ネットワークの Cisco WAE Design からプランファイルを開きます。この例では、送信元ネットワークは NetworkABC_demands です。

2. レイアウトを更新します。

3. ファイルを保存します。この例では、プラン ファイルの名前は template_01.pln です。

```
networks network networkABC_layout
nimo layout-nimo source-network NetworkABC_demands
nimo layout-nimo template-plan-file-path /home/centos/plan_files/template_01.pln
nimo layout-nimo advanced advanced-options -method
value visual
!
!
```

マルチキャストフローデータの収集

マルチキャスト NIMO は、特定のネットワークからマルチキャストフローデータを収集します。これは、次の NIMO の収取です。

- snmp-find-multicast-nimo：SNMP を使用してマルチキャストフローのマルチキャストデータを収集します。
- snmp-poll-multicast-nimo：SNMP を使用してマルチキャストフローのトラフィックデータレートを収集します。

- login-find-multicast-nimo：ルータにログインして、マルチキャストフローデータを取得または解析します。
- login-poll-multicast-nimo：ルータにログインしてマルチキャスト トラフィック レートを取得します。

始める前に

トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作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、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_multicast などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 該当する NIMO を NIMO タイプとして選択します。snmp-find-multicast-nimo、snmp-poll-multicast-nimo、login-find-multicast-nimo、login-poll-multicast-nimo から選択します。

ステップ 6 [NIMO] リンクをクリックして、次の情報を入力します。

- [network-access]：ネットワークのネットワーク アクセス プロファイルを選択します。
- [source-network]：トポロジ情報を含む、該当するネットワークモデルを選択します。

ステップ 7 [詳細設定（advanced）] タブをクリックして、情報を入力します。フィールドにマウスを合わせると、詳細が表示されます。

ステップ 8 [コミット（Commit）] ボタンをクリックします。

ステップ 9 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

例

WAE CLI を使用する場合は、次のコマンドを使用します。

マルチキャスト NIMO を次のように構成し、1 つの NIMO を次の NIMO のソースとして使用します。

```
networks network snmp_find_mc
nimo snmp-find-multicast-nimo network-access network_access
nimo snmp-find-multicast-nimo source-network dare_network
!
networks network login_find_mc
nimo login-find-multicast-nimo network-access network_access
nimo login-find-multicast-nimo source-network snmp_find_mc
!
networks network snmp_poll_mc
nimo snmp-poll-multicast-nimo network-access network_access
nimo snmp-poll-multicast-nimo source-network login_find_mc
!
networks network login_poll_mc
```

```
nimo login-poll-multicast-nimo network-access network_access
nimo login-poll-multicast-nimo source-network snmp_poll_mc
!
```

アグリゲータ設定で、`snmp-find` および `snmp-poll` マルチキャストネットワークを間接としてマークします（direct-source を False に設定）。

```
wae components aggregators aggregator  dare_network
sources source mcast-topo
  nimo topo-igp-nimo
!
dependencies dependency login_find_mc
  nimo login-find-multicast-nimo
!
dependencies dependency login_poll_mc
  nimo login-poll-multicast-nimo
!
dependencies dependency snmp_find_mc
  nimo          snmp-find-multicast-nimo
  direct-source false
!
dependencies dependency snmp_poll_mc
  nimo          snmp-poll-multicast-nimo
  direct-source false
!
final-network mcast-final
!
```

トラフィック需要の収集

traffic-demands-nimo は、ネットワークからトラフィック需要に関する情報を収集します。

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集 （5 ページ） 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、[設定エディタ（Configuration editor）] で、[/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。ソースネットワーク名と NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_traffic_demands_config などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[traffic-demands-nimo] を選択します。

ステップ 6 [traffic-demands-nimo] をクリックし、次のように入力します。

- [source-network]：基本的なトポロジ情報を含む、該当するネットワークモデルを選択します。
- [connection-timeout]：接続のタイムアウトを分単位で入力します。

ステップ 7 [demand-mesh-config] タブで、[demand-mesh-steps] をクリックします。

ステップ 8 [+] をクリックして、ステップを追加します。ステップの名前を入力し、[追加（Add）] をクリックします。

ステップ 9 作成したステップをクリックします。[選択-ツール （Choice-tool] ドロップダウンメニューからツールを選択します。

ステップ 10 ツールをクリックし、必要な詳細を入力します。

ステップ 11 [詳細設定（advanced）]タブをクリックし、詳細を入力します。オプションの説明を表示するには、フィールドの上にマウスポインタを合わせます。

設定にさらに手順を追加するには、手順 9 ～ 11 を繰り返します。

ステップ 12 [コミット（Commit）] ボタンをクリックします。

ステップ 13 **[run-collection]** > **[run-collectionの呼び出し（Invoke run-collection）]** をクリックします。

AS プランファイルのマージ

異なる自律システム（AS）からの計画ファイルは、**inter-as-nimo** を使用してマージできます。**inter-as-nimo** は、計画ファイル間の競合を解決します。ネイティブ形式の計画ファイルもサポートされています。

各 AS は、異なる WAE サーバーに置くことができます。

（注）
- 自律システム（AS）、回路、ノード、インターフェイス、外部エンドポイント、仮想ノードおよび未解決のインターフェイスを持つ外部エンドポイントメンバーのみが対象です。
- 次のデマンドが解決されます。
  - 実ノードで解決される仮想ノードに関連付けられた送信元または宛先。
  - 特定の形式でインターフェイスに関連付けられた送信元または宛先。
  - 外部エンドポイントに関連付けられた送信元または宛先。
- 次のデマンドは解決されていません。
  - ASN 番号のみに関連付けられた送信元または宛先。
- 特定のプランファイルについては、他のプランファイルが外部 AS 番号として認識するものと内部 AS 番号が一致する必要があり、マージされるすべての自律システムは、すべてのプランファイルで検出される必要があります。

始める前に

- さまざまな自律システム（AS）のトポロジとトラフィック情報を収集します。
- 異なる AS からのプランファイルは、同じ WAE サーバーに存在する必要があり、プランファイルへのパスを指定する必要があります。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/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。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_merge_as_plan などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[inter-as-nimo] を選択します。

ステップ 6 [inter-as-nimo] をクリックして、次のように入力します。

- [retain-demands]：true を選択してデマンドをマージします。
- [tag-name]：.pln ファイル内の更新された行の識別に役立つタグ名を入力します。.pln ファイルのタグ列は、変更された行のタグ名で更新されます。
- [path-to-report-file]：マージ後にドロップされた行がレポートされる、レポートファイルへのパスを入力します。

ステップ 7 [送信元（sources）] タブで、[+] をクリックしてネットワークに入ります。[Add] をクリックします。

ステップ 8 [plan-file-path] を入力します。このフィールドが空白の場合、NIMO は指定された名前の送信元を検索します。

ステップ 9 [確定する（Commit）] をクリックします。

ステップ 10 **[merge-inter-as]** > **[Invoke merge-inter-as]** をクリックします。

CLI を使用して AS プランファイルをマージするには、次のコマンドを使用します。

```
networks network <network-name>
 nimo inter-as-nimo retain-demands true
 nimo inter-as-nimo tag-name <tag-name>
 nimo inter-as-nimo path-to-report-file <report-file-path>
 nimo inter-as-nimo sources <source1>
  plan-file-path <source1-plan-file-path>
 !
 nimo inter-as-nimo sources <source2>
  plan-file-path <source2-plan-file-path>
 !
!
```

ネットワークモデルに対する外部スクリプトの実行

external-executable-nimo を使用すると、選択したネットワークモデルに対してカスタマイズされたスクリプトを実行できます。既存の Cisco WAE NIMO が提供しないネットワークからの特定のデータが必要な場合は、これを行うことがあります。この場合、Cisco WAE で作成された既存のモデルを取得し、カスタムスクリプトからの情報を追加して、必要なデータを含む最終ネットワークモデルを作成します。

シスコでは、この NIMO をインベントリ収集、レイアウト情報の適用、需要の作成、およびデマンド推論に使用することをお勧めします。詳細は、次のトピックを参照してください。

- インベントリ収集の設定 （27 ページ）
- ネットワークモデルのレイアウト（可視化） （35 ページ）

別の例が外部スクリプトの実行例 （42 ページ）トピックに記載されています。

始める前に

この NIMO の設定は、ネットワーク モデル コンポーザ を使用して Cisco WAE UI でも利用できます。

ステップ 1 エキスパート モード から、[設定エディタ（Configuration editor）] で、[/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。

NIMO タイプに一致する名前を使用するか、その機能に一致する既存の NIMO の名前を使用することをお勧めします。

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]：送信元ネットワークモデルのプランファイルバージョンを入力します（6.3、6.4 など）。デフォルトは現在のバージョンです。
- [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を指定します。デフォルトは .pln です。
- [argv]：スクリプトの実行に必要な引数を（順番に）入力します。WAE CLI ツールのいずれかを実行している場合は、送信元ネットワークモデルに $$input、結果のネットワークモデル（スクリプトの実行後）に $$output、認証ファイルに $$authfile を入力します（ネットワークアクセスが設定されている必要があります）。$$input、$$output、およびその他の argv 引数は、スクリプトで必要な順序でリストする必要があることに注意することが重要です。例については、外部スクリプトの実行例 （42 ページ）を参照してください。

ステップ 7 [external-executable-nimo] タブで、[実行（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 "<command[s]> <arguments>"
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
```

外部スクリプトの実行例

この例では、WAE CLI で external-executable-nimo を使用する方法について説明します。サンプルの Python スクリプト（ext_exe_eg.py）は、ネットワーク内のすべてのインターフェイスに「私のIGPメトリックは<*value*>（My IGP metric is <value>）」という説明を付加します。別の例については、インベントリ収集の設定 （27 ページ）トピックを参照してください。

ext_exe_eg.py の内容：

```
import sys
from com.cisco.wae.opm.network import Network

src  = sys.argv[1]
dest = sys.argv[2]

srcNet =  Network(src)

for node in srcNet.model.nodes:
    cnt = 1
    for iface in node.interfaces:
        iface.description = 'My IGP metric is ' + str(iface.igp_metric)
        cnt = cnt + 1

srcNet.write(dest)
```

WAE CLI で、次のように入力します。

```
admin@wae(config)# networks network net_dest nimo external-executable-nimo source-network
 net_src
advanced argv "/usr/bin/python3 /home/user1/srcs/br1/mate/package/linux/run/ext_exe_eg.py
 $$input $$output"
admin@wae(config-network-net_dest)# commit
Commit complete.
admin@wae(config-network-net_dest)# exit
admin@wae(config)# exit

admin@wae# networks network net_dest nimo external-executable-nimo run
status true
message Changes successfully applied.
```

Cisco WAE Design のネットワークプランファイルをチェックして、スクリプトが成功したことを確認します。

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。