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 （11 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 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 BGP-LS トポロジ収集（8 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イス/ポートのプロパティは、SNMP を使用して検出されます。	• この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 （15 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 （21 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。
LAG ポートと LMP インターフェイス収集（20 ページ）	port-cfg-parse-nimo	ネットワーク内のルータ構成から LAG ポートとリンク管理プロトコル（LMP）インターフェイスを検出します。	• 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。 • この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。構成解析エージェントの構成 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
マルチレイヤ収集の構成	optical-nimo	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。マルチレイヤ収集のワークフローを参照してください 。
NSO NED を使用した LSP 収集 （15 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 （23 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 PCEP LSP 収集 （19 ページ）	lsp-pcep-xtc-nimo	XTC を使用して PCEP LSP を検出します。	この収集を実行する前に、XTC を使用した BGP-LS トポロジ収集 （8 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 LSP トラフィック収集 （24 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 （11 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。
自律システム（AS）モデルの統合 （13 ページ）	as-merger	AS モデル間で共有されるインターフェイス、回路などを解決して、単一の統合ネットワークモデルを作成します。	• 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			

収集または機能	NIMO	説明	前提条件/注意事項
継続的な収集 （25 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（23 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 （15 ページ） を参照してください。
ネットワークモデルの可視化 （28 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。	• 統合ネットワークモデル（アグリゲータ）です。 • layout-nimo が構成されたら、レイアウトプロパティを含むプランファイルを layout-nimo モデルにインポートして戻す必要があります。
デマンドメッシュの作成 （30 ページ）	demandmesh-creator-nimo	一連の送信元ノードと接続先ノードの間にデマンドメッシュを作成します。	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。
デマンド推論 （29 ページ）	demand-deduction-nimo	測定された SNMP トラフィックを使用し、エンドツーエンドのデマンドのデマンドメッシュビルド（トラフィックマトリックス）をネットワークモデルに適用することにより、デマンド推論を実行します。	この NIMO では、デマンドとインターフェイスの測定値（traffic-poller）が含まれる送信元ネットワークとして統合ネットワークモデル（アグリゲータ）を使用する必要があります。
ネットワークモデルに対する外部スクリプトの実行 （32 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送信元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 （11 ページ）を参照してください。

IGP 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 の前に構成される最初の NIMO です。この NIMO は、完全なトポロジディスカバリを提供し、一般的ではありませんが、インターフェイスまたはポートの詳細を収集しないトポロジディスカバリも提供します。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、構成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 プロファイルを構成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タイプを選択します。

ステップ 3 シードルータの管理 IP アドレスを入力します。

ステップ 4 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。

ステップ 5 [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 6 （オプション）収集から個々のノードを除外するには、[node-blacklist] タブをクリックし、ノードの該当する IP アドレスを入力します。

（注） 詳細オプションについては、IGP トポロジの詳細オプション （6 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 [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。

ステップ 9 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 10 [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。

IGP トポロジの詳細オプション

このトピックでは、IGP 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
igp	
backup-router	自動フェールオーバーに使用するセカンダリシードルータ。
get-segments	IS-IS データベースからセグメント ルーティング データを収集します。Cisco IOS XR ルータの IS-IS にのみ有効です。
ospf-area	単一の OSPF エリアを収集するか、すべてのエリアを収集します。 エリア ID は、整数または IP アドレスとして指定できます。「all」に設定すると、ABR はエリア 0 の情報から識別され、ゼロ以外のエリア情報でログインします。
ospf-proc-id	複数の OSPF プロセスがある場合に使用する OSPF プロセス ID。値は正の整数です。
database-file	raw IGP データベースを書き込むファイル。
オフライン	オフラインモードで IGP 検出を実行します。 オフラインモードでは、ルータへのログインアクセスは実行されません。代わりに、必要な構成をデータベースファイルで提供する必要があります。オフラインモードは、主にテストに使用されます。
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。
ノード	

オプション	説明
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。
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	シスコの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。

オプション	説明
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。

XTC を使用した BGP-LS トポロジ収集

BGP-LS XTC トポロジ（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イス/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、追加の収集用の送信元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 のみで送信元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、**/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 [追加（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 （9 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 （オプション）収集から個々のノードを除外するには、[node-blacklist] タブをクリックし、ノードの該当する IP アドレスを入力します。たとえば、収集内の XTC ノードを表示したくない場合があります。

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送信元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	

オプション	説明
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。
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	シスコの 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。

オプション	説明
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、送信元 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。

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。最初のアグリゲータ構成が完了するまで、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。

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 **/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/dare:aggregators** に移動し、[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 ドロップダウンの接続先リストから、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 送信元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 を追加します。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、繰り返します。

ステップ 7 [OK] をクリックします。

ステップ 8 （オプション）送信元が送信元リストに追加または送信元リストから削除されたとき、または結果のアグリゲータで更新が呼び出されたときに、ルールが自動的に生成されます（送信元 NIMO によって報告された機能に依存します）。default 以外のルールセットを選択するには、**/wae:wae/components/dare:aggregators/aggregator/<*network_name*>/aggregator** に戻り、[rule-set] ドロップダウンリストでオプションを選択します。ルールセットを編集するには、**/wae:wae/components/dare:rule-sets/rule-set** に移動し、[default] または [full] のいずれかを選択します。

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 送信元 NIMO を実行します。最終ネットワークモデルが送信元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 CLI 構成例 （12 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>
# commit
```

アグリゲータが構成されたら、送信元 NIMO を実行します。

アグリゲータとマルチレイヤ収集の CLI 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 を構成する方法の詳細については、IGP トポロジ収集 （5 ページ）およびマルチレイヤ収集の構成を参照してください。

```
# show running-config networks network nimo
```

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：

```
    nimo topo-igp-nimo network-access cisco:access
    nimo topo-igp-nimo seed-router 10.225.121.60
    nimo topo-igp-nimo igp-protocol isis
    nimo topo-igp-nimo collect-interfaces true
    nimo topo-igp-nimo node-blacklist 10.11.255.12
    !
    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
# 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
```

収集が完了すると、ステータスメッセージが表示されます。完了したら、ノードが設定されていることを確認します。

```
# show running-config networks network l3-network model nodes node
```

最終モデルにも L3 情報が入力されていることを確認することもできます。

```
# show running-config networks network l1-l3-final-model model nodes node
```

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
```

最終モデルに L1 情報が入力されていることも確認できます。

```
# show running-config networks network l1-l3-final-model model nodes node
```

WAE Design を開いて、最終ネットワークモデルを表示することもできます（**[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ます）。

自律システム（AS）モデルの統合

as-merger NIMO は、AS モデル間で共有されるインターフェイス、回路などを解決して、単一の統合ネットワークモデルを作成します。

始める前に

- 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。

（注） BGP 収集は、すべての AS モデルの一部である必要があります。

- 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。詳細については、「XTC を使用した BGP-LS トポロジ収集 （8 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、**/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、統合 AS ネットワークモデルの名前を入力します。
ステップ 3 [追加（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、[as-merger-nimo] を選択します。
ステップ 6 [as-merger-nimo] をクリックします。
ステップ 7 [source] から、マージする個々の AS モデルを追加します。
ステップ 8 [subtrees] から、値として **model** を入力します。
ステップ 9 [generate capabilities] から、次のいずれかを選択します。

- [false]：AS 送信元が DARE ネットワークの場合は、このオプションを選択します。
- [true]：AS 送信元が DARE ネットワークではなく、他の NIMO によって作成されている場合は、このオプションを選択します。

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 [merge] 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。

```
# config
# networks network <as-merger-name> nimo as-merger sources [ <AS1-source> <AS2-sourse>
<ASn-source> ]
# networks network <as-merger-name> nimo as-merger subtrees [ model ]
# networks network <as-merger-name> nimo as-merger generate-capabilities <flag>
# commit
```

次に例を示します。

```
# config
# networks network AS100AS200 nimo as-merger sources [ AS6100 AS6200 ]
# networks network AS100AS200 nimo as-merger subtrees [ model ]
# networks network AS100AS200 nimo as-merger generate-capabilities <false>
# 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、**/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 [追加（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。

NSO NED を使用した LSP 収集

lsp-config-nimo は、NED を使用して LSP 構成を検出します。この収集を使用すると、ネットワーク内の LSP、名前付きパス、およびセグメントリストへの変更を展開できます。セグメントリストと LSP パスの作成の構成例については、セグメントルーティング LSP の作成 （17 ページ）を参照してください。

始める前に

- システムおよびベンダーのネットワーク要素ドライバ（NED）にNetwork Services Orchestrator（NSO）がインストールされている必要があります。シスコの担当者に連絡して、適切なNED を入手してください。
- NSO インスタンスを使用した WAE LSA 構成が構成されていることを確認します。詳細については、LSA パッケージのインストールを参照してください。
- 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。

ステップ 1 適切な LSP NIMO 構成ファイルを WAE インストールディレクトリから WAE ランタイム パッケージディレクトリにコピーします。

a) Unix シェルから、***<wae_installation_directory>*/packages/cisco-wae-lsp-config-nimo** ディレクトリに移動します。

b) 適切な LSP NIMO パッケージ（たとえば、**cisco-wae-lsp-config-nimo-rfs**）ディレクトリを <*wae_run_time_directory*>/packages にコピーします。

ステップ 2 適切な NED パッケージを <*wae_run_time_directory*>/packages にコピーします。

ステップ 3 エキスパート モードから、**/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。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_config などです。

ステップ 5 [追加（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。
- 詳細オプションについては、LSP 構成の詳細オプション （16 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 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。

LSP 構成の詳細オプション

このトピックでは、lsp-config-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
属性	
action-timeout	すべての lsp-config-nimo アクションに使用するタイムアウト（分単位）。 デフォルト値は 0 です。大規模なネットワークでは、より大きな値が必要になる場合があります。
create-nodes	device-sync アクションがノードを作成するかどうか。 デフォルト値は「false」です。
perform-sync-from	デフォルトでは、run-collection アクションと device-sync アクションは、最初に device sync-from を実行します。これには、デバイスごとに数秒かかります。大規模なネットワークでは、この値を「false」に設定し、run-collection を実行する前に単一の device sync-from を実行する方が効率的です。 デフォルト値は「false」です。
preserve-device-config	run-collection がデバイスモデルへの変更をコミットするかどうか、および「re-deploy reconcile」がどのように実行されるか。「true」に設定されている場合、調整は「keep-non-service-config」オプションで行われるため、収集はサービスモデルで表されていないデバイスモデル部分も保持します。 デフォルトのオプションは「true」です。
アクション（**Actions**）	
copy-topology	送信元ネットワークのみをこのネットワークにコピーします。 **警告** このアクションにより、すべての LSP が削除されます。
device-sync	トポロジネットワーク内の LSP を検出して調整します。送信元ネットワークはコピーされません。
調整	トポロジネットワーク内の LSP をデバイスモデルと調整します。

セグメントルーティング LSP の作成

次の手順では、セグメントルーティング LSP を作成するコマンドについて説明します。

ステップ 1 WAE ランタイムディレクトリから WAE CLI を起動し、構成モードに入ります。

```
waerun# wae_cli -C
wae@wae# config
wae@wae$#
```

ステップ 2 2 つのホップを持つセグメントリストを作成します。

```
wae@wae(config)# networks network <network_name> model nodes node <node_name>
 segment-lists segment-list <segment_list_name>
```

最初のホップはノードホップです。

```
wae@wae(config-segment-list-<segment_list_name>)# hops hop 1 node node-name <node_name>
wae@wae(config-hop-1)# exit
```

2 番目のホップはインターフェイスホップです。

```
wae@wae(config-segment-list-<segment_list_name>)# hops hop 2 interface node-name <node_name>
interface-name <interface_name>
wae@wae(config-hop-2)# exit
wae@wae(config-segment-list-<segment_list_name>)# exit
```

ステップ 3 2 つの LSP パスを持つ LSP を作成します。

```
wae@wae(config)# networks network <network_name> model nodes node <node_name>
 segment-lists segment-list <segment_list_name>
```

最初のパスは PCE 委任パスです。

```
wae@wae(config-node-<node_name>)# lsps lsp <lsp_name> destination <destination> lsp-paths lsp-path
 1 type segment-routing pce-delegated true
wae@wae(config-lsp-path-1)# exit
```

2 番目のパスは、先ほど作成したセグメントリストを使用します。

```
wae@wae(config-lsp-<lsp_name>)# lsp-paths lsp-path 2 type segment-routing segment-list <segment_list>
wae@wae(config)# commit
```

例

次に例を示します。

```
wae@wae(config)# networks network <network_name> model nodes node <node_name> segment-lists
 segment-list <segment_list_name>
wae@wae(config-segment-list-<segment_list_name>)# hops hop 1 node node-name <node_name>
wae@wae(config-hop-1)# exit
wae@wae(config-segment-list-<segment_list_name>)# hops hop 2 interface node-name
<node_name> interface-name <interface_name>
wae@wae(config-hop-2)# exit
wae@wae(config-segment-list-<segment_list_name>)# exit
wae@wae(config-node-<node_name>)# lsps lsp <lsp_name> destination <destination> lsp-paths
 lsp-path 1 type segment-routing pce-delegated true
wae@wae(config-lsp-path-1)# exit
wae@wae(config-lsp-<lsp_name>)# lsp-paths lsp-path 2 type segment-routing segment-list
<segment_list>
wae@wae(config)# commit
```

XTC を使用した PCEP LSP 収集

XTC（lsp-pcep-xtc-nimo）を使用した PCEP LSP 検出は、bgpls-xtc-nimo から収集されたデータを使用し、PCEP 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。PCEP 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 BGP-LS トポロジ収集 （8 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、**/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 [追加（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、以前に構成された 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 [コミット（Commit）] ボタンをクリックします。

ステップ 10 **[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 11 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 12 [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 13 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 14 [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。

LAG ポートと LMP インターフェイス収集

port-cfg-parse NIMO は、ネットワーク内のルータ構成から LAG ポートとリンク管理プロトコル（LMP）インターフェイスを検出します。この NIMO は、マルチレイヤ収集に使用されます。

（注） WAE UI を使用してこの収集を構成することはできません。

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、「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、**/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 [追加（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 NIMO タイプとして [port-cfg-parse-nimo] を選択します。

ステップ 6 [port-cfg-parse-nimo] 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。
- [cfg-parse-agent]：構成解析エージェントを選択します。

（注） 詳細オプションについては、ポート構成解析の詳細オプション （21 ページ）を参照してください。

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。

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。

ポート構成解析の詳細オプション

このトピックでは、port-cfg-parse NIMO を作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
lag	ポートメンバーの LAG 検出を有効にします。
lmp	LMP インターフェイスの検出を有効にします。

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）を送信元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、**/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_topo_bgp などです。

ステップ 3 [追加（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 （22 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、該当する IPv4 および IPv6 アドレスを入力します。

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。

オプション	説明
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。
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。

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。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_config などです。

ステップ 3 [追加（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。

セグメントルーティング LSP トラフィック収集

セグメントルーティング（SR）LSP トラフィック収集（sr-traffic-matrix-nimo）は、SR LSP 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 トポロジ収集 （5 ページ）またはXTC を使用した BGP-LS トポロジ収集 （8 ページ）を参照してください。
- テレメトリはルータで構成する必要があります。

（注） WAE UI を使用してこの収集を構成することはできません。

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。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_sr_traffic_matrix などです。

ステップ 3 [追加（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に入ります。

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>

# commit
```

継続的な収集

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 （15 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 （23 ページ） を参照してください。

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、**/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_traffic_polling などです。

ステップ 3 [追加（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。

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 （27 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、送信元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 （27 ページ）を参照してください。

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 [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。

トラフィックポーリングの詳細オプション

このトピックでは、継続的な収集（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	
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。
verbosity	ポーラーのロギングレベルを設定します。デフォルト値は 40 です。 • 40：情報 • 50：デバッグ • 60：トレース
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。
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。
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れます）、WMD は RPC API からトラフィック統計をプルします。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。

トラフィックポーリングの調整

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. 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
```

2. 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。
3. 次のテキストを検索してフィルタ処理します。
   - Interface Traffic Poller: Collection complete. Duration:
   - LSP Traffic Poller: Collection complete. Duration:
4. 平均の期間とワーストケースシナリオでの期間に注意してください。

たとえば、インターフェイスポーリングには、平均で 30 秒、ワーストケースシナリオでは 45 秒かかります。LSP ポーリングには、平均で 90 秒、最大で 120 秒かかります。インターフェイスポーリングを効率的に実行するには、ワーストケースの（より大きい）の数値から始めることをお勧めします。この例では、インターフェイスポーラーを 45 秒ごとに開始し、LSP ポーラーを 120 秒ごとに開始します。その後、時間が経過するにつれて、必要に応じてこれらの数値を調整できます。

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（2 回のポーリング実行）が必要です。カウンタは、スライディングウィンドウを使用してメモリに保存されているものから選択されます。詳細な `raw-counter-ttl` オプションは、カウンタを保持する期間を指定します。詳細な `stats-computing-minimum-window-length` および `stats-computing-maximum-window-length` オプションは、スライディングウィンドウのサイズを指定します。これらのパラメータを構成する方法に関する基本的な計算は次のとおりです。

```
stats-computing-minimum-window-length = ( poller duration ) * 5
stats-computing-maximum-window-length = stats-computing-minimum-window-length * 1.5
raw-counter-ttl = stats-computing-minimum-window-length * 3 / 60
```

この例では、LSP 収集にはインターフェイス収集よりも時間がかかるため、LSP のワーストケース期間（120 秒）を使用して、スライディングウィンドウのサイズを決定します。次の数値が得られます。

```
stats-computing-minimum-window-length = 120*5 = 600
stats-computing-maximum-window-length = stats-computing-minimum-window-length * 1.5 =
600 * 1.5 = 900
raw-counter-ttl = stats-computing-minimum-window-length * 3 / 60 = 30
```

トラフィック計算は、30 ～ 45 秒ごとに実行することも、詳細な `scheduler-interval` オプションで構成されるように 120 秒ごとに実行することもできます。CPU を節約するには、120 秒に設定します。大規模なネットワークではトラフィックの計算に時間がかかる場合があるため、適切な **`connect-timeout`** オプションを設定してください。実際の期間は、`logs/ncs-java-vm.log` で確認できます。次に例を示します。

```
Traffic calculation took (ms) 3750
```

これらのパラメータは、積極的にも保守的にも調整できます。より保守的な設定から始めて、必要に応じて調整することをお勧めします。出力からトラフィックがドロップされていることがわかった場合は、それに応じて `stats-computing-maximum-window-length` および `raw-counter-ttl` オプションを増やすことができます。

ネットワークモデルの可視化

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 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のみされます。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

- 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） WAE UI を使用してこの収集を構成することはできません。

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。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 [追加（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、送信元ネットワークを入力します。

（注） 送信元ネットワークには、既存のレイアウトテンプレートを割り当てることはできません。

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

ステップ 9 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 10 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 11 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「Using Layouts」を参照してください。

ステップ 12 パッチを作成して送信します（**[ツール（Tools）]** > **[パッチ（Patches）]** > **[作成（Create）]**）。Cisco WAE Design ユーザー ガイドの章「Patch Files」を参照してください。

ステップ 13 エキスパート モードから、layout-nimo ネットワークモデル（networkABC_layout）に戻ります。

ステップ 14 [layouts] タブをクリックします。

ステップ 15 [layout] をクリックして、テーブルにレイアウトデータが入力されたことを確認します。次回 WAE Design からプランファイルを開くと、保存されたレイアウトプロパティとともにトポロジが表示されます。

デマンド推論

トラフィックは、インターフェイス、インターフェイスキュー、およびLSP で測定できます。デマンド推論を使用し、これらの測定値のいずれかに基づいてデマンドトラフィックを見積もることができます。デマンド推論の詳細については、Cisco WAE Design ユーザー ガイドを参照してください。demand-deduction-nimo は、測定された SNMP トラフィックを使用し、エンドツーエンドのデマンドのデマンドメッシュビルド（トラフィックマトリックス）をネットワークモデルに適用することにより、デマンド推論を実行します。

始める前に

デマンド（networkABC_demandmesh など）とインターフェイス測定（networkABC_traffic_poller など）を備えた統合ネットワークモデル（アグリゲータ NIMO）が存在する必要があります。

（注） WAE UI を使用してこの NIMO を構成することはできません。

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。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_demand_deduction などです。

ステップ 3 [追加（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、[demand-deduction-nimo] を選択します。

ステップ 6 [demand-deduction-nimo] をクリックして、送信元ネットワークを入力します。送信元ネットワークは通常、デマンドとトラフィックポーラー情報を含む統合ネットワークモデル（アグリゲータ NIMO）です。

ステップ 7 [input] タブをクリックします。

- [nodes]：ノードで測定されたトラフィックを使用します。デフォルト値は true です。
- [interfaces]：インターフェイスで測定されたトラフィックを使用します。デフォルト値は true です。
- [lsps]：LSP で測定されたトラフィックを使用します。
- [remove-zero-bw-demands]：トラフィックのない（ゼロの）（または `zero-bw-tolerance` オプション未満の）デマンドをすべて削除します。デフォルトは true です。
- [zero-bw-demands-tolerance]：ゼロトラフィックと見なされる許容値（ゼロ未満）を入力します。
- [demand-upper-bound]：デマンドトラフィックレベルの上限を入力します。上限に達すると、警告が発行されます。デフォルトは 10,000 Mb/s です。

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 [demand-deduction-nimo] タブに戻り、**[run]** > **[runの呼び出し（Invoke run）]** をクリックします。

ステップ 10 デマンド推論が成功したことを確認するには、**/wae:networks/network/<*network_model*>/model/demands** に移動し、[traffic] 列が入力されているかどうかを確認します。

デマンドメッシュの作成

デマンドメッシュは、ネットワークのすべてまたは一部に多数のデマンドを作成するための時間効率のよい方法です。demandmesh-creator-nimo は、一連の送信元ノードと接続先ノードの間にデマンドメッシュを作成します。

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） WAE UI を使用してこの NIMO を構成することはできません。

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。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_demandmesh などです。

ステップ 3 [追加（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、[demandmesh-creator-nimo] を選択します。

ステップ 6 [demandmesh-creator-nimo] をクリックして、送信元ネットワークを入力します。

ステップ 7 [input] タブをクリックして、次のように入力します。

- [source-nodes]：ノードを入力して、デマンドメッシュソースを指定された一連のノード、外部非同期システム、または外部エンドポイントに制限します。**/wae:networks/network/<*network_model*>/model/nodes** からノードを表示できます。
- [both-directories]：接続先から送信元へ、および送信元から接続先へのすべてのデマンドを含みます。デフォルトは true です。
- [service-class]：作成されたデマンドにサービスクラスタイプを割り当てます（たとえば、QoS）。空の場合、デフォルトのクラスが使用されます。
- [topology]：トポロジを入力します。デフォルトでは、デマンドはすべてのトポロジに適用されます。
- [choice-destination: destination-nodes]：送信元として選択されたもの以外の接続先へのデマンドを作成する場合は、このオプションを選択します。
- [choice-destination: destination-equal-source]：[true] に設定すると、接続先ノードは送信元に設定されます。デフォルトは true です。

（注） 完全なデマンドメッシュを取得するには、どのフィールドも編集せず、[destination-equal-source] を [true] に設定します。これにより、ネットワーク内のすべてのノードが選択され、考えられるすべてのデマンド（ノード自体へのデマンドを含む）が作成されます。

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 [demandmesh-creator-nimo] タブに戻り、**[run] > [runの呼び出し（Invoke run）]** をクリックします。

ステップ 10 デマンドが作成されたことを確認するには、**/wae:networks/network/<*network_model*>/model/demands** に移動します。テーブルには、デマンド情報が取り込まれています。

ネットワークモデルに対する外部スクリプトの実行

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 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、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。

始める前に

送信元ネットワークモデルとカスタムスクリプトが必要です。

（注） WAE UI を使用してこの NIMO を構成することはできません。

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_my_script 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]：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 7.0 で、
- [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。送信元ネットワークモデルとして $$input を入力し、得られるネットワークモデルとして $$output を入力します（スクリプトの実行後）。$$input、$$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 （33 ページ）を参照してください。

ステップ 7 [external-executable-nimo] タブで、[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 [ <arg_1> <arg_2> <arg_x> $$input $$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
```

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。

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/python /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.
```

スクリプトが成功したことを確認します。

```
admin@wae# show running-config networks network net_dest model nodes node cr1.atl
interfaces interface to_cr1.hst description
networks network net_dest
 model nodes node cr1.atl
  interfaces interface to_cr1.hst
   description "My IGP metric is 37"
  !
```

!
!

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。