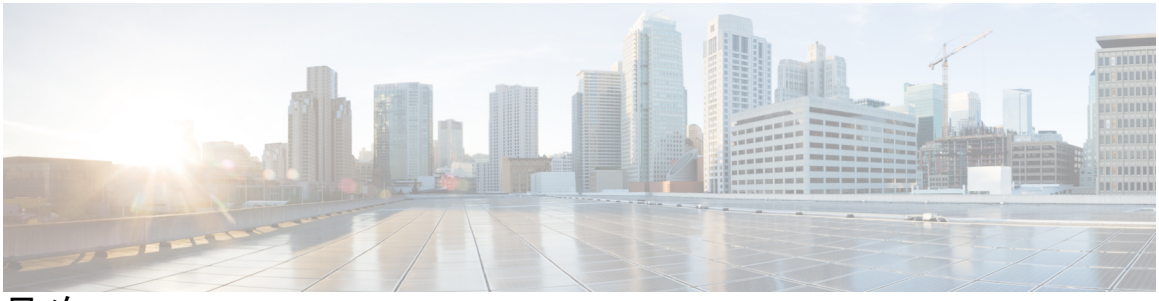
Cisco WAE 7.1 ユーザーガイド

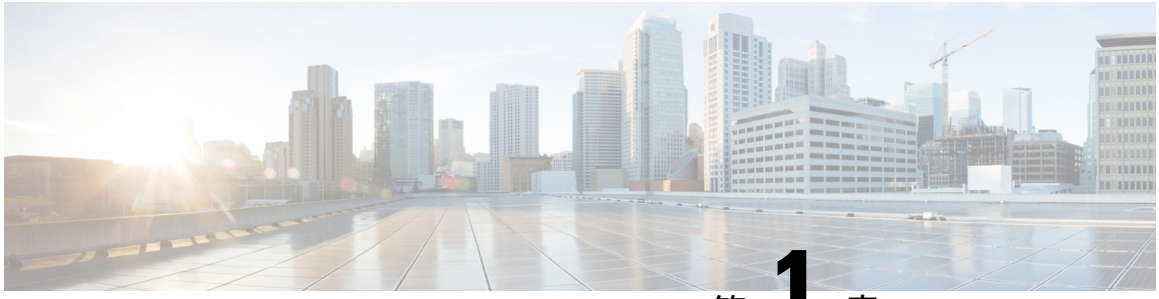
初版：2018 年 1 月 18 日

最終更新：2018 年 4 月 17 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/



目 次



第 1 章

概要

ここでは、次の内容について説明します。

Cisco WAE の概要

Cisco WAN Automation Engine（WAE）のプラットフォームは、ソフトウェアモジュールを相互接続し、ネットワークと通信し、外部アプリケーションとインターフェイスする API を提供するオープンでプログラマブルなフレームワークです。

Cisco WAE は、ネットワークとそのネットワーク上のトラフィック需要の継続的なモニタリングと分析を通じて、現在のネットワークのモデルを作成および維持するためのツールを提供します。このネットワーク モデルには、トポロジ、設定、トラフィック情報など、特定の時点でのネットワークに関するすべての関連情報が含まれています。この情報は、トラフィック要求、パス、ノードとリンクの障害、ネットワークの最適化、またはその他の変更によるネットワークへの影響を分析するための基礎として使用できます。

Cisco WAE プラットフォームには、次のような数多くのユースケースがあります。

- トラフィック エンジニアリングとネットワークの最適化：TE LSP 構成を計算してネットワークパフォーマンスを改善したり、ローカルまたはグローバルな最適化を実行したりします。
- デマンドエンジニアリング：ネットワーク上のトラフィック需要の追加、削除、または変更がネットワーク トラフィック フローに与える影響を調べます。
- トポロジと予測分析：設計またはネットワーク障害によって引き起こされるネットワークトポロジの変更がネットワークパフォーマンスに与える影響を観察します。
- TE トンネルプログラミング：トンネルパスや予約帯域幅などのトンネルパラメータを変更した場合の影響を調べます。

- サービスクラス（CoS）対応のオンデマンド帯域幅：既存のネットワークトラフィックと需要を調べ、ルータ間で一連のサービスクラス固有の需要を許可します。

WAE アーキテクチャ

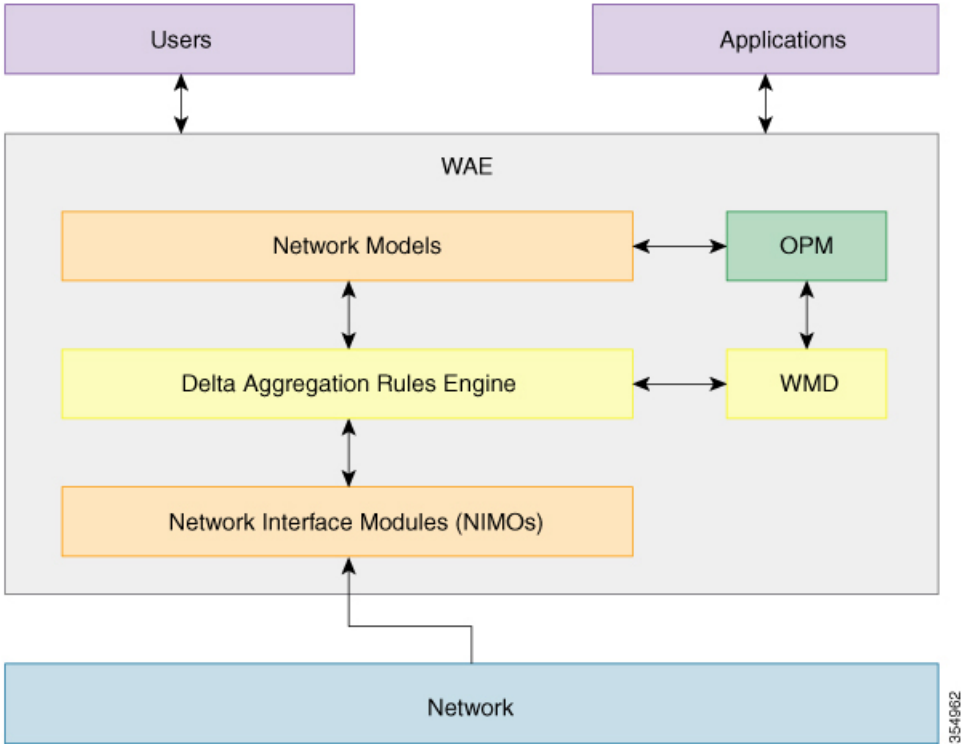
本質的に、WAE は抽象的なネットワークモデルを定義します。このモデルは、ネットワークインターフェイスモジュール（NIMO）をつなぎ合わせることによって実際のネットワークから構築できます。

WAE ネットワークモデルは YANG で定義され、標準の YANG メカニズムを介して拡張できます。WAE 自体は、YANG モデルから API（NETCONF、RESTConf、CLI）を自動的に生成する YANG ランタイムシステムの上に実装されます。

最適化および予測モジュール（OPM）は、ネットワークモデルを操作するための強力な Python API を提供します。OPM API を使用すると、デバイス固有のプロパティを気にすることなくネットワーク上で操作できます。基になるルータが別のベンダーのルータに置き換えられても、API 呼び出しはまったく同じままです。

OPM API は、強力な「仮定の問題評価」機能を提供します。たとえば、OPM API を使用すると、次の質問に答えることができます。

- メンテナンスのためにこのルータを停止すると、どのような影響がありますか。
- この回路のキャパシティを増やすとどうなりますか。
- 私のネットワークは現在、データセンターのバックアップを処理できますか。



最適化および予測モジュール

最適化および予測モジュール（OPM）は、ネットワークモデルを操作するための強力なPython API を提供します。OPM API を使用すると、デバイス固有のプロパティを気にすることなくネットワーク上で操作できます。基になるルータが別のベンダーのルータに置き換えられても、API 呼び出しはまったく同じままです。

OPM API は、強力な**「仮定の問題評価」**機能を提供します。たとえば、OPM API を使用すると、次の質問に答えることができます。

- メンテナンスのためにこのルータを停止すると、どのような影響がありますか。
- 特定の回路のキャパシティを増やすとどうなりますか。
- 私のネットワークは現在、データセンターのバックアップを処理できますか。

ネットワーク インターフェイス モジュール

ネットワークインターフェイスモジュール（NIMO）は、抽象ネットワークモデルの一部を設定する WAE パッケージであり、そのためにネットワークにクエリを実行する可能性があります。ほとんどの NIMO は次のように動作します。

1. 送信元ネットワークモデル（または単に送信元モデル）を読み取ります。
2. 実際のネットワークから取得した情報で送信元モデルを拡張します。

3. 結果のモデルを使用して接続先ネットワークモデル（または単に接続先モデル）を生成します。

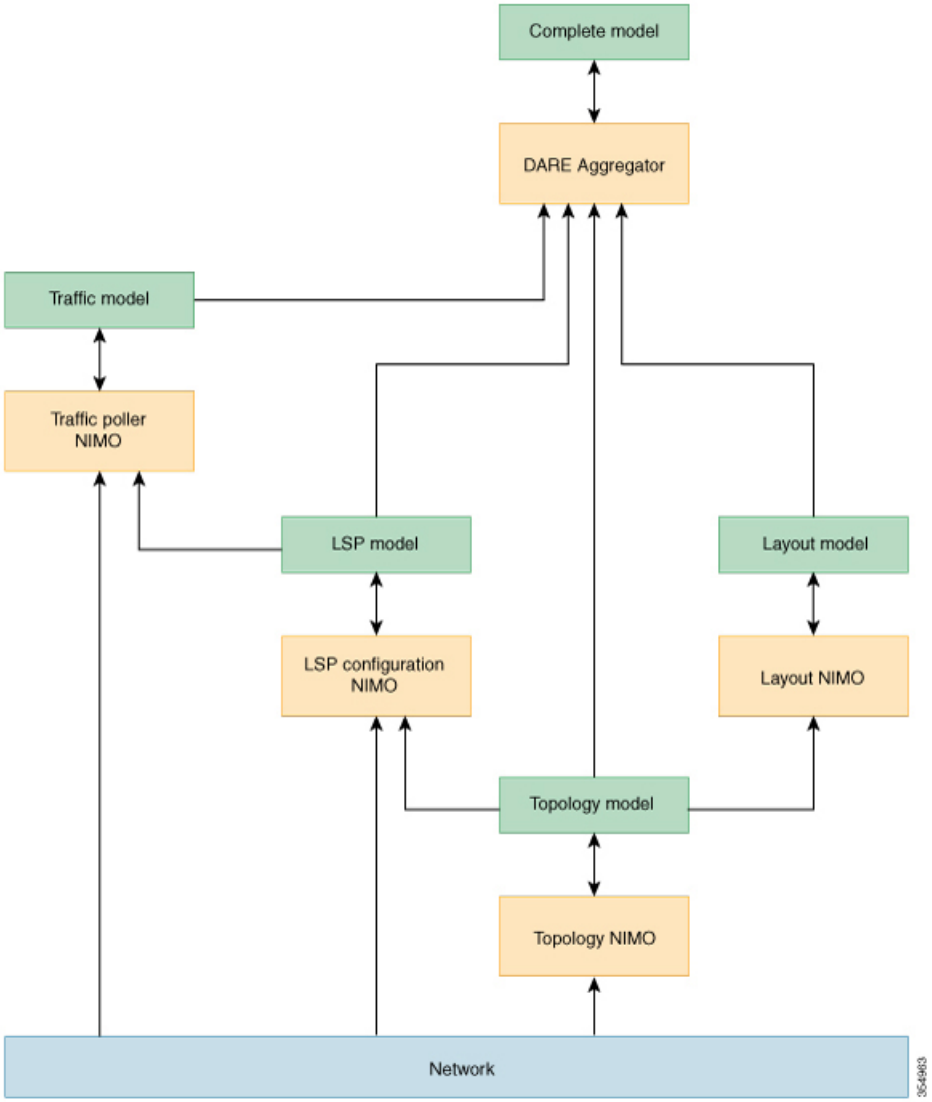
WAE には、次のようないくつかの異なる NIMO が含まれています。

- トポロジ NIMO：SNMP クエリによって拡張済みの検出された IGP データベースに基づいて、トポロジ情報（ノード、インターフェイス、回路）を基本的なネットワークモデルに入力します。トポロジ NIMO には送信元モデルがありません。
- LSP 構成 NIMO：LSP 情報で送信元モデルを拡張し、追加情報で接続先モデルを生成します。
- トラフィックポーラーNIMO：ネットワークからポーリングされたトラフィック統計で送信元モデルを拡張し、追加情報で新しい接続先モデルを生成します。
- レイアウトNIMO：送信元モデルにレイアウトプロパティを追加して、可視化を改善します。追加のレイアウト情報を使用して、新しい接続先モデルを生成します。NIMOはレイアウトプロパティの変更を記録するため、送信元モデルが変更され、接続先モデルが更新されると、それに応じて接続先モデルのレイアウトプロパティが更新されます。

ネットワークモデル

モデル構築チェーンは、必要な情報を備えたネットワークモデルを生成するように編成されたNIMOの配置です。たとえば、前述のNIMOの場合、1つのチェーンの構成でトポロジNIMO、その後に LSP 構成 NIMO、さらにトラフィックポーラー NIMO が続きます。このチェーンには3つのモデルが含まれ、一部のモデルは他のモデルよりも多くの情報を持っています。モデル構築チェーンを編成すると、さまざまなユースケースに合わせてさまざまなモデルを作成できます。チェーンを分岐して、独立したモデル構築トラックを持つことができます。

次の図は、DARE アグリゲータによって結び付けられたチェーンを示しています。

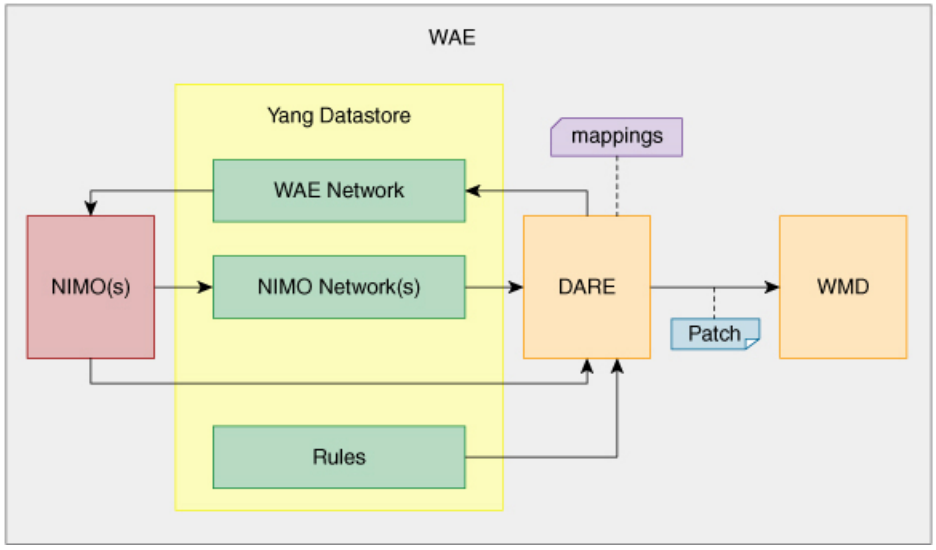


前の図に示すように、モデル構築チェーンは分岐でき、独立した並列モデル構築タスクを可能にします。その結果、ブランチごとに異なるモデル情報が含まれます。前の例では、1 つのブランチがLSPとトラフィック測定で終了し、もう一方のブランチがより可視化できるモデルで修了します。

DAREアグリゲータは、さまざまなモデル構築チェーンブランチを1つにまとめ、それぞれからモデル情報を選択し、その情報を接続先モデルに統合するWAEコンポーネントです。前の例は、チェーン内のすべてのモデルを調べるように構成されています。示されている構成では、アグリゲータはトポロジモデルへの変更を即座にピックアップします。接続がない場合、変更は、最上位モデルに処理される前に、モデル構築チェーンブランチに伝播する必要があります。アグリゲータは、接続先モデルへの変更を正しいダウンストリームNIMOにルーティングします。前の例では、最上位モデルでLSPを作成すると、アグリゲータはその変更をLSPモデルに転送します。

デルタ集約ルールエンジン

デルタ集約ルールエンジン（DARE）は、さまざまなモデル構築チェーンブランチを 1 つにまとめ、それぞれからモデル情報を選択し、その情報を接続先モデル（最終ネットワークモデル）に統合する WAE コンポーネントです。NIMO モデルへの変更を収集するために、通知に登録し、NIMO が変更を直接公開するための API を提供します。これらの変更は、構成されたルールに基づいて集約されます。さらに、変更は WAE モデルデーモン（WMD）にパッチの形式で同時に送信されます。DARE は、集約に必要な独自の状態をファイルシステム上のさまざまなマップに保存します。



（注） DARE は変更を前提に機能しているため、NIMO モデルに変更を加える前に構成する必要があります。

DARE を使用するようにアグリゲータを構成する方法については、NIMO 収集の統合 （61 ページ）を参照してください。

WAE モデリングデーモン（WMD）

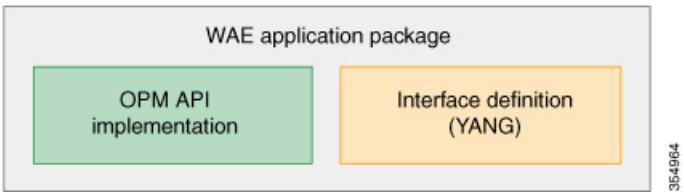
WMD は、DARE から変更を受け取り、スケジュールされた NIMO の実行と、XTC エージェントからパッチモジュールへの事後対応の更新を組み込みます。また、測定されたトラフィック更新をトラフィックポーラー NIMO からインメモリモデルに挿入するようスケジュールします。すべての更新は、ネットワークのほぼリアルタイムのマスターモデルに統合されます。WAE アプリケーション（次のセクションで説明）は、WMD に接続し、このほぼリアルタイムモデルのコピーにアクセスして、WAE OPM API 機能を利用することができます。

WMD の構成方法については、WAE モデリングデーモン（WMD）の構成 （85 ページ）を参照してください。

WAE アプリケーション

WAE は、柔軟で強力なアプリケーション開発インフラストラクチャを提供します。単純な WAE アプリケーションは、次のもので構成されます。

- アプリケーション インターフェイス。YANG モデルで定義されています。通常、このインターフェイスにはRPC とデータモデルが含まれます。YANG モデルは、必要に応じて、WAE ネットワークモデルを拡張して、新しいデータタイプを追加できます。
- アプリケーションロジック。OPM API を使用して実装されます。



WAE は YANG 定義から API を自動的に生成するため、WAE アプリケーションには自動的に公開された API があります。WAE アプリケーションは、ある意味で、WAE 機能のシームレスな拡張です。

オンデマンド帯域幅アプリケーション

Bandwidth on Demand（BWoD）アプリケーションは、WMD によって提供されるほぼリアルタイムのネットワーク モデルを利用して、XTC から WAE に委任された帯域幅制約を含む SR ポリシーのパスを計算して維持します。帯域幅制約を含む SR ポリシーで使用可能な最短パスを計算し、パスに輻輳がないことを確認するには、パス計算要素（PCE）によってネットワーク上のトラフィック負荷が認識される必要があります。WAE BWoD アプリケーションは、SR ポリシーの帯域幅認識パス計算の委任を新しい XTC REST API を介して副次的に WAE に委任できるようにすることで、XTC の既存のトポロジ対応PCE機能を拡張します。ユーザーは、ネットワーク使用率のしきい値（輻輳の定義）やパス最適化基準の設定などのアプリケーションオプションを選択して、BWoDアプリケーションの動作を微調整し、計算するパスに影響を与えることができます。

BWoDアプリケーションの構成方法については、オンデマンド帯域幅の構成ワークフロー（123 ページ）を参照してください。

Bandwidth Optimization アプリケーション

Bandwidth Optimization アプリケーションとは、ネットワーク トラフィックを管理するアプローチで、ネットワークで特定の成果を達成するために少数のLSP を展開することに重点を置いています。この種の戦術的なトラフィック エンジニアリングの例として、輻輳が発生しているリンクからトラフィックを移動する LSP の展開、優先度の高い音声またはビデオ トラフィック用の低遅延 LSP の確立、特定のノードまたはリンクを回避する LSP の展開などがあります。

WAE は、ネットワークの状態の変化に対応してトラフィックを管理する Bandwidth Optimization アプリケーションを提供します。

帯域幅最適化アプリケーションの構成方法については、Bandwidth Optimizationアプリケーション ワークフロー （129 ページ）を参照してください。

Cisco WAE インターフェイス

Cisco WAE には、ネットワーク デルの構成に使用できる 3 つのインターフェイスがあります。

WAE UI

WAE UI は、ネットワークのモデル構築チェーンを作成する複雑さを隠す、使いやすいインターフェイスを提供します。WAE UI は、複数のデータ収集の構成を 1 つのネットワークの下でまとめ、統合されたデータを含む単一のプランファイルを生成できます。ただし、WAE UI では実行できない特定の操作があります。WAE エキスパートモードまたは CLI を使用して実行された構成は、WAE UI 構成画面に表示されないことがあります。ネットワークモデルの構成：WAE UI （11 ページ）および特記事項 （12 ページ）を参照してください。

エキスパート モード

エキスパート モードは、WAE UI では利用できない可能性のある追加のデバイスおよびサービス機能を備えた YANG モデルブラウザです。また、各操作のすべてのオプションがエキスパート モードに表示されるため、WAE CLI を介してエキスパート モードを使用することもできます。ネットワークモデルの構成：エキスパートモード （21 ページ） を参照してください。

WAE CLI

WAE CLI は、ユーザーがコマンドを入力してビジュアルプロンプトに応答するインターフェイスです。システム応答が返されます。これは、すべての WAE 構成に必要な最低限のインターフェイスです。エキスパート モードで使用できる操作は、CLI でも使用できます。ネットワークモデルの構成：WAE CLI （31 ページ） を参照してください。

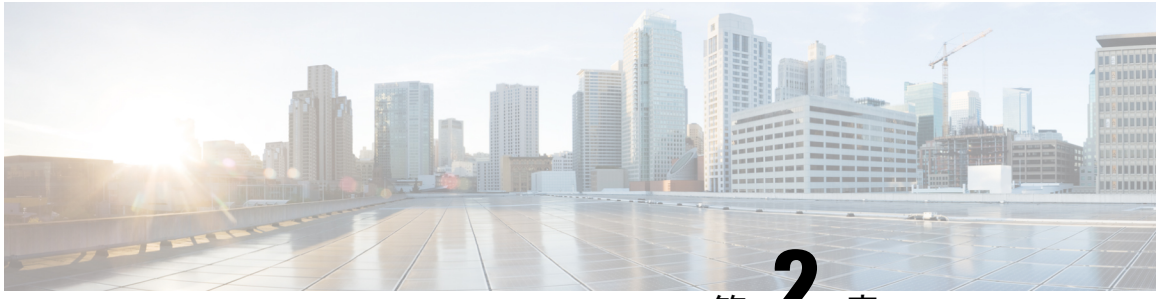
ネットワークモデル作成ワークフロー

以下は、個々のネットワークモデルを構成する方法に関するワークフローの概要です。詳細な手順は、使用するインターフェイスのタイプ（エキスパート モード、WAE UI、または WAE CLI）によって異なります。

複数の NIMO を実行して情報を 1 つの最終的なネットワークに統合することを計画している場合は、アグリゲータ NIMO を設定するまで収集を実行しないでください。詳細については、「NIMO 収集の統合 （61 ページ）」を参照してください。

1. デバイス認証グループ、SNMP グループ、およびネットワーク プロファイル アクセスを構成します。

2. （オプション）エージェントを構成します。この手順は、XTC、LAG およびポートインターフェイス、またはマルチレイヤ情報を収集する場合にのみ必要です。
3. 基本的なトポロジ収集を使用してネットワークを構成します。
4. 収集を実行します。
5. 追加のネットワーク収集を構成します。
6. （オプション）収集をいつ実行するかをスケジュールします。
7. プランファイルが定期的に保存されるアーカイブファイルシステムの場所と間隔を構成します。
8. （オプション）WAE アプリケーションでプランファイルを表示します。



第 2 章

ネットワークモデルの構成：WAE UI

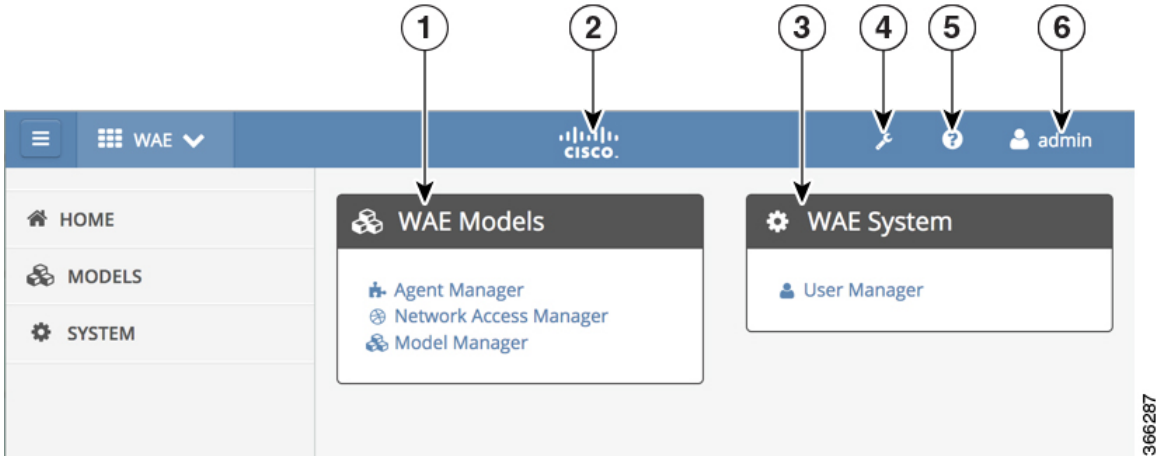
ここでは、次の内容について説明します。

WAE UI の概要

WAE UI には、デバイスとネットワークのアクセス、ネットワーク収集、ユーザー管理、および XTC エージェントのための使いやすい構成ツールが用意されています。モデルマネージャでは、ネットワークモデルを構成するときの複雑さがウィザードにより隠されます。ウィザードは、単一のネットワークに対する基本的なトポロジ収集、高度な収集、およびその他の NIMO 機能の構成を案内します。

基本的なネットワークモデル構成については、モデルマネージャで始めることをお勧めします。WAE UI を使用して実行できない特定の操作があります（テレメトリ、解析エージェント、一部のスケジューラジョブ、および LSP 変更の構成）。これらのタスクでは、エキスパート モードまたは WAE CLI に切り替えることができます。使用するインターフェイスに関係なく、最後にコミットされた構成が保存されます。

図 *1 : WAE UI*



引き出し線番号	名前	説明
1	WAE モデル	ネットワークモデルの構成情報が含まれています。 • エージェントマネージャ：XTC エージェントの構成。 • ネットワーク アクセス マネージャ：デバイスとネットワークのログイン情報の構成。 • モデルマネージャ：NIMO を使用したネットワークモデルの構成。
2	WAE ホーム（シスコのアイコン）	WAE ホームページが表示されます。
3	WAE システム	システムタスクおよび管理タスクが含まれます。
4	エキスパート モード	エキスパート モードを別のウィンドウで起動します。
5	ヘルプ	www.cisco.com/go/wae を開きます。サポートエリアに WAE のドキュメントがあります。
6	ユーザー	WAE にログインしている現在のユーザーを表示します。クリックすると、ユーザーをログアウトできます。

特記事項

WAE UI を使用する場合は、次の点に注意してください。

- ネットワークモデルがエキスパートモードまたは CLI を使用して作成された場合：
  - モデルマネージャを使用して、そのネットワークモデル内で NIMO を追加または削除しないでください。
  - モデルマネージャを使用して、そのネットワークモデル内で既存の NIMO の編集のみを行えます。
- モデルマネージャは、ネットワークモデル構築の構成を 2 つのフェーズに分割します。最初のフェーズでは、さまざまなネットワークトポロジ送信元（ノード、インターフェイス、LSP、VPN、BGP）を収集して統合します。このフェーズでは、<*network_model_name*>_topology（例：NetworkABC_topology）という命名規則で統合ネットワークモデル（アグリゲータ NIMO）が自動的に生成されます。2 番目のフェーズでは、追加データ（トラフィック統計、トラフィック需要、および視覚的レイアウト）で <*network_model_name*>_topology を補強します。このフェーズでは、<*network_model_name*>（例：NetworkABC）という命名規則で最終統合モデル（アグリゲータ NIMO）が自動的に生成されます。
- スケジューリングまたはアーカイブが WAE UI を使用して構成されている場合、モデルマネージャではネットワークモデルごとにスケジューラにタスクが自動的に作成されます。タスクには、WAE CLI またはエキスパート モードで次の命名規則があります。
  - _WAE_PLAN_ARCHIVE_<*network_model_name*>
  - _WAE_SCHEDULER_<*network_model_name*>
- WAE エキスパートモードまたは CLI を使用して実行されたスケジューラ構成は、WAE UI に表示されません。

WAE UI を使用したネットワークモデルの構成

このワークフローでは、WAE UI を使用してネットワークモデルを作成する手順の概要について説明します。

XTC を使用してネットワークの情報を収集している場合は、ネットワークモデルを構成する前に XTC エージェントを構成する必要があります。

ステップ	詳細
1. デバイスログイン情報（ネットワーク認証グループと SNMP グループ）を構成します。	ネットワーク アクセスの設定 （14 ページ）
2. ネットワークモデル名を作成し、プランアーカイブ設定を構成します。	ネットワークの作成 （15 ページ）
3. 基本的なトポロジ収集を構成します。	基本的なトポロジ収集の構成 （15 ページ）
4. 追加のデータ収集を構成します。	追加 NIMO の構成 （26 ページ）

ステップ	詳細
5. 収集を実行し、ネットワークの詳細を表示します。	ネットワークモデルの詳細の表示 （17 ページ）
6. （オプション）収集をいつ実行するかをスケジュールします。	ネットワーク収集のスケジュール （17 ページ）
7. （オプション）プランアーカイブを構成および表示します。	アーカイブの構成およびプランファイルの表示 （18 ページ）

ネットワーク アクセスの設定

このタスクでは、ネットワーク アクセス プロファイルを作成して、グローバルデバイスログイン情報を定義します。

始める前に

グローバル ネットワーク デバイスのログイン情報を把握します。

ステップ 1 WAE モデルから、[ネットワークアクセスマネージャ（Network Access Manager）] をクリックします。

ステップ 2 [+ネットワークアクセスの追加（+ Add Network Access）] をクリックします。

ステップ 3 グローバルデバイスログイン情報を入力します。

- [名前（Name）]：ネットワーク アクセス プロファイルの名前を入力します。
- [ログインタイプ（Login Type）]：使用するログインプロトコルを [SSH] または [Telnet] から選択します。SSH プロトコルはより安全です。Telnet プロトコルは、ユーザー名とパスワードを暗号化しません。
- **[ユーザー名（Username）]**
- **Password**
- **イネーブルパスワード（Enable Password）**
- [SNMPアクセスオプション（SNMP Access Options）]：[SNMPv2c] または [SNMPv3] のいずれかを選択します。

  [SNMPv2c] の場合、パスワードとして機能する SNMP RO コミュニティストリングを入力します。これは、ノードとシードルータの間で送信されるメッセージを認証するために使用されます。

  [SNMPv3] の場合、次のデフォルトのログイン情報を入力します。

  - [セキュリティレベル（Security Level）]：次のいずれかを選択します。
    - [noAuthNoPriv]：認証も暗号化も実行しないセキュリティレベル。このレベルは、SNMPv3 ではサポートされていません。
    - [authNoPriv]：認証は実行するが、暗号化を実行しないセキュリティレベル。
    - [authPriv]：認証と暗号化の両方を実行するセキュリティレベル。

- [認証プロトコルとパスワード（Authentication Protocol and Password）]：次のいずれかを選択します。
  - [md5]：HMAC-MD5-96 認証プロトコル
  - [sha]：HMAC-SHA-96 認証プロトコル
- [暗号化プロトコルとパスワード（Encryption Protocol and Password）]：priv オプションで、SNMP セキュリティ暗号化方式として、DES または 128 ビット AES 暗号化を選択できます。priv オプションと aes-128 トークンを併用すると、このプライバシー パスワードは 128 ビットの AES キー番号を生成するためのパスワードになります。AES priv パスワードは、8 文字以上の長さにできます。パスフレーズをクリア テキストで指定する場合、最大 64 文字を指定できます。ローカライズド キーを使用する場合は、最大 130 文字を指定できます。

ステップ 4 [保存（**Save**）] をクリックします。

次のタスク

モデルマネージャを使用してネットワークモデルを作成します。

ネットワークの作成

完全なネットワークモデルを構成する最初の手順は、topo-igp-nimo または topo-bgpls-xtc-nimo を使用するトポロジ収集で新しいネットワークを作成することです。

ステップ 1 モデルマネージャから、[新しいネットワークの追加（Add New Network）] をクリックし、次のように入力します。

a) [ネットワークモデル名（Network model name）]：この名前は保存後に変更できません。
b) （オプション）WAE がプランファイルを保存するディレクトリ。
c) （オプション）プランファイルをアーカイブする頻度。15 分から始めることをお勧めします。

ステップ 2 [保存（**Save**）] をクリックします。

次のタスク

グローバル デバイス クレデンシャルを設定します。ネットワーク アクセスの設定 （14 ページ） を参照してください。

基本的なトポロジ収集の構成

このタスクでは、追加のネットワーク収集の送信元ネットワークとなるトポロジ収集を構成します。最初の収集後、ノード IP アドレステーブルにデータが入力され、管理 IP アドレスを追

加できます。基本的なトポロジ収集の詳細については、基本的なトポロジ収集 （55 ページ）を参照してください。

始める前に

- 特定のネットワークの **[モデルマネージャ（Model Manager）]** > **[ネットワーク検出（Network Discovery）]** タブにいる必要があります。
- XTC（topo-bgpls-xtc-nimo）を実行しているネットワークでトポロジ収集を構成する場合、モデルマネージャを使用してネットワークモデルを作成する前に、XTC エージェントを構成して実行する必要があります。XTC エージェントの構成 （18 ページ） を参照してください。

ステップ 1 [Add Discovery] をクリックします。

ステップ 2 次のいずれかの NIMO をクリックします。

- [topo-igp-nimo]：IGP データベースを使用してトポロジ情報を収集します。IGP トポロジ収集 （55 ページ） を参照してください。
- [topo-bgpls-xtc-nimo]：XTC を実行しているネットワークからトポロジ情報を収集します。XTC を使用した BGP-LS トポロジ収集 （58 ページ）を参照してください。

ステップ 3 **[保存（Save）]** をクリックします。

ステップ 4 [NIMOの実行（Run NIMO）] をクリックします。ウィンドウの右上隅にステータスが表示されることに注意してください。状態が [実行中（Running）] から [実行していない（Not Running）] になったら収集完了です。

ステップ 5 構成した NIMO の横にあるテーブルアイコンをクリックします。ノードの IP アドレスが入力されたテーブルが表示されます。

ステップ 6 テーブルに管理 IP アドレスを直接追加し、[保存（Save）] をクリックします。

（注） いつでもテーブルをクリックして、更新されたノードリストを表示できます。

次のタスク

追加の収集を構成します。

追加のネットワーク収集の構成

このトピックでは、追加の NIMO を構成する一般的な手順について説明します。NIMO の説明については、NIMO の説明 （51 ページ）を参照してください。

始める前に

- 基本的なトポロジ収集が構成され、収集が完了していることを確認します。

- 特定のネットワークの **[モデルマネージャ（Model Manager）]** > **[収集（Collection）]** タブにいる必要があります。

ステップ 1 モデルマネージャから、[収集（Collection）] をクリックします。

ステップ 2 [NIMOの追加（Add NIMO）] をクリックします。使用可能なすべての NIMO が表示されます。

ステップ 3 NIMOをクリックして、適切なオプションを入力します。特定のNIMOの構成オプションを表示するには、ネットワーク インターフェイス モジュール（NIMO） （51 ページ）を参照してください。

ステップ 4 [保存（**Save**）] をクリックします。

次のタスク

他のNIMOを使用してより多くの収集を構成したり機能（継続的なポーリング、デマンドメッシュの作成など）を追加したりして、完全なネットワークモデルを作成できます。

ネットワークモデルの詳細の表示

ネットワークのノードの詳細、インターフェイス、およびLSP（該当する場合）を表示するには、**[モデルマネージャ（Model Manager）]** > **[モデルの表示（View Model）]** に移動します。このページには、統合された NIMO 収集データが表示されます。

ネットワーク収集のスケジュール

この手順では、さまざまなネットワーク収集を実行するようにスケジュールする方法、および該当する場合はさまざまな時刻に実行する方法について説明します。

始める前に

特定のネットワークの **[モデルマネージャ（Model Manager）]** > **[収集のスケジュール（Schedule Collection）]** タブにいる必要があります。

ステップ 1 複数のネットワーク収集を異なる時刻に実行するようにスケジュールするには、次の手順を実行します。

a) [タスクの追加（Add Task）] をクリックします。

b) 収集を実行するタスク名と時間間隔を入力します。

c) [構成されたNIMOの選択（Select Configured NIMOs）] ドロップダウンリストから、実行する収集を選択します。

（注） 各収集は、リストおよび構成された順序で実行されます。

ステップ 2 [保存（**Save**）] をクリックします。

次のタスク

ネットワークモデルの詳細を表示し、プランファイルを表示します。

アーカイブの構成およびプランファイルの表示

ネットワークモデルを作成し、収集を実行した後、プランファイルを取得して表示するオプションがあります。プランファイルは、特定の時点でのネットワークに関するすべての関連情報をキャプチャし、トポロジ、トラフィック、ルーティング、および関連情報が含まれます。

アーカイブは、プランファイルのリポジトリです。アーカイブの構成 （48 ページ）も参照してください。WAE CLI を使用してアーカイブを構成する方法について説明しています。

ステップ 1 モデルマネージャから、プランファイルを表示するネットワークモデルをクリックします。

ステップ 2 [アーカイブ（Archive）] タブをクリックします。

ステップ 3 ネットワークが最初に作成されたときに構成した場合は、アーカイブのパスと間隔のフィールドに値が入力されている可能性があります。そうでない場合、またはそれらを変更する場合は、新しい値を入力します。

ステップ 4 [日付/時間範囲（Date/Time Range）] フィールドには、プランファイルを入手できる時間範囲が表示されます。リンクをクリックします。

ステップ 5 取得するプランファイルのタイムゾーンを選択します。

ステップ 6 プランファイルの開始日と終了日（時刻を含む）を入力し、[プランファイルの一覧表示（List Plan Files）] をクリックします。プランファイルのリストが表示されます。

ステップ 7 適切なプランファイルをクリックし、ダウンロードするプランファイルのフォーマットを選択して、[ダウンロード（Download）] をクリックします。

WAE UI を使用した XTC エージェントの構成

エージェントは情報収集タスクを実行するため、特定のネットワーク収集操作の前に構成する必要があります。このセクションでは、WAE UI を使用して XTC エージェントを構成する方法について説明します。構成解析エージェントを構成するには、構成解析エージェントの構成（28 ページ）を参照してください。

XTC エージェントの構成

XR Transport Controller（XTC）エージェントは、XTC から定期的に情報を収集し、未加工の正規化されたデータとして保持します。このデータは、トポロジや LSP などを抽出するためにさまざまな NIMO によって使用されます。ネットワーク内のすべての XTC ノードに対してエージェントを構成する必要があります。ネットワーク収集を実行する前に、XTC を使用するネットワークに対して XTC エージェントを構成する必要があります。

始める前に

[WAE UI] > **[エージェントマネージャ（Agent Manager）]** ウィンドウにいる必要があります。

ステップ 1 [XTC] をクリックします。

ステップ 2 [新規エージェントの追加（Add New Agent）] をクリックします。

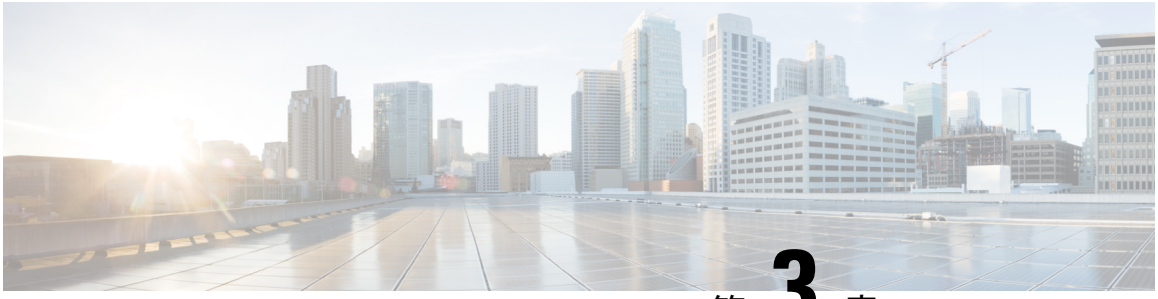
ステップ 3 次の情報を入力します。

- XTC エージェント名。
- XTC ルータのホスト IP アドレス。
- XTC ホストへの REST 呼び出しに使用するポート番号。デフォルトは 8080 です。
- 定義済みのログイン情報で HTTP Basic 認証を使用するには、ドロップダウンリストから [true] を選択します。
- XTC ログイン情報：ユーザー名、パスワード、およびイネーブルパスワード。

ステップ 4 **[保存（Save）]** をクリックします。

次のタスク

XTC を使用するネットワークの収集を構成します。詳細については、NIMO の説明 （51 ページ）を参照してください。



第 3 章

ネットワークモデルの構成：エキスパートモード

ここでは、次の内容について説明します。

エキスパートモードの概要

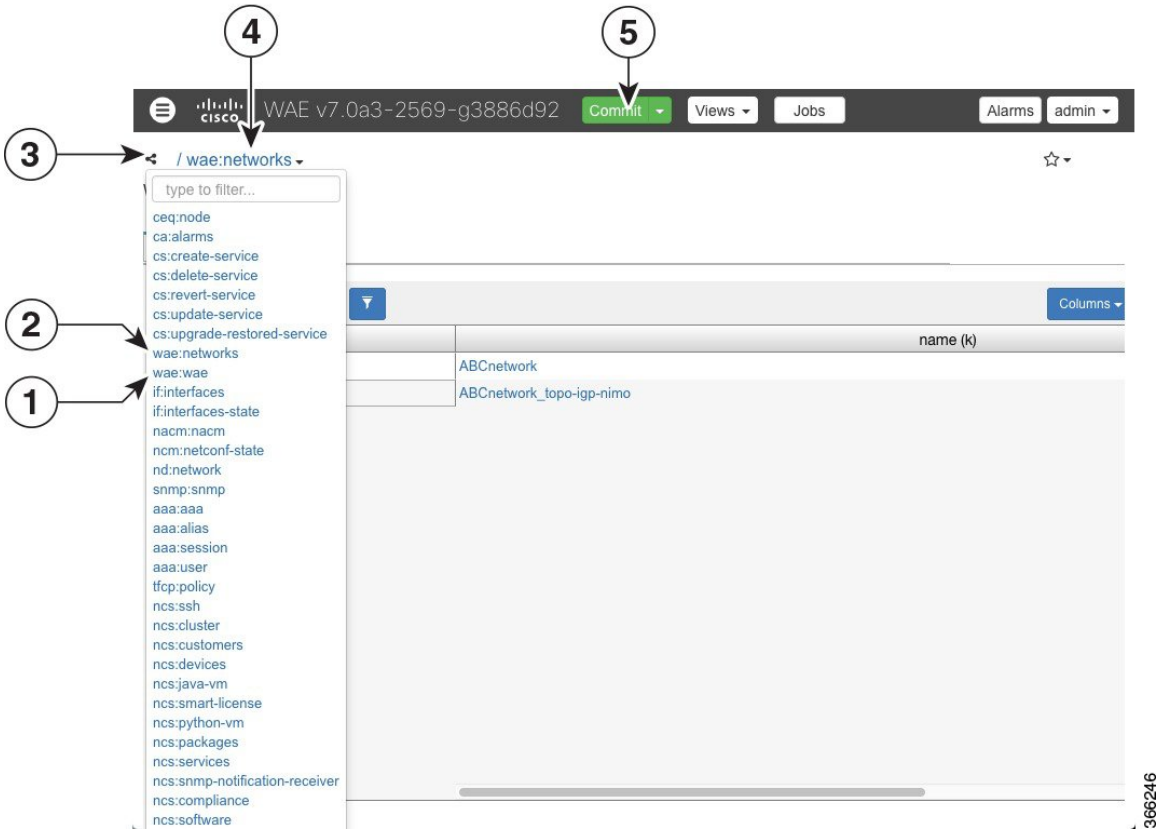
エキスパート モードは、WAE UI では利用できない可能性のある追加のデバイスおよびサービス機能を備えた YANG モデルブラウザです。また、各操作のすべてのオプションがエキスパート モードに表示されるため、WAE CLI を介してエキスパート モードを使用することもできます。

エキスパートモードは、カスタムビルドのウィジェットと、基礎となるデバイス、サービス、およびネットワークモデルからの自動レンダリングの組み合わせです。エキスパート モードは、新しいデバイス、NIMO、またはネットワークモデルがシステムに追加されるとすぐに更新されます。

WAE UI（https://server-ip:8443）の右上隅で、ツールアイコンをクリックしてエキスパート モードにアクセスします。

このセクションの目的は、エキスパートモードについて、および実行できるようにするための手順を説明することです。このセクションでは、詳細な構成については説明しません。基本的な手順を理解すると、より複雑な操作を構成できるようになることを前提としています。

図 *2:* エキスパートモードのインターフェイス



引き出し線番号	名前	説明
1	wae:wae	このパスに移動して、グローバル設定とエージェントを構成します。
2	wae:networks	このパスに移動して、ネットワーク設定と NIMO を構成します。
3	ルート ディレクトリ	このアイコンをクリックして、wae:networks および wae:wae パスにアクセスします。
4	トピックパス（パンくずリスト）	ディレクトリ構造のどこにいるかを示します。
5	[コミット（Commit）] ボタン	このボタンをクリックして、構成の変更をコミットします。

ナビゲーションとコミット

オブジェクトタイプ（ネットワークインスタンスなど）を選択すると、関連するすべてのオブジェクトインスタンスのリストが表示されます。ネットワーク構成操作を実行するときは、[コミット（Commit）] ボタンをクリックして変更を保存します。コミット機能の詳細については、コミットフラグ （171 ページ）を参照してください。

エキスパートモードを使用したネットワークモデルの構成

このワークフローでは、エキスパートモードを使用してネットワークモデルを作成する構成手順の概要について説明します。

ステップ	詳細
1. デバイス認証グループと SNMP グループを構成します。	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 （24 ページ）
2. ネットワーク アクセス プロファイルを構成します。	ネットワーク アクセスの設定 （25 ページ）
3. エージェントを構成します。 （注） この手順は、XTC またはマルチレイヤ情報を収集する場合にのみ必要です。	• エキスパートモードを使用した XTC エージェントの構成 （27 ページ） • 構成解析エージェントの構成 （28 ページ）
4. ネットワークを作成し、基本的なトポロジデータを収集します。 （注） ネットワークモデルを統合し、基本的なトポロジを超えるものを収集する（たとえば、topo-bgpls-xtc-nimo および lsp-pcep-xtc-nimo 情報を 1 つの最終ネットワークモデルにマージする）予定の場合は、DARE を構成し、収集が実行されていないネットワークを作成します。詳細については、「NIMO 収集の統合 （61 ページ）」を参照してください。	ネットワークモデルの作成 （25 ページ）
5. 追加のデータ収集を構成します。	追加 NIMO の構成 （26 ページ）
6. （オプション）スケジューラを構成します。	スケジューラ構成 （133 ページ）

ステップ	詳細
7. プランアーカイブを構成して表示します。	• WAE UI から：アーカイブの構成およびプランファイルの表示 （18 ページ） • WAE CLI から：アーカイブの構成 （48 ページ）

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

Cisco WAE は、デバイスへのログインおよび SNMP アクセスに認証グループを使用します。次の手順では、エキスパートモードを使用して認証グループとネットワークアクセスを構成する方法について説明します。

ステップ 1 エキスパート モードから、認証グループを設定します。

a) **/ncs:devices** に移動し、[authgroups] タブをクリックします。
b) [group] をクリックします。
c) プラス（[+]）記号をクリックし、認証グループ名を入力して、[追加（Add）] をクリックします。
d) [default-map] をクリックして、デフォルトの認証パラメータを入力します。たとえば、ドロップダウンリストから [remote-name] を選択し、[default-map] チェックボックスをオンにして、リモート名文字列ログイン情報を入力します。

（注） リモートのセカンダリパスワードが表示されていない場合は、下にスクロールして表示し、入力します。

ステップ 2 SNMP グループを設定します。

a) **/ncs:devices/authgroups** に戻り、[snmp-group] タブをクリックします。
b) プラス（[+]）記号をクリックし、SNMP グループ名を入力します。
c) [default-map] をオンにし、デフォルトの SNMP ログイン情報を入力します。たとえば、ドロップダウンリストから [community-name] を選択し、[default-map] チェックボックスをオンにして、コミュニティストリング ログイン情報を入力します。
d) SNMPv3 を構成する場合は、[usm] タブをクリックし、該当するユーザーベース セキュリティ モデル（USM）の値（リモートユーザー、セキュリティレベル、認証、およびプライバシープロトコル）を入力します。USM 値の詳細については、WAE UI 手順トピックネットワーク アクセスの設定 （14 ページ）で説明されている SNMPv3 オプションを参照してください。
e) [コミット（Commit）] ボタンをクリックします。

次のタスク

ネットワーク アクセス プロファイルを作成します。ネットワーク アクセスの設定 （25 ページ） を参照してください。

ネットワーク アクセスの設定

ステップ 1 **/wae:wae** に移動し、[nimos] タブをクリックします。

ステップ 2 [network_access] をクリックします。

ステップ 3 プラス（[+]）記号をクリックして、ネットワークアクセス名を入力します。

ステップ 4 適切なネットワークアクセスの詳細を選択して入力します。

以前に構成したデフォルトの認証グループと SNMP グループが、ドロップダウンリストに表示されます。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 （24 ページ）」を参照してください。

ステップ 5 [node-access] タブをクリックして、ルータの管理 IP アドレスを入力します。

a) プラス（[+]）記号をクリックし、IP アドレスを入力して、[追加（Add）] をクリックします。

b) 関連する管理 IP を入力します。

すべての管理 IP について、必要に応じてこれらの手順を繰り返します。

ステップ 6 [コミット（Commit）] ボタンをクリックします。

次のタスク

このタスクの完了後、ネットワークを作成して基本的なデータ収集を実行できます。

ネットワークモデルの作成

ネットワークを作成するときは、topo-igp-nimo または topo-bgpls-xtc-nimo を使用して基本的なトポロジ収集も構成する必要があります。次の手順では、エキスパートモードを使用した最初の構成手順について説明します。

始める前に

- デバイスアクセスとネットワークアクセスが構成されていることを確認します。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 （24 ページ）およびネットワーク アクセスの設定 （25 ページ）を参照してください。
- XTC を実行するネットワークを作成する場合は、XTC エージェントが構成されていることを確認します。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 （27 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。

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、次のいずれかのオプションを選択します。

- [topo-igp-nimo]：IGP データベースを使用してトポロジ情報を収集します。オプションについては、IGP トポロジ収集 （55 ページ）およびIGP トポロジの詳細オプション （56 ページ）を参照してください。
- [topo-bgpls-xtc-nimo]：XTC を実行しているネットワークからトポロジ情報を収集します。この NIMO には、構成されたエージェントが必要です。詳細については、XTC を使用した BGP-LS トポロジ収集 （58 ページ）およびエキスパートモードを使用した XTC エージェントの構成 （27 ページ）を参照してください。

ステップ 6 対応するリンクをクリックします。たとえば、[topo-igp-nimo] を選択した場合は、[topo-igp-nimo] リンクをクリックして、該当するパラメータを入力します。

ステップ 7 [コミット（Commit）] ボタンをクリックします。このネットワークモデルは、追加のネットワーク収集の送信元ネットワークとして使用できるようになりました。

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、追加のネットワーク収集を構成します。詳細については、NIMO の説明 （51 ページ）を参照してください。

追加 NIMO の構成

このトピックでは、さまざまなタイプの高度なネットワークデータ収集を構成するための一般的な手順についてのみ説明します。NIMOは、さまざまなタイプのデータを収集するために使用されます。一部の NIMO では、エージェントの構成が必要です。詳細については、「NIMO の説明 （51 ページ）」を参照してください。

始める前に

送信元ネットワークとして使用するには、基本的な収集を含むネットワークモデルが必要です。詳細については、「ネットワークモデルの作成 （25 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/network/*network_name*** に移動します。

ステップ 2 [nimo] タブをクリックします。

ステップ 3 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、構成する NIMO を選択します。

ステップ 4 選択した NIMO に適したパラメータを入力します。

ステップ 5 [コミット（Commit）] ボタンをクリックします。

ステップ 6 [run-collection] > [run-collectionの呼び出し（Invoke run-collection）] をクリックします。

エキスパートモードを使用したエージェントの構成

エージェントは情報収集タスクを実行するため、特定のネットワーク収集操作の前に構成する必要があります。このセクションでは、エキスパートモードを使用してこれらのエージェントを設定する方法について説明します。

エキスパートモードを使用した XTC エージェントの構成

XR Transport Controller（XTC）エージェントは、XTC から定期的に情報を収集し、未加工の正規化されたデータとして保持します。エージェントは、XTC の REST インターフェイスに接続し、PCE トポロジを取得するために使用されます。このデータは、トポロジや LSP などを抽出するために、さまざまなアプリケーション（オンデマンド帯域幅）や NIMO（topo-bgpls-xtc-nimo および lsp-pcep-xtc-nimo）によって消費されます。ネットワーク内のすべての XTC ノードに対してエージェントを構成する必要があります。ネットワーク収集を実行する前に、XTC を使用するネットワークに対して XTC エージェントを構成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、[wae:wae] に移動し、[agents] タブをクリックします。

ステップ 2 [xtc] をクリックします。

ステップ 3 プラス（[+]）アイコンをクリックして、エージェントを追加します。

ステップ 4 次の情報を入力します。

- XTC エージェント名。
- [xtc-host-ip]：XTC ルータのホスト IP アドレス。
- [xtc-rest-port]：XTC ホストへの REST 呼び出しに使用するポート番号。デフォルトは 8080 です。
- [use-auth]：定義済みのログイン情報で HTTP Basic 認証を使用するには、ドロップダウンリストから [true] を選択します。
- [auth-group]：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（24 ページ）で定義された XTC ログイン情報。
- [batch-size]：各メッセージで送信するノードの数。デフォルトは 1000 です。
- [keep-alive]：キープアライブメッセージを送信する間隔（秒単位）。デフォルトは 10 です。
- [max-lsp-history]：送信する LSP エントリの数。デフォルトは 0 です。
- [enabled]：XTC エージェントを有効にします。デフォルトは [true] です。

[enabled] オプションが [true] に設定されている限り、XTC エージェントは構成後または WAE の起動時にすぐに開始します。同様に、WAE が停止した場合、または [enabled] オプションが [false] に設定されている場合、構成が削除されると XTC エージェントが停止します。

ステップ 5 [確定する（Commit）] をクリックします。

ステップ 6 すべての XTC ノードに対してこれらの手順を繰り返します。

ステップ 7 raw データを表示するには、**/wae:wae/agents/xtc-agent:xtc/xtc/<*agent-name*>** に戻り、[pce] タブをクリックします。

ステップ 8 適切なデータコンテナ（topology-nodes、tunnel-detail-infos、および xtc-topology-objects）をクリックして、raw データを表示します。

データが正常に収集されたことを確認するには、**/wae:wae/agents/xtc-agent:xtc/xtc/<*agent-name*>** に移動し、[status] タブをクリックします。最後に成功した収集のタイムスタンプを表示できます。

次のタスク

XTC を使用するネットワークの収集を構成します。詳細については、NIMO の説明 （51 ページ）を参照してください。

構成解析エージェントの構成

構成解析エージェントは、Cisco、Juniper、Huawei のルータ構成ファイルからデータを収集（run-config-get）および解析（run-config-parse）できます。このエージェントは、マルチレイヤ収集に使用される port-cfg-parse-nimo および optimal-nimo を構成する前に構成する必要があります。

エージェントは、ルータのタイプ/ベンダーを判別し、構成を解析することにより、構成を取得できます。この情報を解析した後、ツールは IGP メッシュ内で対応するインターフェイスを照合して、ネットワークトポロジを作成します。エージェントが読み取り可能なルータ構成のタイプについては、ルータ構成情報 （29 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:wae] に移動し、[agents] タブをクリックします。

ステップ 2 [cfg-parse] をクリックします。

ステップ 3 プラス（[+]）アイコンをクリックしてエージェントを追加し、構成解析エージェント名を入力します。これは任意の名前にできます。

ステップ 4 ネットワークの構成がすでに保存されている場合は、構成が保存されているディレクトリを入力します。たとえば、`/home/user1/wae/etc/configs/gc_out` などです。または、エージェントを使用して構成を取得している場合は、構成が保存されているパスを入力します。

ステップ 5 構成を取得する場合は、[get] タブをクリックして、送信元ネットワークとネットワークアクセスを選択します。既存の構成を解析するだけの場合は、ステップ 8 にスキップします。

ステップ 6 [確定する（Commit）] をクリックします。

ステップ 7 [cfg-parse] タブに戻り、[run-config] > [run-config-parseの呼び出し（Invoke run-config-parse）] をクリックします。

ステップ 8 [parse] タブをクリックします。

ステップ 9 次の情報を入力します。

- [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。

ステップ10 [include-object] をクリックして、収集タイプを追加します。

ステップ11 プラス（[+]）アイコンをクリックし、ドロップダウンリストから収集タイプを選択します。少なくとも、base、lag、およびlmpを追加する必要があります。

ステップ12 [確定する（Commit）] をクリックします。

ステップ13 [cfg-parse] タブに戻り、[run-config-parse] > [run-config-parseの呼び出し（Invoke run-config-parse）] をクリックします。

ステップ14 エージェントが正常に実行されていることを確認するには、[model] タブ > [nodes] をクリックします。ノードのリストが表示されます。

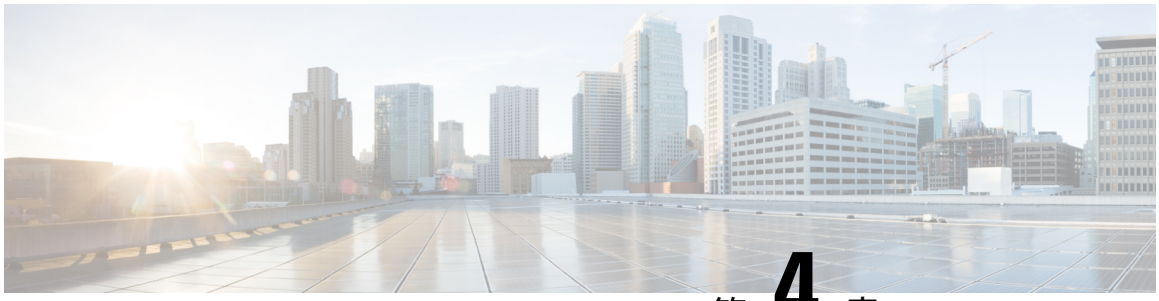
次のタスク

マルチレイヤ収集用にport-cfg-parse-nimoとoptical-nimoを構成します。詳細については、NIMOの説明 （51 ページ）を参照してください。

ルータ構成情報

次のルータ構成情報は、構成解析エージェントによって読み取ることができます。

• ルータ名 • ルータのIPアドレス（ループバック） • インターフェイス名（IGPトポロジ内） • インターフェイスのIPアドレス • インターフェイスのキャパシティ（利用可能な場合）	• IGPタイプとメトリック（IS-ISまたはOSPF） • RSVP予約可能帯域幅（MPLS） • LAGポート（イーサネット用）およびバンドルポート（各種リンクタイプ用）



第 4 章

ネットワークモデルの構成：WAE CLI

ここでは、次の内容について説明します。

WAE CLI の概要

WAE は、WAE YANG ファイルで記述されたデータモデルを使用して自動的にレンダリングされるネットワーク CLI を提供します。CLI には、ネットワーク構成を操作するためのコマンドが含まれています。CLI は完全にデータモデル駆動型です。YANG モデルは構成要素の階層を定義します。CLI はこのツリーに従います。CLI には、管理対象デバイスのハードウェアおよびネットワーク接続を構成するためのさまざまなコマンドが用意されています。

CLI は 2 つのモードをサポートしています。WAE ノードの状態を監視するための動作モードと、ネットワークの状態を変更するための構成モードです。プロンプトは、CLI がどのモードにあるかを示します。**configure** コマンドを使用して動作モードから構成モードに移行すると、プロンプトが **user@wae#** から **user@wae(config)#** に変化します。プロンプトは、wae.conf ファイルの c-prompt1 および c-prompt2 設定を使用して構成できます。

次に例を示します。

```
admin@wae# configure
Entering configuration mode terminal
admin@wae(config)#
```

動作モード（Operational Mode）

動作モードは、CLI へのログインに成功した後の初期モードです。これは主に、システムステータスの表示、CLI 環境の制御、ネットワーク接続の監視とトラブルシューティング、および構成モードの開始に使用されます。

次のコマンドは、動作モードで使用できる基本コマンドです。追加のコマンドは、ロードされた YANG ファイルからレンダリングされます。

アクションを呼び出します。

<path> <parameters>

指定された *parameters* を使用して、*path* で見つかったアクションを呼び出します。このコマンドはYANGファイルから自動生成されます。たとえば、YANGファイルに次のアクション指定があるとします。

```
tailf:action shutdown {
  tailf:actionpoint actions;
  input {
    tailf:constant-leaf flags {
      type uint64 {
        range "1 .. max";
      }
      tailf:constant-value 42;
    }
    leaf timeout {
      type xs:duration;
      default PT60S;
    }
    leaf message {
      type string;
    }
    container options {
      leaf rebootAfterShutdown {
        type boolean;
        default false;
    }
    leaf forceFsckAfterReboot {
      type boolean;
      default false;
    }
    leaf powerOffAfterShutdown {
      type boolean;
      default true;
    }
  }
 }
}
```

このアクションは、次の方法で呼び出すことができます。

```
user@wae> shutdown timeout 10s message reboot options { \
forceFsckAfterReboot true }
```

組み込みの動作モードコマンド

コマンド	説明
commit (abort|confirm)	保留中の確認コミットを中止または確認します。**commit confirm**を実行せずにCLIセッションが終了した場合、保留中の確認コミットは中止されます。デフォルトは **confirm** です。例： `user@wae# commit abort`

config (exclusive|terminal) [no-confirm]	構成モードを開始します。デフォルトは **terminal** です。
	• terminal：実行構成のプライベートコピーを編集します。ロックはかかりません。
	• no-confirm：確認ダイアログを無視して、構成モードを開始します。
	例：
	`user@wae# config terminal`
	`Entering configuration mode terminal`

file list <*directory*>	ディレクトリ内のファイルを一覧表示します。例：
	`user@wae# file list /config`
	`rollback10001`
	`rollback10002`
	`rollback10003`
	`rollback10004`
	`rollback10005`

file show <*file*>	ファイルの内容を表示します。例：
	`user@wae# file show /etc/skel/.bash_profile`
	`# /etc/skel/.bash_profile`
	`# This file is sourced by bash for login shells. The following line`
	`# runs our .bashrc and is recommended by the bash info pages.`
	`[[ -f ~/.bashrc ]] && . ~/.bashrc`

help <*command*>	コマンドのヘルプテキストが表示されます。例：
	`user@wae# help job`
	`Help for command: job`
	`  Job operations`

job stop <*job id*>	特定のバックグラウンドジョブを停止します。デフォルトの CLI では、バックグラウンドジョブを作成するコマンドは **monitor start** だけです。例：
	`user@wae# monitor start /var/log/messages`
	`[ok][...]`
	`admin@ncs# show jobs`
	`JOB COMMAND`
	`3 monitor start /var/log/messages`
	`admin@ncs# job stop 3`
	`admin@ncs# show jobs`
	`JOB COMMAND`

logout session <*session ID*>	WAE から特定のユーザーセッションをログアウトします。ユーザーが**構成排他**ロックを保持している場合、ロックは解放されます。例：
	
	`user@wae# who`
	`Session User Context From Proto Date Mode`
	`25 oper cli 192.0.2.254 ssh 12:10:40 operational`
	`*24 admin cli 192.0.2.254 ssh 12:05:50 operational`
	`user@wae# logout session 25`
	`user@wae# who`
	`Session User Context From Proto Date Mode`
	`*24 admin cli 192.0.2.254 ssh 12:05:50 operational`

logout user <*username*>	WAE から特定のユーザーをログアウトします。ユーザーが**構成排他**ロックを保持している場合、ロックは解放されます。例：
	
	`user@wae# who`
	`Session User Context From Proto Date Mode`
	`25 oper cli 192.0.2.254 ssh 12:10:40 operational`
	`*24 admin cli 192.0.2.254 ssh 12:05:50 operational`
	`user@wae# logout user oper`
	`user@wae# who`
	`Session User Context From Proto Date Mode`
	`*24 admin cli 192.0.2.254 ssh 12:05:50 operational`

script reload	scripts/command ディレクトリにあるスクリプトをリロードします。新しいスクリプトが追加されます。スクリプトファイルが削除されている場合、対応する CLI コマンドは消去されます。

send (all | <*user*>**)** <*message*>	デバイスにログインしているすべてのユーザーの画面または特定の画面にメッセージを表示します。
	• all：現在ログインしているすべてのユーザーにメッセージを表示します。
	• <*user*>：特定のユーザーにメッセージを表示します。
	
	例：
	
	`user@wae# send oper "I will reboot system in 5 minutes."`
	
	oper ユーザーの画面には、次のメッセージが表示されます。
	
	`oper@wae# Message from user@wae at 13:16:41...`
	`I will reboot system in 5 minutes.`
	`EOF`

show cli	CLI プロパティを表示します。例：
	
	`user@wae# show cli`
	`autowizard false`
	`complete-on-space true`
	`display-level 99999999`
	`history 100`
	`idle-timeout 1800`
	`ignore-leading-space false`
	`output-file terminal`
	`paginate true`
	`prompt1 \h\M#`
	`prompt2 \h(\m)#`
	`screen-length 71`
	`screen-width 80`
	`service prompt config true`
	`show-defaults false`
	`terminal xterm-256color`
	`timestamp disable`

show history [<*limit*>]	CLI コマンドの履歴を表示します。デフォルトでは、最新の 100 個のコマンドが一覧表示されます。履歴リストのサイズは、history CLI 設定を使用して構成されます。履歴制限が指定されている場合、その制限までの最新のコマンドのみが表示されます。例：
	
	`user@wae# show history`
	`06-19 14:34:02 -- ping router`
	`06-20 14:42:35 -- show running-config`
	`06-20 14:42:37 -- who`
	`06-20 14:42:40 -- show history`
	`user@wae# show history 3`
	`14:42:37 -- who`
	`14:42:40 -- show history`
	`14:42:46 -- show history 3`

show jobs	現在バックグラウンドで実行中のジョブを表示します。例：
	
	`user@wae# show jobs`
	`JOB COMMAND`
	`3 monitor start /var/log/messages`

show log <*file*>	ログファイルの内容を表示します。例： `user@wae# show log messages`
show parser dump <*command prefix*>	指定されたコマンドプレフィックスで始まるすべての可能なコマンドを表示します。

show running-config [<*path filter*> [**sort-by** <*idx*>]]	現在の設定を表示します。デフォルトでは、構成全体が表示されます。パスフィルタを指定することで、表示される内容を制限できます。パスフィルタでは、特定のインスタンスを指すパスを使用することも、またはインスタンスIDを省略した場合は、省略されたインスタンス以降の部分をフィルタとして扱うこともできます。
	
	sort-by引数は、パスフィルタがセカンダリインデックスを持つリスト要素を指す場合に使用できます。セカンダリインデックスの名前は*idx*です。使用すると、テーブルはセカンダリインデックスで定義された順序で並べ替えられます。これにより、インスタンスを表示する順序を制御できます。
	
	たとえば、管理者ユーザーのaaa設定を表示するには：
	
	`user@wae# show running-config aaa authentication users user admin`
	`aaa authentication users user admin`
	`  uid 1000`
	`  gid 1000`
	`  password $1$JA.1O3Tx$Zt1ycpnMlg1bVMqM/zSZ7/`
	`  ssh_keydir /var/ncs/homes/admin/.ssh`
	`  homedir /var/ncs/homes/admin`
	`!`
	
	グループID 1000のすべてのユーザーを表示し、ユーザーIDを省略して代わりにgid 1000を指定するには：
	
	`user@wae# show running-config aaa authentication users user * gid 1000`
	`...`

show <*path*> [**sort-by** <*idx*>]	パスがリスト要素につながり、データをテーブルとしてレンダリングできる（つまり、テーブルが画面に収まる）場合、構成をテーブルとして表示します。|**tab**パイプコマンドを使用して、リストのテーブルフォーマットを強制することもできます。
	
	sort-by引数は、パスがセカンダリインデックスを持つリスト要素を指す場合に使用できます。セカンダリインデックスの名前は*idx*です。使用すると、テーブルはセカンダリインデックスで定義された順序で並べ替えられます。これにより、インスタンスを表示する順序を制御できます。例：
	
	`user@wae# show devices device-module`
	`NAME REVISION URI DEVICES`
	`---------------------------------------------------------------------------------------`
	`junos - http://xml.juniper.net/xnm/1.1/xnm [ pe2 ]`
	`tailf-ned-cisco-ios - urn:ios [ ce1 ce0 ]`
	`tailf-ned-cisco-ios-stats - urn:ios-stats [ ce1 ce0 ]`
	`tailf-ned-cisco-ios-xr - http://tail-f.com/ned/cisco-ios-xr [ p1 p0 ]`

source <*file*>	ユーザーが入力したかのように、指定されたファイルからコマンドを実行します。ファイルからコマンドを実行する場合、autowizardは無効になります。

timecmd <*command*>	コマンドの実行時間を測定して表示します。**timecmd** は、CLI セッション設定で devtools が true に設定されている場合にのみ使用できることに注意してください。例：
	
	`user@wae# timecmd id`
	`user = admin(501), gid=20, groups=admin, gids=12,20,33,61,79,80,81,98,100`
	`Command executed in 0.00 sec`
	`user@wae#`

who	現在、ログオンしているユーザーを表示します。現在のセッション（**show status** コマンドを実行しているセッション）には、アスタリスクが付いています。例：
	
	`user@wae# who`
	`Session User Context From Proto Date Mode`
	`25 oper cli 192.0.2.254 ssh 12:10:40 operational`
	`*24 admin cli 192.0.2.254 ssh 12:05:50 operational`
	`admin@ncs#`

構成モード

構成モードは、動作モードで configure コマンドを入力することで開始できます。ネットワーク構成に対するすべての変更は、アクティブな構成のコピーに対して行われます。これらの変更は、コミットまたはコミット確認コマンドが正常に入力されるまで有効になりません。

次のコマンドは、構成モードで使用できる基本コマンドです。追加のコマンドは、ロードされた YANG ファイルからレンダリングされます。

値を構成します。

<path> **[***<value>***]**

パラメータを設定します。新しい識別子が作成され、autowizard が enabled の場合、CLI は、その識別子のすべての必須サブ要素についてユーザーにプロンプトを表示します。このコマンドは YANG ファイルから自動生成されます。

<*value*> が提供されていない場合、CLI はその値についてユーザーにプロンプトを表示します。<*path*> が tailf-common.yang データモデルで記述されている MD5DigestString、DESDigestString、DES3CBCEncryptedString、または AESCFB128EncryptedString 型の暗号化された値である場合、入力された値のエコーは発生しません。

組み込みの構成モードコマンド

コマンド	説明
annotate <*statement*> <*text*>	注釈を特定の構成に関連付けます。注釈を削除するには、テキストを空のままにします。このコマンドは、システムが属性を有効にして構成されている場合にのみ使用できます。

commit (**check** | **and-quit** | **confirmed** |**to-startup**) [**comment** <*text*>] [**label** <*text*>]	現在の構成をコミットして running にします。 • check：現在の構成を検証します。 • and-quit：コミットして running にし、構成モードを終了します。 • comment <*text*>：コミットに関連付けます。コメントは、ロールバックファイルを調べるときに表示されます。 • label <text>：ラベルをコミットに関連付けます。ラベルは、ロールバックファイルを調べるときに表示されます。 （注） 便利なコマンドは、`commit dry-run` です。このコマンドは、構成の変更を検証して表示しますが、実際のコミットは実行しません。使用可能な `commit` コマンドの詳細については、コミットフラグ （171 ページ）を参照してください。
copy <*instance path*> <*new id*>	インスタンスのコピーを作成します。

copy cfg [**merge** | **overwrite**] <*src path*> **to** <*dest path*>	ある構成ツリーから別の構成ツリーにデータをコピーします。接続先で意味のあるデータのみがコピーされます。コピーできないデータについてはエラーメッセージは生成されず、操作はエラーメッセージが生成されずに完全に失敗する可能性があります。たとえば、デバイス構成の一部からテンプレートを作成するには、最初にデバイスを構成してから、構成をテンプレート構成ツリーにコピーします。例：
	`user@wae(config)# devices template host_temp`
	`user@wae(config-template-host_temp)# exit`
	`user@wae(config)# copy cfg merge devices device ce0 config \`
	`  ios:ethernet to devices template host_temp config ios:ethernet`
	`user@wae(config)# show configuration diff`
	`+devices template host_temp`
	`+ config`
	`+ ios:ethernet cfm global`
	`+ !`
	`+!`

copy compare <*src path*> **to** <*dest path*>	2つの任意の構成ツリーを比較します。送信元ツリーにのみ表示される項目は無視されます。
delete <*path*>	データ要素を削除します。
do <*command*>	コマンドを動作モードで実行します。
edit <*path*>	サブ要素を編集します。パスに欠落している要素が作成されます。
exit (**level** | **configuration-mode**)	• level：このレベルを終了します。トップレベルで実行すると、構成モードを終了します。これは、オプションが指定されていない場合のデフォルトです。 • configuration-mode：編集レベルに関係なく構成モードを終了します。
help <*command*>	コマンドのヘルプテキストを表示します。

hide <*hide-group*>	非表示グループに属する要素とアクションを再度非表示にします。非表示にするのにパスワードは必要ありません。このコマンドは非表示であり、コマンドの完了時に表示されません。
insert <*path*>	新しい要素を挿入します。要素がすでに存在し、データモデルにindexedViewオプションが設定されている場合、古い要素の名前がelement+1に変更され、新しい要素がその場所に挿入されます。
insert <*path*>[**first** | **last** | **before** <*key*> | **after** <*key*>]	順序付きリストに新しい要素を挿入します。要素は、先頭、末尾（デフォルト）、別の要素の前または後に追加できます。

load (**merge** | **override**) (**terminal** | <*file*>)	ファイルまたは端末から構成をロードします。
	• merge：ファイルまたは端末の内容を現在の構成とマージします。
	• override：現在の構成をファイルまたは端末からの構成で置き換えます。
	たとえば、現在の構成が次のようであるとします。
	`devices device p1`
	` config`
	`  cisco-ios-xr:interface GigabitEthernet 0/0/0/0`
	`   shutdown`
	`  exit`
	`  cisco-ios-xr:interface GigabitEthernet 0/0/0/1`
	`   shutdown`
	` !`
	`!`
	エントリ `GigabitEthernet 0/0/0/0` の **shutdown** 値が削除されます。構成ファイルは、間にコメントが挿入された一連のコマンドであるため、構成ファイルは次のようになります。
	`devices device p1`
	` config`
	`  cisco-ios-xr:interface GigabitEthernet 0/0/0/0`
	`   no shutdown`
	`  exit`
	` !`
	`!`
	その後、このファイルをコマンド **load merge** *FILENAME* で使用して、目的の結果を得ることができます。

move <*path*>[**first**|**last** | **before** <*key*> | **after** <*key*>]	既存の要素を順序付きリストの新しい位置に移動します。要素は、先頭、末尾（デフォルト）、別の要素の前または後に移動できます。
rename <*instance path*> <*new id*>	インスタンスの名前を変更します。
revert	実行構成を現在の構成にコピーし、コミットされていないすべての変更を削除します。

rload(**merge**|**override**) (**terminal** | <*file*>)	現在のサブモードに関連するファイルをロードします。たとえば、ファイルにデバイス構成がある場合、1 つのデバイスに入り、**rload merge**/**override** <*file*> コマンドを発行してそのデバイスの構成をロードし、次に別のデバイスに入り、**rload** を使用して同じ構成ファイルをロードできます。 **load** コマンドも参照してください。 • merge：ファイルまたは端末の内容を現在の構成とマージします。 • override：現在の構成をファイルまたは端末からの構成で置き換えます。
rollback configuration [<*number*>] [<*path*>]	構成を以前にコミットされた構成に戻します。wae.confファイルに保存する古い構成の数を構成できます。保存する構成がしきい値を超えると、新しい構成を作成する前に最も古い構成が削除されます。構成の変更はロールバックファイルに保存され、最新の変更は最大の番号 N を持つファイル rollbackN に保存されます。 デルタのみがロールバックファイルに保存されます。構成をロールバックNにロールバックすると、rollback10001 ～ rollbackN に保存されているすべての変更が適用されます。オプションの path 引数を使用すると、構成ツリーの残りの部分を変更せずに、サブツリーをロールバックできます。 このコマンドは、wae.confでロールバックが有効になっている場合にのみ使用できます。例： `user@wae(config)# rollback configuration 10005`
rollback selective [<*number*>] [<*path*>]	rollback10001 から rollbackN までのすべての変更を元に戻す代わりに、特定のロールバックファイルに保存されている変更のみを元に戻すことができます。場合によっては、ロールバックファイルの適用に失敗したり、構成を有効にするために追加の変更が必要になったりすることがあります。 オプションのpath引数を使用すると、構成ツリーの残りの部分を変更せずに、サブツリーをロールバックできます。
show full-configuration [<*pathfilter*> [**sort-by** <*idx*>]]	ローカルの変更を考慮して、現在の構成を表示します。パスフィルタを指定することにより、showコマンドを構成の一部に制限できます。**sort-by**引数は、パスフィルタがセカンダリインデックスを持つリスト要素を指す場合に指定できます。セカンダリインデックスの名前は*idx*です。使用すると、テーブルはセカンダリインデックスで定義された順序で並べ替えられます。これにより、インスタンスを表示する順序を制御できます。
show configuration [<*pathfilter*>]	構成に対する現在の編集を表示します。
show configuration merge [<*pathfilter*> [**sort-by** <*idx*>]]	ローカルの変更を考慮して、現在の構成を表示します。パスフィルタを指定することにより、showコマンドを構成の一部に制限できます。**sort-by**引数は、パスフィルタがセカンダリインデックスを持つリスト要素を指す場合に指定できます。セカンダリインデックスの名前は*idx*です。使用すると、テーブルはセカンダリインデックスで定義された順序で並べ替えられます。これにより、インスタンスを表示する順序を制御できます。

show configuration commit changes [<*number*> [<*path*>]]	コミットに対して作成されたロールバック番号によって識別される、そのコミットに関連付けられた編集を表示します。変更を元に戻すためのコマンドを表示する **show configuration rollback changes** とは対照的に、変更は前方変更として表示されます。オプションの path 引数を使用すると、特定のサブツリーに関連する編集のみをリストできます。
show configuration commit list [<*path*>]	ロールバックファイルをリストします。オプションの path 引数を使用すると、特定のサブツリーに関連するロールバックファイルのみをリストできます。
show configuration rollback listed [<*number*>]	ロールバックファイルに関連付けられたコミットで実行された変更を元に戻すために必要な操作を表示します。これらは、構成がそのロールバック番号にロールバックされた場合に適用される変更です。
show configuration running [<*pathfilter*>]	コミットされていない変更を考慮せずに実行構成を表示します。オプションのパスフィルタを指定して、表示する内容を制限できます。
show configuration diff [<*pathfilter*>]	追加および削除された構成行の前に + と - を付けて、実行構成に対するコミットされていない変更を差分形式で表示します。
show parser dump <*command prefix*>	コマンドプレフィックスで始まるすべての可能なコマンドを表示します。
tag add <*statement*> <*tag*>	構成ステートメントにタグを追加します。このコマンドは、システムが属性を有効にして構成されている場合にのみ使用できます。
tag del <*statement*> <*tag*>	構成ステートメントからタグを削除します。このコマンドは、システムが属性を有効にして構成されている場合にのみ使用できます。
tag clear <*statement*>	構成ステートメントからすべてのタグを削除します。このコマンドは、システムが属性を有効にして構成されている場合にのみ使用できます。

timecmd <*command*>	コマンドの実行時間を測定して表示します。このコマンドは、CLI セッション設定で devtools が true に設定されている場合にのみ使用できます。例：
	
	`user@wae# timecmd id`
	`user = admin(501), gid=20, groups=admin, gids=12,20,33,61,79,80,81,98,100`
	`Command executed in 0.00 sec`
	`user@wae#`

top [<*command*>]	構成の最上位に戻るか、構成の最上位でコマンドを実行します。
unhide <*hide-group*>	非表示グループに属するすべての要素とアクションを再表示します。パスワードが必要な場合があります。このコマンドは非表示であり、コマンドの完了時に表示されません。
validate	現在の構成を検証します。**commit check** と同じ動作です。

xpath [**ctx** <*path*>] (**eval** | **must** | **when**) <*expression*>	XPath 式を評価します。context-path は、式の評価のための現在のコンテキストとして使用できます。context-path が指定されていない場合、現在のサブモードが context-path として使用されます。パイプコマンドトレースを使用して、デバッグ情報またはトレース情報を表示できます。このコマンドは、CLI セッション設定で devtools が true に設定されている場合にのみ使用できます。 • eval：XPath 式を評価します。 • must：式を YANG *must* 式として評価します。 • when：式を YANG *when* 式として評価します。

エキスパートモードと WAE CLI の比較

このガイドでは、エキスパートモードを使用した多くの構成について説明していますが、エキスパート モードと CLI インターフェイスを互換的に使用できることに注意することが重要です。GUI ではなくエキスパート モードを使用する利点は、構成に使用可能なすべてのフィールドが表示されることです。CLI で、使用可能なすべてのオプションを表示するには、パラメータを把握しているか、CLI コマンドのヘルプを表示する必要があります。

CLI での構成は、エキスパートモードでのナビゲートと同じパス構造に従います。次の表に、サンプルデータを使用して同等である CLI コマンドとエキスパート モード構成を示します。

設定の種類	エキスパート モード	CLI での同等コマンド
デバイスにログイン情報を使用してデバイス認証グループを作成します。	1. **/ncs:devices** に移動し、[authgroups] タブをクリックします。 2. [group] をクリックします。 3. プラス（[+]）記号をクリックし、認証グループ名として **groupABC** を入力して、[追加（Add）] をクリックします。 4. [default-map] をクリックし、次の認証パラメータを入力します。[remote-name]：**rpc1**、[remote-password]：**XLydrf**、[remote-secondary-password]：**XLydrr**。	`# set devices authgroups group` **groupABC** `default-map remote-name` **rpc1** `remote-password` **XLydrf** `remote-secondary-password` **XLydrr**

<table>
<tr><th>設定の種類</th><th>エキスパート モード</th><th>CLI での同等コマンド</th></tr>
<tr>
<td>XTC（topo-bgpls-xtc-nimo）を使用してネットワークを検出して、ネットワークモデルを作成します。
（注） この例では、ネットワークアクセスと XTC エージェントが構成され、実行されていることを前提としています。</td>
<td>1. /wae:networks に移動し、プラス（[+]）記号をクリックして、as54001_topo と入力します。
2. [追加（Add）]をクリックします。
3. [nimo] タブをクリックし、NIMO タイプとして [topo-bgpls-xtc-nimo] を選択します。
4. 次を入力します。
<table>
<tr><th>フィールド</th><th>ユーザ入力（User Input）</th></tr>
<tr><td>network-access</td><td>as54001</td></tr>
<tr><td>xtc-host</td><td>xt11</td></tr>
<tr><td>backup-xtc-host</td><td>xt12</td></tr>
<tr><td>asn</td><td>54001</td></tr>
<tr><td>igp-protocol</td><td>isis</td></tr>
<tr><td>extended-topology-discovery</td><td>true</td></tr>
</table>
</td>
<td># set networks network as54001_topo nimo topo-bgpls-xtc-nimo network-access as54001 xtc-host xtc11

backup-xtc-host xtc12 igp-protocol isis extended-topology-discovery true asn 54001</td>
</tr>
<tr>
<td>ネットワークモデルを統合します。</td>
<td>1. /wae:networks に移動し、プラス（[+]）記号をクリックして、as54001 と入力します。
2. [追加（Add）]をクリックします。
3. [nimo] タブをクリックし、[aggregator] を選択します。
4. [aggregator] > プラス（[+]）記号をクリックし、送信元 NIMO として [as54001_topo]、[as54001_xtclsp]、[as54001_conflsp]、および [as54001_snmplsp] を選択します。</td>
<td># set networks network as54001 nimo aggregator sources as54001_topo
set networks network as54001 nimo aggregator sources as54001_xtclsp
set networks network as54001 nimo aggregator sources as54001_conflsp
set networks network as54001 nimo aggregator sources as54001_snmplsp</td>
</tr>
</table>

WAE CLI を使用したネットワークモデルの構成

このワークフローでは、エキスパートモードを使用してネットワークモデルを作成する構成手順の概要について説明します。

ステップ	詳細
1. デバイス認証グループと SNMP グループを構成します。	CLI を使用したデバイスアクセスの構成 （44 ページ）
2. ネットワーク アクセス プロファイルを構成します。	ネットワーク アクセス プロファイルの構成 （45 ページ）
3. エージェントを構成します。 （注） この手順は、XTC またはマルチレイヤ情報を収集する場合にのみ必要です。	• エキスパートモードを使用した XTC エージェントの構成 （27 ページ） • 構成解析エージェントの構成 （28 ページ）
4. ネットワークを作成し、基本的なトポロジデータを収集します。	ネットワークモデルの作成 （46 ページ）
5. 追加のデータ収集または NIMO 機能を構成します。	追加 NIMO の構成 （48 ページ）
6. （オプション）スケジューラを構成します。	スケジューラ構成 （133 ページ）
7. プランアーカイブを構成して表示します。	アーカイブの構成 （48 ページ）

CLI を使用したデバイスアクセスの構成

WAE は、デバイスへのログインおよび SNMP アクセスに認証グループを使用します。次の手順では、構成モードで CLI を使用してデバイスアクセスを構成する方法について説明します。

ステップ 1 デバイスログイン情報を使用してデバイス認証グループを作成します。

```
# set devices authgroups group <group_name>
# default-map remote-name <username>
# default-map remote-password <user_password>
# default-map remote-secondary-password <secondary_password>
# commit
```

ステップ 2 SNMP ツールを実行できるように SNMP グループを作成します。

```
# set devices authgroups snmp-group <group_name>
# default-map community-name <community_name>
# commit
```

SNMPv3 の場合、次のオプションを設定できます。

```
# set devices authgroups snmp-group <group_name>
# default-map community-name <community_name>
# default-map usm remote-name <remote_user>
# default-map usm security-level <auth-priv or auth-no-priv or no-auth-no-priv>
# default-map usm auth <auth_protocol> remote-password <remote_password>
# default-map usm priv <priv_protocol> remote-password <remote_password>
# commit
```

例

例（デモンストレーションの目的で単純な名前とパスワードを使用）：

```
user@wae(config)# set devices authgroups group ABCgroup default-map remote-name anyuser
 remote-password password123 remote-secondary-password mypassword
user@wae(config)# commit

user@wae(config)# set devices authgroups snmp-group snmp_v2 default-map community-name
mycompany
user@wae(config)# commit

user@wae(config)# set devices authgroups snmp-group snmp_v3_01
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User1
user@wae(config)# default-map usm security-level auth-priv
user@wae(config)# default-map usm auth md5 remote-password pass_a123
user@wae(config)# default-map usm priv aes remote-password pass_a123
user@wae(config)# commit

user@wae(config)# set devices authgroups snmp-group snmp_v3_02
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User2
user@wae(config)# default-map usm security-level auth-no-priv
user@wae(config)# default-map usm auth sha remote-password pass_b456
user@wae(config)# commit

user@wae(config)# set devices authgroups snmp-group snmp_v3_03
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User2
user@wae(config)# default-map usm security-level no-auth-no-priv
user@wae(config)# commit
```

ネットワーク アクセス プロファイルの構成

始める前に

認証およびSNMPグループが構成されていることを確認します。詳細については、「CLIを使用したデバイスアクセスの構成 （44 ページ）」を参照してください。

ステップ1 次のコマンドを入力します。

```
# set wae nimos network-access network-access <network-access-ID> auth-group <auth-group-ID>
# set wae nimos network-access network-access <network-access-ID> snmp-group <snmp-group-ID>
```

ステップ 2 次のコマンドを繰り返して、各管理 IP アドレスを入力します。

```
# set wae nimos network-access network-access <network-access-IP> node-access <node-access-ID-1>
auth-group <auth_group_ID> default-snmp-group <snmp-group-ID>
ip-manage <ip-address-1>
```

ステップ 3 設定をコミットします。

```
# commit
```

例

次に例を示します。

```
# set wae nimos network-access network-access as64001 auth-group ABCgroup
# set wae nimos network-access network-access as64001 snmp-group snmp_v3_01
# set wae nimos network-access network-access as64001 node-access 1.1.1.1 ip-manage
10.18.20.121
# set wae nimos network-access network-access as64001 node-access 2.2.2.2 ip-manage
10.18.20.122
# commit

# set wae nimos network-access network-access netaccess_01 auth-group ABCgroup
# set wae nimos network-access network-access netaccess_01 snmp-group snmp_v2
node-access 122.168.200.2 ip-manage 192.18.20.2
```

ネットワークモデルの作成

ネットワークを作成するときは、topo-igp-nimo または topo-bgpls-xtc-nimo を使用して基本的なトポロジ収集も構成する必要があります。詳細については、IGP トポロジ収集 （55 ページ）およびXTC を使用した BGP-LS トポロジ収集 （58 ページ）を参照してください。

（注） 既存のプランファイルをロードしてネットワークモデルを作成するオプションがあります。プランファイルのロード （47 ページ） を参照してください。

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。
- XTC を実行するネットワークを作成する場合は、XTC エージェントが構成されていることを確認します。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 （27 ページ）」を参照してください。

次のコマンドを入力します。

```
# networks network <topo-network-model-name> nimo <NIMO-name>
network-access <network-access> <parameter-1>
```

```
<parameter-1-option> <parameter-2> <parameter-2-option>
<parameter-x> <parameter-x-option>
# commit
```

例

次の例は、topo-bgpls-xtc-nimo を設定する 2 つの方法を示しています。

例 1：

```
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo network-access
 TTE_lab_access
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo xtc-host
TTE-xtc11
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo backup-xtc-host
 xtc12
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo asn 62001
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo igp-protocol
 isis
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo
extended-topology-discovery true
```

例 2：

```
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo
network-access TTE_lab xtc-host TTE-xtc11
backup-xtc-host TTE-xtc12 igp-protocol isis
extended-topology-discovery true asn 62001
```

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、追加のネットワーク収集を構成します。詳細については、NIMO の説明 （51 ページ）を参照してください。

プランファイルのロード

プランファイルを読み込んでネットワークモデルを作成できます。これは、たとえば、収集されたトポロジとデマンドの情報を含むプランファイルがすでにある場合に役立ちます。基本的なトポロジ収集を使用して新しいネットワークモデルを作成してデマンドで拡張することによってゼロから始める代わりに、既存のプランファイルをロードできます。

次のコマンドを使用して、プランファイルからネットワークモデルを作成します。

```
# wae components load-plan run plan-file <plan-file-location> network-name
<network-model-name>
```

次に例を示します。

```
# wae components load-plan run plan-file /home/tommy/us_atlanta_wan1.txt network-name
NetworkABC_topo_demands
```

追加 NIMO の構成

このトピックでは、さまざまなタイプの高度なネットワークデータ収集を構成するための一般的な手順について説明します。NIMOは、さまざまなタイプのデータを収集するために使用されます。一部のNIMOでは、エージェントの構成が必要です。詳細については、「NIMOの説明 （51ページ）」を参照してください。

始める前に

送信元ネットワークとして使用するには、基本的な収集を含むネットワークモデルが必要です。

次のコマンドを入力します。

```
# networks network <network-model-name> nimo <NIMO-name> source-network <source-network>

# networks network <network-model-name> nimo <NIMO-name> <parameter-x> <parameter-x-option>
```

例

次の例は、lsp-config-nimo 構成を示しています。

```
# networks network NetworkABC_lsp-config-nimo nimo lsp-config-nimo source-network
NetworkABC_topo-bgpls-xtc-nimo
# networks network NetworkABC_lsp-config-nimo nimo lsp-config-nimo in-sync true
# commit
```

アーカイブの構成

WAE UI を使用して、アーカイブを構成することもできます。アーカイブの構成およびプランファイルの表示 （18ページ） を参照してください。

ステップ 1 WAE CLI を起動し、構成モードに入ります。

```
# wae_cli -C
# config
(config)#
```

ステップ 2 アーカイブディレクトリを構成します。

```
(config)# networks network <network_model_name> plan-archive archive-dir <archive_directory>
(config)# commit
```

次に例を示します。

```
(config)# networks network Network_123 plan-archive archive-dir /archive/planfiles/Network_123
(config)# commit
```

ステップ 3 アーカイブを実行します。これにより、現在のネットワークモデルがプランファイル（.pln フォーマット）で指定したアーカイブディレクトリに保存されます。

```
(config)# networks network <network_model_name> plan-archive run
```

次に例を示します。

```
(config)# networks network Network_123 plan-archive run
status true
message Successfully archived plan file 20170131_1919_UTC.pln for network Network_123
```

ステップ 4 アーカイブディレクトリに移動して、プランファイルが保存されたことを確認します。アーカイブディレクトリは、年、月、日といったサブフォルダに分かれています。

次に例を示します。

```
(config)# ls /Network_123/2017/01/31
20170131_0100_UTC.pln 20170131_0330_UTC.pln 20170131_1012_UTC.pln
20170131_1312_UTC.pln 20170131_1919_UTC.pln
```

次のタスク

プランファイルをアーカイブに保存する頻度をスケジュールします。アーカイブの構成およびプランファイルの表示 （18 ページ） を参照してください。

アーカイブ内のプランファイルの管理

アーカイブが構成され、アーカイブディレクトリが作成されたことを確認します。詳細については、「アーカイブの構成 （48 ページ）」を参照してください。

（注） WAE UI を使用して、アーカイブを構成し、プランファイルを表示できます。アーカイブの構成およびプランファイルの表示 （18 ページ） を参照してください。

アーカイブでは、次のタスクを実行できます。

- すべてのプランファイルを一覧表示するには：

  (config)# networks network *<network_model_name>* plan-archive list

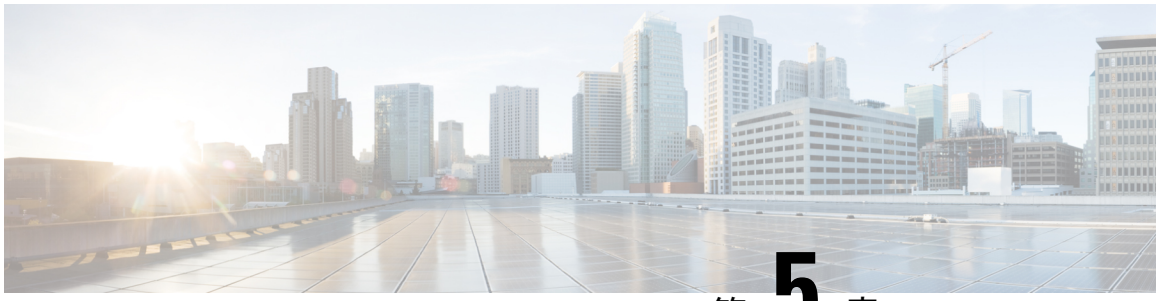
- ネットワークモデルをアーカイブに保存するには：

  (config)# networks network *<network_model_name>* plan-archive run

- プランファイルを取得するには：

  (config)# networks network *<network_model_name>* plan-archive get

（注） 既存のプランファイルを使用してネットワークモデルを作成できます。プランファイルのロード （47 ページ） を参照してください。



第 5 章

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 （61 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（55 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（58 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 （27 ページ） 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 （65 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 （71 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（70 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 （28 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
マルチレイヤ収集の構成 （93 ページ）	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 （90 ページ） を参照してください 。
NSO NED を使用した LSP 収集 （65 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 （73 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 （69 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 （58 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 （74 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 （61 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 （63 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 （75 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（73 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 （65 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 （78 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 （80 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 （79 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 （82 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 （61 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 （25 ページ）」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 （25 ページ） 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 （56 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 （51 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 （24 ページ）およびネットワーク アクセスの設定 （25 ページ）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 （27 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。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 （59 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 （51 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 （4 ページ）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 （123 ページ）」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 （62 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 （55 ページ）およびマルチレイヤ収集の構成 （93 ページ）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 （58 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 （25 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_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 （67 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 （145 ページ）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 （66 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 （58 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 （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 **[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 （25 ページ） 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 （28 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_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 （71 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 （51 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（25 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_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 （72 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 （55 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 （55 ページ）またはXTC を使用した BGP-LS トポロジ収集 （58 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 （65 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 （73 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 （77 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 （77 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（55 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 （55 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 （83 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。

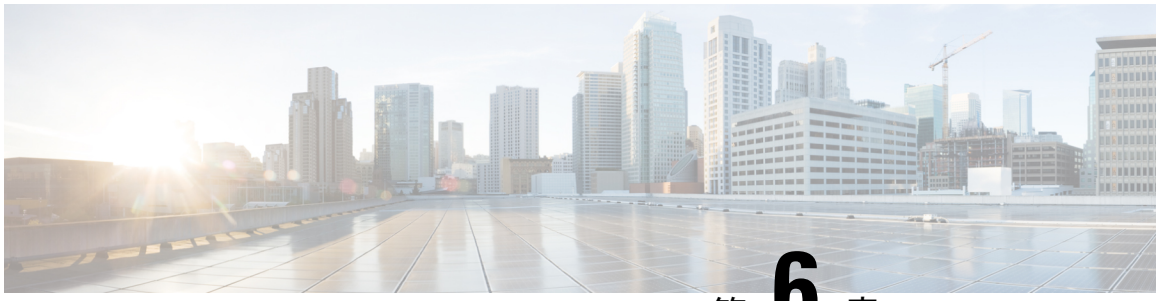
```
admin@wae(config)# networks network net_dest nimo external-executable-nimo source-network
 net_src
advanced argv [ /usr/bin/python /home/user1/srcs/br1/mate/package/linux/run/ext_exe_eg.py
 $$input $$output ]
admin@wae(config-network-net_dest)# commit
Commit complete.
admin@wae(config-network-net_dest)#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

!
!



第 6 章

WAE モデリングデーモン（WMD）の構成

WMD は、メモリ内にリアルタイムのネットワークモデルを提供します。DARE は（NIMO および XATP モジュールから）ネットワークの変更を受け取り、これらの変更を含むパッチを WMD に送信します。WMD および DARE の動作の詳細については、概要 （1 ページ）の章を参照してください。

DARE、WMD、および XATP モジュールを構成するには、次のトピックを参照してください。

WAE モデリングデーモン（WMD）の構成

WMD は、メモリ内のネットワークのほぼリアルタイムの表現（モデル）を提供して、アプリケーションがそのモデルにアクセスできるようにします。これは、DARE、XTC Agent to Patch モジュール、および NIMO からのスケジュールされたフィードから、変更を取得します。

始める前に

次の情報が手元にあるか、構成されている必要があります。

- 最終ネットワークモデル名
- 設計 RPC
- 継続的なポーリングを使用している場合は、traffic-poll-nimo ネットワークの名前

ステップ 1 エキスパートモードから、**/wae:wae/components/wmd:wmd** に移動し、[config] をクリックします。
ステップ 2 [network-name] ドロップダウンリストから、最終ネットワークモデルを選択します。
ステップ 3 [enable] ドロップダウンリストから [true] を選択して WMD を有効化します。
ステップ 4 [rpc-connection] をクリックして、設計 RPC 値を入力します。

ステップ 5 [app-subscriber-connections] をクリックし、すべての自動化アプリケーション接続のホストとポートの情報を入力します。

ステップ 6 [measured-traffic-source] をクリックし、継続的なポーリング情報を入力します。

ステップ 7 [dare] をクリックして、次の値を入力します。

- [dare-destination]：最終ネットワークモデルを選択します。
- [connection-attempts]：接続が再確立されるまでの再接続の試行回数を入力します。
- [connection-retry-delay]：接続試行の間隔（秒単位）を入力します。

ステップ 8 （オプション）デマンドのメッシュと推論を有効にするには、[demands] をクリックして次の値を入力します。

- [add-demands]：デマンドを有効にするには、[true] を選択します。有効にすると、WMD は、WMD を使用するすべてのアプリケーションに対してデマンドメッシュとデマンド推論を実行するように設定されます。したがって、継続的なポーラーが WMD を更新すると、WMD はデマンドをトリガーします。
- [demand-mesh-config]：該当するデマンドメッシュオプションを入力します。フィールドの詳細については、デマンドメッシュの作成 （80 ページ）を参照してください。
- [demand-deduction-config]：該当するデマンドメッシュオプションを入力します。フィールドの詳細については、デマンド推論 （79 ページ）を参照してください。

例

WAE CLI（構成モード）の例：

```
# wae components wmd config network-name <final_model_name> dare dare-destination
<final_model_name>
# wae components wmd config network-name <final_model_name> demands add-demands true
demand-mesh-config dest-equals-source true
```

XTC Agent to Patch モジュールの構成

XTC Agent to Patch（XATP）モジュールは XTC エージェントに接続し、XTC の変更または PCEP/LSP をパッチとして DARE に送信します。構成の一部として、WMD をポイントして最新の WAE モデルを取得する必要があります。

始める前に

次の情報が手元にあるか、構成されている必要があります。

- 最終ネットワークモデル名
- XTC エージェント
- WMD

ステップ 1 エキスパートモードから、**/wae:wae/components/xatp:xatp** に移動し、[config] をクリックします。

ステップ 2 [enable] ドロップダウンリストから [true] を選択して XATP を有効化します。

ステップ 3 [xtc-agent] ドロップダウンリストから送信元 XTC エージェントを選択します。

ステップ 4 [WMD] をクリックして、次の値を設定します。

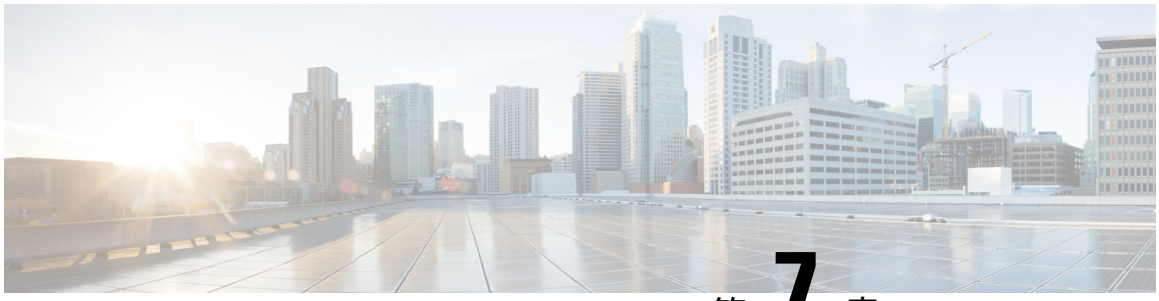
- [host]：関連付けられた WMD インスタンスを入力します。
- [port]：WMD ポートを入力します。
- [connection-attempts]：接続が失われた場合に、XATP が WMD に再接続を試行する回数を入力します。0 を入力すると、XATP は接続が再確立されるまで再接続を試みます。
- [connection-retry-delay]：接続試行の間隔（秒単位）を入力します。

ステップ 5 [dare] をクリックします。パッチの送信先となる最終ネットワークモデルと、送信元 NIMO を選択します。

例

WAE CLI（構成モード）の例：

```
# wae components xatp config wmd connection-attempts 0
# wae components xatp config xtc-agent <xtc-agent-name>
# wae components xatp config dare aggregator-network <aggregator-network-name>
# wae components xatp config dare topo-bgpls-xtc-nimo-network
<topo-bgpls-xtc-nimo-network-name>
# wae components xatp config dare pcep-lsp-xtc-nimo-network
<pcep-lsp-xtc-nimo-network-name>
# wae components xatp config enable true
```



第 7 章

マルチレイヤ収集

マルチレイヤ（レイヤ 1 およびレイヤ 3）ネットワーク収集は、高度な収集構成です。この項では、マルチレイヤネットワークから収集を構成する方法について説明します。

（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。

この手順の後、次の情報を収集してモデル化できるはずです。

- 非ユーザー ネットワーク インターフェイス（UNI）回路で Generalized Multiprotocol Label Switching（GMPLS）をサポートする DWDM ネットワークのトポロジ
- L1 回路パス
- 増幅器の有無にかかわらず L1 トポロジ
- 保護されていない復元可能なパス
- 実際の L1 回路パスホップ
- フィージビリティのメトリックと制限
- 非アクティブ L1 リンク
- L1 ノードおよび L1 リンク SRLG
- サイト情報
- ユーザ プロパティ（User properties）
- エイジング情報と前回表示日。エージングを構成するには、エージングの構成（138 ページ）を参照してください。

ここでは、次の内容について説明します。

- マルチレイヤ収集の構成 （93 ページ）

マルチレイヤ収集の制限事項

マルチレイヤ（L1-L3）収集には次の制限事項が存在します。

- 収集は、次のプラットフォームでのみサポートされています。
  - L1 デバイス用のバージョン 10.61 を実行している Cisco Network Convergence System（NCS）2000 プラットフォーム。
  - Cisco アグリゲーション サービス ルータ（ASR）9000、Cisco Carrier Routing System（CRS）、および L3 デバイス用の IOS-XR を実行している Cisco NCS 5500 プラットフォーム。
- マルチレイヤ収集は、保護されていない回路の収集に限定されます。
- 非 WSON 回路の収集はサポートされていません。
- LMP による L3-L1 マッピングは、コントローラ インターフェイス名が実際の L3 インターフェイス名と同じ場合、または「dwdmx/x/x/x」の形式（「x/x/x/x」の添え字が対応する L3 インターフェイスのものと一致）の場合にのみサポートされます。
- Lambda マッピングは現在、回路パスに対してのみサポートされ、パスホップに対してはサポートされていません。

マルチレイヤ収集のワークフロー

このワークフローでは、マルチレイヤ収集を構成するための手順の概要について説明します。

ステップ	詳細
1. マルチレイヤ収集の制限事項を確認します。	マルチレイヤ収集の制限事項 （90 ページ）
2. オプティカルプラグインを構成して実行します。	オプティカルプラグインの構成 （91 ページ）
3. L1 - L3 マッピング情報を取得して構成します。	L3-L1 マッピング情報の構成 （91 ページ）
4. マルチレイヤ収集を構成します。	マルチレイヤ収集の構成 （93 ページ）

L3-L1 マッピング情報の構成

（注） リンク管理プロトコル（LMP）による L3-L1 マッピングは、コントローラ インターフェイス名が実際の L3 インターフェイス名と同じ場合、または「dwdmx/x/x/x」の形式（「x/x/x/x」の添え字が対応する L3 インターフェイスのものと一致）の場合にのみサポートされます。

構成解析エージェントを実行して L3-L1 情報を取得できます。構成解析エージェントの構成（28 ページ） を参照してください。構成解析エージェントは、オプティカル NIMO で networks/*<multilayer_network_name>*/nimo/optical-nimo/advanced/cfg-parse-agent のように指定する必要があります。

構成解析エージェントを実行していない場合は、L3 ノードとインターフェイスの L1 ノードとポートへのマッピングを手動で入力します。

1. エキスパート モードから、**/wae:wae/nimos** に移動し、[l3-l1-mappings] タブをクリックします。
2. プラス（[+]）記号をクリックし、L3 - L1 マッピンググループの任意の名前を入力して、[追加（Add）] をクリックします。
3. [l3-l1-mapping] タブをクリックし、プラス（[+]）記号をクリックして各マッピングを入力します。この手順を繰り返して、すべての L3 - L1 マッピングを入力します。
4. [コミット（Commit）] ボタンをクリックします。

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@ncs(config)# wae nimos l3-l1-mappings l3-l1-mappings <mapping_name> l3-l1-mapping
<l3_node> <l3_interface> <l1_node> <l1_interface>
wae@ncs(config)# networks network <ml_network> optical-nimo optical-agents [<agent_name>]
 advanced use-configured-l3-l1-mapping true l3-l1-mapping <mapping_name>
wae@ncs(config-networks-<ml_network_name>)# commit
```

値の使用例：

```
# wae nimos l3-l1-mappings l3-l1-mappings bgl-phys
l3-l1-mapping Samurai-1 TenGigE0/0/1/0 SIte1_IP40 OCH_PORT:Unit-2/81
!
l3-l1-mapping Samurai-1 TenGigE0/0/1/1 SIte1_IP40 OCH_PORT:Unit-2/79
# commit
```

オプティカルプラグインの構成

オプティカルプラグインにより、ネットワーク内のオプティカル情報を収集できます。オプティカルプラグインの構成ファイルは、正しいログイン情報とシードレイヤ 1 ノードを使用して更新する必要があります。オプティカルプラグインは、シードノードへの接続を開始して、オプティカルネットワークの詳細を取得します。

（注） HTTPS サポートをアクティブ化するには、**config/profile.properties** ファイルの次のキーで https を設定し、http を削除します：# HTTP instead HTTPS Profile spring.profiles.active=dwdm,https。**config/spring-jetty.xml** ファイルには、https の構成プロファイルが含まれています。ここで、restconf リクエストをプラグインに送信できるポートを構成する必要があります。

ステップ 1 CLI から、*<wae-installation-directory>*/packages/optical-ctc-plugin/config/optical-ctc-plugin.properties ファイルを次の情報で編集して保存します。

- **network.id**：オプティカルネットワーク名。<*optical_network_name*>:network のように入力する必要があります。たとえば、cisco:network などです。
- **network.nodes.vendor**：ノードベンダー。
- **restconf.http.port** および **restconf.https.port**：L1 ネットワークが実行されているポート番号。デフォルトでは、WAE は http ポート 9000 および https ポート 8445 にリクエストを送信して情報を収集し、オプティカルネットワークに展開します。
- **network.discovery.start.node**：検出シードノードのレイヤ 1 IP アドレス。
- **network.discovery.start.node.login**：シードノードにログインするための ID。
- **network.discovery.start.node.password**：シードノードにアクセスするためのパスワード。セキュリティを強化するために、ログイン情報を WAE でデバイスログイン情報として入力し（認証グループを作成）、オプティカルエージェントの構成中に選択することができます。
- **network.discovery.inactivity.period**：ネットワークへのアクセスがない場合に検出がタイムアウトする時間（ミリ秒）。

（注） この情報は、後でマルチレイヤ収集（optical-nimo）を構成するときにも必要になります。

ステップ 2 オプティカル プラグイン スクリプトを実行します。

```
# ./run.sh
```

例

オプティカルプラグイン構成ファイルの例：

```
network.id=cisco:network
network.nodes.vendor=cisco

restconf.http.port=9000

network.discovery.start.node=10.89.204.17
network.discovery.start.node.login=CISCO15
network.discovery.start.node.password=pwcisco+11
network.discovery.inactivity.period=5000
```

マルチレイヤ収集の構成

このタスクでは、次のことを行います。

- L3 および L1 ネットワークモデルを構成します。アグリゲータの構成が完了するまで、これらの収集を実行しないでください。
- オプティカルエージェントを選択、構成、および実行します。
- L3 および L1 収集を最終ネットワークモデルに統合するようにアグリゲータを構成します。
- L3 および L1 収集を実行します。
- L3 および L1 収集が最終ネットワークモデルに統合されていることを確認します。

始める前に

マルチレイヤ収集のワークフロー （90 ページ）のすべての準備作業が完了したことを確認します。

ステップ 1 次のインターフェイスオプションを **true** に設定して、L3 IGP トポロジネットワークモデルを構成しますが、収集は実行しません。

- lag
- get-physical-ports

（注） 詳細については、「IGP トポロジ収集 （55 ページ）」を参照してください。

ステップ 2 **/wae:wae/agents** に移動し、[optical-agents] をクリックします。

ステップ 3 [追加（+）（Add (+)）] をクリックし、エージェント名を入力します。

ステップ 4 [agent-type] ドロップダウンリストから [Cisco] を選択します。

ステップ 5 [cisco-optical-agent] をクリックし、[optical-plugin-config] タブをクリックします。

（注） オプティカルエージェント名は、オプティカルプラグイン構成ファイルで構成された「network.id」と一致する必要があります。オプティカルプラグインの構成 （91 ページ） を参照してください。

ステップ 6 次の値を入力します。

この情報は、オプティカルプラグイン構成ファイルで構成されています。

- [optical-plugin-ip]：オプティカルプラグインがインストールされている場所の IP アドレスを入力します。
- [optical-plugin-port]：オプティカルプラグインが実行されている http または https ポートを入力します。
- [optical-plugin-protocol]：[http] または [https] を選択します。
- [seed-node]：シードノードの IP アドレスを入力します。

- [seed-node-access]：適切なアクセスグループを選択します。

（注） [advanced] タブをクリックして、再試行の回数と間隔の長さ、データの記録（net-recorder が record に設定されている場合、ファイルは net-record-file が設定されているディレクトリに保存されます）、接続タイムアウト設定などのオプションを表示します。

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 Lambda ID マッピング（チャネル ID、中心周波数、または波長を Lambda ID にマッピングするかどうかを設定できます）を利用する場合は、Lambda ID 構成ファイルをロードする必要があります。次のコマンドを入力します。

```
# ncs_load -lmj /wae/agents/optical-agents/optical-agent <agent-name>/lambda-mappings
```

ステップ 9 [cisco-optical-agent] タブに移動し、**[run-optical-collection]** > **[run-optical-collectionの呼び出し（Invoke run-optical-collection）]** をクリックします。

ステップ 10 L1 オプティカル収集ネットワークモデルを作成します。

a) **/wae:networks** に移動します。
b) プラス（[+]）記号をクリックし、使用される L1 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名と NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_L1 などです。
c) [追加（Add）] をクリックします。
d) [nimo] タブをクリックします。
e) [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、[optical-nimo] を選択します。
f) [optical-nimo] をクリックして、次の情報を入力します。

- [source-network]：いずれかのトポロジ NIMO を使用して収集された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 グループを選択します。

ステップ 11 [optical-agents] タブをクリックし、作成されたエージェントを追加します。

ステップ 12 [advanced] タブをクリックして次の構成を行います。

- [cfg-parse-agent]：L1-L3 マッピングに使用された場合の構成解析エージェント名を選択します。
- [lag]：構成解析エージェントを使用する場合は [true] を選択します。
- [lmp]：構成解析エージェントを使用する場合は [true] を選択します。
- [retain-amplifiers]：増幅器を収集の一部として含める場合は [true] を選択します。
- [map-lambdas]：[true] に設定すると、Lambda マッピング値（Lambda ID、チャネル ID、中心周波数、および波長）を表示するユーザーテーブルが作成されます。その後、これらの値はプルして、WAE Design に表示できます。96 チャネルをサポートする L1 リンクを持つネットワークから情報を収集する場合、この値を [true] に設定することを推奨します。
- [map-lambda-id-to]：チャネル ID、中心周波数、または波長を Lambda ID にマッピングするかどうかを設定します。プランファイルがインポートされると、このフィールドは WAE Design によって [L1 回路パス（L1 Circuit Paths）] テーブルに表示されます。
- [use-configured-l3-l1-mapping]：l3-l1 マッピングを手動で構成した場合は、[true] を選択します（L3-L1 マッピング情報の構成（91 ページ）を参照）。構成解析エージェントを実行している場合は、[false] を選択します。

- [l3-l1-mapping]：以前に設定したl3-l1マッピンググループを選択します。構成解析エージェントを実行している場合は、何も選択しないでください。

ステップ 13 [コミット（Commit）] ボタンをクリックします。

ステップ 14 作成したばかりのL1およびL3ネットワークモデルを統合するようにアグリゲータを構成します。適切な送信元ネットワークからデータを選択するためのアグリゲータルールの例を参照してください。この手順の残りのCLI構成例を表示するには、アグリゲータとマルチレイヤ収集のCLI構成例（62ページ）を参照してください。

a) 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。たとえば、networkABC_L3L1です。また、展開する場合は、アグリゲータNIMOを選択して送信元を追加します。

b) [/wae:wae/components/dare:aggregators/aggregator] タブに移動します。

c) プラス（[+]）記号をクリックし、[destination] ドロップダウンリストから作成したばかりのマルチレイヤネットワーク（networkABC_L3L1）を選択します。

d) [sources] タブで [source] をクリックし、収集を結合するL1およびL3ネットワークモデルを追加します。

e) [確定する（Commit）] をクリックします。

ステップ 15 L3収集を実行します。

a) **/wae:networks/network/<*network-name*> nimo/topo-igp-nimo** に移動します。

b) [topo-igp-nimo] タブで、[run-collection] をクリックします。

ステップ 16 L1収集を実行します。

a) **/wae:networks/network/<*network-name*> nimo/optical-nimo** に移動します。

b) [optical-nimo] タブから、L1送信元ネットワークを選択し、[build-optical-topology] をクリックします。

ステップ 17 マージが成功したことを確認するには、WAE Designからネットワークを開きます（**[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）。

例

次に、（WAE CLIからの）アグリゲータルールの例を示します。

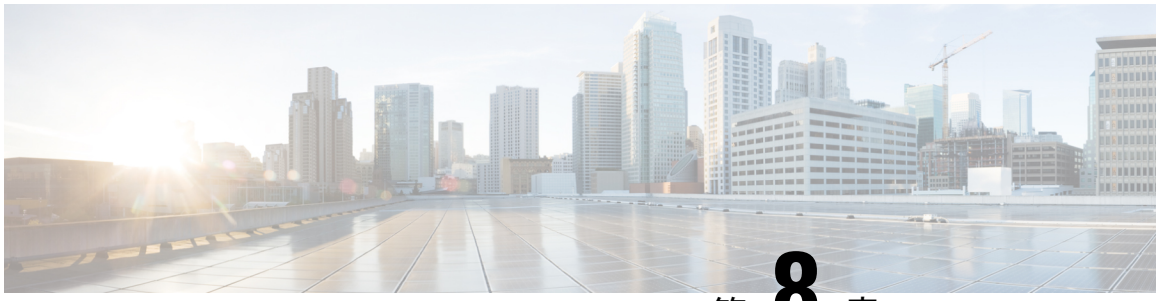
```
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/area source-ownership source [not(../ml-collected)]
 source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/area source-ownership source [../ml-collected]
source l1-source operation collect_deploy
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/igp-metric source-ownership source
[not(../ml-collected)] source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/igp-metric source-ownership source [../ml-collected]
 source l1-source operation collect_deploy
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/ip-addresses/ip-address/ip-address source-ownership
```

```
 source [not(../ml-collected)] source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/ip-addresses/ip-address/ip-address source-ownership
 source [../ml-collected] source l1-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/ip-addresses/ip-address/prefix-length
source-ownership source [not(../ml-collected)] source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/ip-addresses/ip-address/prefix-length
source-ownership source [../ml-collected] source l1-source operation collect_deploy
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/isis-level source-ownership source
[not(../ml-collected)] source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/isis-level source-ownership source [../ml-collected]
 source l1-source operation collect_deploy
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/name source-ownership source [not(../ml-collected)]
 source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/interfaces/interface/name source-ownership source [../ml-collected]
source l1-source operation collect_deploy
networks network final-network nimo aggregator rules rule
model/nodes/node/ports/port/interface/interface-name source-ownership source
[not(../../l1-port/l1-port-name)] source l3-source operation collect_only
networks network final-network nimo aggregator rules rule
model/nodes/node/ports/port/interface/interface-name source-ownership source
[../../l1-port/l1-port-name] source l1-source operation collect_deploy
```

ここで、`final-network` は最終ネットワーク、`l1-source` は optical-nimo ネットワーク、`l3-source` は topo-igp-nimo ネットワークです。インターフェイスに関する上記のルールは、オプティカル nimo によってのみ入力されるカスタム [ml-collected] フィールドに基づいていることに注意してください。これらのルールは、アグリゲータの送信元ネットワークが 2 つ（topo-igp-nimo ネットワークと optical-nimo ネットワーク）しかない場合にのみ適用されます。ルールの詳細については、Cisco WAE の担当者にお問い合わせください。

次のタスク

WAE Design で、プランファイルを開いて L1 および L3 トポロジを表示できます。



第 8 章

NetFlow データ収集

ここでは、次の内容について説明します。

NetFlow データ収集

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

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

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

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

NetFlow 収集アーキテクチャ

フロー収集アーキテクチャには 2 つのタイプがあります。

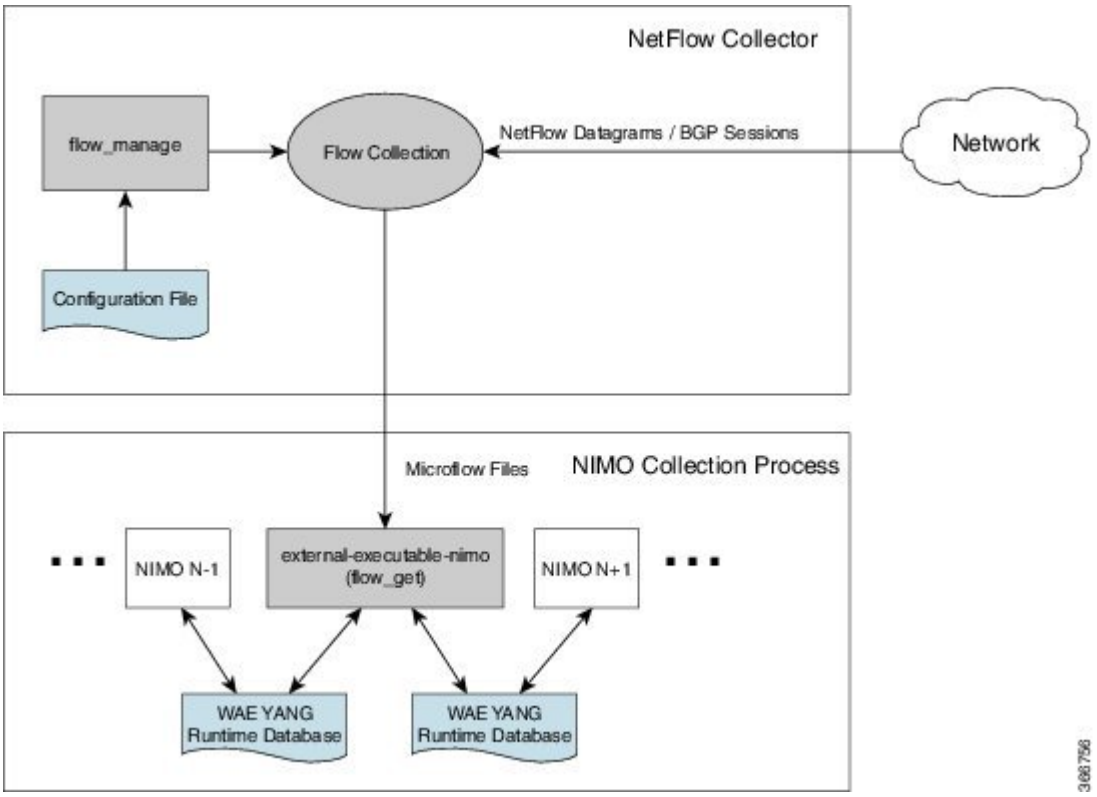
（注） 展開する収集アーキテクチャは、ネットワークからの NetFlow トラフィックエクスポートの測定レートまたは推定レート（Mbps または fps 単位）によって異なります。

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

CNF 収集

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

図 *3:* 一元的な収集とデマンドの作成



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

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

DNF 収集

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

図 *4 : DNF* アーキテクチャ

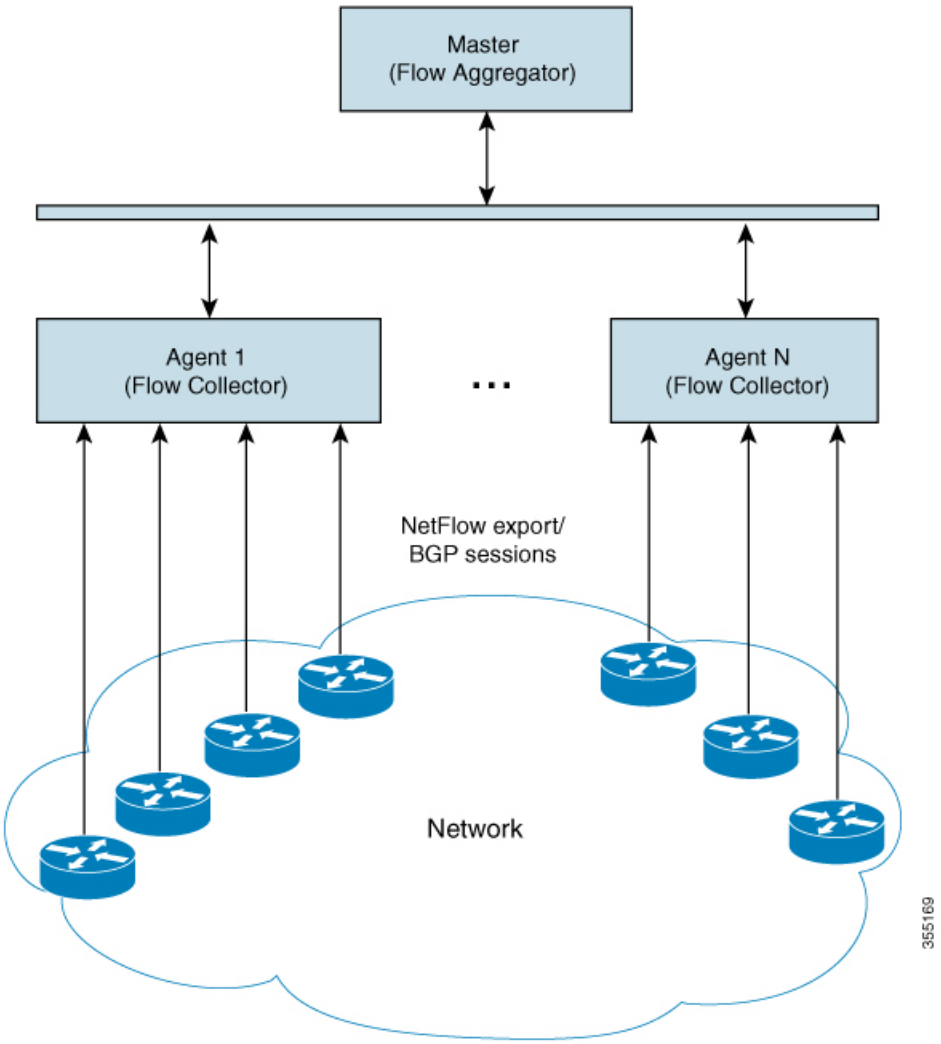
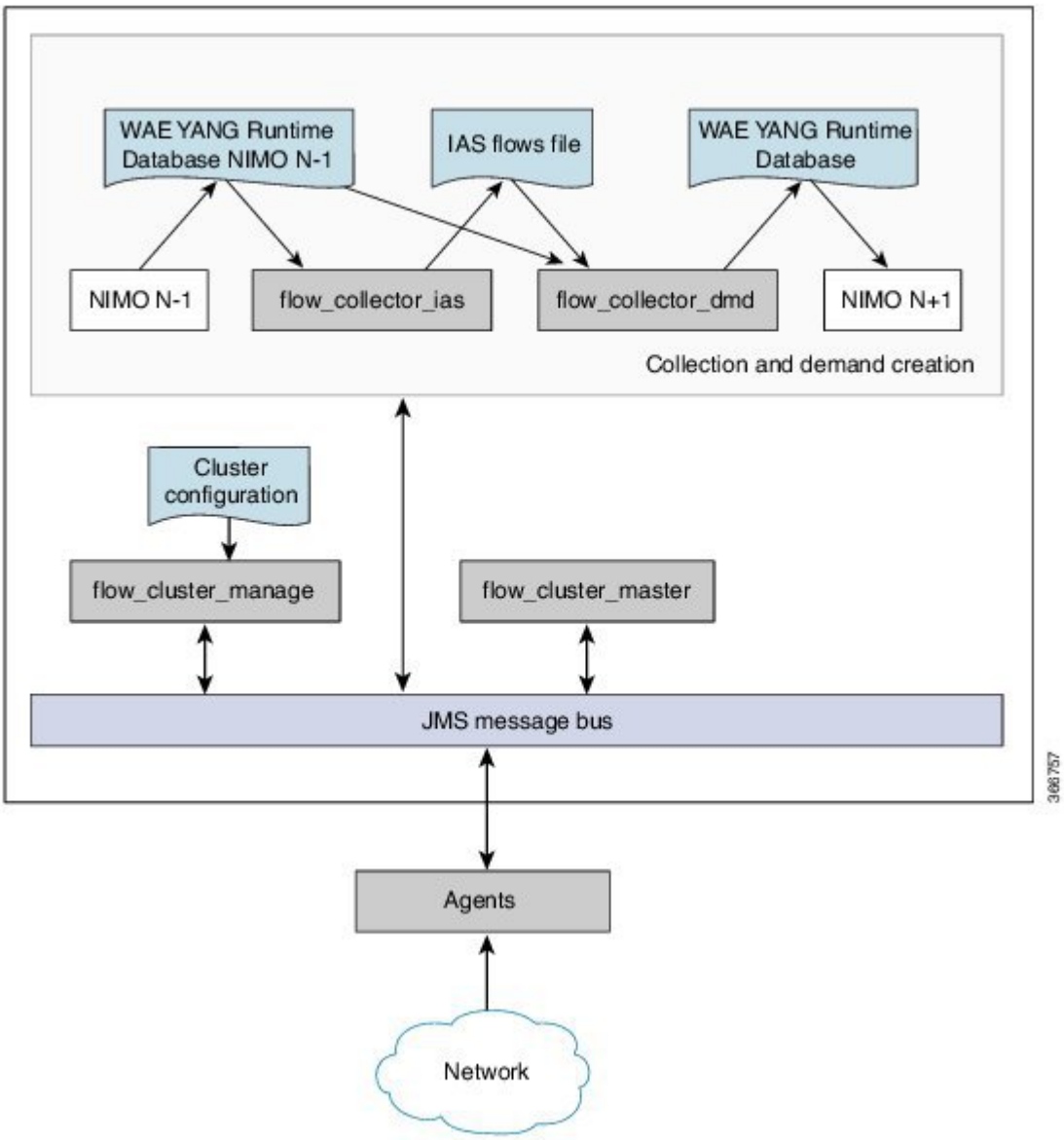


図 *5 : DNF* 収集ワークフロー



- **flow_cluster_manage**：このCLIツールは、クラスタの構成とステータスの取得に使用されます。クラスタ構成ファイルを受け取り、構成をクラスタに送信します。詳細については、「DNF 構成ファイルの使用（flow_cluster_manage の実行）（118 ページ）」を参照してください。

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

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

たとえば、クラスタ構成を取得するには：

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

たとえば、クラスタ構成を設定するには：

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

たとえば、クラスタのステータスを取得するには：

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

- **flow_cluster_master**：マスターサービスは、すべてのエージェントからのすべてのフローデータ結果を収集し、データを集約して、flow_collector_ias に返します。詳細については、「マスターとエージェント （109 ページ）」を参照してください。
- **flow_cluster_agent**：エージェントサービスは、関連付けられたフローコレクタのステータスを管理および追跡します。各エージェントは、対応する収集サーバーからフローデータを受信して計算します。
- **flow_cluster_broker**：（図には示されていない）JMS ブローカサービスは、マスターとエージェントを含むアーキテクチャ内のすべてのコンポーネント間の通信を可能にします。詳細については、「Java メッセージサーバー（JMS）ブローカ （109 ページ）」を参照してください。
- **flow_collector_ias**：この CLI ツールは、nimo_flow_collector_ias_and_dmd.sh ファイル内で構成され、external-executable-nimo 内で実行され、マスターからフローデータを受信し、IAS フローファイルを生成します。詳細については、「flow_collector_ias および flow_collector_dmd の構成 （120 ページ）」を参照してください。
- **flow_collector_dmd**：この CLI ツールは、NetFlow デマンドとデマンドトラフィックを WAE YANG ランタイムデータベースに送信します。nimo_flow_collector_ias_and_dmd.sh ファイル内で構成され、external-executable-nimo 内で実行されます。

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

集中型 NetFlow 構成ワークフロー

CNF を構成して収集を開始するには、次の手順を実行します。

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

ステップ 1 CNF NetFlow の要件 （103 ページ）が満たされていることを確認します。
ステップ 2 CNF 用のオペレーティングシステムの準備 （103 ページ）
ステップ 3 CNF 構成ファイルの作成 （104 ページ）
ステップ 4 CNF 構成ファイルの使用（flow_manage の実行） （105 ページ）
ステップ 5 CNF 収集の構成 （106 ページ）
a) flow_get の構成 （106 ページ）
b) CNF 用の external-executable-nimo の構成 （107 ページ）

CNF NetFlow の要件

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

ライセンシング

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

CNF 用のオペレーティングシステムの準備

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

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

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

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

（注） このコマンドの実行後、サーバーを再起動する必要があります。

NetFlow 収集の構成

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

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

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

CNF 構成ファイルの作成

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

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

表 1 : <NodeFlowConfigs> テーブルの列

カラム	説明
名前	ノード名

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

以下は、<NodeFlowConfigs> テーブルの例です。

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

CNF 構成ファイルの使用（flow_manage の実行）

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

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

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

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

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

サンプル構成ファイル

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

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

CNF 収集の構成

flow_get の構成

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

編集する前に、このファイルの権限を変更します。

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

`nimo_flow_get.sh` を次のように編集します。

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

`nimo_flow_get.sh` の例：

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

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

CNF 用の external-executable-nimo の構成

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

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他のNIMO収集を含む最終ネットワークモデルです。
- 集中型NetFlow構成ワークフロー（102ページ）の準備作業が完了したことを確認します。

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

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

ステップ 3 [nimo] タブをクリックします。

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

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

ステップ 6 [advanced] タブで、以下の情報を入力します。

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

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

例

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

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

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

次のタスク

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

DNF NetFlow 構成ワークフロー

DNF を構成して収集を開始するには、次の手順を実行します。

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

ステップ 1 分散 NetFlow の要件 （108 ページ）が満たされていることを確認します。

ステップ 2 DNF クラスタのセットアップ （110 ページ）

a) DNF 構成ファイルの変更 （110 ページ）

b) DNF クラスタの展開 （114 ページ）

ステップ 3 DNF クラスタの構成 （115 ページ）

a) DNF クラスタ構成ファイルの作成 （115 ページ）

b) DNF 構成ファイルの使用（flow_cluster_manage の実行） （118 ページ）

ステップ 4 DNF 収集の構成 （120 ページ）

a) flow_collector_ias および flow_collector_dmd の構成 （120 ページ）

b) DNF 用の external-executable-nimo の構成 （121 ページ）

分散 NetFlow の要件

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

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

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

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

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

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

- フロー収集プロセスは、入力方向のルータによってキャプチャおよびエクスポートされる IPv4 および IPv6 フローをサポートしています。また、IPv4 および IPv6 iBGP ピアリングもサポートしています。ルータは、フローをフロー収集サーバーにエクスポートし、フロー収集サーバーとの BGP ピアリングを確立するように構成する必要があります。詳細については、「NetFlow 収集の構成 （104 ページ）」を参照してください。

ライセンシング

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

Java メッセージサーバー（JMS）ブローカ

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

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

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

マスターとエージェント

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

Master

マスターノードは、クラスタ内で次のサービスを提供します。

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

- クラスタからの構成およびステータスリクエストを処理します。

エージェント（Agents）

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

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

DNF クラスタのセットアップ

DNF 構成ファイルの変更

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

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

group_vars/all

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

次のオプションを編集します。

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

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

例（コメントは削除）：

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

hosts

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

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

次に例を示します。

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

prepare-agents.yml

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

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

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

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

startup.yml

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

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

次に例を示します。

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

service_conf

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

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

次のオプションを編集します。

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

オプション	説明

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

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

次に例を示します。

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

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

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

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

DNF クラスタの展開

DNF クラスタを展開するには、次の手順を実行します。

ステップ 1 ブローカ、マスター、およびエージェントをインストールします。

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

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

ステップ 2 DNF のエージェントを準備して再起動します。

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

ステップ 3 マスター、ブローカ、およびエージェントを開始します。

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

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

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

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

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

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

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

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

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

DNF クラスタの構成

DNF クラスタ構成ファイルの作成

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

次に例を示します。

ステップ 1 テンプレート構成ファイルを生成します。

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

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

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

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

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

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

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

2 番目のエージェントの情報はここから始まります。

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

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

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

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

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

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

（注） クラスタの構成には約 1 分かかります。

クラスタステータスの結果例：

```
CLUSTER STATUS - BEGIN

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

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

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

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

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

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

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

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

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

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

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

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

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

CLUSTER STATUS - END
```

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

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

DNF 収集の構成

flow_collector_ias および flow_collector_dmd の構成

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

編集する前に、このファイルの権限を変更します。

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

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

`nimo_flow_collector_ias_dmd.sh` の例：

```
#!/bin/bash

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

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

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

DNF 用の external-executable-nimo の構成

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

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他の NIMO 収集を含む最終ネットワークモデルです。
- DNF NetFlow 構成ワークフロー （108 ページ）の準備作業が完了したことを確認します。

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

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

ステップ 3 [nimo] タブをクリックします。

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

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

ステップ 6 [advanced] タブで、以下の情報を入力します。

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

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

例

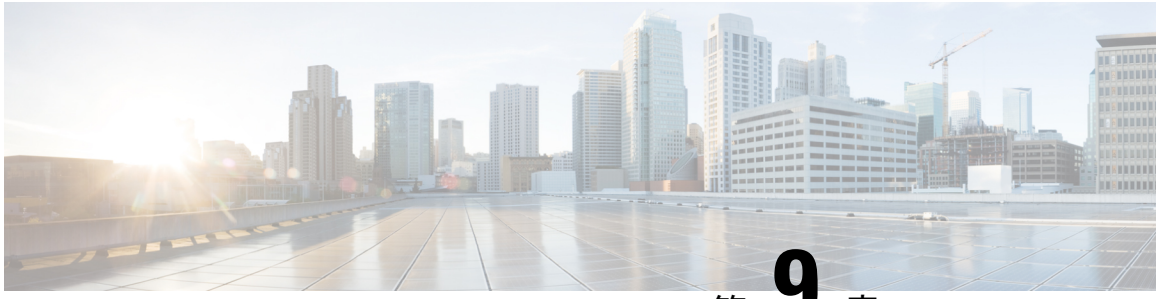
WAE CLI を（構成モードで）使用している場合は、次のように入力します。

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

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

次のタスク

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



第 9 章

自動化アプリケーション

ここでは、次の内容について説明します。

自動化アプリケーション

自動化アプリケーションは、リアルタイムのネットワークモデルを使用することに依存しています。アプリケーションは、最新のネットワークモデル（マスターモデル）のコピーを取得し、アプリケーションの目的または機能に基づいて最適化を実行するか操作します。

その後、WAE Design（**[ファイル（File）] > [開く場所（Open From）] > [WAEモデリングデーモン（WAE Modeling Daemon）]**）を使用して、ネットワークモデルを表示できます。

オンデマンド帯域幅の構成ワークフロー

オンデマンド帯域幅アプリケーションは、新しいサービスの影響をモデル化し、予測します。このアプリケーションは、永続的な帯域幅と特定の IGP または TE メトリック需要を必要とする新しいサービスをプロビジョニングするときに使用されます。アプリケーションは、ネットワークで委任されているSRポリシーのパスを見つけます。オンデマンド帯域幅アプリケーションの詳細については、オンデマンド帯域幅アプリケーション （7 ページ）を参照してください。

このワークフローでは、オンデマンド帯域幅およびその他のコンポーネントを構成するための構成手順の概要について説明します。

（注） オンデマンド帯域幅を有効にする前に、Bandwidth Optimizationアプリケーションが実行されていないことを確認してください。両方のアプリケーションを同時に実行することはできません。

手順	詳細
1. デバイス認証グループと SNMP グループの構成	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 （24 ページ）。
2. ネットワーク アクセス プロファイルの構成	ネットワーク アクセスの設定 （25 ページ）
3. XTC エージェントの構成	エキスパートモードを使用した XTC エージェントの構成 （27 ページ）
4. アグリゲータの構成	NIMO 収集の統合 （61 ページ）
5. WAE モデリングデーモン（WMD）の構成	WAE モデリングデーモン（WMD）の構成 （85 ページ）
6. XTC Agent to Patch（XATP）モジュールの構成	XTC Agent to Patch モジュールの構成 （86 ページ）
7. トポロジと追加の NIMO の実行	ネットワーク インターフェイス モジュール（NIMO） （51 ページ）
8. オンデマンド帯域幅アプリケーションと SR ポリシーの構成	オンデマンド帯域幅の設定 （125 ページ）
9. ネットワークモデルを開く	WAE Design を使用して、視覚的なネットワーク モデル レイアウトを取得できます。WAE Design から、**[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE Modeling Daemon）]** に移動し、最終ネットワークモデルを選択します。 （注） 初期構成後、いつでも NIMO を実行でき、オンデマンド帯域幅アプリケーションはネットワークモデルを更新します。

オンデマンド帯域幅の設定

始める前に

この手順では、オンデマンド帯域幅アプリケーションの構成オプションについて説明します。完全な構成ワークフローについては、オンデマンド帯域幅の構成ワークフロー（123ページ）を参照してください。ワークフロー全体の例については、初期オンデマンド帯域幅のCLI構成例（126ページ）を参照してください。

（注） Bandwidth Optimizationアプリケーションは、オンデマンド帯域幅アプリケーションと同時に実行できません。他のアプリケーションを使用するには、一方のアプリケーションを無効にする（`enable = false`）必要があります。Bandwidth Optimizationアプリケーションを適切にシャットダウンする方法については、帯域幅最適化のシャットダウン（131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:wae/components/bw-on-demand:bw-on-demand**に移動し、[config]タブをクリックします。

ステップ2 次の値を入力します。

- [xtc-host]：XTCホストのIPアドレスを入力します。このエントリは、REST APIを使用してXTCに接続する方法を決定します。これは永続的な接続であり、有効にすると、委任されたPCEP要求を受け取れるようになります。
- [xtc-port]：XTCポートを入力します。
- [keepalive]：キープアライブインターバルを入力します。オンデマンド帯域幅アプリケーションとXTCは、永続的な接続を維持するためにキープアライブメッセージを交換します。接続が失敗した場合、オンデマンド帯域幅アプリケーションはシャットダウンし、現在の状態をクリアして再接続を試み、SRポリシーを再委任します。
- [priority]：オンデマンド帯域幅アプリケーション インスタンスが複数ある場合は、このインスタンスの優先度を入力します。XTCは、優先度に応じてインスタンスを委任します。
- [util-threshold]：輻輳制約を入力します（パーセンテージ）。オンデマンド帯域幅アプリケーションは、委任されているポリシーのパスを検索するときに、輻輳使用率のしきい値を超える可能性のあるパスを回避します。
- [enable]：オンデマンド帯域幅アプリケーションを有効にするには、[true]を選択します。

（注） [advanced]オプションについては、Cisco WAEの担当者にお問い合わせください。

ステップ3 [コミット（Commit）]をクリックして、構成を保存します。

ステップ4 デバイスの帯域幅とIGPまたはTEメトリックタイプを使用して、新しいSRポリシーを構成します。特定のデバイス構成については、該当するCisco IOS XRのドキュメントを参照してください（たとえば、『Configure SR-TE Policies』）。デバイス構成の例：

```
segment-routing
    traffic-eng
        policy BWOD_2TO3_IGP
```

```
bandwidth 10000
color 100 end-point ipv4 198.51.100.3
candidate-paths
    preference 10
        dynamic mpls
            pce
                address ipv4 198.51.100.1
                exit
            metric
                type igp
```

ステップ 5 WAE Design（**[WAE Design]** > **[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE Modeling Daemon）]**）を使用して、得られたネットワークモデルを開きます。

初期オンデマンド帯域幅の CLI 構成例

以下は、Cisco Virtual Internet Routing Lab（VIRL）テスト環境内の初期オンデマンド帯域幅の CLI 構成の例です。初期構成後、いつでも NIMO を実行でき、オンデマンド帯域幅アプリケーションはネットワークモデルを更新します。

デバイスとネットワークの検出を構成します。

```
# config
# devices authgroups group virl_test default-map
# devices authgroups group virl_test default-map remote-name cisco
# devices authgroups group virl_test default-map remote-password cisco
# devices authgroups group virl_test default-map remote-secondary-password cisco
# devices authgroups snmp-group virl_test default-map
# devices authgroups snmp-group virl_test default-map community-name cisco
# wae nimos network-access network-access virl_test default-auth-group virl_test
# wae nimos network-access network-access virl_test default-snmp-group virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 ip-manage
192.0.2.131
# wae nimos network-access network-access virl_test node-access 198.51.100.2 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.2 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.2 ip-manage
192.0.2.132
# wae nimos network-access network-access virl_test node-access 198.51.100.3 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.3 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.3 ip-manage
192.0.2.133
# wae nimos network-access network-access virl_test node-access 198.51.100.4 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.4 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.4 ip-manage
192.0.2.134
# wae nimos network-access network-access virl_test node-access 198.51.100.5 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.5 snmp-group
```

```
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.5 ip-manage
192.0.2.135
# wae nimos network-access network-access virl_test node-access 198.51.100.6 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.6 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.6 ip-manage
192.0.2.136
# wae nimos network-access network-access virl_test node-access 198.51.100.7 auth-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.7 snmp-group
 virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.7 ip-manage
192.0.2.137
```

XTC エージェントを構成します。

```
# wae agents xtc xtc virl enabled xtc-host-ip 192.0.2.131
```

BGP ネットワーク（topo-bgpls-xtc-nimo）を構成します。

```
# networks network virl_bgpls nimo topo-bgpls-xtc-nimo xtc-host virl igp-protocol isis
extended-topology-discovery true backup-xtc-host virl network-access virl_test advanced
 nodes remove-node-suffix virl.info
```

LSP PCEP ネットワーク（lsp-pcep-xtc-nimo）を構成します。

```
# networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo xtc-hosts virl xtc-host virl
# networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo source-network virl_bgpls advanced
 sr-use-signaled-name true
```

アグリゲータが書き込むネットワークを設定します。

```
# networks network virl_final_model
```

継続的なポーリング（traffic-poll-nimo）を構成します。

```
# networks network virl_cp nimo traffic-poll-nimo network-access virl_max source-network
 virl_dare iface-traffic-poller enabled
# networks network virl_cp nimo traffic-poll-nimo lsp-traffic-poller enabled
# networks network virl_cp nimo traffic-poll-nimo advanced snmp-traffic-population
scheduler-interval 0
```

送信元ネットワークにサブスクライブするようにアグリゲータを構成します。

```
# wae components aggregators aggregator virl_final_model sources source virl_bgpls
# wae components aggregators aggregator virl_final_model sources source virl_pcep_lsp
```

WMD を構成します。この例では、WMD は、WMD を使用するすべてのアプリケーションに対してデマンドメッシュとデマンド推論を実行するように設定されます。したがって、継続的なポーラーが WMD を更新すると、WMD はデマンド推論をトリガーします。

```
# wae components wmd config network-name virl_final_model dare dare-destination
virl_final_model
# wae components wmd config network-name virl_final_model demands add-demands true
demand-mesh-config dest-equals-source true
```

XATP モジュールを構成します。この例では、connection-attempts パラメータは 0 に設定されています。つまり、WMD 接続が失敗した場合、アプリケーションは接続が成功するまで WMD への接続を試行し続けます。

```
# wae components xatp config xtc-agent virl dare aggregator-network virl_dare
pcep-lsp-xtc-nimo-network virl_pcep_lsp topo-bgpls-xtc-nimo-network virl_bgpls
wae components xatp config wmd connection-attempts 0
```

WMD と XATP を有効にします。

```
# wae components wmd config enable true
# wae components xatp config enable true
# commit
# exit
```

NIMO（ネットワーク収集）を実行します。

```
networks network virl_bgpls nimo topo-bgpls-xtc-nimo run-xtc-collection
networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo run-collection
```

オンデマンド帯域幅を構成します。

```
# configure
# wae components bw-on-demand config xtc-host 192.0.2.131 xtc-port 8080 util-threshold
90.0
# wae components bw-on-demand config advanced lsp-traffic max-simulated-requested
primary-objective min-metric private-new-lsps true
# commit
# exit
```

WAE Design（**[WAE Design]** > **[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE Modeling Daemon）]**）を使用して基本的なネットワークモデルを開き、結果のネットワークモデルを比較します（SR ポリシーが構成され、オンデマンド帯域幅アプリケーションが実行された後）。

デバイスで SR ポリシーを構成します。

```
# configure
# segment-routing
# traffic-eng
# policy BWOD_2TO3_IGP
# bandwidth 1000
# color 100 end-point ipv4 192.0.2.132
# candidate-paths
# preference 10
# dynamic mpls
# pce
# address ipv4 192.0.2.130
# exit
# metric
# type igp
# commit
# end
```

SR ポリシー構成がコミットされると、WMD が更新され、オンデマンド帯域幅アプリケーションは、輻輳抑制と IGP メトリックを考慮してベストパスを計算します。WAE Design（**[WAE Design]** > **[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE**

Modeling Daemon）]）を使用して結果として得られるネットワークモデルを開き、ベースライン ネットワーク モデルを新しいネットワークモデルと比較します。

オンデマンド帯域幅のシャットダウン

オンデマンド帯域幅アプリケーションを適切にシャットダウンするには、次の手順を順番に実行する必要があります。

ステップ 1 オンデマンド帯域幅を停止します。

```
# wae components bw-on-demand config enable false
# commit
```

ステップ 2 XTC Agent to Patch モジュールを停止します。

```
# wae components xatp config enable false
# commit
```

ステップ 3 WAE モデリングデーモンを停止します。

```
# wae components wmd config enable false
# commit
```

ステップ 4 XTC エージェントを停止します。

```
# wae agents xtc xtc <network_name> disable xtc-host-ip <xtc_ip_address>
# commit
```

Bandwidth Optimizationアプリケーション ワークフロー

Bandwidth Optimizationアプリケーションは、ネットワークの状態の変化に対応してトラフィックを管理するように設計されています。ネットワーク状態の変化が輻輳を引き起こすかどうかを判断します。その場合、Bandwidth OptimizationアプリケーションはLSPを計算し、展開のために XTC に送信します。

このワークフローでは、Bandwidth Optimizationアプリケーションおよびその他のコンポーネントを構成するために必要な構成手順の概要について説明します。

手順	詳細
1. デバイス認証グループと SNMP グループの構成	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 （24 ページ）。
2. ネットワーク アクセス プロファイルの構成	ネットワーク アクセスの設定 （25 ページ）
3. XTC エージェントの構成	エキスパートモードを使用したXTCエージェントの構成 （27 ページ）

手順	詳細
4. アグリゲータの構成	NIMO 収集の統合 （61 ページ）
5. WAE モデリングデーモン（WMD）の構成	WAE モデリングデーモン（WMD）の構成 （85 ページ）
6. XTC Agent to Patch（XATP）モジュールの構成	XTC Agent to Patch モジュールの構成 （86 ページ）
7. トポロジと追加の NIMO の実行	ネットワーク インターフェイス モジュール（NIMO） （51 ページ）
8.Bandwidth Optimizationアプリケーションの構成	Bandwidth Optimization の設定 （130 ページ）
9. ネットワークモデルを開く	WAE Design を使用して、視覚的なネットワーク モデル レイアウトを取得できます。WAE Design から、**[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE Modeling Daemon）]** に移動し、最終ネットワークモデルを選択します。 （注） 初期構成後、いつでも NIMO を実行でき、オンデマンド帯域幅アプリケーションはネットワークモデルを更新します。

Bandwidth Optimization の設定

この手順では、Bandwidth Optimizationアプリケーションの構成オプションについて説明します。完全な構成ワークフローについては、Bandwidth Optimizationアプリケーション ワークフロー （129 ページ）を参照してください。

始める前に

Bandwidth Optimizationアプリケーションは、オンデマンド帯域幅アプリケーションと同時に実行できません。他のアプリケーションを使用するには、一方のアプリケーションを無効にする（`enable = false`）必要があります。オンデマンド帯域幅アプリケーションを適切にシャットダウンする方法については、オンデマンド帯域幅のシャットダウン （129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:wae/components/bw-opt** に移動し、[config] タブをクリックします。

ステップ 2 次の値を入力します。

- [enable]：Bandwidth Optimizationアプリケーションを有効にするには、[true] を選択します。

- [util-threshold]：最適化を行う場合に超える必要があるパーセンテージを入力します。デフォルトは 100% です。
- [xtc-host]：XTC ホストのホスト名または IP アドレスを入力します。
- [xtc-port]：XTC ホストポートを入力します。
- [color]：XTC の SR ポリシーを表す色。詳細については、Cisco WAE の担当者にお問い合わせください。

ステップ 3 [コミット（Commit）] をクリックして、構成を保存します。

ステップ 4 ツールの実行後、[created-lsps] をクリックして、最適化されたルーティング用に作成された SR LSP を表示できます。

ステップ 5 WAE Design（**[WAE Design]** > **[ファイル（File）]** > **[開く場所（Open From）]** > **[WAEモデリングデーモン（WAE Modeling Daemon）]**）を使用して、得られたネットワークモデルを開きます。

例

CLI（構成モード）の例：

```
# wae components bw-opt config color 2000 enable false threshold 90 xtc-host 192.0.2.131
 xtc-port 8080
```

WAE SR ポリシーの制限事項

SR ポリシーに関連付けられた最新の IOS-XR SR 機能を使用する場合、次の WAE 制限事項が存在します。

- `candidate-paths` オプションで 2 つのパスが指定されている場合、最初のパスのみが考慮されます。
- SR LSP が WAE を介して作成される場合、デフォルトの色が SR LSP に設定されます。
- 複数の LSP が同じ色、送信元、および接続先を使用することはできません。

帯域幅最適化のシャットダウン

帯域幅最適化アプリケーションを適切にシャットダウンするには、次の手順を順番に実行する必要があります。

ステップ 1 帯域幅最適化を停止します。

```
# wae components bw-opt config enable false
# commit
```

ステップ 2 XTC Agent to Patch モジュールを停止します。

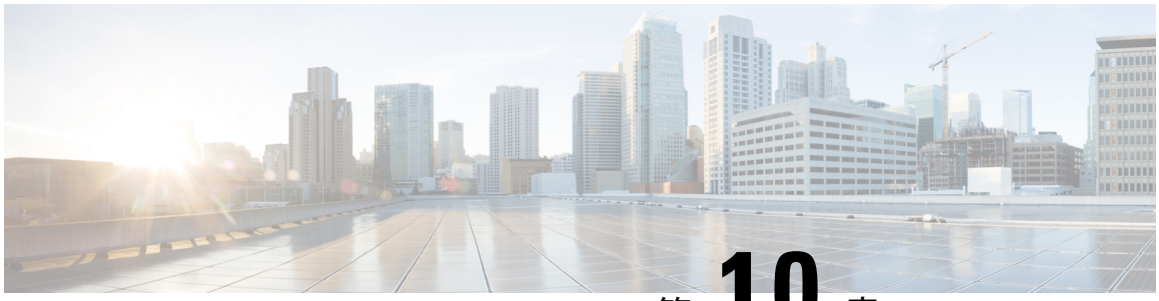
```
# wae components xatp config enable false
# commit
```

ステップ 3 WAE モデリングデーモンを停止します。

```
# wae components wmd config enable false
# commit
```

ステップ 4 XTC エージェントを停止します。

```
# wae agents xtc xtc <network_name> disable xtc-host-ip <xtc_ip_address>
# commit
```



第 10 章

スケジューラ構成

このセクションでは、cron ジョブとサブスクリプションジョブをスケジュールする方法の例と手順を示します。

スケジューラの概要

スケジューラは、次の 2 種類のスケジューリングジョブを実行します。

- Cron ジョブ：特定の操作を特定の日時に実行できるようにする時間ベースのジョブスケジューラ。たとえば、収集を定期的に実行するようにスケジュールできます。
- サブスクリプションジョブ：指定された送信元からのトリガーによって定義されたイベント通知をリッスンする、イベントベースのジョブスケジューラ。たとえば、スケジューラにプッシュされるネットワークモデル変更です。

スケジューラの構成

この手順では、エキスパートモードを使用して cron およびサブスクリプションベースのジョブをスケジュールする方法について説明します。

（注） WAE エキスパートモードまたは WAE CLI を使用したスケジューラ構成は、WAE UI に表示されません。WAE UI を使用してネットワーク収集をスケジュールするには、ネットワーク収集のスケジュール （17 ページ）を参照してください。

始める前に

依存するアクションまたはイベントの構成は、スケジューラに追加する前に完了する必要があります。たとえば、ネットワークトポロジ収集を特定の間隔または時間に実行するようにスケジュールする場合は、このタスクを続行する前に、そのネットワーク収集を構成しておく必要があります。

ステップ 1 エキスパート モードから、[wae:wae] > [components] タブ > [scheduler] > [task] に移動します。

ステップ 2 プラス（[+]）アイコンをクリックし、スケジューラジョブ名を入力します。

ステップ 3 [追加（Add）] をクリックします。

ステップ 4 スケジューラジョブが有効になっているときに実行するアクションを定義します。

a) [action] タブで、プラス（[+]）アイコンをクリックし、アクション名を入力します。
b) [追加（Add）] をクリックします。
c) [選択 - action（Choice-action）] ドロップダウンリストから、[rpc] を選択します。
d) [rpc] をクリックし、パス名を入力します。パス名では、呼び出す操作を指定します。たとえば、ネットワーク収集を呼び出すには、次のパスを入力します：**`/wae:networks/network{<network_model_name>}/nimo/<nimo_name>/run-collection`**
e) （オプション）アクションを呼び出すために特定のパラメータを満たす必要がある場合は、[params] タブをクリックし、要件の順にパラメータを追加します。

ステップ 5 アクションをトリガーするイベントのタイプを特定します（トリガーが複数ある場合、いずれかのトリガーが呼び出されるとアクションが実行されます）。

a) [trigger] タブから、プラス（[+]）アイコンをクリックし、トリガー名を入力します。
b) [追加（Add）] をクリックします。
c) [選択 - trigger-spec（Choice-trigger-spec）] ドロップダウンリストから、トリガータイプとして [cron] または [subscription] を選択します。

ステップ 6 サブスクリプションベースのジョブを構成している場合は、[subscription] リンクをクリックして、次の手順を実行します。

a) トリガーの送信元のパスを入力します。たとえば、イベントの送信元が回路変更の場合は、「**`/wae:networks/network{<network_model_name>}/model/circuits/circuit`**」と入力します。
b) [subscription-type] ドロップダウンリストから、次のいずれかのオプションを選択します。

- [operational]：トラフィックポーリングなど、すべての動作（読み取り専用）変更に適用されます。これらの変更は、ユーザーが開始したものではありません。
- [configuration]：ユーザーが開始した LSP 構成変更など、構成変更（ネットワークでの追加、削除、または変更）に適用されます。

ステップ 7 cron ベースのジョブを構成している場合は、[cron] リンクをクリックして、アクションをいつ実行するかを定義する適切なパラメータを入力します。

ステップ 8 さらにトリガーを追加するには、前の手順を繰り返します（トリガーが複数ある場合、いずれかのトリガーが呼び出されるとアクションが実行されます）。

ステップ 9 [確定する（Commit）] をクリックします。

トポロジ収集を実行するためのトリガーの構成例

この例では、トポロジ収集の実行をトリガーするサブスクリプションベースのジョブを構成します。次の手順では、ネットワークモデルに変更が発生したときにBGP-LS収集を実行するようにスケジューラを構成します。詳細については、「XTC を使用した BGP-LS トポロジ収集（58 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:wae] > [components] タブ > [scheduler] > [task] に移動します。

ステップ 2 プラス（[+]）アイコンをクリックし、スケジューラジョブ名として **run-topo-bgpls** と入力します。

ステップ 3 [追加（Add）] をクリックします。

ステップ 4 スケジューラジョブが有効になっているときに実行するアクションを定義します。

a) [action] タブで、プラス（[+]）アイコンをクリックし、アクション名として **run-xtc-topo** と入力します。

b) [追加（Add）] をクリックします。

c) [選択 - action（Choice-action）] ドロップダウンリストから、[rpc] を選択します。

d) [rpc] をクリックし、パス名を入力します。パス名では、呼び出す操作を指定します。たとえば、ネットワーク収集を呼び出すには、次のパスを入力します：**/wae:networks/network{NetworkABC_topo-bgpls-xtc-nimo}/nimo/topo-bgpls-xtc-nimo/run-collection**。

ステップ 5 このアクションをトリガーするイベントのタイプを特定します。

a) [trigger] タブから、プラス（[+]）アイコンをクリックし、トリガー名として **xtc-objects** と入力します。

b) [追加（Add）] をクリックします。

c) [選択 - trigger-spec（Choice-trigger-spec）] ドロップダウンリストから、[subscription] を選択します。

ステップ 6 [subscription] リンクをクリックして、次の手順を実行します。

a) 送信元パスを入力します（この例では、XTC リンクステータスが変更される場所です）：**/wae/agents/xtc/xtc{TTE-xtc11}/pce/xtc-topology-objects/xtc-links**。

b) [subscription-type] ドロップダウンリストから、[operational] を選択します。

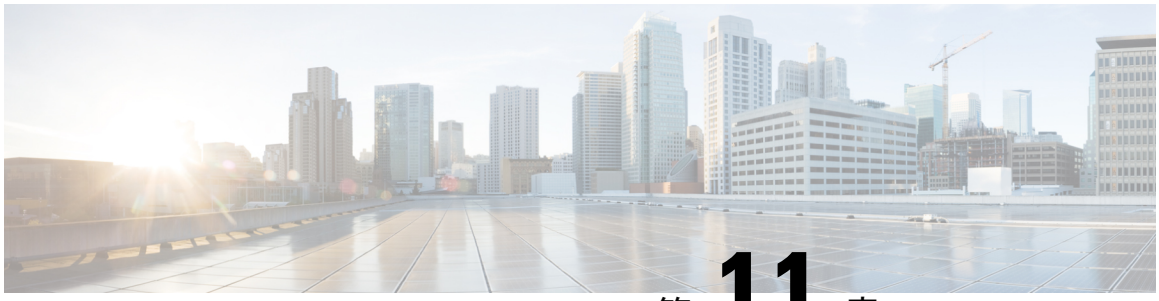
ステップ 7 [確定する（Commit）] をクリックします。

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。

```
# wae components scheduler tasks task run-topo-bgpls action run-xtc-topo rpc path
"/wae:networks/network{NetworkABC_topo-bgpls-xtc-nimo}/nimo/topo-bgpls-xtc-nimo/run-xtc-collection"
# wae components scheduler tasks task run-topo-bgpls triggers trigger xtc-objects
subscription node "/wae/agents/xtc/xtc{TTE-xtc11}/pce/xtc-topology-objects/xtc-links"
```

```
# wae components scheduler tasks task run-topo-bgpls triggers trigger xtc-objects
subscription subscription-type operational
# commit
```



第 11 章

WAE 管理

ここでは、次の内容について説明します。

ユーザーの管理

WAE 7.0 では、すべてのユーザーが管理者のロールを持っています。次の手順では、ユーザーを作成および削除する方法について説明します。

ステップ 1 WAE UI から、**[システム（System）] > [ユーザーマネージャ（User Manager）]** を選択します。

ステップ 2 ユーザーを追加するには、[+ユーザーを追加（+Add User）] をクリックして、該当するすべてのフィールドに入力します。

ステップ 3 ユーザーのパスワードを変更するには、次の手順を実行します。

a) ユーザーの行から、鉛筆アイコンをクリックします。

b) パスワードフィールドを更新します。

c) **[保存（Save）]** をクリックします。

ステップ 4 ユーザーを削除するには、次の手順を実行します。

a) ユーザーの行から、ごみ箱アイコンをクリックします。

エージングの構成

デフォルトでは、回路、ポート、ノード、またはリンクがネットワークから消失すると、永久に削除され、再検出する必要があります。消失したこれらの要素を WAE が保持してからネットワークから完全に削除されるまでの期間を設定するには、次の手順を実行します。

（注） これは、すべてのネットワークに構成されるグローバルオプションです。

ステップ 1 エキスパート モードから、**/wae:wae** に移動し、[nimos] タブをクリックします。

ステップ 2 [aging] をクリックします。

ステップ 3 WAE が要素（回路、ノード、ポート、またはリンク）を保持する必要がある分数を適切なフィールドに入力します。

ステップ 4 [確定する（Commit）] をクリックします。

wae.conf

`wae.conf` は、YANG モデル `tailf-ncsconfig.yang` で正式に定義されている XML 構成ファイルです。この YANG ファイルは、コメント付きの `wae.conf.example` ファイルと同様に、WAE ディストリビューションに含まれています。

`wae.conf` ファイルは、WAE ランタイムの基準設定を制御します。`wae.conf` ファイルで特定の構成パラメータを変更できます。たとえば、WAE が動作するデフォルトポート（ポート 8080）を別のポートに変更できます。

WAE デーモンを起動またはリロードするたびに、`./wae.conf` または *<waeruntime-directory>*`/etc/wae.conf` から構成を読み取ります。

<waeruntime-directory>`/etc/wae.conf` の内容を次の例に示します。

```
<!-- -*- nxml -*- -->
<!-- Example configuration file for wae. -->

<ncs-config xmlns="http://tail-f.com/yang/tailf-ncs-config">

  <!-- WAE can be configured to restrict access for incoming connections -->
  <!-- to the IPC listener sockets. The access check requires that -->
  <!-- connecting clients prove possession of a shared secret. -->
  <ncs-ipc-access-check>
    <enabled>false</enabled>
    <filename>${NCS_DIR}/etc/ncs/ipc_access</filename>
  </ncs-ipc-access-check>

  <!-- Where to look for .fxs and snmp .bin files to load -->

  <load-path>
```

```
  <dir>./packages</dir>
  <dir>${NCS_DIR}/etc/ncs</dir>

  <!-- To disable northbound snmp altogether -->
  <!-- comment out the path below -->
  <dir>${NCS_DIR}/etc/ncs/snmp</dir>
</load-path>

<!-- Plug and play scripting -->
<scripts>
  <dir>./scripts</dir>
  <dir>${NCS_DIR}/scripts</dir>
</scripts>

<state-dir>./state</state-dir>

<notifications>
  <event-streams>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications whenever the alarm table is changed. -->
    <!-- See tailf-ncs-alarms.yang -->
    <!-- If you are not interested in WAE northbound netconf notifications -->
    <!-- remove this item since it does consume some CPU -->
    <stream>
      <name>wae-alarms</name>
      <description>WAE alarms according to tailf-ncs-alarms.yang</description>
      <replay-support>false</replay-support>
      <builtin-replay-store>
        <enabled>false</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications for internal events. -->
    <!-- See tailf-ncs-devices.yang -->
    <!-- Required for cluster mode. -->
    <stream>
      <name>wae-events</name>
      <description>WAE event according to tailf-ncs-devices.yang</description>
      <replay-support>true</replay-support>
      <builtin-replay-store>
        <enabled>true</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications forwarded from devices. -->
    <!-- See tailf-event-forwarding.yang -->
    <stream>
      <name>device-notifications</name>
      <description>WAE events forwarded from devices</description>
      <replay-support>true</replay-support>
      <builtin-replay-store>
        <enabled>true</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
```

```
      </builtin-replay-store>
    </stream>

    <!-- This is the builtin stream used by WAE to generate northbound -->
    <!-- notifications for plan state transitions. -->
    <!-- See tailf-ncs-plan.yang -->
    <stream>
      <name>service-state-changes</name>
      <description>Plan state transitions according to
      tailf-ncs-plan.yang</description>
      <replay-support>false</replay-support>
      <builtin-replay-store>
        <enabled>false</enabled>
        <dir>./state</dir>
        <max-size>S10M</max-size>
        <max-files>50</max-files>
      </builtin-replay-store>
    </stream>
    <stream>
      <name>XtcNotifications</name>
      <description>Xtc object change notifications</description>
      <replay-support>false</replay-support>
    </stream>
  </event-streams>
 </notifications>

 <!-- Where the database (and init XML) files are kept -->
 <cdb>
   <db-dir>./ncs-cdb</db-dir>
   <!-- Always bring in the good system defaults -->
   <init-path>
     <dir>${NCS_DIR}/var/ncs/cdb</dir>
   </init-path>
 </cdb>


 <!--&#xa;      These keys are used to encrypt values of the types&#xa;
tailf:des3-cbc-encrypted-string and tailf:aes-cfb-128-encrypted-string.&#xa;      For a
 deployment install it is highly recommended to change&#xa;      these numbers to something
 random (done by WAE "system install")&#xa;  -->
 <encrypted-strings>
   <DES3CBC>
     <key1>0123456789abcdef</key1>
     <key2>0123456789abcdef</key2>
     <key3>0123456789abcdef</key3>
     <initVector>0123456789abcdef</initVector>
   </DES3CBC>

   <AESCFB128>
     <key>0123456789abcdef0123456789abcdef</key>
     <initVector>0123456789abcdef0123456789abcdef</initVector>
   </AESCFB128>
 </encrypted-strings>


 <logs>
   <syslog-config>
     <facility>daemon</facility>
     <udp>
       <enabled>false</enabled>
       <host>syslogsrv.example.com</host>
     </udp>
   </syslog-config>
```

```
<ncs-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/wae.log</name>
    <enabled>true</enabled>
  </file>
  <syslog>
    <enabled>true</enabled>
  </syslog>
</ncs-log>

<developer-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/devel.log</name>
    <enabled>true</enabled>
  </file>
</developer-log>
<developer-log-level>trace</developer-log-level>

<audit-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/audit.log</name>
    <enabled>true</enabled>
  </file>
</audit-log>

<netconf-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/netconf.log</name>
    <enabled>true</enabled>
  </file>
</netconf-log>

<snmp-log>
  <enabled>true</enabled>
  <file>
    <name>./logs/snmp.log</name>
    <enabled>true</enabled>
  </file>
</snmp-log>

<webui-browser-log>
  <enabled>true</enabled>
  <filename>./logs/webui-browser.log</filename>
</webui-browser-log>

<webui-access-log>
  <enabled>true</enabled>
  <dir>./logs</dir>
</webui-access-log>


 <!-- This log is disabled by default if wae is installed using -->
<!-- the 'system-install' flag. It consumes a lot of CPU power -->
<!-- to have this log turned on, OTOH it is the best tool to -->
<!-- debug must expressions in YANG models -->

<xpath-trace-log>
  <enabled>false</enabled>
  <filename>./logs/xpath.trace</filename>
</xpath-trace-log>
```

```
  <error-log>
    <enabled>true</enabled>
    <filename>./logs/wae-err.log</filename>
  </error-log>

</logs>

<ssh>
  <algorithms>
    <mac>hmac-sha1,hmac-sha2-256,hmac-sha2-512</mac>
    <encryption>aes128-ctr,aes192-ctr,aes256-ctr</encryption>
  </algorithms>
</ssh>

<aaa>
  <ssh-server-key-dir>${NCS_DIR}/etc/ncs/ssh</ssh-server-key-dir>

  <!-- Depending on OS - and also depending on user requirements -->
  <!-- the pam service value value must be tuned. -->

  <pam>
    <enabled>true</enabled>
    <service>common-auth</service>
  </pam>
  <external-authentication>
    <enabled>false</enabled>
    <executable>my-test-auth.sh</executable>
  </external-authentication>

  <local-authentication>
    <enabled>true</enabled>
  </local-authentication>

</aaa>

<!-- Hash algorithm used when setting leafs of type ianach:crypt-hash, -->
<!-- e.g. /aaa/authentication/users/user/password -->
<crypt-hash>
  <algorithm>sha-512</algorithm>
</crypt-hash>

<!-- Disable this for performance critical applications, enabling -->
<!-- rollbacks means additional disk IO for each transaction -->
<rollback>
  <enabled>true</enabled>
  <directory>./logs</directory>
  <history-size>50</history-size>
</rollback>


<cli>
  <enabled>true</enabled>

  <!-- Use the builtin SSH server -->
  <ssh>
    <enabled>true</enabled>
    <ip>0.0.0.0</ip>
    <port>2024</port>
  </ssh>

  <prompt1>\u@wae> </prompt1>
  <prompt2>\u@wae% </prompt2>
```

```
  <c-prompt1>\u@wae# </c-prompt1>
  <c-prompt2>\u@wae(\m)# </c-prompt2>

  <show-log-directory>./logs</show-log-directory>
  <show-commit-progress>true</show-commit-progress>
  <suppress-commit-message-context>maapi</suppress-commit-message-context>
  <suppress-commit-message-context>system</suppress-commit-message-context>
</cli>

<webui>
  <absolute-timeout>PT0M</absolute-timeout>
  <idle-timeout>PT30M</idle-timeout>
  <enabled>true</enabled>
  <transport>
    <tcp>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8080</port>
     <redirect>https://@HOST@:8443</redirect>
    </tcp>
    <ssl>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>8443</port>
      <key-file>${NCS_DIR}/var/ncs/webui/cert/host.key</key-file>
      <cert-file>${NCS_DIR}/var/ncs/webui/cert/host.cert</cert-file>
    </ssl>
  </transport>

  <cgi>
    <enabled>true</enabled>
    &lt;php>
      <enabled>false</enabled>
    &lt;/php>
  </cgi>
</webui>

<rest>
  <enabled>true</enabled>
</rest>

<restconf>
  <enabled>true</enabled>
</restconf>

<netconf-north-bound>
  <enabled>true</enabled>

  <transport>
    <ssh>
      <enabled>true</enabled>
      <ip>0.0.0.0</ip>
      <port>2022</port>
    </ssh>
    <tcp>
      <enabled>false</enabled>
      <ip>127.0.0.1</ip>
      <port>2023</port>
    </tcp>
  </transport>
</netconf-north-bound>

<!-- <ha> -->
<!--   <enabled>true</enabled> -->
```

```
  <!-- </ha> -->

  <large-scale>
    <lsa>
      <!-- Enable Layered Service Architecture, LSA. This requires&#xa;          a
separate Cisco Smart License.&#xa;      -->
      <enabled>true</enabled>
    </lsa>
  </large-scale>

</ncs-config>
```

多くの構成パラメータのデフォルト値は、YANG ファイルで定義されています。wae.conf 構成パラメータ （174 ページ） を参照してください。

LSA 構成

階層化されたサービスアーキテクチャ（LSA）の基本的な考え方は、サービスを上位層と 1 つまたは複数の下位レベルの部分に分割することです。これは、サービスを顧客向け（CFS）部分とリソース向け（RFS）部分に分割すると見なすことができます。CFS コード（上位レベル）は 1 つ（または複数）の NSO cfs ノードで実行され、RFS コード（下位レベル）は多くの NSO rfs ノードの 1 つで実行されます。rfs ノードのそれぞれには管理対象デバイスの一部が /devices ツリーにマウントされていて、cfs ノードには NSO rfs ノードが /devices ツリーにマウントされています。

このセクションでは、以下を想定しています。

- 展開、デバイス構成、および LSA に関して Cisco Network Service Orchestrator（NSO）を理解していること。詳細については、NSO 4.4 のドキュメントを参照してください。具体的には、『NSO Getting Started』ガイドおよび『NSO Layered Service Architecture』ガイドを参照してください。
- マシン上に 1 つ以上の NSO インストールがあること（各 LSA リソース向け（RFS）ノード用）。各インスタンスのデバイスオプション構成は、手順に記載されています。

LSA 構成ワークフロー

このワークフローでは、WAE で LSA を構成するための手順の概要について説明します。

ステップ	詳細
1. WAE LSA 構成用の追加パッケージをインストールします。	LSA パッケージのインストール （145 ページ）
2. LSA をサポートするように WAE を構成します。	LSA のための WAE の構成 （145 ページ）
3. RFS モデルをブートストラップします。	LSA のための RFS モデルのブートストラップ （146 ページ）

LSA パッケージのインストール

階層化されたサービスアーキテクチャ（LSA）をサポートするように WAE を構成する前に、追加の WAE NSO パッケージをインストールし、ncs.conf ファイルで LSA を有効にする必要があります。

ステップ 1 *<wae_installation_directory>*/packages/lsa から、cisco-wae-rfs を NSO ノードの run/packages ディレクトリにコピーします。

ステップ 2 （オプション）LSP 収集に lsp-config-nimo を使用する場合は、次の手順を実行します。

a) *<wae_installation_directory>*/packages/lsa から、cisco-wae-lsp-rfs を NSO ノードの NSO run/packages ディレクトリにコピーします。

b) 必要な RFS NED を NSO ノードにコピーします。例：
<wae_installation_directory>/packages/cisco-wae-lsp-rfs/cisco-wae-lsp-rfs-iosxr。

ステップ 3 RFS LSP CONFIG NIMO パッケージがインストールされていることを確認します。
<wae_installation_directory>/cisco-wae-lsp-config-nimo/cisco-wae-lsp-config-nimo-rfs を WAE ノードの *<wae_run_time_directory>*/packages ディレクトリにコピーします。

ステップ 4 ncs.conf ファイルで LSA 機能を有効にします。

```
<large-scale>
    <lsa>
      <!-- Enable Layered Service Architecture, LSA. This requires
           a separate Cisco Smart License.
      -->
      <enabled>true</enabled>
    </lsa>
  </large-scale>
```

ステップ 5 [コミット（Commit）] ボタンをクリックします。

ステップ 6 LSA のための WAE の構成 （145 ページ）に記載されている手順を完了します。

LSA のための WAE の構成

LSA をサポートするように WAE を構成するには、特定のデバイスオプションを構成する必要があります。

この手順では、以下を想定しています。

- 展開、デバイス構成、および LSA に関して Cisco Network Service Orchestrator（NSO）を理解していること。詳細については、NSO 4.4 のドキュメントを参照してください。具体的には、『NSO Getting Started』ガイドおよび『NSO Layered Service Architecture』ガイドを参照してください。
- マシン上に 1 つ以上の NSO インストールがあること（各 LSA リソース向け（RFS）ノード用）。各インスタンスのデバイスオプション構成は、手順に記載されています。

ステップ 1 WAE で、次の構成オプションを使用して、LSA RFS ノードごとに LSA NETCONF デバイスを作成します。

エキスパートモードから、**/ncs:devices/device/*rfs_node*** に移動すると、オプションを簡単に表示できます。

- [device] タブから：
  - [address]：LSA RFS ノードの IP アドレス。
  - [port]：ncs.conf の LSA RFS ノードで構成された netconf-north-bound ポート。デフォルトは 2022 ですが、LSA RFS ノードインスタンスごとに異なるポート番号を持つように構成できます。
  - [use-lsa]：LSA を有効にするには、このボックスをオンにします。
- [device-type] タブから：
  - [netconf]：このチェックボックスがオンになっていることを確認してください。
  - [ned-id]：[ned:lsa-netconf] を選択します。
- [ssh] タブで、[fetch-host-keys] をクリックします。
- [state] タブの [admin-state] ドロップダウンリストで、[unlocked] を選択します。
- [device] タブで、[sync-from] をクリックします。

ステップ 2 [コミット（Commit）] ボタンをクリックします。

ステップ 3 エキスパートモードから、**/wae:wae/lsa** に移動し、構成したすべての LSA RFS ノードを [rfs-node] ドロップダウンリストに追加します。

ステップ 4 [コミット（Commit）] ボタンをクリックします。

LSA のための RFS モデルのブートストラップ

RFS モデルをブートストラップすると、使用可能なデバイスから RFS モデルのノードが設定されます。これには、ベンダー、オペレーティングシステム、および管理 IP 情報が含まれます（LSP 関連の情報ではありません）。`wae-rfs services wae-rfs run-collection` コマンドを呼び出すことにより、RFS ノードから情報を取り込むことができます。これにより、ノードのリストが更新されます（存在しないノードが削除される可能性があります）。

ステップ 1 RFS ノードから RFS 収集を呼び出します。

```
admin@nso# wae-rfs services wae-rfs run-collection
status true
message Wae-rfs Collection done
admin@nso#
```

ステップ 2 デバイス同期（`devices sync-from`）を実行して、WAE を新しい RFS モデルで更新し、LSA RFS ノードに存在するデバイスが RFS モデルに読み込まれていることを確認します。

```
admin@wae# show running-config devices device rfs1 config wae-rfs:wae-rfs rfs-model nodes node XRv-2
devices device rfs1
 config
  wae-rfs:wae-rfs rfs-model nodes node XRv-2
   ip-manage 192.0.2.24
   platform vendor Cisco
   platform os "IOS XR"
  !
 !
!
```

（注） NSO インスタンスのデバイスモデルは、同期状態であることが想定されます。展開中のデバイス同期エラーを回避するために、デバイスを同期します（**devices sync-from**）。

ステップ 3 （LSP 展開の場合）RFS モデルの LSP 部分を設定するには、次の手順を実行します。

a) 便宜上、リモートアクションを介して CFS ノードから RFS LSP モデルを設定できますが、これにより NED 読み取りタイムアウトエラーが発生する可能性があります。エラーを回避するには、lsp-config-nimo `perform-sync-from` 詳細オプションを無効にします。

b) コマンド `wae-rfs services lsp-rfs run-collection` を使用して、RFS ノードから RFS モデルの RFS LSP 情報を直接入力します。

c) デバイスを再度同期して（**devices sync-from**）、WAE を更新します。

次のタスク

次のいずれかを実行できます。

- NSO NED を使用した LSP 収集 （65 ページ）
- マルチレイヤ収集 （89 ページ）

LSA のトラブルシューティング

（注） <*wae_runtime_directory*>/logs でログを表示できます。

LSA の構成設定が完了し、LSP 収集（lsp-config-nimo）が呼び出されると、次の処理が行われます。

1. 送信元ネットワークがコピーされます。
2. LSA RFS ネットワークモデルからの LSP 収集が読み込まれます。
    1. LSA RFS モデルが LSP 情報で更新されます（**`wae-rfs services lsp-rfs run-collection`**）。

2. デバイス同期（**devices device** *<rfs_device_name>* sync-from）によって、RFS の WAE デバイスモデルが更新されます。
3. WAE ネットワークモデルの LSP 部分が RFS モデルから取り込まれます。
4. WAE ネットワークモデルの新しく取り込まれた LSP 部分がコミットされます。

3. RFS モデルのサービスメタデータが更新されます（**re-deploy reconcile**）。

上記の手順で必要な処理により、タイムアウトが発生する可能性があります。RFS モデルの非同期エラーと wae-java-vm.log ファイルのタイムアウトエラーは、多くの場合、タイムアウトエラーを示しています。一般に、開始時のタイムアウト値は、各 RFS ノードのデバイスごとに少なくとも 4 秒にする必要があります。タイムアウト時間がエラーの原因であると思われる場合は、必要に応じて次の値を変更します。

- **lsp-config-nimo run-collection** アクションタイムアウト：**lsp-config-action** が完了するために YANG ランタイム（YRT）によって割り当てられる時間。この値は、wae.conf ファイルで YRT インスタンスのすべてのアクションに対して指定される可能性があります。この値は次のコマンドで編集できます：**networks network** *<network_name>* **nimo lsp-config-nimo advanced action-timeout** *<timeout_value>*
- **wae-rfs services lsp-rfs run-collection** アクションタイムアウト：**lsp-rfs** アクションが完了するために RFS モデルによって割り当てられる時間。この値は、ncs.conf ファイルで RFS NSO インスタンスに対して指定される可能性があります。この値は次のコマンドで編集できます：**wae-rfs services lsp-rfs advanced action-timeout***<timeout_value>*

例：YRT で 5 分のタイムアウトを設定する

```
admin@wae# config
Entering configuration mode terminal
Current configuration users:
admin tcp (maapi from 127.0.0.1) on since 2017-08-09 09:46:21 terminal mode
admin@wae(config)# networks network lsp-network nimo lsp-config-nimo advanced
action-timeout 5
admin@wae(config-network-lsp-network)# top
admin@wae(config)# devices device rfs config wae-rfs:wae-rfs services lsp-rfs advanced
action-timeout 5
admin@wae(config-config)# commit
Commit complete.
admin@wae(config-config)#
```

WAE CLI ロギングについて

WAE は豊富なロギング機能を備えています。WAE は `wae.conf` ファイルで指定されたディレクトリにログを記録します。最も有用なログファイルは次のとおりです。

- `wae.log`：WAE デーモンログ。syslog に構成できます。

- wae_err.log.1、wae_err.log.idx、wae_err.log.siz：WAE デーモンに問題がある場合、このログにはサポートのためのデバッグ情報が含まれています。コマンド**wae --printlog wae_err.log** で内容を表示します。
- audit.log：すべてのノースバウンド インターフェイスをカバーする中央監査ログ。syslog に構成できます。
- localhost:8080.access：デーモンへのすべての http リクエスト。組み込み Web サーバーのアクセスログです。このファイルは、Apache などで定義されている Common Log Format に準拠しています。このログはデフォルトで無効であり、ローテーションされません。そのため、**logrotate(8)** を使用してください。
- devel.log：ユーザー作成コードをトラブルシューティングするためのデバッグログ。このログはデフォルトで有効であり、ローテーションされません。そのため、**logrotate(8)** を使用してください。このログは、java-vm または python-vm ログとともに使用してください。ユーザーコードは vm ログに記録され、対応するライブラリは devel.log に記録されます。実稼働システムではこのログを無効にしてください。syslog に構成できます。
- wae-java-vm.log、wae-python-vm.log：サービスアプリケーションなど、Java または Python VM で実行されるコードのログ。Java および Python コードを作成する開発者は、このログを（devel.log と組み合わせて）デバッグに使用します。
- netconf.log、snmp.log：ノースバウンドエージェントのログ。syslog に構成できます。
- rollback*NNNNN*：すべての WAE コミットは、対応するロールバックファイルを生成します。wae.conf で、ロールバックファイルの最大数とファイル番号を構成できます。
- xpath.trace：XPATH は、XML テンプレートなど多くの場所で使用されます。このログファイルには、すべての XPATH 式の評価が示されます。テンプレートの XPATH をデバッグするには、代わりに CLI で pipe-target debug を使用します。
- ned-cisco-ios-xr-pe1.trace：デバイストレースがオンになっている場合、デバイスごとにトレースファイルが作成されます。ファイルの場所は wae.conf では構成されませんが、CLI などでデバイストレースがオンになっているときに構成されます。

Syslog

WAE では、BSD または IETF syslog フォーマット（RFC5424）を使用して、ローカルまたはリモートの syslog サーバーに syslog を送信できます。wae.conf ファイルを使用して、syslog に保存するログ（ncs.log、devel.log、netconf.log、または snmp.log）を選択できます。

次の例は、一般的な syslog 構成を示しています。

```
<syslog-config>
  <facility>daemon</facility>
```

```
  <udp>
     <enabled>false</enabled>
     <host>127.0.0.1</host>
     <port>895</port>
  </udp>

  <syslog-servers>
    <server>
      <host>127.0.0.2</host>
      <version>1</version>
    </server>
    <server>
      <host>127.0.0.3</host>
      <port>7900</port>
      <facility>local4</facility>
      </server>
   </syslog-servers>
  </syslog-config>

<ncs-log>
  <enabled>true</enabled>
  <file>
   <name>./logs/ncs.log</name>
   <enabled>true</enabled>
  </file>
  <syslog>
    <enabled>true</enabled>
  </syslog>
</ncs-log>
```

Syslog のメッセージと形式

次の表に、WAE syslog メッセージとそのフォーマットを示します。

記号	フォーマット文字列	備考
DAEMON_DIED	"Daemon ~s died"	外部データベースデーモンがその制御ソケットを閉じました。
DAEMON_TIMEOUT	"Daemon ~s timed out"	外部データベースデーモンがクエリに応答しませんでした。
NO_CALLPOINT	"no registration found for callpoint ~s of type=~s"	ConfD は XML ツリーにデータを入力しようとしましたが、関連するコールポイントにコードが登録されていませんでした。
CDB_DB_LOST	"CDB: lost DB, deleting old config"	CDB はデータスキーマファイルを検出しましたが、データファイルは検出しませんでした。空のデータベースから開始して CDB をリカバリしました。

記号	フォーマット文字列	備考
CDB_CONFIG_LOST	"CDB: lost config, deleting DB"	CDB はデータファイルを検出しましたが、スキーマファイルは検出しませんでした。空のデータベースから開始して CDB をリカバリしました。
CDB_UPGRADE_FAILED	"CDB: Upgrade failed: ~s"	自動 CDB アップグレードに失敗しました。つまり、サポートされていない方法でデータモデルが変更されました。
CDB_INIT_LOAD	"CDB load: processing file: ~s"	CDB は初期化ファイルを処理しています。
CDB_OP_INIT	"CDB: Operational DB re-initialized"	アップグレードまたは破損したファイルが原因で、運用データベースが削除され、再初期化されました。
CDB_CLIENT_TIMEOUT	"CDB client (~s) timed out, waiting for ~s"	CDB クライアントがタイムアウト時間内に応答できず、切断されました。
INTERNAL_ERROR	"Internal error: ~s"	ConfD 内部エラーが発生しました。シスコテクニカルサポートに報告する必要があります。
AAA_LOAD_FAIL	"Failed to load AAA: ~s"	外部データベースの動作に問題があるか、AAA のマウントまたは入力が不適切なため、AAA データをロードできませんでした。
EXTAUTH_BAD_RET	"External auth program (user=~s) ret bad output: ~s"	認証は外部であり、外部プログラムが不適切な形式のデータを返しました。
BRIDGE_DIED	"confd_aaa_bridge died - ~s"	ConfD が confd_aaa_bridge を開始するように構成され、C プログラムが停止しました。
PHASE0_STARTED	"ConfD phase0 started"	ConfD は開始フェーズ 0 を開始しました。
PHASE1_STARTED	"ConfD phase1 started"	ConfD は開始フェーズ 1 を開始しました。
STARTED	"ConfD started vsn: ~s"	ConfD が開始しました。
UPGRADE_INIT_STARTED	"Upgrade init started"	インサービスアップグレードの初期化が開始しました。

記号	フォーマット文字列	備考
UPGRADE_INIT_SUCCEEDED	"Upgrade init succeeded"	インサービスアップグレードの初期化に成功しました。
UPGRADE_PERFORMED	"Upgrade performed"	インサービスアップグレードが実行されましたが、まだコミットされていません。
UPGRADE_COMMITTED	"Upgrade committed"	インサービスアップグレードがコミットされました。
UPGRADE_ABORTED	"Upgrade aborted"	インサービスアップグレードは中止されました。
CONSULT_FILE	"Consulting daemon configuration file ~s"	ConfD は構成ファイルを読み取っています。
STOPPING	"ConfD stopping (~s)"	ConfD が停止しています（たとえば、**confd --stop** のため）。
RELOAD	"Reloading daemon configuration"	デーモン構成のリロードを開始しました。
BADCONFIG	"Bad configuration: ~s:~s: ~s"	confd.conf に不正なデータが含まれています。
WRITE_STATE_FILE_FAILED	"Writing state file failed: ~s: ~s (~s)"	状態ファイルの書き込みに失敗しました。
READ_STATE_FILE_FAILED	"Reading state file failed: ~s: ~s (~s)"	状態ファイルの読み取りに失敗しました。
SSH_SUBSYS_ERR	"ssh protocol subsys - ~s"	クライアントは **\"subsystem\"** コマンドを正しく送信しませんでした。
SESSION_LIMIT	"Session limit of type '~s' reached, rejected new session request"	セッション制限に達しました。新しいセッションリクエストは拒否されました。
CONFIG_TRANSACTION_LIMIT	"Configuration transaction limit of type '~s' reached, rejected new transaction request"	構成トランザクション制限に達しました。新しいトランザクションリクエストは拒否されました。
ABORT_CAND_COMMIT	"Aborting candidate commit, request from user, reverting configuration"	ユーザーのリクエストにより、候補コミットを中止します。構成を元に戻しています。

記号	フォーマット文字列	備考
ABORT_CAND_COMMIT_TIMER	"Candidate commit timer expired, reverting configuration"	候補コミットタイマーが期限切れになりました。構成を元に戻しています。
ABORT_CAND_COMMIT_TERM	"Candidate commit session terminated, reverting configuration"	候補コミットセッションが終了しました。構成を元に戻しています。
ROLLBACK_REMOVE	"Found half created rollback0 file - removing and creating new"	半分しか作成されていないことがわかった rollback0 ファイルを削除して再作成しています。
ROLLBACK_REPAIR	"Found half created rollback0 file - repairing"	半分しか作成されていないことがわかった rollback0 ファイルを修復しています。
ROLLBACK_FAIL_REPAIR	"Failed to repair rollback files"	ロールバックファイルの修復に失敗しました。
ROLLBACK_FAIL_CREATE	"Error while creating rollback file: ~s: ~s"	ロールバックファイルの作成中にエラーが発生しました。
ROLLBACK_FAIL_RENAME	"Failed to rename rollback file ~s to ~s: ~s"	ロールバックファイルの名前を変更できませんでした。
NS_LOAD_ERR	"Failed to process namespace ~s: ~s"	システムは、ロードされた名前空間を処理できませんでした。
NS_LOAD_ERR2	"Failed to process namespaces: ~s"	システムは、ロードされた名前空間を処理できませんでした。
FILE_LOAD_ERR	"Failed to load file ~s: ~s"	システムは、そのロードパスにファイルをロードできませんでした。
FILE_LOADING	"Loading file ~s"	システムは、ファイルのロードを開始しています。
SKIP_FILE_LOADING	"Skipping file ~s: ~s"	システムは、ファイルをスキップしました。
FILE_LOAD	"Loaded file ~s"	システムは、ファイルをロードしました。
LISTENER_INFO	"~s to listen for ~s on ~s:~s"	ConfD は、着信接続をリッスンするために開始または停止します。
NETCONF_HDR_ERR	"Got bad NETCONF TCP header"	ユーザーとグループが正しくフォーマットされていないことを示すクリアテキストのヘッダー。

記号	フォーマット文字列	備考
LIB_BAD_VSN	"Got library connect from wrong version (~s, expected ~s)"	ConfD に接続しているアプリケーションで、ConfD バージョンと一致しないライブラリバージョン（たとえば、古いバージョンのクライアントライブラリ）が使用されました。
LIB_BAD_SIZES	"Got connect from library with insufficient keypath depth/keys support (~s/ ~s, needs ~s/~s)"	ConfD に接続しているアプリケーションで、データモデルで使用されるキーの深さと数を処理できないライブラリバージョンが使用されました。
LIB_NO_ACCESS	"Got library connect with failed access check: ~s"	アプリケーションが ConfD に接続したときにアクセスチェックエラーが発生しました。
SNMP_NOT_A_TRAP	"SNMP gateway: Non-trap received from ~s"	トラップ受信ポートで UDP パッケージを受信しましたが、SNMP トラップではありません。
SNMP_TRAP_V1	"SNMP gateway: V1 trap received from ~s"	トラップ受信ポートで SNMPv1 トラップを受信しましたが、v1 トラップの転送はサポートされていません。
SNMP_TRAP_NOT_FORWARDED	"SNMP gateway: Can't forward trap from ~s; ~s"	SNMP トラップが転送されませんでした。
SNMP_TRAP_UNKNOWN_ SENDER	"SNMP gateway: Not forwarding trap from ~s; the sender is not recognized"	SNMP トラップが転送されるはずでしたが、送信者が confd.conf にリストされていませんでした。
SNMP_TRAP_OPEN_PORT	"SNMP gateway: Can't open trap listening port ~s: ~s"	SNMP トラップをリッスンするためのポートを開けませんでした。
SNMP_TRAP_NOT_RECOGNIZED	"SNMP gateway: Can't forward trap with OID ~s from ~s; There is no notification with this OID in the loaded models"	トラップ受信ポートで SNMP トラップを受信しましたが、その定義が不明です。
XPATH_EVAL_ERROR1	"XPath evaluation error: ~s for ~s"	xpath 式の評価中にエラーが発生しました。
XPATH_EVAL_ERROR2	"XPath evaluation error: '~s' resulted in ~s for ~s"	xpath 式の評価中にエラーが発生しました。

記号	フォーマット文字列	備考
CANDIDATE_BAD_FILE_FORMAT	"Bad format found in candidate db file ~s; resetting candidate"	候補データベースファイルのフォーマットが正しくありません。候補データベースが空のデータベースにリセットされます。
CANDIDATE_CORRUPT_FILE	"Corrupt candidate db file ~s; resetting candidate"	候補データベースファイルが壊れているため、読み取ることができません。候補データベースが空のデータベースにリセットされます。
MISSING_DES3CBC_SETTINGS	"DES3CBC keys were not found in confd.conf"	confd.conf に DES3CBC キーが見つかりませんでした。
MISSING_AESCFB128_ SETTINGS	"AESCFB128 keys were not found in confd.conf"	confd.conf に AESCFB128 キーが見つかりませんでした。
SNMP_MIB_LOADING	"Loading MIB: ~s"	SNMP エージェントが MIB ファイルをロードしています。
SNMP_CANT_LOAD_MIB	"Can't load MIB file: ~s"	SNMP エージェントが MIB ファイルのロードに失敗しました。
SNMP_WRITE_STATE_ FILE_FAILED	"Write state file failed: ~s: ~s"	SNMP エージェント状態ファイルの書き込みに失敗しました。
SNMP_READ_STATE_ FILE_FAILED	"Read state file failed: ~s: ~s"	SNMP エージェント状態ファイルの読み取りに失敗しました。
SNMP_REQUIRES_CDB	"Can't start SNMP. CDB is not enabled"	SNMP エージェントを開始する前に、CDB を有効にする必要があります。
FXS_MISMATCH	"Fxs mismatch, slave is not allowed"	スレーブは、異なる fxs ファイルを持つマスターに接続されています。
TOKEN_MISMATCH	"Token mismatch, slave is not allowed"	スレーブは、不正な認証トークンでマスターに接続されています。
HA_SLAVE_KILLED	"Slave ~s killed due to no ticks"	スレーブノードは、ティックを生成しませんでした。
HA_DUPLICATE_NODEID	"Nodeid ~s already exists"	すでに存在するノード ID を持つスレーブが到着しました。
HA_FAILED_CONNECT	"Failed to connect to master: ~s"	スレーブがマスターに接続できなかったため、ライブラリをスレーブ呼び出しにしようとしましたが失敗しました。

記号	フォーマット文字列	備考
HA_BAD_VSN	"Incompatible HA version (~s, expected ~s), slave is not allowed"	スレーブは、互換性のない HA プロトコルバージョンでマスターに接続されています。
NETCONF	"~s"	NETCONF トラフィックログメッセージ。
DEVEL_WEBUI	"~s"	デベロッパー Web UI ログメッセージ。
DEVEL_AAA	"~s"	デベロッパー AAA ログメッセージ。
DEVEL_CAPI	"~s"	デベロッパー C API ログメッセージ。
DEVEL_CDB	"~s"	デベロッパー CDB ログメッセージ。
DEVEL_CONFD	"~s"	デベロッパー ConfD ログメッセージ。
DEVEL_SNMPGW	"~s"	デベロッパー SNMP ゲートウェイログメッセージ。
DEVEL_SNMPA	"~s"	デベロッパー SNMP エージェントログメッセージ。
NOTIFICATION_REPLAY_ STORE_FAILURE	"~_s"	組み込みの通知再生ストアで障害が発生しました。
EVENT_SOCKET_TIMEOUT	"Event notification subscriber with bitmask ~s timed out, waiting for ~s"	イベント通知サブスクライバは、構成されたタイムアウト期間内に応答しませんでした。
EVENT_SOCKET_WRITE_BLOCK	"~s"	イベントソケットへの書き込みが長時間ブロックされました。
COMMIT_UN_SYNCED_DEV	"Committed data towards device ~s which is out of sync"	同期状態が不良または不明なデバイスに対してデータがコミットされました。
NCS_SNMP_INIT_ERR	"Failed to locate snmp_init.xml in loadpath ~s"	ロードパスで snmp_init.xml が見つかりませんでした。
NCS_JAVA_VM_START	"Starting the NCS Java VM"	NCS Java VM を起動しています。
NCS_JAVA_VM_FAIL	"The NCS Java VM ~s"	NCS Java VM の障害またはタイムアウトが発生しました。
NCS_PACKAGE_SYNTAX_ERROR	"Failed to load NCS package: ~s; syntax error in package file"	パッケージファイルの構文エラー。

記号	フォーマット文字列	備考
NCS_PACKAGE_DUPLICATE	"Failed to load duplicate NCS package ~s: (~s)"	重複するパッケージが見つかりました。
NCS_PACKAGE_COPYING	"Copying NCS package from ~s to ~s"	パッケージがロードパスからプライベートディレクトリにコピーされました。
NCS_PACKAGE_UPGRADE_ ABORTED	"NCS package upgrade failed with reason '~s'"	CDB のアップグレードが中止されました。これは、CDB が変更されていないことを意味します。ただし、パッケージの状態は変更されました。
NCS_PACKAGE_BAD_ NCS_VERSION	"Failed to load NCS package: ~s; requires NCS version ~s"	パッケージの NCS バージョンが正しくありません。
NCS_PACKAGE_BAD_ DEPENDENCY	"Failed to load NCS package: ~s; required package ~s of version ~s is not present (found ~s)"	NCS パッケージの依存関係が正しくありません。
NCS_PACKAGE_CIRCULAR_ DEPENDENCY	"Failed to load NCS package: ~s; circular dependency found"	NCS パッケージに循環依存関係があります。
CLI_CMD	"CLI '~s'"	ユーザーが CLI コマンドを実行しました。
CLI_DENIED	"CLI denied '~s'"	権限が原因で、ユーザーは CLI コマンドの実行を拒否されました。
BAD_LOCAL_PASS	"Provided bad password"	ローカルに構成されたユーザーが間違ったパスワードを入力しました。
NO_SUCH_LOCAL_USER	"no such local user"	存在しないローカルユーザーがログインしようとしました。
PAM_LOGIN_FAILED	"pam phase ~s failed to login through PAM: ~s"	ユーザーが PAM 経由でログインできませんでした。
PAM_NO_LOGIN	"failed to login through PAM: ~s"	ユーザーが PAM 経由でログインできませんでした。
EXT_LOGIN	"Logged in over ~s using externalauth, member of groups: ~s~s"	外部認証されたユーザーがログインしました。
EXT_NO_LOGIN	"failed to login using externalauth: ~s"	ユーザーの外部認証に失敗しました。
GROUP_ASSIGN	"assigned to groups: ~s"	ユーザーは一連のグループに割り当てられました。

記号	フォーマット文字列	備考
GROUP_NO_ASSIGN	"Not assigned to any groups - all access is denied"	ユーザーはログインしましたが、どのグループにも割り当てられていません。
MAAPI_LOGOUT	"Logged out from maapi ctx=~s (~s)"	管理エージェント API（MAAPI）ユーザーがログアウトされました。
SSH_LOGIN	"logged in over ssh from ~s with authmeth:~s"	ユーザーが ConfD の組み込み SSH サーバーにログインしました。
SSH_LOGOUT	"Logged out ssh <~s> user"	ユーザーが ConfD の組み込み SSH サーバーからログアウトされました。
SSH_NO_LOGIN	"Failed to login over ssh: ~s"	ユーザーが ConfD の組み込み SSH サーバーにログインできませんでした。
NOAAA_CLI_LOGIN	"logged in from the CLI with aaa disabled"	ユーザーが --noaaa フラグを confd_cli に使用しました。
WEB_LOGIN	"logged in through Web UI from ~s"	ユーザーが Web UI を介してログインしました。
WEB_LOGOUT	"logged out from Web UI"	Web UI ユーザーがログアウトしました。
WEB_CMD	"WebUI cmd '~s'"	ユーザーが Web UI コマンドを実行しました。
WEB_ACTION	"WebUI action '~s'"	ユーザーが Web UI アクションを実行しました。
WEB_COMMIT	"WebUI commit ~s"	ユーザーが Web UI コミットを実行しました。
SNMP_AUTHENTICATION_FAIL	"ESDNMP authentication failed: ~s"	SNMP 認証に失敗しました。
LOGIN_REJECTED	"~s"	ユーザーの認証がアプリケーションのコールバックによって拒否されました。
COMMIT_INFO	"commit ~s"	構成変更に関する情報が実行中のデータストアにコミットされました。
CLI_CMD_DONE	"CLI done"	CLI コマンドが正常に完了しました。
CLI_CMD_ABORTED	"CLI aborted"	CLI コマンドが中止されました。

記号	フォーマット文字列	備考
NCS_DEVICE_OUT_OF_SYNC	"NCS device-out-of-sync Device '~s' Info '~s'"	check-sync アクションで、デバイスの非同期が報告されました。
NCS_SERVICE_OUT_OF_SYNC	"NCS service-out-ofsync Service '~s' Info '~s'"	check-sync アクションで、サービスの非同期が報告されました。
NCS_PYTHON_VM_START	"Starting the NCS Python VM"	NCS Python VM を起動しています。
NCS_PYTHON_VM_FAIL	"The NCS Python VM ~s"	NCS Python VM が失敗したか、タイムアウトしました。
NCS_SET_PLATFORM_ DATA_ERRORS	"NCS Device '~s' failed to set platform data Info '~s'"	デバイスは、接続時にプラットフォーム運用データを設定できませんでした。
NCS_SMART_LICENSING_ START	"Starting the NCS Smart Licensing Java VM"	NCS スマートライセンス Java VM を起動しています。
NCS_SMART_LICENSING_FAIL	"The NCS Smart Licensing Java VM ~s"	NCS スマートライセンス Java VM が失敗したか、タイムアウトしました。
NCS_SMART_LICENSING_ GLOBAL_NOTIFICATION	"Smart Licensing Global Notification: ~s"	スマートライセンスのグローバル通知。
NCS_SMART_LICENSING_ ENTITLEMENT_NOTIFICATION	"Smart Licensing Entitlement Notification: ~s"	スマートライセンス資格の通知。
NCS_SMART_LICENSING_EVALUATION_COUNTDOWN	"Smart Licensing evaluation time remaining: ~s"	スマートライセンス評価の残り時間。
DEVEL_SLS	"~s"	デベロッパー スマートライセンス API ログメッセージ。
JSONRPC_REQUEST	"JSON-RPC: '~s' with JSON params ~s"	JSON-RPC メソッドがリクエストされました。
DEVEL_ECONFD	"~s"	デベロッパー econfd API ログメッセージ。
CDB_FATAL_ERROR	"fatal error in CDB: ~s"	CDB で回復不能なエラーが発生しました。
LOGGING_STARTED	"Daemon logging started"	ロギングサブシステムが開始されました。
LOGGING_SHUTDOWN	"Daemon logging terminating, reason: ~s"	ロギングサブシステムが終了しました。

記号	フォーマット文字列	備考
REOPEN_LOGS	"Logging subsystem, reopening log files"	ロギングサブシステムがログファイルを再度開きました。
OPEN_LOGFILE	"Logging subsystem, opening log file '~s' for ~s"	特定タイプのロギングのターゲットファイルを示します。
LOGGING_STARTED_TO	"Writing ~s log to ~s"	サブシステムのログを特定のファイルに書き込みます。
LOGGING_DEST_CHANGED	"Changing destination of ~s log to ~s"	ターゲットログファイルが別のファイルに変更されます。
LOGGING_STATUS_CHANGED	"~s ~s log"	サブシステムのロギングステータス（有効/無効）の変更を通知します。
ERRLOG_SIZE_CHANGED	"Changing size of error log (~s) to ~s (was ~s)"	エラーログのログサイズの変更を通知します。
CGI_REQUEST	"CGI: '~s' script with method ~s"	CGI スクリプトがリクエストされました。
MMAP_SCHEMA_FAIL	"Failed to setup the shared memory schema"	共有メモリスキーマの設定に失敗しました。
KICKER_MISSING_SCHEMA	"Failed to load kicker schema"	キッカースキーマのロードに失敗しました。
JSONRPC_REQUEST_IDLE_TIMEOUT	"Stopping session due to idle timeout: ~s"	JSON-RPC アイドルタイムアウト。
JSONRPC_REQUEST_ ABSOLUTE_TIMEOUT	"Stopping session due to absolute timeout: ~s"	JSON-RPC 絶対タイムアウト。

データベースのロック

このセクションでは、WAE に存在するさまざまなロックと、それらがどのように相互作用するかについて説明します。

グローバルロック

WAE 管理バックプレーンは、データストア（実行中）をロックし続けます。このロックはグローバルロックと呼ばれ、データストアへの排他的アクセスを許可するメカニズムを提供します。グローバルロックは、NETCONF <lock> 操作や Maapi.lock() 呼び出しなどノースバウンドエージェントを介して明示的に取得できる唯一のロックです。

グローバルロックは、データストア全体に対して行うことも、部分的なロックにする（データモデルのサブセットに対して行う）こともできます。部分ロックは、NETCONFおよびMAAPIを介して公開されます。

エージェントは、グローバルロックを要求して、排他的な書き込みアクセスを確保できます。エージェントがグローバルロックを保持している場合、他の誰もそのデータストアに書き込むことはできません。この動作は、トランザクションエンジンによって強制されます。他のロック所有者（部分ロックを含む）がなく、すべてのデータプロバイダーがロック要求を承認した場合に、実行中のグローバルロックがエージェントに付与されます。各データプロバイダー（CDB または外部データプロバイダー）には、ロックを拒否または受け入れるために呼び出される lock() コールバックがあります。**ncs --status** の出力には、ロックステータスが含まれます。

トランザクションロック

ノースバウンドエージェントは、WAE 管理バックプレーンに対するユーザーセッションを開始します。各ユーザーセッションは、複数のトランザクションを開始できます。トランザクションは、読み取り/書き込みまたは読み取り専用です。

トランザクションエンジンには、実行中のデータストアに対する内部ロックがあります。これらのトランザクションロックは、データストアに対する構成の更新をシリアル化するために存在し、グローバルロックとは別のものです。

ノースバウンドエージェントが実行中のデータストアを新しい構成で更新する場合、トランザクションロックを暗黙的に取得して解放します。トランザクションエンジンは、トランザクション ステート マシンを通過するときにロックを管理します。ノースバウンドエージェントにトランザクションロックを公開する API はありません。

トランザクションエンジンがトランザクションのロックを取得する場合（たとえば、検証状態に入るとき）、最初に他のトランザクションがロックを保持していないことを確認します。次に、そのデータストアにグローバルロックが設定されているユーザーセッションがないことを確認します。最後に、transLock() コールバックを使用して各データプロバイダーを呼び出します。

ノースバウンドエージェントとグローバルロック

暗黙的なトランザクションロックとは対照的に、一部のノースバウンドエージェントは、グローバルロックへの明示的なアクセスを公開します。管理 API は、Maapi.lock() および Maapi.unlock() メソッド（および部分ロック用の対応する Maapi.lockPartial() Maapi.unlockPartial()）を提供することにより、グローバルロックを公開します。ユーザーセッションが確立（または接続）されると、これらの関数を呼び出すことができます。

CLI では、次のように、さまざまな構成モードに入るときにグローバルロックが取得されます。

- **config exclusive**：実行中のデータストアのグローバルロックを取得します。
- **config terminal**：ロックを取得しません。

CLI は、構成モードが終了するまでグローバルロックを保持します。

エキスパート モードは、CLI と同じように動作し、前述の CLI モードに対応する [プライベート編集（Edit private）] および [排他編集（Edit exclusive）] と呼ばれる編集タブがあります。

NETCONF エージェントは、<lock> 操作を、リクエストされたデータストアのグローバルロックのリクエストに変換します。部分ロックも partial-lock rpc を通じて公開されます。

外部データプロバイダーと CDB

外部データプロバイダーは、lock() および unlock() コールバックを実装する必要はありません。WAE は、グローバルロックが取得されている間、データプロバイダーへの transLock() 状態遷移の開始を試みません。データプロバイダーがロック用のコールバックを実装する理由は、他の誰かがデータプロバイダーのデータベースに書き込むことができる場合です。

CDB は、lock() コールバックと unlock() コールバックを無視します（データ プロバイダー インターフェイスが唯一の書き込みインターフェイスであるため）。

CDB には、データベースに独自の内部ロックがあります。実行中のデータストアには、1 つの書き込みロックと複数の読み取りロックがあります。データストアにアクティブな読み取りロックがある場合、データストアの書き込みロックを取得することはできません。CDB のロックは、リーダーが常にデータの一貫したビューを取得できるようにするために存在します（YANG リストエントリの getNext() の呼び出しの間に別のユーザーが構成ノードを削除すると、混乱が生じます）。

トランザクション中、transLock() はトランザクションのデータストアに対して CDB 読み取りロックを取得しますが、writeStart() は読み取りロックを解放し、代わりに書き込みロックを取得しようとします。CDB 外部リーダークライアントは、Cdb.startSession() と Cdb.endSession() の間で暗黙的に CDB 読み取りロックを取得します。つまり、CDB クライアントが読み取りを行っている間、トランザクションは writeStart() を通過できません。逆に、トランザクションが writeStart() と commit() または abort() の間にある間は、CDB リーダーを開始できません。

CDB のオペレーショナルストアにはロックがありません。WAE のトランザクションエンジンは、そこからのみ読み取ることができます。CDB クライアントの書き込みは、書き込み操作単位でアトミックです。

ユーザーセッションへのロックの影響

セッションがロックされているデータストアを変更しようとすると、失敗します。たとえば、CLI は次のように出力します。

```
admin@wae(config)# commit
Aborted: the configuration database is locked
```

一部のロックは持続時間が短いため（CDB 読み取りロックなど）、WAE はデフォルトで、失敗した操作を構成可能な時間だけ再試行するように構成されています。この時間が経過してもデータストアがロックされたままの場合、操作は失敗します。

再試行タイムアウトを構成するには、wae.conf で /ncs-config/commit-retry-timeout 値を設定します。

セキュリティ

WAE には、特定のタスクを実行する権限が必要です。ターゲットシステムによっては、次のタスクにルート権限が必要になる場合があります。

- 特権ポートへのバインド。wae.conf 構成ファイルは、WAE が bind(2) する必要があるポート番号を指定します。ポート番号が 1024 より小さい場合、ターゲット オペレーティング システムで WAE が非ルートユーザーとしてこれらのポートにバインドすることを許可しない限り、通常、WAE はルート権限を必要とします。
- PAM を認証に使用する場合、$NCS_DIR/lib/ncs/priv/pam/epam としてインストールされたプログラムが PAM クライアントとして機能します。ローカルの PAM 構成によっては、このプログラムにルート権限が必要になる場合があります。PAM がローカルの passwd ファイルを読み取るように構成されている場合、プログラムはルートとして実行するか、setuid root である必要があります。ローカル PAM 構成で、WAE に pam_radius_auth などを実行するように指示している場合、ローカル PAM のインストールによっては、ルート権限が必要ない場合があります。
- CLI を使用して実行可能ファイルを実行する CLI コマンドを作成する場合は、$NCS_DIR/lib/ncs/priv/ncs/cmdptywrapper プログラムのアクセス許可を変更します。

ルートまたは特定のユーザーとして実行可能ファイルを実行するには、cmdptywrapper を setuid root にします。

```
# chown root cmdptywrapper
# chmod u+s cmdptywrapper
```

これに失敗すると、すべてのプログラムは WAE デーモンを実行しているユーザーとして実行されます。そのユーザーがルートの場合、上記の chmod 操作を実行する必要はありません。

これに失敗すると、すべてのプログラムは confd デーモンを実行しているユーザーとして実行されます。そのユーザーがルートの場合、上記の chmod 操作を実行する必要はありません。

アクションを介して実行される実行可能ファイルの場合、$NCS_DIR/lib/ncs/priv/ncs/cmdwrapper プログラムのアクセス許可を変更します。

```
# chown root cmdwrapper
# chmod u+s cmdwrapper
```

WAE は、クリアテキスト TCP を介して NETCONF を終了するように指示できます。これは、デバッグに役立ちます（NETCONF トラフィックをキャプチャして分析できます）。また、SSH 以外のローカル独自のトランスポートメカニズムを提供する場合にも役立ちます。クリアテキスト TCP による終了は認証されません。クリアテキストクライアントは、セッションを

実行するユーザーを WAE に通知するだけです。認証は、SSH サーバーなどの外部エンティティによってすでに行われていることが前提です。クリアテキスト TCP が有効になっている場合、WAE はこれらの接続のために localhost（127.0.0.1）にバインドする必要があります。

クライアントライブラリは WAE に接続します。たとえば、CDB API は TCP ベースであり、CDB クライアントは WAE に接続します。WAE は、`wae.conf` パラメータ `/ncs-config/ncs-ipc-address/ip`（デフォルトのアドレスは 127.0.0.1）および `/ncs-config/ncs-ipcaddress/port`（デフォルトのポートは4565）を介して、これらの接続に使用するアドレスを学習します。

WAE は、同じソケット上でさまざまな種類の接続を多重化します（IP とポートの組み合わせ）。次のプログラムはソケットに接続します。

- **ncs --reload** などのリモートコマンド。
- CDB クライアント。
- 外部データベース API クライアント。
- 管理エージェント API（MAAPI）クライアント。
- ncs_cli プログラム。

デフォルトでは、上記のプログラムは信頼できると見なされます。MAAPI クライアントと ncs_cli は、WAE に接続する前にユーザーを認証します。CDB クライアントと外部データベース API クライアントは信頼できると見なされるため、認証は必要ありません。

ncs-ipc-address ソケットはシステムへの完全な非認証アクセスを許可するため、信頼できないネットワークからソケットにアクセスできないようにすることが重要です。アクセスチェックを使用して、ncs-ipc-address ソケットへのアクセスを制限することもできます。IPC ポートへのアクセスの制限 （164 ページ） を参照してください。

IPC ポートへのアクセスの制限

デフォルトでは、IPC ポートに接続するクライアントは信頼できると見なされます。認証は必要ありません。リモートアクセスを防止するために、WAE は `/ncs-config/ncs-ipc-address/ip` に 127.0.0.1 を使用します。ただし、アクセスチェックを構成することで、IPC ポートへのアクセスを制限できます。

アクセスチェックを有効にするには、`wae.conf` 要素 `/ncs-config/ncs-ipc-accesscheck/enabled` を **true** に設定し、`/ncs-config/ncs-ipc-accesscheck/filename` にファイル名を指定します。ファイルには、共有秘密（ランダムな文字による文字列）が含まれている必要があります。IPC ポートに接続するクライアントは、WAE 機能へのアクセス権が付与される前に、チャレンジハンドシェイクを提供する必要があります。

（注） このファイルのアクセス許可は、IPC ポートへの接続が許可されている WAE デーモンおよびクライアントプロセスによってのみファイルが読み取られるように、OS ファイル権限によって制限する必要があります。たとえば、デーモンとクライアントの両方が root として実行されている場合、ファイルは root によって所有され、「所有者による読み取り」権限（モード 0400）のみを持つことができます。別の方法は、デーモンとクライアントのみが属するグループを作成し、ファイルのグループ ID をそのグループに設定し、「グループによる読み取り」（モード 040）権限のみを持つことです。

クライアントライブラリに秘密を提供し、アクセス チェック ハンドシェイクを使用するように指示するには、環境変数 NCS_IPC_ACCESS_FILE を、シークレットを含むファイルのフルパス名に設定します。上記のすべてのクライアントにはこれで十分です。このチェックを有効にするためにアプリケーションコードを変更する必要はありません。

（注） アクセスチェックは、デーモンとクライアントの両方に対して有効または無効にする必要があります。たとえば、`wae.conf` 要素 `/ncsconfig/ncs-ipc-access-check/enabled` が **true** に設定されていない場合に、秘密が含まれるファイルを環境変数 NCS_IPC_ACCESS_FILE で指してクライアントが起動される場合、クライアント接続は失敗します。

WAE 運用データのクリア

データベースから WAE 運用データを消去するには、それぞれの NIMO ネットワークモデルからモデル l1-model を削除する必要があります。その後、デバイスツリーを削除します。NIMO ネットワークモデルにレイアウトがある場合は、それらのレイアウトを NIMO ネットワークモデルから削除します。

次のコマンド例は、as64002 ネットワークモデルとデバイスツリーから運用データを消去する方法を示しています。

```
delete networks network as64002 model
delete networks network as64002 layouts
delete networks network as64002 l1-model
delete devices device *
commit
```

WAE 構成のバックアップと復元

YANG ランタイムフレームワークを使用すると、WAE 構成を簡単にバックアップおよび復元できます。収集を開始する前（つまり、運用データが読み込まれる前）に、WAE 構成をバックアップすることをお勧めします。

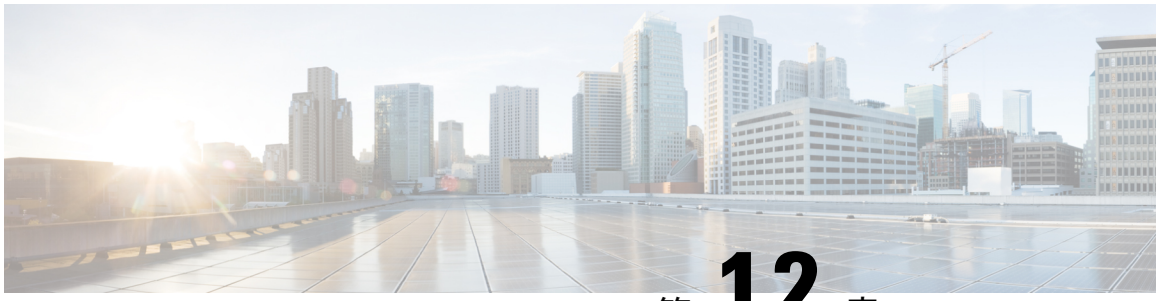
- WAE 構成をバックアップするには、次の手順を実行します。

```
admin@wae% save /home/wae/wae-backup.cfg
```

上記のコマンドは、構成データと運用データの両方をバックアップします。構成データのみをバックアップするには、WAE 運用データのクリア （165 ページ）の説明に従って、データベースから運用データを消去する必要があります。すべての運用データが削除されるため、実稼働環境で運用データを消去する前に注意してください。

- WAE 構成を復元するには、次の手順を実行します。

```
[wae@wae ~]$ ncs_load -l -m -F j wae-backup.cfg
```



第 12 章

セキュリティ

主要なセキュリティ概念

製品のセキュリティの最適化を目指す管理者は、次のセキュリティ概念をよく理解しておく必要があります。

HTTPS

Hypertext Transfer Protocol Secure（HTTPS）では、チャネルを介して送信されるデータの暗号化に、セキュア ソケット レイヤ（SSL） またはその後続の標準規格である Transport Layer Security（TLS）が使用されます。SSL で複数の脆弱性が見つかったため、 では現在 TLS のみがサポートされています。

（注） TLS は大まかに SSL と呼ばれることが多いため、本ガイドでもこの表記に従います。

SSL は、プライバシー、認証、およびデータ整合性を組み合わせることで、クライアントとサーバーの間のデータ転送を保護します。これらのセキュリティ メカニズムを有効にするために、SSL は証明書、秘密キー/公開キー交換ペア、および Diffie-Hellman 鍵共有パラメータを使用します。

SSL 証明書

SSL 証明書と秘密キー/公開キー ペアは、ユーザー認証および通信パートナーの ID 検証に使われるデジタル ID の一種です。VeriSign や Thawte などの認証局（CA）は、エンティティ（サーバーまたはクライアント）を識別するための証明書を発行します。クライアントまたはサーバー証明書には、発行認証局の名前とデジタル署名、シリアル番号、証明書が発行されたクライアントまたはサーバーの名前、公開キー、および証明書の有効期限が含まれます。CA は、1 つ以上の署名証明書を使用して SSL 証明書を作成します。各署名証明書には、CA 署名の作成に使用される照合秘密キーがあります。CA は署名付き証明書（公開キーが埋め込まている）

を簡単に入手できるようにしているため、誰でもその証明書を使用して、SSL 証明書が実際に特定の CA によって署名されたことを確認できます。

一般に、証明書の設定には次の手順が含まれます。

1. サーバーの ID 証明書を生成する。
2. サーバーに ID 証明書をインストールする。
3. 対応するルート証明書をクライアントまたはブラウザにインストールする。

実行する必要がある具体的なタスクは、ご利用の環境によって異なります。

1 方向 SSL 認証

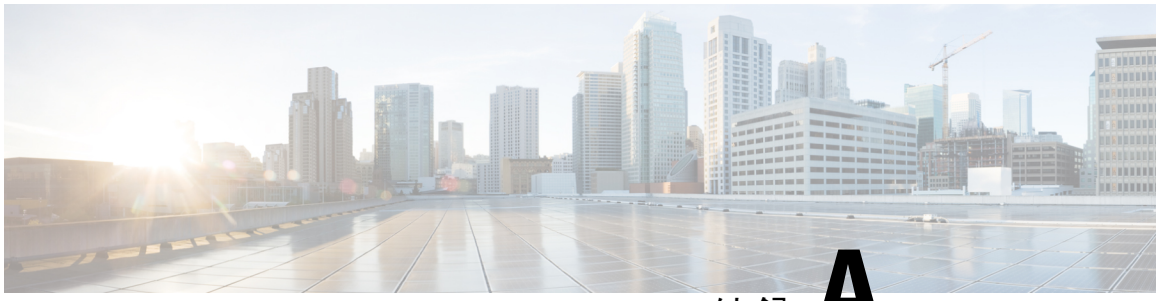
これは、クライアントが適切なサーバー（中間サーバーではなく）に接続していることを保証する必要がある場合に使用される認証方法で、オンライン バンキングの Web サイトなどのパブリック リソースに適しています。認証は、クライアントがサーバー上のリソースへのアクセスを要求したときに開始されます。リソースが存在するサーバーは、その ID を証明するために、サーバー証明書（別名 SSL 証明書）をクライアントに送信します。クライアントは受信したサーバー証明書を、クライアントまたはブラウザにインストールする必要がある別の信頼できるオブジェクト（サーバー ルート証明書）と照合して検証します。サーバーの検証後、暗号化された（つまりセキュアな）通信チャネルが確立されます。ここで、サーバは HTML フォームへの有効なユーザ名とパスワードの入力を求めます。SSL 接続が確立された後にユーザー クレデンシャルを入力すると、未認証の第三者による傍受を防ぐことができます。最終的に、ユーザー名とパスワードが受け入れられた後、サーバー上に存在するリソースへのアクセスが許可されます。

（注） クライアントは複数のサーバーとやり取りするために、複数のサーバー証明書を格納する必要がある場合があります。

クライアントにルート証明書をインストールする必要があるかどうかを判断するには、ブラウザの URL フィールドでロック アイコンを探します。通常このアイコンが表示される場合は、

必要なルート証明書がすでにインストール済みであることを示します。多くの場合、これはより大きいいずれかの認証局（CA）によって署名されたサーバー証明書に該当します。一般的なブラウザではこれらの CA からのルート証明書が含まれているからです。

クライアントがサーバー証明書に署名した CA を認識しない場合は、接続がセキュリティで保護されていないことを意味します。これは必ずしも大きな問題ではなく、接続するサーバーの ID が検証されていないことを示しているだけです。この時点で、次の 2 つの操作のいずれかを実行できます。1 つは必要なルート証明書をクライアントまたはブラウザにインストールすることです。ブラウザの URL フィールドにロック アイコンが表示された場合は、証明書が正常にインストールされたことを意味します。もう 1 つは、クライアントに自己署名証明書をインストールできることです。信頼できる CA によって署名されたルート証明書とは異なり、自己署名証明書は作成者である個人またはエンティティによって署名されます。自己署名証明書を使用して暗号化チャネルを作成できますが、接続するサーバの ID が検証されていないため、固有のリスクが伴うことを理解しておいてください。



付録 A

その他の WAE CLI コマンド

ここでは、次の内容について説明します。

コミットフラグ

コミットフラグはトランザクションのセマンティクスを変更します。**commit** コマンドを発行するときにコミットフラグを使用します。

commit *<flag>*

次の表に、一般的に使用されるフラグの一部を示します。

コマンド	説明
and-quit	コミット後に CLI 動作モードを終了します。
bypass-commit-queue	コミットキューをバイパスして、直接コミットを試みます。このフラグは、コミットキューが（構成項目 /devices/global-settings/commit-queue/enabled-bydefault によって）デフォルトで使用されている場合にのみ関連します。 コミットキューにコミットされるトランザクションと同じデバイスに影響するエントリが含まれている場合、操作は失敗します。
check	保留中の構成変更を検証します。**validate** コマンドと同等です。
comment|**label**	コンプライアンスレポート、ロールバックファイルなどに表示されるコミットコメントまたはラベルを追加します。
dry-run	構成の変更を検証して表示しますが、実際のコミットは実行しません。CDB もデバイスも影響を受けません。さまざまな出力フォーマットがサポートされています。

no-networking	構成の変更を検証し、CDB を更新しますが、実際のデバイスは更新しません。これは、最初に admin-state の状態を southbound-locked に設定してから、標準のコミットを発行することと同じです。どちらの場合も、構成の変更は実際のデバイスにコミットされません。 コミットが変更を意味する場合、デバイスは非同期になります。
no-out-of-sync-check	デバイスが非同期であってもコミットします。このフラグは、変更がデバイスの内容と競合しないことがわかっていて、最初に **sync-from** を実行したくないシナリオで使用できます。**device compare-config** を使用して結果を確認します。 コミットが変更を意味する場合、デバイスは非同期になります。
no-revision-drop	デバイスに古いデバイスモデルがある場合は失敗します。WAE が NETCONF デバイスに接続すると、デバイスデータモデルのバージョンが検出されます。ネットワーク内のデバイスが異なれば、バージョンが異なる場合があります。WAE が構成をデバイスに送信する場合、デフォルトでは、デバイスがサポートするモデルよりも新しいモデルにのみ存在する構成はすべて破棄されます。
through-commit-queue	構成の変更は CDB にすぐにコミットされますが、実際のデバイスにはコミットされません。代わりに、トランザクションのスループットを向上させる目的で、構成変更は最終的なコミットのためにキューに入れられます。これにより、デフォルトで有効にしなくても、個々のコミットコマンドに対してコミットキュー機能を使用できるようになります。

すべての WAE コマンドには、パイプコマンドを含めることができます。たとえば、**details** パイプコマンドは、コミットで実行されたステップに関するフィードバックを提供します。

```
wae% commit | details
```

すべてのテンプレートでデバッグを有効にするには、**debug** パイプコマンドを使用します。

```
wae% commit | debug template
```

構成中に多くのテンプレートを使用すると、デバッグ出力が膨大になる可能性があります。次の *l3vpn* という名前のテンプレートの例に示すように、デバッグ情報を 1 つのテンプレートだけに制限できます。

```
wae% commit | debug template l3vpn
```

デバイス アクション

デバイスのアクションは /devices パスでグローバルに実行でき、個々のデバイスは /devices/device/name で実行できます。多くのアクションは、デバイスグループとデバイス範囲に対しても使用できます。

次の表に、デバイスアクションを示します。

コマンド	説明
check-sync	デバイス構成の WAE コピーが実際のデバイス構成と同期しているかどうかを確認します。この操作は、デバイスからの構成の署名のみを比較します。構成全体を比較するわけではありません。 署名は、transaction-id、time-stamp、または hash-sum として実装されます。対応する NED が機能をサポートしている必要があります。出力にサポートされていないことが示されている場合は、完全な **device compare-config** コマンドを使用する必要があります。
check-yang-modules	WAE とデバイスに互換性のある YANG モジュールがあるかどうかを確認します。
clear-trace	すべてのトレースファイルをクリアします。
commit-queues	キューに入れられたコミットのリストを表示します。
connect	ロック解除されたデバイスへのセッションを設定します。WAE はオンデマンドで接続を自動的に確立するため、このアクションは実際の運用シナリオでは使用されません。ただし、このアクションは、新しい NED をインストールするとき、デバイスを追加するときなどのテスト目的で役立ちます。
disconnect	デバイスへのセッションを閉じます。
sync-from	実際のデバイス構成を読み取って、デバイス構成の WAE コピーを同期します。変更は WAE にすぐにコミットされ、ロールバックできません。 いずれかのサービスがデバイス上で構成を作成した場合、対応するサービスは非同期になる可能性があります。この不一致を調整するには、コマンド **service check-sync** および **service re-deploy** を使用します。
sync-to	WAE コピーをデバイスにプッシュして、デバイス構成を同期します（このアクションはロールバックできません）。

サービスアクション

前述のデバイス操作の多くは、オプション **no-networking** と組み合わせることができます。このオプションは、構成データベースでのみすべての更新を実行し、デバイスは非同期になります。更新は後でネットワークにプッシュできます（このアクションは、デバイスを admin-state southbound-locked に設定するのと同じです）。

次の表に、サービスアクションを示します。

コマンド	説明

check-sync	サービスとそれに関連付けられているデバイス構成が同期していることを確認します。相違点は、選択した出力フォーマットで表示されます。 構成の変更がアウトオブバンドで行われた場合は、非同期状態を検出するために **deep-check-sync** が必要です。
deep-check-sync	実際のデバイスがサービスに従って構成されているかどうかを検証します。**re-deploy** を使用してサービスを調整します。
get-modifications	サービスによって作成された構成データを取得します。
re-deploy	すべてのサービスデータを考慮してサービスロジックを再実行し、構成データベースのデバイス構成を使用して差分を生成します。構成差分をデバイスに送信します。このアクションは、次の場合に役立ちます。 • 帯域外の変更を組み込むために、**device sync-from** アクションが実行された。 • サービスによって参照されるデータ（トポロジ情報、QoS ポリシー定義など）が変更された。 このアクションはべき等です。構成差分が存在しない場合は、何も実行される必要はありません。最小変更に関する WAE の一般原則が適用されます。
un-deploy	ネットワーク上のサービスの影響を元に戻します。このアクションにより、実際のデバイスおよび WAE 構成データベースから構成が削除されます。

wae.conf 構成パラメータ

次の表に、`wae.conf` 構成パラメータとそのタイプ（丸カッコ内）およびデフォルト値（角カッコ内）をリストします。パラメータは、それらが互いにどのように関係しているかを簡単に確認できるように、パス表記を使用して記述されます。

パラメータ	説明
/ncs-config	WAE 構成。
/ncs-config/db-mode (running) [running]	この機能は廃止されました。WAE は、running db-mode のみをサポートします。 このリーフの設定は必須ではありません。これは、後方互換性のためにのみ保持されています。

パラメータ	説明
/ncs-config/ncs-ipc-address	WAE は、CDB、MAAPI、CLI、外部データベース API などの WAE クライアントライブラリからの着信 TCP 接続、および ncs スクリプトからのコマンド（「ncs --reload」など）を、デフォルトで 127.0.0.1:4569 でリッスンします。IP アドレスとポートは変更可能です。変更された場合は、MAAPI、CDB などを使用するすべてのクライアントを再コンパイルして、処理できるようにする必要があります。 **注意** WAE が localhost 以外に bind(2) するように指示されている場合、セキュリティに重大な影響が生じます。WAE がすべての IPv4 アドレスで listen(2) するようにする場合は、IP 0.0.0.0 を使用します。
/ncs-config/ncs-ipc-address/ip (ipv4-address | ipv6-address) [127.0.0.1]	WAE が Java ライブラリからの着信接続をリッスンする IP アドレス。
/ncs-config/ncs-ipc-address/port (port-number) [4569]	WAE が Java ライブラリからの着信接続をリッスンするポート番号。
/ncs-config/ncs-ipc-extra-listen-ip (ipv4-address | ipv6-address)	このパラメータは複数回指定できます。WAE IPC リスナーをバインドする追加の IP をリストします。これは、WAE IPC を特定のインターフェイスに公開することがないように、ワイルドカード 0.0.0.0 アドレスを使用しない場合に役立ちます。
/ncs-config/ncs-ipc-access-check	WAE は、IPC リスナーソケットへの着信接続のアクセスを制限するように構成できます。アクセスチェックでは、接続しているクライアントが共有秘密を所有していることを証明する必要があります。
/ncs-config/ncs-ipc-access-check/enabled (boolean) [false]	「true」の場合、IPC 接続のアクセスチェックが有効になります。
/ncs-config/ncs-ipc-access-check/filename (string)	このパラメータは必須です。*filename* は、IPC アクセスチェックの共有秘密を含むファイルへのフルパスです。ファイルは、IPC リスナーソケットへの接続が許可されている WAE デーモンおよびクライアントプロセスによってのみ読み取られるように、OS ファイル権限によって保護する必要があります。
/ncs-config/enable-shared-memory-schema (boolean) [true]	*enabled* は true または false のいずれかです。true の場合、C プログラムが起動し、スキーマを共有メモリにロードします（これにより、たとえば Python からアクセスできます）。
/ncs-config/load-path	—

パラメータ	説明
/ncs-config/load-path/dir（文字列）	このパラメータは複数回指定できます。*load-path* 要素には、任意の数の *dir* 要素が含まれます。各 *dir* 要素は、デーモンの起動時にコンパイルおよびインポートされた YANG ファイル（.fxs ファイル）およびコンパイルされた clispec ファイル（.ccl ファイル）が検索されるディスク上のディレクトリパスを指します。また、WAEは、最初の起動時、または /packages/reload アクションによってリクエストされたときに、パッケージのロードパスを検索します。
/ncs-config/state-dir (string)	このパラメータは必須です。これは、WAE が永続的な状態データを書き込む場所です。ロードパスで見つかったすべてのパッケージのプライベートコピーを、「packages-in-use.cur」をルートとする（また、シンボリックリンク「packages-in-use」によって参照される）ディレクトリツリーに保存します。また、実行中のデータベースステータスが無効な場合にのみ存在する状態ファイル「running.invalid」にも使用されます。この状態は、2フェーズコミットプロトコル中にデータベース実装の 1 つが失敗した場合に発生します。また、再起動後も保持する必要があるデータを格納するために使用される「global.data」にも使用されます。
/ncs-config/commit-retry-timeout (xs:duration | infinity) [infinity]	WAE バックプレーンでのコミットタイムアウト。このタイムアウトは、別のエンティティがデータベースをロックしているときに（たとえば、別のコミットが進行中の場合や、管理対象オブジェクトがデータベースをロックしている場合）、CLI および JSON-RPC API でのコミット操作が完了しようとする時間を制御します。
/ncs-config/max-validation-errors (uint32 | unbounded) [1]	一度に収集してユーザーに表示する検証エラーの数を制御します。
/ncs-config/notifications	NETCONF ノースバウンド通知設定を定義します。
/ncs-config/notifications/event-streams	使用可能なすべての通知イベントストリームをリストします。
/ncs-config/notifications/event-streams/stream	単一の通知イベントストリームのパラメータ。
/ncs-config/notifications/event-streams/stream/name (string)	特定のイベントストリームに関連付けられた名前。
/ncs-config/notifications/event-streams/stream/description (string)	このパラメータは必須です。特定のイベントストリームに関連付けられた説明テキスト。
/ncs-config/notifications/event-streams/stream/replay-support (boolean)	このパラメータは必須です。特定のイベントストリームで再生サポートが利用可能かどうかを通知します。

パラメータ	説明
/ncs-config/notifications/event-streams/stream/builtin-replay-store	このイベントストリームの組み込み再生ストアのパラメータ。 再生サポートが有効になっている場合、WAE はすべての通知をディスクに自動的に保存し、NETCONF マネージャが記録されている通知を要求した場合に再生できるようにします。再生ストアは、ディスク上の（特定の数とサイズの）ラップログファイルのセットを使用して通知を保存します。 特定の時間範囲で通知を高速に再生するには、各ラップログファイルの最大サイズが大きすぎないようにする必要があります。可能であれば、代わりに多数のラップログファイルを使用します。確信が持てない場合は、推奨設定を使用してください（以下を参照）。
/ncs-config/notifications/event-streams/stream/builtin-replay-store/enabled (boolean) [false]	「false」の場合、アプリケーションは独自の再生サポートを実装する必要があります。
/ncs-config/notifications/event-streams/stream/builtin-replay-store/dir (string)	このパラメータは必須です。ラップログファイルのディスクの場所。
/ncs-config/notifications/event-streams/stream/builtin-replay-store/max-size (tailf:size)	このパラメータは必須です。各ログラップファイルの最大サイズ。推奨設定は S10M 程度です。
/ncs-config/notifications/event-streams/stream/builtin-replay-store/max-files (int64)	このパラメータは必須です。ログラップファイルの最大数。推奨設定は 50 ファイル前後です。
/ncs-config/opcache	運用データキャッシュの動作を制御します。
/ncs-config/opcache/enabled (boolean) [false]	「true」の場合、キャッシュは有効です。
/ncs-config/opcache/timeout (uint64)	このパラメータは必須です。データをキャッシュに保持する時間（秒単位）。
/ncs-config/hide-group	再表示できる非表示グループをリストします。構成には、0個、1個、または複数の hide-group エントリを含めることができます。 非表示グループに hide-group エントリがない場合、CLI の「unhide」コマンドを使用して再表示することはできません。ただし、hide-group エントリを ncs.conf ファイルに追加してから、**ncs -- reload** を使用して、CLI で使用できるようにすることができます。これは、たとえば、パスワードを使用してもアクセス可能にしない診断非表示グループを有効にする場合に役立ちます。
/ncs-config/hide-group/name (string)	非表示グループの名前。「tailf:hidden」を指定して YANG モジュールで定義された非表示グループ名に対応する必要があります。

パラメータ	説明
/ncs-config/hide-group/ password (tailf:md5-digest-string) []	オプションで、非表示グループにパスワードを指定できます。パスワードまたはコールバックが指定されていない場合、非表示グループはパスワードを指定せずに再表示できます。パスワードが指定されている場合、パスワードを入力しないと非表示グループを有効にすることはできません。 非表示グループを完全に無効にするには（つまり、再表示できないようにするには）、その非表示グループの hide-group コンテナ全体を削除します。
/ncs-config/hide-group/ callback (string)	オプションで、非表示グループにコールバックを指定できます。コールバックまたはパスワードが指定されていない場合、非表示グループはパスワードを指定せずに再表示できます。コールバックが指定されている場合、パスワードを入力して検証されないと非表示グループを有効にすることはできません。コールバックは、非表示グループの名前、unhide コマンドを発行したユーザーの名前、およびパスワードを受け取ります。コールバックを使用すると、有効期間の短い再表示パスワードやユーザー単位の再表示パスワードを使用できます。
/ncs-config/cdb	—
/ncs-config/cdb/db-dir (string)	*db-dir* は、CDB がストレージおよび一時ファイルに使用するディスク上のディレクトリです。また、CDB が初期化ファイルを検索するディレクトリでもあります。
/ncs-config/cdb/init-path	—
/ncs-config/cdb/init-path/dir (string)	このパラメータは複数回指定できます。*init-path* 要素には、任意の数の *dir* 要素を含めることができます。各 *dir* 要素は、CDB が *db-dir* 内を検索する前に .xml ファイルを検索するディレクトリパスを指します。ディレクトリは、リストされている順序で検索されます。
/ncs-config/cdb/client-timeout (xs:duration | infinity) [infinity]	CDB がクライアントの応答を待機してから応答していないと見なすまでの時間を指定します。クライアントがタイムアウト期間内に Cdb.syncSubscriptionSocket() の呼び出しに失敗した場合、CDB はこの失敗の syslog を記録し、クライアントが停止していると見なして、ソケットを閉じてサブスクリプション通知を続行します。infinity に設定すると、CDB はクライアントからの応答待機をタイムアウトすることはありません。
/ncs-config/cdb/subscription-replay	—
/ncs-config/cdb/subscription-replay/enabled (boolean) [false]	有効にすると、新しい CDB サブスクライバへの以前のサブスクリプション通知の再生を要求できます。

パラメータ	説明
/ncs-config/cdb/replication (async | sync) [sync]	CDB レプリケーションが有効になっている場合（つまり高可用性モードが有効になっている場合。/ncs-config/ha を参照）、CDB 構成ストアを非同期または同期的にレプリケートできます。非同期レプリケーションでは、接続されたスレーブに更新が送信されるとすぐに、構成を更新するトランザクションを完了することができます。デフォルトの同期レプリケーションでは、更新がスレーブに完全に反映され、スレーブのサブスクライバ（存在する場合）がサブスクリプション通知を確認するまで、トランザクションは中断されます。
/ncs-config/cdb/journal-compaction (automatic | manual) [automatic]	CDB 構成ストアがジャーナル圧縮を行う方法を制御します。cdb_initiate_journal_compaction() API 呼び出しを使用して圧縮を制御する外部メカニズムがない限り、デフォルトの「automatic」以外には設定しないでください。
/ncs-config/cdb/operational	運用データは、外部コールバックによって実装するか、CDB に格納できます（またはその両方を組み合わせます）。運用データストアは、データが CDB に格納されるときに使用されます。
/ncs-config/cdb/operational/ db-dir (string)	*db-dir* は、CDB 操作がストレージおよび一時ファイルに使用するディスク上のディレクトリです。設定しない場合（デフォルト）、CDB の *db-dir* と同じディレクトリが使用されます。
/ncs-config/encrypted-strings	*encrypted-strings* は、タイプ tailf:des3-cbc-encryptedstring および tailf:aes-cfb-128-encrypted-string に準拠する文字列の暗号化に使用されるキーを定義します。
/ncs-config/encrypted-strings/DES3CBC	DES3CBC では、3 つの 64 ビット（8 バイト）キーとランダムな初期ベクトルが文字列の暗号化に使用されます。initVector リーフは、以前のバージョンからアップグレードする場合にのみ使用されますが、後方互換性のために保持されています。
/ncs-config/encrypted-strings/ DES3CBC/key1 (hex8-value-type)	このパラメータは必須です。
/ncs-config/encrypted-strings/ DES3CBC/key2 (hex8-value-type)	このパラメータは必須です。
/ncs-config/encrypted-strings/ DES3CBC/key3 (hex8-value-type)	このパラメータは必須です。
/ncs-config/encrypted-strings/ DES3CBC/initVector (hex8-value-type)	—

パラメータ	説明
/ncs-config/encrypted-strings/ AESCFB128	AESCFB128 では、1 つの 128 ビット（16 バイト）キーとランダムな初期ベクトルが文字列の暗号化に使用されます。initVector リーフは、以前のバージョンからアップグレードする場合にのみ使用されますが、後方互換性のために保持されています。
/ncs-config/encrypted-strings/ AESCFB128/key (hex16-value-type)	このパラメータは必須です。
/ncs-config/encrypted-strings/ AESCFB128/initVector (hex16-value-type)	—
/ncs-config/crypt-hash	*crypt-hash* は、タイプ ianach:crypt-hash、tailf:sha-256-digest-string、および tailf:sha-512-digest-string のリーフについて、クリアテキスト値をハッシュする方法を指定します。
/ncs-config/crypt-hash/algorithm (md5 | sha-256 | sha-512) [md5]	*algorithm* を値「md5」、「sha-256」、または「sha-512」のいずれかに設定して、ianach:crypt-hash タイプのクリアテキスト入力のハッシュに対応するハッシュアルゴリズムを選択できます。
/ncs-config/crypt-hash/rounds (crypt-hash-rounds-type) [5000]	ianach:crypt-hash タイプの「sha-256」および「sha-512」アルゴリズムの場合、および tailf:sha-256-digest-string および tailf:sha-512-digest-string タイプの場合、*rounds* はハッシュループを何回実行する必要があるかを指定します。デフォルトの 5000 以外の値が指定されている場合、ハッシュされたフォーマットには「rounds=N$」が含まれます。N は指定された値で、ソルトの前に付加されます。このパラメータは、ianach:crypt-hash の「md5」アルゴリズムでは無視されます。
/ncs-config/logs	—
/ncs-config/logs/syslog-config	syslog に記録する方法の共有設定。ログは、ファイルまたは syslog に記録するように構成できます。ログが syslog に記録されるように構成されている場合、/ncs-config/logs/syslog-config の下の設定が使用されます。
/ncs-config/logs/syslog-config/version (bsd | 1) [bsd]	*version* は、「bsd」（従来の syslog）または「1」（新しい IETF syslog フォーマット：RFC 5424）のいずれかです。「1」は、/ncs-config/logs/syslog-config/udp/enabled を true に設定する必要があることを意味します。
/ncs-config/logs/syslog-config/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32) [daemon]	このファシリティ設定はデフォルトのファシリティです。異なるログに個々のファシリティを設定することもできます。
/ncs-config/logs/syslog-config/udp	—

パラメータ	説明
/ncs-config/logs/syslog-config/ udp/enabled (boolean) [false]	「false」の場合、メッセージはローカルの syslog デーモンに送信されます。
/ncs-config/logs/syslog-config/ udp/host (string | ipv4-address | ipv6-address)	このパラメータは必須です。*host* は、ドメイン名または IPv4/IPv6 ネットワークアドレスのいずれかです。UDP syslog メッセージがこのホストに送信されます。
/ncs-config/logs/syslog-config/ udp/port (port-number) [514]	*port* は、/ncs-config/logs/syslog-config/udp/host と組み合わせて使用される有効なポート番号です。
/ncs-config/logs/syslog-config/ syslog-servers	これは、UDP syslog サーバーを指定する別の方法です。/ncs-config/logs/syslog-config/udp コンテナを構成すると、このコンテナの構成はすべて無視されます。
/ncs-config/logs/syslog-config/ syslog-servers/server	すべての syslog メッセージのコピーを受信する一連の syslog サーバー。
/ncs-config/logs/syslog-config/ syslog-servers/server/host (string | ipv4-address | ipv6-address)	*host* は、ドメイン名または IPv4/IPv6 ネットワークアドレスのいずれかです。UDP syslog メッセージがこのホストに送信されます。
/ncs-config/logs/syslog-config/ syslog-servers/server/port (port-number) [514]	*port* は、この syslog サーバーがリッスンしている UDP ポート番号です。
/ncs-config/logs/syslog-config/ syslog-servers/server/version (bsd | 1) [bsd]	*version* は、「bsd」（従来の syslog）または「1」（新しい IETF syslog フォーマット：RFC 5424）のいずれかです。
/ncs-config/logs/syslog-config/ syslog-servers/server/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32) [daemon]	—
/ncs-config/logs/syslog-config/ syslog-servers/server/enabled (boolean) [true]	「false」の場合、この syslog サーバーは UDP メッセージを受信しません。
/ncs-config/logs/ncs-log	ncs-log は WAE のデーモンログです。WAE デーモン自体の起動の問題については、このログを確認してください。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/ncs-log/ enabled (boolean) [true]	「true」の場合、ログは有効です。
/ncs-config/logs/ncs-log/file	—

パラメータ	説明
/ncs-config/logs/ncs-log/ file/name (string)	*name* は、実際のログファイルへのフルパスです。
/ncs-config/logs/ncs-log/file/ enabled (boolean) [false]	「true」の場合、ファイルのロギングが有効になります。
/ncs-config/logs/ncs-log/syslog	—
/ncs-config/logs/ncs-log/ syslog/enabled (boolean) [false]	「true」の場合、syslog メッセージが送信されます。
/ncs-config/logs/ncs-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)	このオプションの値は、指定されたログの /ncs-config/logs/syslog-config/facility をオーバーライドします。
/ncs-config/logs/developer-log	*developer-log* は、ユーザー作成の Java コードをトラブルシューティングするためのデバッグログです。このログを有効にして、検証コードに問題がないか確認します。デフォルトでこのログは有効です。他のすべての点では、ncs-log として構成できます。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/developer-log/ enabled (boolean) [true]	「true」の場合、ログは有効です。
/ncs-config/logs/developer-log/ file	—
/ncs-config/logs/developer-log/ file/name (string)	*name* は、実際のログファイルへのフルパスです。
/ncs-config/logs/developer-log/ file/enabled (boolean) [false]	「true」の場合、ファイルのロギングが有効になります。
/ncs-config/logs/developer-log/ syslog	—
/ncs-config/logs/developer-log/ syslog/enabled (boolean) [false]	「true」の場合、syslog メッセージが送信されます。
/ncs-config/logs/developer-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)	このオプションの値は、指定されたログの /ncs-config/logs/syslog-config/facility をオーバーライドします。
/ncs-config/logs/developer-log-level (error | info | trace) [info]	デベロッパーログに出力するデベロッパーメッセージのレベルを制御します。

パラメータ	説明
/ncs-config/logs/audit-log	*audit-log* は、WAE バックプレーンへのログインの成功と失敗を記録する監査ログです。デフォルトでこのログは有効です。他のすべての点では、/ncs-config/logs/ncs-log として構成できます。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/audit-log/ enabled (boolean) [true]	「true」の場合、ログは有効です。
/ncs-config/logs/audit-log/file	—
/ncs-config/logs/audit-log/ file/name (string)	*name* は、実際のログファイルへのフルパスです。
/ncs-config/logs/audit-log/ file/enabled (boolean) [false]	「true」の場合、ファイルのロギングが有効になります。
/ncs-config/logs/audit-log/ syslog	—
/ncs-config/logs/audit-log/ syslog/enabled (boolean) [false]	「true」の場合、syslog メッセージが送信されます。
/ncs-config/logs/audit-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)	このオプションの値は、指定されたログの /ncs-config/logs/syslog-config/facility をオーバーライドします。
/ncs-config/logs/audit-log-commit (boolean) [false]	監査ログに、実行中のデータストアへのコミットごとに結果として生じる構成変更に関するメッセージを含めるかどうかを制御します。
/ncs-config/logs/netconf-log	*netconf-log* は、フィルタ操作でリクエストされたデータを返さなかった理由を確認するなど、ノースバウンド NETCONF 操作をトラブルシューティングするためのログです。デフォルトでこのログは有効です。他のすべての点では、/ncs-config/logs/ncs-log として構成できます。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/netconf-log/ enabled (boolean) [true]	「true」の場合、ログは有効です。
/ncs-config/logs/netconf-log/ file	—
/ncs-config/logs/netconf-log/ file/name (string)	*name* は、実際のログファイルへのフルパスです。
/ncs-config/logs/netconf-log/ file/enabled (boolean) [false]	「true」の場合、ファイルのロギングが有効になります。
/ncs-config/logs/netconf-log/syslog	—

パラメータ	説明
/ncs-config/logs/netconf-log/ syslog/enabled (boolean) [false]	「true」の場合、syslog メッセージが送信されます。
/ncs-config/logs/netconf-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)	このオプションの値は、指定されたログの /ncs-config/logs/syslog-config/facility をオーバーライドします。
/ncs-config/logs/snmp-log	—
/ncs-config/logs/snmp-log/ enabled（boolean）[true]	「true」の場合、ログは有効です。
/ncs-config/logs/snmp-log/file	—
/ncs-config/logs/snmp-log/ file/name (string)	*name* は、実際のログファイルへのフルパスです。
/ncs-config/logs/snmp-log/ file/enabled (boolean) [false]	「true」の場合、ファイルのロギングが有効になります。
/ncs-config/logs/snmp-log/ syslog	—
/ncs-config/logs/snmp-log/ syslog/enabled (boolean) [false]	「true」の場合、syslog メッセージが送信されます。
/ncs-config/logs/snmp-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)	このオプションの値は、指定されたログの /ncs-config/logs/syslog-config/facility をオーバーライドします。
/ncs-config/logs/snmp-log-level (error | info) [info]	SNMP ログに出力される SNMP PDU のレベルを制御します。値「error」は、エラーステータスが「noError」と等しくない PDU のみが出力されることを意味します。
/ncs-config/logs/webui-browser-log	*webui-browser-log* を使用すると、ブラウザのエラーコンソールだけでなく、ターゲットデバイスのログファイルに Java スクリプトのエラー/例外を記録できます。このログはデフォルトで有効ではなく、ローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/webui-browser-log/ enabled (boolean) [false]	「true」の場合、ブラウザログが使用されます。
/ncs-config/logs/webui-browser-log/filename (string)	このパラメータは必須です。ブラウザのログエントリが書き込まれるファイル名へのパス。

パラメータ	説明
/ncs-config/logs/webui-access-log	*webui-access-log* は、組み込み WAE Web サーバーのアクセスログです。このファイルは、Apache などで定義されている Common Log Format に準拠しています。このログはデフォルトで有効ではなく、ローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/webui-access-log/enabled (boolean) [false]	「true」の場合、アクセスログが使用されます。
/ncs-config/logs/webui-access-log/traffic-log (boolean) [false]	「true」の場合、組み込み Web サーバーへのすべての HTTP(S) トラフィックは、traffic.trace という名前のログファイルに記録されます。このログはデフォルトで有効ではなく、ローテーションされません。logrotate(8) を使用してください。 **注意** このログを実稼働環境で使用しないでください。
/ncs-config/logs/webui-access-log/dir (string)	このパラメータは必須です。アクセスログが書き込まれるディレクトリへのパス。
/ncs-config/logs/netconf-trace-log	*netconf-trace-log* は、ノースバウンド NETCONF プロトコルの相互作用を理解し、トラブルシューティングするためのログです。このログを有効にすると、WAE との間で送受信されるすべての NETCONF トラフィックがファイルに保存されます。デフォルトでは、すべての XML が整形されて出力されます。これにより NETCONF サーバーの速度が低下するため、このログを有効にするときは注意してください。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/netconf-trace-log/enabled (boolean) [false]	「true」の場合、すべての NETCONF トラフィックがログに記録されます。
/ncs-config/logs/netconf-trace-log/filename (string)	このパラメータは必須です。NETCONF トラフィックのトレースログが書き込まれるファイルの名前。
/ncs-config/logs/netconf-trace-log/format (pretty | raw) [pretty]	値「pretty」は、XML データが整形されて出力されることを意味します。値「raw」は、整形されないで出力されることを意味します。
/ncs-config/logs/xpath-trace-log	*xpath-trace-log* は、xpath 評価を理解し、トラブルシューティングするためのログです。このログを有効にすると、WAE によって評価されたすべての xpath クエリがファイルに記録されます。これにより WAE の速度が低下するため、このログを有効にするときは注意してください。このログはローテーションされません。logrotate(8) を使用してください。
/ncs-config/logs/xpath-trace-log/enabled (boolean) [false]	「true」の場合、すべての xpath 実行がログに記録されます。
/ncs-config/logs/xpath-trace-log/filename (string)	このパラメータは必須です。xpath のトレースログが書き込まれるファイルの名前。

その他の WAE CLI コマンド

パラメータ	説明
/ncs-config/logs/error-log	*error-log* は、WAE デーモンからの内部ロギングに使用されるエラーログです。WAEデーモン自体のトラブルシューティングに使用され、通常は無効にする必要があります。このログは、WAEデーモンによってローテーションされます。
/ncs-config/logs/error-log/ enabled (boolean) [false]	「true」の場合、エラーログが実行されます。
/ncs-config/logs/error-log/ filename (string)	このパラメータは必須です。*filename* は、実際のログファイルへのフルパスです。エラーログが有効な場合は、このパラメータを設定する必要があります。
/ncs-config/logs/error-log/max-size (tailf:size) [S1M]	*max-size* は、ローテーションされる前の個々のログファイルの最大サイズです。5 つのログが使い果たされると、ログファイル名が再利用されます。
/ncs-config/logs/error-log/ debug	—
/ncs-config/logs/error-log/ debug/enabled (boolean) [false]	—
/ncs-config/logs/error-log/ debug/level (uint16) [2]	—
/ncs-config/logs/error-log/ debug/tag (string)	このパラメータは複数回指定できます。
/ncs-config/candidate	—
/ncs-config/candidate/ filename (string)	candidate db-mode は削除されました。このリーフは WAE 構成に影響を与えなくなりました。このリーフと candidate コンテナは、後方互換性のために保持されています。
/ncs-config/sort-transactions (boolean) [true]	このパラメータは、新しく作成され、まだコミットされていないリストエントリを WAE がリストする方法を制御します。この値が「false」に設定されている場合、WAE は既存のデータをリストする前にすべての新しい要素をリストします。この値が「true」に設定されている場合、WAE は新しいエントリと既存のエントリをマージし、データがソートされた 1 つのビューを提供します。この動作は、構成データの保存に CDB が使用されている場合には適切に機能しますが、外部データプロバイダーが使用されている場合、WAE はソート順を認識せず、新しいエントリを正しくマージできません。構成データに外部データプロバイダーが使用されていて、ソート順が CDB のソート順と異なる場合、このパラメータを「false」に設定する必要があります。

パラメータ	説明
/ncs-config/enable-attributes (boolean) [true]	このパラメータは、WAE の属性機能を有効にするかどうかを制御します。注釈とタグの 2 つの属性があります。これらはノースバウンドインターフェイス（CLI の annotate コマンド、および NETCONF の annotation XML 属性）で使用できますが、使用するには基礎となる構成データプロバイダーからのサポートが必要です。CDB は属性をサポートしていますが、外部データプロバイダーが構成データに使用されていて、属性のコールバックをサポートしていない場合は、このパラメータを「false」に設定する必要があります。
/ncs-config/enable-inactive (boolean) [true]	このパラメータは、WAE の非アクティブ機能を有効にするかどうかを制御します。この機能では、enableAttributes も有効にする必要があります。WAE を使用して Juniper ルータを制御する場合、この機能が必要です。
/ncs-config/session-limits	WAE への同時アクセスを制限します。
/ncs-config/session-limits/max-sessions (uint32 | unbounded) [unbounded]	WAE への合計同時セッション数を制限します。
/ncs-config/session-limits/session-limit	特定のコンテキストでの WAE への同時アクセスを制限します。このコンテナ要素には複数のインスタンスがあり、それぞれが特定のコンテキストのパラメータを指定します。
/ncs-config/session-limits/ session-limit/context (string)	context は、cli、netconf、webui、snmp、または MAAPI を使用して定義されたその他のコンテキスト文字列です。たとえば、MAAPI を使用して WAE への CORBA インターフェイスを実装する場合、MAAPI プログラムは文字列「corba」をコンテキストとして送信できます。
/ncs-config/session-limits/ session-limit/max-sessions (uint32 | unbounded)	このパラメータは必須です。WAE への合計同時セッション数を制限します。
/ncs-config/session-limits/ max-config-sessions (uint32 | unbounded) [unbounded]	WAE への合計同時構成セッション数を制限します。
/ncs-config/session-limits/ config-session-limit	特定のコンテキストでの WAE への同時読み書きトランザクションを制限します。このコンテナ要素には複数のインスタンスがあり、それぞれが特定のコンテキストのパラメータを指定します。
/ncs-config/session-limits/ config-session-limit/context (string)	context は、cli、netconf、webui、snmp、または MAAPI を使用して定義されたその他のコンテキスト文字列です。たとえば、MAAPI を使用して WAE への CORBA インターフェイスを実装する場合、MAAPI プログラムは文字列「corba」をコンテキストとして送信できます。

パラメータ	説明
/ncs-config/session-limits/ config-session-limit/max-sessions (uint32 |unbounded)	このパラメータは必須です。対応するコンテキストの WAE への合計同時構成セッション数を制限します。
/ncs-config/aaa	—
/ncs-config/aaa/ssh-login-grace-time (xs:duration) [PT10M]	クライアントが自身を正常に認証しなかった場合、WAE サーバーはこの時間の後に SSH 接続を閉じます。値が 0 の場合、クライアント認証の時間制限はありません。これは、WAE のすべての SSH サーバーに対するグローバル値です。この値を変更すると、変更後に確立された SSH 接続にのみ影響します。
/ncs-config/aaa/ssh-max-auth-tries (uint32 | unbounded) [unbounded]	WAE サーバーは、クライアントがこの回数の認証試行に失敗すると、SSH 接続を閉じます。これは、WAE のすべての SSH サーバーに対するグローバル値です。この値を変更すると、変更後に確立された SSH 接続にのみ影響します。
/ncs-config/aaa/ssh-server-key-dir (string)	*ssh-server-key-dir* は、WAE SSH デーモンによって使用されるキーがあるディレクトリファイルパスです。NETCONF または CLI に対して SSH が有効になっている場合は、このパラメータを設定する必要があります。SSH が有効になっている場合、WAE によって使用されるサーバーキーは、openssh によって使用されるサーバーキーと同じフォーマット（つまり、「ssh-keygen」によって生成されるものと同じフォーマット）になります。 DSA タイプおよび RSA タイプのキーのみが WAE SSH デーモンで使用できます。それぞれのキーは、「-t dsa」および「-t rsa」スイッチを使用して「ssh-keygen」で生成されます。キーは、空のパスフレーズを使用し、DSA タイプのキーの場合は「ssh_host_dsa_key」という名前で、RSA タイプのキーの場合は「ssh_host_rsa_key」という名前で保存する必要があります。SSH サーバーは、使用可能なキーファイルがあるキータイプや、必要なアルゴリズムが有効になっているキータイプのサポートをアドバタイズします。/ncs-config/ssh/algorithms/server-host-key リーフを参照してください。

パラメータ	説明
/ncs-config/aaa/ssh-pubkey-authentication (none | local | system) [system]	WAE SSH デーモンが公開キー認証用のユーザーキーを見つける方法を制御します。 「none」に設定すると、公開キー認証が無効になります。 「local」に設定されていて、ユーザーが /aaa/authentication/users に存在する場合、ユーザーの「ssh_keydir」ディレクトリ内のキーが使用されます。 「system」に設定すると、ユーザーは最初に /aaa/authentication/users で検索されますが、これは /ncs-config/aaa/local-authentication/enabled が「true」に設定されている場合のみです。ローカル認証が無効になっている場合、またはユーザーが /aaa/authentication/users に存在しないが OS パスワードデータベースに存在する場合、ユーザーの $HOME/.ssh ディレクトリにあるキーが使用されます。
/ncs-config/aaa/default-group (string)	ユーザーグループが AAA サブシステムで見つからない場合、ログインユーザーはデフォルトグループのメンバーになります（指定されている場合）。ユーザーがログインしていて、グループメンバーシップを確立できない場合、そのユーザーのアクセス権はゼロです。
/ncs-config/aaa/auth-order (string)	認証のデフォルトの順序は「local-authentication pam external-authentication」です。このパラメータを使用して、この順序を変更することができます。
/ncs-config/aaa/expiration-warning (ignore | display | prompt) [ignore]	PAM または外部認証が使用されている場合、認証メカニズムにより、ユーザーのパスワードの有効期限が近づいているという警告が表示される場合があります。このパラメータは、WAE デーモンがその警告メッセージを処理する方法を制御します。 「ignore」に設定すると、警告は無視されます。 「display」に設定すると、ログイン時に対話型ユーザーインターフェイスに警告メッセージが表示されます。 「prompt」に設定すると、ログイン時に対話型ユーザーインターフェイスに警告メッセージが表示されます。ユーザーは、続行する前にメッセージを確認する必要があります。

パラメータ	説明
/ncs-config/aaa/audit-user-name (always | known | never) [known]	失敗した認証の試行が監査ログに記録されるときのユーザー名のロギングを制御します。 「always」に設定すると、ユーザー名は常にログに記録されます。 「known」に設定すると、ユーザー名は有効であることがわかっている場合（つまり、ローカル認証を試行し、ユーザーが /aaa/authentication/users に存在する場合）にのみログに記録されます。それ以外の場合は、「[withheld]」としてログに記録されます。 「never」に設定すると、ユーザー名は常に「[withheld]」としてログに記録されます。
/ncs-config/aaa/pam	ログインに PAM を使用する場合、WAE デーモンは通常、root として実行する必要があります。
/ncs-config/aaa/pam/enabled (boolean) [false]	「true」に設定すると、WAE は認証に PAM を使用します。
/ncs-config/aaa/pam/service (string) [common-auth]	ログイン NETCONF/SSH CLI 手順に使用する PAM サービス。これは、/etc/pam.d ディレクトリにインストールされている任意のサービスです。Unix が異なると、/etc/pam.d にインストールされるサービスは異なります。既存のサービスを選択するか、新しいサービスを作成します。
/ncs-config/aaa/pam/timeout (xs:duration) [PT10S]	認証が PAM からの応答を待機する最大時間。タイムアウトに達すると、PAM 認証は失敗しますが、認証は /ncs-config/aaa/authOrder に構成されている他のメカニズムで試行されます。デフォルトは PT10S（10 秒）です。
/ncs-config/aaa/external-authentication	—
/ncs-config/aaa/external-authentication/enabled (boolean) [false]	「true」に設定すると、外部認証が使用されます。

パラメータ	説明
/ncs-config/aaa/external-authentication/executable (string)	外部認証が有効になっている場合、ローカルホスト上の実行可能ファイルを起動してユーザーを認証できます。実行可能ファイルは、標準入力でユーザー名とクリアテキストパスワードを受け取ります。形式は「[${USER};${PASS};]\n」です。たとえば、ユーザーが「bob」でパスワードが「secret」の場合、実行可能ファイルは、標準入力で行「[bob;secret;]」とそれに続く新規改行を受け取ります。プログラムでこの行を解析する必要があります。 外部プログラムのタスクは、ユーザーを認証し、ユーザーとグループのマッピングを提供することです。「bob」が「oper」および「lamers」グループのメンバーである場合、プログラムは標準出力に「accept oper lamers」をエコーします。ユーザーが認証に失敗した場合、プログラムは標準出力に「reject ${reason}」をエコーします。
/ncs-config/aaa/external-authentication/use-base64 (boolean) [false]	「true」に設定すると、実行可能ファイルに渡されるデータの${USER}と ${PASS} は base64 でエンコードされ、パスワードに「;」文字を含めることができます。たとえば、ユーザーが「bob」でパスワードが「secret」の場合、実行可能ファイルは、文字列「[Ym9i;c2VjcmV0;]」とそれに続く新規改行を受け取ります。
/ncs-config/aaa/external-authentication/include-extra (boolean) [false]	「true」に設定すると、送信元の IP アドレスとポート、コンテキスト、およびプロトコルの追加情報項目が実行可能ファイルに提供されます。完全なフォーマットは「[${USER};${PASS};${IP};${PORT};${CONTEXT};${PROTO};]\n」です。 例：「[bob;secret;192.168.1.1;12345;cli;ssh;]\n」。
/ncs-config/aaa/local-authentication	—
/ncs-config/aaa/local-authentication/enabled (boolean) [true]	「true」に設定すると、WAE はローカル認証を使用します。aaa 名前空間に保持されているユーザーデータがユーザーの認証に使用されます。「false」に設定すると、別の認証メカニズム（PAM や外部認証など）が使用されます。
/ncs-config/aaa/authentication-callback	—
/ncs-config/aaa/authentication-callback/enabled (boolean) [false]	「true」に設定すると、認証が成功または失敗したときに、WAE はアプリケーション コールバックを呼び出します。コールバックは、成功したはずの認証を拒否する場合があります。コールバックが登録されていない場合、認証の試行はすべて失敗します。
/ncs-config/aaa/authorization	—
/ncs-config/aaa/authorization/enabled (boolean) [true]	「false」に設定すると、ncs_cli の -noaaa フラグと同様に、すべての承認チェックがオフになります。

パラメータ	説明
/ncs-config/aaa/authorization/ callback	—
/ncs-config/aaa/authorization/callback/enabled (boolean) [false]	「true」に設定すると、WAEは承認のためにアプリケーションコールバックを呼び出します。コールバックが登録されていない場合、すべての承認チェックが却下されます。
/ncs-config/aaa/namespace (string) [http://tail-f.com/ns/aaa/1.1]	AAA データを別のユーザー定義の名前空間に移動するには、その名前空間をここで指定します。
/ncs-config/aaa/prefix (string) [/]	AAA データを別のユーザー定義の名前空間に移動するには、WAE AAA 名前空間がマウントされているその名前空間のプレフィックスパスを指定します。
/ncs-config/rollback	ロールバックファイルを作成するかどうか、および作成する場所を制御する設定。ロールバックファイルには、システム構成のコピーが含まれています。現在の実行構成は常に rollback0 に、その直前のバージョンはrollback1 に、というように保存されます。保存された構成のうち最も古いもののサフィックスが最も大きくなります。
/ncs-config/rollback/ enabled (boolean) [false]	「true」に設定すると、実行構成が変更されるたびにロールバックファイルが作成されます。
/ncs-config/rollback/ directory (string)	このパラメータは必須です。ロールバックファイルが作成される場所。
/ncs-config/rollback/ history-size (uint32) [35]	保存する古い構成の数。
/ncs-config/rollback/ type (delta) [delta]	このパラメータは廃止されます。WAE は、タイプ「delta」のみをサポートします。このパラメータの値を設定する必要はありません。これは、後方互換性のためにのみ保持されています。タイプ「delta」は、変更のみがロールバックファイルに保存されることを意味します。ロールバックファイル 0 には、最後の構成コミットからの変更が含まれています。これは、大規模な構成ではスペースと時間の効率が高くなります。
/ncs-config/rollback/ rollback-numbering (rolling | fixed) [fixed]	*rollback-numbering* は、「fixed」または「rolling」のいずれかです。「rolling」に設定すると、ロールバックファイル「0」には常に最後のコミットが含まれます。「fixed」に設定すると、ロールバックごとに番号が増加します。
/ncs-config/ssh	WAE に組み込まれた SSH サーバーの動作を制御します。

パラメータ	説明
/ncs-config/ssh/idle-connection-timeout (xs:duration) [PT10M]	SSH サーバーへの認証済み接続が、オープンチャネルなしで存在できる最大時間。タイムアウトに達すると、SSH サーバーは接続を閉じます。デフォルトは PT10M（10 分）です。値 0 は、タイムアウトしないことを示します。
/ncs-config/ssh/algorithms	組み込みの SSH 実装で使用できるアルゴリズムのカスタムリストを定義します。アルゴリズムのタイプごとに、空の値は、サポートされているすべてのアルゴリズムが使用可能であることを意味します。空でない値（アルゴリズム名のカンマ区切りリスト）は、サポートされているアルゴリズムと構成されたアルゴリズムとで共通しているものが使用可能であることを意味します。
/ncs-config/ssh/algorithms/server-host-key (string) []	サポートされている serverHostKey アルゴリズム（libcrypto に実装されている場合）は「ssh-dss」および「ssh-rsa」ですが、SSH サーバーでは /ncs-config/aaa/ssh-server-key-dir で指定されたディレクトリにホストキーがインストールされているアルゴリズムに限定されます。使用可能な serverHostKey アルゴリズムを「ssh-dss」に制限するには、この値を「ssh-dss」に設定するか、sshServerKeyDir に ssh-dss 以外のタイプのキーをインストールしないようにします。
/ncs-config/ssh/algorithms/kex (string) []	サポートされているキー交換アルゴリズム（ハッシュ関数が libcrypto に実装されている場合）は、「diffie-hellman-group-exchange-sha256」、「diffie-hellman-group-exchange-sha1」、「diffie-hellmangroup14-sha1」、および「diffie-hellman-group1-sha1」です。使用可能なキー交換アルゴリズムを「diffie-hellman-group14-sha1」および「diffie-hellmangroup-exchange-sha256」に（この順序で）制限するには、この値を「diffie-hellman-group14-sha1, diffie-hellmangroup-exchange-sha256」に設定します。
/ncs-config/ssh/algorithms/dh-group	「diffie-hellman-groupexchange」中に SSH サーバーがクライアントに応答する許容グループサイズの範囲。範囲は、クライアントがリクエストするものとの共通部分です。存在しない場合、キー交換は中止されます。
/ncs-config/ssh/algorithms/ dh-group/min-size (dh-group-size-type) [2048]	p の最小サイズ（ビット単位）。
/ncs-config/ssh/algorithms/dh-group/max-size (dh-group-size-type) [4096]	p の最大サイズ（ビット単位）。
/ncs-config/ssh/algorithms/mac (string) []	サポートされている mac アルゴリズム（libcrypto で実装されている場合）は、「hmac-md5」、「hmac-sha1」、「hmacsha2-256」、「hmac-sha2-512」、「hmac-sha1-96」、および「hmac-md5-96」です。

パラメータ	説明
/ncs-config/ssh/algorithms/encryption (string) []	サポートされている暗号化アルゴリズム（libcrypto で実装されている場合）は、「aes128-ctr」、「aes192-ctr」、「aes256-ctr」、「aes128-cbc」、「aes256-cbc」、および「3des-cbc」です。
/ncs-config/ssh/client-alive-interval (xs:duration | infinity) [infinity]	接続されたクライアントからこの時間データを受信しなかった場合、クライアントからの応答を必要とするリクエストが SSH トランスポートを介して送信されます。
/ncs-config/ssh/client-alive-count-max (uint32) [3]	この回数の連続したクライアントアライブ間隔が経過してもクライアントからデータを受信しなかった場合、接続は切断されます。
/ncs-config/cli	CLI パラメータ。
/ncs-config/cli/enabled (boolean) [true]	「true」の場合、CLI サーバーが開始されます。
/ncs-config/cli/allow-implicit-wildcard (boolean) [true]	「true」の場合、ユーザーはリストのすべてのインスタンスを表示するために、リストのキーの代わりに明示的に * を入力する必要はありません。「false」の場合、すべてのリストインスタンスを表示するには、ユーザーは明示的に * を入力する必要があります。
/ncs-config/cli/completion-show-max (cli-max) [100]	補完の実行時に提示する可能な選択肢の最大数。
/ncs-config/cli/style (j | c)	スタイルは「j」または「c」です。「j」に設定すると、CLI は Juniper スタイルの CLI として表示されます。「c」の場合、CLI は Cisco XR スタイルとして表示されます。
/ncs-config/cli/ssh	—
/ncs-config/cli/ssh/enabled (boolean) [true]	*enabled* は「true」または「false」のいずれかです。「true」の場合、WAE CLI は組み込みの SSH サーバーを使用します。
/ncs-config/cli/ssh/ip (ipv4-address | ipv6-address) [0.0.0.0]	*ip* は、WAE CLI が SSH 接続をリッスンする IP アドレスです。0.0.0.0 は、マシン上のすべての IPv4 アドレスのポート（/ncs-config/cli/ssh/port）でリッスンすることを意味します。
/ncs-config/cli/ssh/port (port-number) [2024]	CLI SSH のポート番号。
/ncs-config/cli/ssh/banner (string) []	*banner* は、組み込みの SSH サーバー経由で CLI にログインするときに、認証前にクライアントに提示される文字列です。
/ncs-config/cli/ssh/banner-file (string) []	*banner-file* は、組み込みの SSH サーバー経由で CLI にログインするときに、認証前に（banner ディレクティブで指定された文字列の後で）クライアントに提示されるコンテンツのファイル名です。

パラメータ	説明
/ncs-config/cli/ssh/extra-listen	WAE CLI が SSH 接続をリッスンする追加の IP アドレスとポートのペアのリスト。
/ncs-config/cli/ssh/extra-listen/ip (ipv4-address | ipv6-address)	—
/ncs-config/cli/ssh/extra-listen/port (port-number)	—
/ncs-config/cli/top-level-cmds-in-sub-mode (boolean) [false]	topLevelCmdsInSubMode は「true」または「false」です。「true」の場合、I および C スタイルの CLI のすべてのトップレベルコマンドがサブモードで使用できます。
/ncs-config/cli/completion-meta-info (false | alt1 | alt2) [false]	completionMetaInfo は「false」、「alt1」、または「alt2」です。「alt1」に設定した場合、補完候補として表示される選択肢には、次のようなプレフィックスが付きます。 コンテナ > リスト + リーフリスト + 次に例を示します。 補完候補：... > applications + apply-groups ... + dns-servers ... 「alt2」に設定した場合、補完候補には次のようなプレフィックスが付きます。 コンテナ > 子を持つリスト +> 子のないリスト + 次に例を示します。 補完候補：... > applications +>apply-groups ... + dns-servers ...
/ncs-config/cli/allow-abbrev-keys (boolean) [false]	allowAbbrevKeys は「true」または「false」です。「false」の場合、キー要素は CLI で省略できません。これは、コマンド「delete」および「edit」を使用するときの J スタイルの CLI に関連しています。これは、C/I スタイルの CLI で「no」コマンドや「show configuration」コマンドを使用する場合、およびサブモードに入るコマンドに関連します。
/ncs-config/cli/j-align-leaf-values (boolean) [true]	j-align-leaf-values は「true」または「false」です。「true」の場合、コンテナまたはリスト内のすべての兄弟のリーフ値が整列されます。

パラメータ	説明
/ncs-config/cli/enter-submode-on-leaf (boolean) [true]	enterSubmodeOnLeaf は「true」または「false」です。「true」（デフォルト）の場合、親モードからサブモードのリーフを設定すると、コマンドの完了後にサブモードに入ります。「false」の場合、サブモードに入るための明示的なコマンドが必要です。たとえば、構成モードのトップレベルからコマンド **interface FastEthernet 1/1/1 mtu 1400** を実行している場合です。enterSubmodeOnLeaf が「true」の場合、コマンドの実行後、CLI は「interface FastEthernet 1/1/1」サブモードになります。「false」の場合、CLI はトップレベルのままです。「false」に設定されているときにサブモードに入るには、コマンド **interface FastEthernet 1/1/1** が必要です。C スタイルの CLI に適用されます。
/ncs-config/cli/table-look-ahead (int64) [50]	tableLookAhead 要素は、テーブルを表示するときにプリフェッチする行数を confd に指示します。プリフェッチされた行は、テーブルに必要な列幅を計算するために使用されます。小さい数値に設定する場合は、clispec ファイルで列幅を明示的に構成する必要があります。
/ncs-config/cli/more-buffer-lines (uint32 | unbounded) [unbounded]	moreBufferLines は、more プロセスによって実行されるバッファリングを制限するために使用されます。「unbounded」またはバッファする最大行数を表す正の整数にすることができます。
/ncs-config/cli/show-all-ns (boolean) [false]	showAllNs が「true」の場合、CLI ですべての要素名の前に名前空間プレフィックスが付けられます。これは、値を設定するとき、および構成を表示するときに表示されます。
/ncs-config/cli/suppress-fast-show (boolean) [false]	suppressFastShow は「true」または「false」です。「true」の場合、C スタイルの CLI で高速表示の最適化が抑制されます。高速表示の最適化はやや実験的な機能であり、特定の操作を中断する可能性があります。
/ncs-config/cli/use-expose-ns-prefix (boolean) [true]	「true」の場合、tailf:cli-expose-ns-prefix で注釈が付けられたすべてのノードは、名前空間プレフィックスが表示/必須になります。「false」の場合、tailf:cli-expose-ns-prefix 注釈は無視されます。コンテナ /devices/device/config には、この注釈があります。
/ncs-config/cli/show-defaults (boolean) [false]	*show-defaults* は「true」または「false」です。「true」の場合、構成を表示するときにデフォルト値が表示されます。デフォルト値は、値と同じ行のコメント内に表示されます。デフォルト値の表示は、動作モードコマンド **set show defaults true** を使用して、セッションごとに CLI で有効にすることもできます。
/ncs-config/cli/default-prefix (string) []	*default-prefix* は、構成がコメントとしてデフォルト値とともに表示されるときに、デフォルト値の前に配置される文字列です。

パラメータ	説明
/ncs-config/cli/commit-retry-timeout (xs:duration | infinity) [PT0S]	CLI のコミットタイムアウト。このタイムアウトは、他のエンティティがデータベースをロックしているときに、コミット操作が操作の完了を試みる時間を制御します。同様の構成パラメータ /ncs-config/commit-retry-timeout は、JSON-RPC API の WAE トランザクションのタイムアウトを設定します。
/ncs-config/cli/timezone (utc | local) [local]	CLI の時刻は、ローカル（ホストでの構成に従う）または UTC にすることができます。
/ncs-config/cli/with-defaults (boolean) [false]	*with-defaults* は「true」または「false」です。「false」の場合、ユーザーが「show」コマンドに「details」オプションを指定しない限り、構成を表示するときにデフォルト値を持つリーフノードは表示されません。使用頻度の少ない設定が多い場合に便利です。「false」の場合、ユーザーが実際に変更した値のみが表示されます。
/ncs-config/cli/banner (string) []	CLI の開始時にユーザーに表示されるバナー。デフォルトは空欄です。
/ncs-config/cli/banner-file (string) []	CLI の開始時に（「banner」ディレクティブで設定された文字列の後で）ユーザーに表示されるコンテンツのファイル。デフォルトは空欄です。
/ncs-config/cli/prompt1 (string) [\u@\h\M>]	動作モードで使用されるプロンプト。文字列には、バックスラッシュでエスケープされた特殊文字がいくつか含まれている場合があり、次のように復号化されます。 • \d：「YYYY-MM-DD」フォーマットの日付（たとえば、「2006-01-18」）。 • \h：最初の「.」（または、promptHostnameDelimiter で定義されたデリミタ）までのホスト名。 • \H：24 時間制の HH:MM:SS フォーマットによる現在時刻。 • \T：12 時間制の HH:MM:SS フォーマットによる現在時刻。 • \@：12 時間制の AM/PM フォーマットによる現在時刻。 • \A：24 時間制の HH:MM フォーマットによる現在時刻。 • \u：現在のユーザーのユーザー名。 • \m：モード名（XR スタイルでのみ使用）。 • \M：モードに入っている場合、括弧内にモード名。
/ncs-config/cli/prompt2 (string) [\u@\h\M%]	構成モードで使用されるプロンプト。文字列には、バックスラッシュでエスケープされた特殊文字がいくつか含まれている場合があり、prompt1 の説明に従って復号化されます。

パラメータ	説明
/ncs-config/cli/c-prompt1 (string) [\u@\h\M>]	Cisco XR スタイルの CLI の動作モードで使用されるプロンプト。文字列には、バックスラッシュでエスケープされた特殊文字がいくつか含まれている場合があり、prompt1 の説明に従って復号化されます。
/ncs-config/cli/c-prompt2 (string) [\u@\h\M%]	Cisco XR スタイルの CLI の構成モードで使用されるプロンプト。文字列には、バックスラッシュでエスケープされた特殊文字がいくつか含まれている場合があり、prompt1 の説明に従って復号化されます。
/ncs-config/cli/prompt-hostname-delimiter (string) [.]	プロンプトで\h トークンを使用すると、promptHostnameDelimiter が最初に出現するまでのホスト名の先頭部分が使用されます。
/ncs-config/cli/show-log-directory (string) [/var/log]	**show log** コマンドがログファイルを検索する場所。
/ncs-config/cli/idle-timeout (xs:duration) [PT30M]	CLI セッションを終了するまでの最大アイドル時間。デフォルトは PT30M（30 分）です。
/ncs-config/cli/prompt-sessions-cli (boolean) [false]	promptSessionsCLI は「true」または「false」です。「true」の場合、ユーザーが新しい CLI セッションを開始しようとしているときにセッションの最大数に達すると、現在の CLI セッションのみが表示されます。コンテキストが「cli」に設定されている MAAPI セッションは、CLI セッションと見なされ、そのようにリストされることに注意してください。
/ncs-config/cli/suppress-ned-errors (boolean) [false]	NED デバイスからのエラーを抑制します。WAE とそのデバイス間のログ通信をよりサイレントにします。興味深いエラーも抑制される可能性があるため、このオプションには注意してください。
/ncs-config/cli/disable-idle-timeout-on-cmd (boolean) [true]	*disable-idle-timeout-on-cmd* は「true」または「false」です。「false」の場合、CLI でコマンドが実行されている場合でも、アイドルタイムアウトがトリガーされます。「true」の場合、アイドルタイムアウトは、ユーザーが CLI プロンプトでアイドリングしている場合にのみトリガーされます。
/ncs-config/cli/command-timeout (xs:duration | infinity) [infinity]	グローバル コマンド タイムアウト：コマンドがタイムアウト内に完了しない限り、コマンドを終了します。この機能の使用はお勧めしません。通常のコマンドが完了するまでに時間がかかるような負荷の高いシステムで望ましくない影響を与える可能性があるためです。このタイムアウトは、ncs.cli ファイルで指定されたコマンド固有のタイムアウトによってオーバーライドできます。
/ncs-config/cli/space-completion	—
/ncs-config/cli/space-completion/enabled (boolean)	—

パラメータ	説明
/ncs-config/cli/ignore-leading-whitespace (boolean)	「false」の場合、行の最初の文字として TAB または SPACE を入力すると、CLI は補完ヘルプを表示します。「true」の場合、先頭の SPACE と TAB は無視されます。代替選択肢のリストについては、「?」を入力します。値を「true」に設定すると、スクリプトを CLI に簡単に貼り付けることができます。
/ncs-config/cli/auto-wizard	CLI の autowizard のデフォルト値。ユーザーは、各セッションでいつでも autowizard を有効または無効にすることができます。これは、初期セッション値を制御します。
/ncs-config/cli/auto-wizard/enabled (boolean) [true]	**enabled** は「true」または「false」です。「true」の場合、CLI は、新しい識別子が作成されるときに、ユーザーに必須属性の入力を求めます。
/ncs-config/cli/restricted-file-access (boolean) [false]	「*restricted-file-access*」は「true」または「false」です。「true」の場合、CLI ユーザーはホームディレクトリツリーの外部にあるファイルとディレクトリにアクセスできません。
/ncs-config/cli/restricted-file-regexp (string) []	*limited-file-regexp* は、空の文字列または正規表現（AWK スタイル）のいずれかです。空でない場合、作成またはアクセスされるすべてのファイルとディレクトリは正規表現と一致する必要があります。これは、作成されたファイルで特定のシンボルが発生しないようにするために使用できます。
/ncs-config/cli/history-save (boolean) [true]	「true」の場合、CLI 履歴は CLI セッション間で保存されます。履歴は state ディレクトリに保存されます。
/ncs-config/cli/history-remove-duplicates (boolean) [false]	「true」の場合、CLI で繰り返されるコマンドは、履歴に 1 回だけ保存されます。コマンドを呼び出すたびに、最後のエントリの日付のみが更新されます。「false」の場合、重複が履歴に保存されます。
/ncs-config/cli/history-max-size (int64) [1000]	構成可能な履歴の最大サイズを設定します。
/ncs-config/cli/message-max-size (int64) [10000]	ユーザーメッセージの最大サイズを設定します。
/ncs-config/cli/show-commit-progress (boolean) [true]	*show-commit-progress* は「true」または「false」です。「true」の場合、CLI でのコミット操作は進行状況情報を提供します。
/ncs-config/cli/commit-message (boolean) [true]	コミットが実行されると、CLI はメッセージを出力します。
/ncs-config/cli/use-double-dot-ranges (boolean) [true]	*use-double-dot-ranges* は「true」または「false」です。「true」の場合、範囲式は 1..3 のように指定します。「false」の場合、範囲は 1-3 のように指定します。

パラメータ	説明
/ncs-config/cli/allow-range-expression-all-types (boolean) [true]	*allow-range-expression-all-types* は「true」または「false」です。「true」の場合、タイプに関係なく、すべてのキー値に対して範囲式が許可されます。
/ncs-config/cli/suppress-range-keyword (boolean) [false]	*suppress-range-keyword* は「true」または「false」です。「true」の場合、「range」キーワードはCおよびIスタイルでは範囲式で許可されません。
/ncs-config/cli/commit-message-format (string) [System message at $(time)... Commit performed by $(user) via $(proto) using $(ctx).]	CLI コミットメッセージのフォーマット。
/ncs-config/cli/suppress-commit-message-context (string)	このパラメータは複数回指定できます。コミットメッセージが表示されないコンテキストのリスト。適切な値は [system] です。これにより、システムによって生成されたすべてのコミットがCLIで認識されなくなります。コンテキストは、エージェントの名前（CLI、Web UI、NETCONF、SNMP）、またはトランザクションがMAAPIから開始された場合は自由形式のテキスト文字列です。
/ncs-config/cli/show-subsystem-messages (boolean) [true]	*show-subsystem-messages* は「true」または「false」です。「true」の場合、CLI は、接続されたデーモンが開始または停止するたびにシステムメッセージを表示します。
/ncs-config/cli/show-editors (boolean) [true]	*show-editors* は「true」または「false」です。「true」の場合、ユーザーが構成モードに入ると、現在のエディタのリストが表示されます。
/ncs-config/cli/rollback-aaa (boolean) [false]	「true」の場合、ロールバックファイルがロードされるときに AAA ルールが適用されます。現在のユーザーが権限を持たない変更を他のユーザーが行った場合、ロールバックができない可能性があります。
/ncs-config/cli/rollback-numbering (rolling | fixed) [fixed]	*rollback-numbering* は、「fixed」または「rolling」です。「rolling」の場合、ロールバックファイル「0」には常に最後のコミットが含まれます。「fixed」の場合、ロールバックごとに番号が増加します。
/ncs-config/cli/show-service-meta-data (boolean) [false]	「true」の場合、構成を表示するときに、デフォルトでバックポインタと参照カウントが表示されます。デフォルトは、パイプフラグ「display service-meta」および「hide service-meta」によってオーバーライドできます。
/ncs-config/rest	組み込み WAE Web サーバーが TCP および SSL に関してどのように動作するかを制御します。
/ncs-config/rest/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、Web サーバーが開始されます。

パラメータ	説明
/ncs-config/rest/custom-headers	—
/ncs-config/rest/custom-headers/header	—
/ncs-config/rest/custom-headers/header/name (string)	—
/ncs-config/rest/custom-headers/header/value (string)	このパラメータは必須です。
/ncs-config/restconf	RESTCONF API の設定を制御します。
/ncs-config/restconf/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、Web UI によって使用される Web サーバーで RESTCONF API が有効になります。Web UI も有効にする必要があることに注意してください。
/ncs-config/restconf/root-resource (string) [restconf]	RESTCONF ルートリソースパス。
/ncs-config/webui	組み込み WAE Web サーバーが TCP および SSL に関してどのように動作するかを制御します。
/ncs-config/webui/custom-headers	*custom-headers* には、RFC7230 で定義されている有効な header-field とともに、任意の数の header 要素が含まれています。ヘッダーは、「/login.html」、「/index.html」、および「/jsonrpc」の HTTP レスポンスの一部です。
/ncs-config/webui/custom-headers/header	—
/ncs-config/webui/custom-headers/header/name (string)	—
/ncs-config/webui/custom-headers/header/value (string)	このパラメータは必須です。
/ncs-config/webui/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、Web サーバーが開始されます。
/ncs-config/webui/server-name (string) [localhost]	Web サーバーが提供するホスト名。
/ncs-config/webui/match-host-name (boolean) [false]	Web サーバーが、上で定義された server-name に準拠する URL のみを提供するかどうかを指定します。デフォルトでは、server-name は「localhost」であり、match-host-name は「false」です。URL には任意のサーバー名を指定できます。サーバーが server-name に準拠する URL のみを受け入れるようにする場合は、この設定を有効にします。

パラメータ	説明
/ncs-config/webui/cache-refresh-secs (uint64) [0]	WAE Web サーバーは、静的コンテンツに RAM キャッシュを使用します。エントリは（アクセス時に）ディスクから再読み取りされる前に、キャッシュに数秒間保持されます。デフォルトは 0 です。
/ncs-config/webui/max-ref-entries (uint64) [100]	leafref および keyref エントリは、自動生成された Web UI のドロップダウンメニューとして表されます。デフォルトでは、フェッチされるエントリは 100 以下です。この要素は、この番号を構成可能にします。
/ncs-config/webui/docroot (string)	ディスク上のドキュメントルートの場所。この構成可能項目が省略されている場合、docroot は代わりに WAE ディストリビューションの次世代の docroot を指します。
/ncs-config/webui/login-dir (string)	*login-dir* は、Web UI へのログインに使用される HTML コードを含む代替ログインディレクトリを示します。このディレクトリは https://<*ip-address*>/login にマップされています。この要素が指定されていない場合、代わりに docroot のデフォルトの login/ ディレクトリが使用されます。
/ncs-config/webui/X-Frame-Options (DENY | SAMEORIGIN | ALLOW-FROM) [DENY]	デフォルトでは、*X-Frame-Options* ヘッダーは /login.html および /index.html ページで DENY に設定されています。このヘッダーを使用すると、代わりに SAMEORIGIN または ALLOW-FROM に設定できます。
/ncs-config/webui/disable-auth	—
/ncs-config/webui/disable-auth/dir (string)	このパラメータは複数回指定できます。*disable-auth* 要素には、任意の数の *dir* 要素が含まれます。各 *dir* 要素は、AAA エンジンによって制限されるべきではない docroot 内のディレクトリパスを指します。*dir* 要素が指定されていない場合、次のディレクトリおよびファイルは AAA エンジンによって制限されません：「/login」および「/login.html」。
/ncs-config/webui/allow-symlinks (boolean) [true]	docroot ディレクトリでシンボリックリンクを許可します。
/ncs-config/webui/transport	Web サーバーがリッスンするトランスポートサービス（TCP または SSL など）を制御します。
/ncs-config/webui/transport/tcp	Web サーバーの TCP トランスポートサービスがどのように動作するかを制御します。
/ncs-config/webui/transport/tcp/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、Web サーバーはトランスポートサービスとしてクリアテキストの TCP を使用します。

その他の WAE CLI コマンド

パラメータ	説明
/ncs-config/webui/transport/tcp/ redirect (string)	ユーザーを指定された URL にリダイレクトします。@HOST@ と @PORT@ の 2 つのマクロを指定できます。次に例を示します。 https://@HOST@:443 または https://192.12.4.3:@PORT@
/ncs-config/webui/transport/ tcp/ip (ipv4-address | ipv6-address) [0.0.0.0]	Web サーバーがリッスンする必要がある IP アドレス。0.0.0.0 は、マシン上のすべての IPv4 アドレスのポート（/ncsconfig/webui/transport/tcp/port）でリッスンすることを意味します。
/ncs-config/webui/transport/ tcp/port (port-number) [8008]	*port* は、/ncs-config/webui/transport/tcp/ip のアドレスと組み合わせて使用する有効なポート番号です。
/ncs-config/webui/transport/tcp/extra-listen	Web サーバーもリッスンする必要がある追加の IP アドレスとポートのペアのリスト。
/ncs-config/webui/ transport/tcp/extra-listen/ip (ipv4-address | ipv6-address)	—
/ncs-config/webui/ transport/tcp/extra-listen/port (port-number)	—
/ncs-config/webui/ transport/ssl	Web サーバーの SSL トランスポートサービスがどのように動作するかを制御します。SSL はインターネットで広く展開されています。事実上、すべてのオンラインショッピングと銀行取引が SSL 暗号化を使用して行われます。SSL について詳しく説明している優れたソースはたくさんあります。たとえば、http://www.tldp.org/HOWTO/SSL-Certificates-HOWTO/ では、証明書とキーの管理方法が説明されています。
/ncs-config/webui/ transport/ssl/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、Web サーバーはトランスポートサービスとして SSL を使用します。
/ncs-config/webui/transport/ ssl/redirect (string)	ユーザーを指定された URL にリダイレクトします。@HOST@ と @PORT@ の 2 つのマクロを指定できます。次に例を示します。 http://@HOST@:80 または http://192.12.4.3:@PORT@
/ncs-config/webui/transport/ssl/ip (ipv4-address | ipv6-address) [0.0.0.0]	Web サーバーが着信 SSL 接続をリッスンする IP アドレス。0.0.0.0 は、マシン上のすべての IPv4 アドレスのポート（/ncs-config/webui/transport/ssl/port）でリッスンすることを意味します。

パラメータ	説明
/ncs-config/webui/ transport/ssl/port (port-number) [8888]	*port* は、/ncs-config/webui/transport/tcp/ip と組み合わせて使用する有効なポート番号です。
/ncs-config/webui/transport/ssl/extra-listen	Web サーバーが着信 SSL 接続をリッスンする追加の IP アドレスとポートのペアのリスト。
/ncs-config/webui/ transport/ssl/extra-listen/ip (ipv4-address | ipv6-address)	—
/ncs-config/webui/ transport/ssl/extra-listen/port (port-number)	—
/ncs-config/webui/transport/ ssl/key-file (string)	証明書の秘密キーを含むファイルを指定します。証明書の詳細については、/ncs-config/webui/ transport/ssl/cert-file を参照してください。この構成可能変数が省略されている場合、keyFile は、代わりに WAE ディストリビューションの組み込みの自己署名証明書/キーを指します。注：この証明書/キーはテスト目的でのみ使用してください。
/ncs-config/webui/transport/ ssl/cert-file (string)	サーバー証明書を含むファイルを指定します。証明書は、自己署名テスト証明書、または認証局（CA）から購入した正規の検証済み証明書のいずれかです。この構成可能変数が省略されている場合、keyFile は、代わりに WAE ディストリビューションの組み込みの自己署名証明書/キーを指します。注：この証明書/キーはテスト目的でのみ使用してください。 WAE ディストリビューションには、テストに使用できるサーバー証明書が付属しています（${NCS_DIR}/var/ncs/webui/cert/host.{cert,key}）。このサーバー証明書は、ローカル CA 証明書を使用して生成されています。 $ openssl OpenSSL> genrsa -out ca.key 4096 OpenSSL> req -new -x509 -days 3650 -key ca.key - out ca.cert OpenSSL> genrsa -out host.key 4096 OpenSSL> req -new -key host.key -out host.csr OpenSSL> x509 -req -days 365 -in host.csr -CA ca.cert \ -CAkey ca.key -set_serial 01 -out host.cert
/ncs-config/webui/transport/ ssl/ca-cert-file (string)	クライアント認証時、およびサーバー証明書チェーンを構築するときに使用する、信頼できる証明書を含むファイルを指定します。このリストは、証明書がリクエストされたときにクライアントに渡される、受け入れ可能な CA 証明書のリストでも使用されます。 WAE ディストリビューションには、テストに使用できる CA 証明書が付属しています（${NCS_DIR}/var/ncs/webui/ca_cert/ca.cert）。この CA 証明書は、上記のように生成されています。

パラメータ	説明

/ncs-config/webui/transport/ssl/verify (1 | 2 | 3) [1]	サーバーがクライアント証明書に対して行う検証のレベルを指定します。
	• 1：検証なし。
	• 2：サーバーはクライアントに証明書を要求しますが、クライアントが証明書を提供しない場合でも失敗になりません。
	• 3：サーバーはクライアントがクライアント証明書を提供することを要求します。
	ca-cert-file が上で生成された ca.cert ファイルに設定されている場合は、次を使用して動作することを確認できます。
	$ openssl s_client -connect 127.0.0.1:8888 \ -cert client.cert -key client.key
	これが機能するには、上記の ca.cert を使用して client.cert が生成されている必要があります。
	$ openssl OpenSSL> genrsa -out client.key 4096 OpenSSL> req -new -key client.key -out client.csr OpenSSL> x509 -req -days 3650 -in client.csr -CA ca.cert \ -CAkey ca.key -set_serial 01 -out client.cert

/ncs-config/webui/transport/ssl/depth (uint64) [1]	クライアント証明書を検証するときにサーバーが従う準備ができている証明書チェーンの深さを指定します。

パラメータ	説明
/ncs-config/webui/transport/ ssl/ciphers (string) [DEFAULT]	サーバーが使用する暗号スイートを指定します。暗号は、次のセットから選択されたコロン区切りリストです。 ECDHEECDSA-AES256-SHA384、ECDHE-RSA-AES256-SHA384、ECDH-ECDSA-AES256-SHA384、ECDH-RSA-AES256-SHA384、DHE-RSA-AES256-SHA256、DHE-DSS-AES256-SHA256、AES256-SHA256、ECDHE-ECDSA-AES128-SHA256、ECDHE-RSA-AES128-SHA256、ECDHECDSA-AES128-SHA256、ECDH-RSA-AES128-SHA256、DHE-RSA-AES128-SHA256、DHEDSS-AES128-SHA256、AES128-SHA256、ECDHE-ECDSA-AES256-SHA、ECDHE-RSA-AES256-SHA、DHE-RSA-AES256-SHA、DHE-DSS-AES256-SHA、ECDH-ECDSA-AES256-SHA、ECDHRSA-AES256-SHA、AES256-SHA、ECDHE-ECDSA-DES-CBC3-SHA、ECDHE-RSA-DES-CBC3-SHA、EDH-RSA-DES-CBC3-SHA、EDH-DSS-DES-CBC3-SHA、ECDH-ECDSA-DES-CBC3-SHA、ECDH-RSA-DES-CBC3-SHA、DES-CBC3-SHA、ECDHE-ECDSA-AES128-SHA、ECDHE-RSAAES128-SHA、DHE-RSA-AES128-SHA、DHE-DSS-AES128-SHA、ECDH-ECDSA-AES128-SHA、ECDH-RSA-AES128-SHA、AES128-SHA、ECDHE-ECDSA-RC4-SHA、ECDHE-RSA-RC4-SHA、RC4-SHA、RC4-MD5、EDH-RSA-DES-CBC-SHA、ECDH-ECDSA-RC4-SHA、ECDH-RSA-RC4-SHA、および DES-CBC-SHA、または「DEFAULT」という単語（DES、RC4、またはMD5アルゴリズムを使用するスイートを除き、リストされているセットを使用してください） 暗号スイートの定義については、OpenSSL のマニュアルページ ciphers(1) を参照してください。注：ciphers(1) で説明されている一般的な暗号リスト構文はサポートされていません。
/ncs-config/webui/transport/ ssl/protocols (string) [DEFAULT]	サーバーが使用する SSL/TLS プロトコルバージョンを、sslv3 tlsv1 tlsv1.1 tlsv1.2 のセットから空白区切りリストとして、または「DEFAULT」という単語を指定します（sslv3 を除くすべてのサポートされているプロトコルバージョンを使用してください）。
/ncs-config/webui/cgi	CGI スクリプトのサポート。
/ncs-config/webui/cgi/ enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、CGI スクリプトのサポートが有効になります。
/ncs-config/webui/cgi/ dir (string) [cgi-bin]	CGI スクリプトの場所へのディレクトリパス。
/ncs-config/webui/cgi/ request-filter (string)	正規表現で指定されていない文字をサイレントに除外することを指定します。

パラメータ	説明
/ncs-config/webui/cgi/ max-request-length (uint16)	リクエストの最大文字数を指定します。この制限を超えるすべての文字は、サイレントに無視されます。
/ncs-config/webui/cgi/php	PHP サポート。
/ncs-config/webui/cgi/php/ enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、PHP サポートが有効になります。
/ncs-config/webui/ idle-timeout (xs:duration) [PT30M]	Web UI セッションを終了するまでの最大アイドル時間。PT0M はタイムアウトがないことを意味します。デフォルトは PT30M（30 分）です。
/ncs-config/webui/ absolute-timeout (xs:duration) [PT60M]	Web UI セッションを終了するまでの最大絶対時間。PT0M はタイムアウトがないことを意味します。デフォルトは PT60M（60 分）です。
/ncs-config/webui/ rate-limiting (uint64) [1000000]	1 時間ごとに許可される JSON-RPC リクエストの最大数。0 は無限を意味します。デフォルトは 100 万です。
/ncs-config/webui/ audit (boolean) [true]	*audit* は「true」または「false」です。「true」の場合、JSON-RPC/CGI リクエストは監査ログに記録されます。
/ncs-config/japi	Java-API パラメータ。
/ncs-config/japi/new-session-timeout (xs:duration) [PT30S]	データプロバイダーが制御ソケットリクエストに応答するためのタイムアウト。DpTrans を参照してください。Dp が所定の時間内に応答しない場合、切断されます。
/ncs-config/japi/query-timeout (xs:duration) [PT120S]	データプロバイダーがワーカーソケットクエリに応答するためのタイムアウト。DpTrans を参照してください。Dp が所定の時間内に応答しない場合、切断されます。
/ncs-config/japi/connect-timeout (xs:duration) [PT60S]	ソケットを WAE サーバーに接続した後、データプロバイダーが最初のメッセージを送信するためのタイムアウト。Dp が所定の時間内に接続を開始できない場合、切断されます。
/ncs-config/japi/object-cache-timeout (xs:duration) [PT2S]	getObject() および iterator(),nextObject() コールバックリクエストによって使用されるキャッシュのタイムアウト。WAE はこれらの呼び出しの結果をキャッシュし、ノースバウンドエージェントからの getElem() リクエストをキャッシュから処理します。 このタイムアウトの設定が低すぎると、コールバックが機能しなくなります。たとえば、ノースバウンドエージェントからの getElem() リクエストごとに getObject() を呼び出すことができます。

パラメータ	説明
/ncs-config/japi/event-reply-timeout (xs:duration) [PT120S]	応答を必要とする通知のイベント通知サブスクライバからの応答に対するタイムアウト。Notif クラスを参照してください。サブスクライバが所定の時間内に応答しなかった場合、イベント通知ソケットは閉じられます。
/ncs-config/netconf-north-bound	NETCONF エージェントが NETCONF および SSH に関してどのように動作するかを制御します。
/ncs-config/netconf-north-bound/ enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、NETCONF エージェントが開始されます。
/ncs-config/netconf-north-bound/ transport	NETCONF エージェントがリッスンするトランスポートサービス（TCP または SSH）を制御します。
/ncs-config/netconf-north-bound/ transport/ssh	NETCONF SSH トランスポートサービスがどのように動作するかを制御します。
/ncs-config/netconf-north-bound/ transport/ssh/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、NETCONF エージェントはトランスポートサービスとして SSH を使用します。
/ncs-config/netconf-north-bound/ transport/ssh/ip (ipv4-address | ipv6-address) [0.0.0.0]	*ip* は、WAE NETCONF エージェントがリッスンする IP アドレスです。0.0.0.0 は、マシン上のすべての IPv4 アドレスのポート（/ncs-config/netconf-north-bound/transport/ssh/port）でリッスンすることを意味します。
/ncs-config/netconf-north-bound/ transport/ssh/port (port-number) [2022]	*port* は、/ncs-config/netconf-north-bound/transport/ssh/ip と組み合わせて使用する有効なポート番号です。SSH 経由の NETCONF の標準ポートは 830 です。
/ncs-config/netconf-north-bound/ transport/ssh/extra-listen	WAE NETCONF エージェントがリッスンする追加の IP アドレスとポートのペアのリスト。
/ncs-config/netconf-north-bound/ transport/ssh/extra-listen/ip (ipv4-address | ipv6-address)	—
/ncs-config/netconf-north-bound/ transport/ssh/extra-listen/port (port-number)	—
/ncs-config/netconf-north-bound/ transport/tcp	NETCONF over TCP は標準化されていませんが、開発中には役立ちます（たとえば、スクリプトに netcat を使用する場合）。また、独自のトランスポートを使用する場合にも役立ちます。localhost でリッスンするように NETCONF エージェントを設定し、トランスポート サービス モジュールからプロキシすることができます。

パラメータ	説明
/ncs-config/netconf-north-bound/ transport/tcp/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、NETCONF エージェントはトランスポートサービスとしてクリアテキストの TCP を使用します。
/ncs-config/netconf-north-bound/ transport/tcp/ip (ipv4-address | ipv6-address) [0.0.0.0]	*ip* は、WAE NETCONF エージェントがリッスンする IP アドレスです。0.0.0.0 は、マシン上のすべての IPv4 アドレスのポート（/ncs-config/netconf-north-bound/transport/tcp/port）でリッスンすることを意味します。
/ncs-config/netconf-north-bound/ transport/tcp/port (port-number) [2023]	*port* は、/ncs-config/netconf-north-bound/transport/tcp/ip と組み合わせて使用する有効なポート番号です。
/ncs-config/netconf-north-bound/ transport/tcp/extra-listen	WAE NETCONF エージェントがリッスンする追加の IP アドレスとポートのペアのリスト。
/ncs-config/netconf-north-bound/ transport/tcp/extra-listen/ip (ipv4-address | ipv6-address)	—
/ncs-config/netconf-north-bound/ transport/tcp/extra-listen/port (portnumber)	—
/ncs-config/netconf-north-bound/ extended-sessions (boolean) [false]	extended-sessions が有効になっている場合は、<kill-session> を使用してすべての WAE セッションを終了できます。他の NETCONF セッションだけでなく、CLI セッション、Web UI セッションなども終了できます。セッションがロックを保持している場合、「0」の代わりにそのセッション ID が <lock-denied> で返されます。 この拡張は NETCONF 仕様の対象外です。したがって、デフォルトでは false です。
/ncs-config/netconf-north-bound/ idle-timeout (xs:duration) [PT0S]	NETCONF セッションを終了するまでの最大アイドル時間。セッションが通知を待っているか、保留中の確認済みコミットがある場合、アイドルタイムアウトは使用されません。デフォルト値は 0 で、タイムアウトがないことを意味します。
/ncs-config/netconf-north-bound/ rpc-errors (close | inline) [close]	*rpc-errors* が「inline」であり、WAE がデータプロバイダーからデータをフェッチしようとしているときに <get> または <get-config> リクエストの処理でエラーが発生した場合、WAE は障害のある要素に rpc-error 要素を生成し、次の要素の処理を続行します。エラーが発生し、*rpc-errors* が「close」の場合、WAE は NETCONF トランスポートを閉じます。

パラメータ	説明
/ncs-config/netconf-north-bound/ max-batch-processes (uint32 | unbounded) [unbounded]	同時 NETCONF バッチプロセスの数を制御します。新しい NETCONF 操作がバッチ操作として実装されている場合、エージェントはバッチプロセスを開始できます。
/ncs-config/netconf-north-bound/ capabilities	有効にする NETCONF 機能を制御します。
/ncs-config/netconf-north-bound/ capabilities/url	サポートする URL 機能オプションをオンにします。
/ncs-config/netconf-north-bound/ capabilities/url/enabled (boolean) [false]	*enabled* は「true」または「false」です。「true」の場合、URL NETCONF 機能が有効になります。
/ncs-config/netconf-north-bound/ capabilities/url/file	URL ファイルサポートの動作方法を制御します。
/ncs-config/netconf-north-bound/ capabilities/url/file/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、URL ファイルスキームが有効になります。
/ncs-config/netconf-north-bound/ capabilities/url/file/root-dir (string)	*root-dir* は、ConfD が URL 機能を使用した NETCONF 操作の結果を保存する、ディスク上のディレクトリパスです。ファイル URL スキームが有効になっている場合は、このパラメータを設定する必要があります。
/ncs-config/netconf-north-bound/ capabilities/url/ftp	URL FTP スキームの動作方法を制御します。
/ncs-config/netconf-north-bound/ capabilities/url/ftp/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、URL FTP スキームが有効になります。
/ncs-config/netconf-north-bound/ capabilities/url/sftp	URL SFTP スキームの動作方法を制御します。
/ncs-config/netconf-north-bound/ capabilities/url/sftp/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、URL SFTP スキームが有効になります。
/ncs-config/netconf-north-bound/ capabilities/inactive	非アクティブな機能オプションを制御します。
/ncs-config/netconf-north-bound/ capabilities/inactive/enabled (boolean) [true]	*enabled* は「true」または「false」です。「true」の場合、「http://tail-f.com/ns/netconf/inactive/1.0」機能が有効になります。

パラメータ	説明
/ncs-config/southbound-source-address	WAE からデバイスへのサウスバウンド接続に使用する送信元アドレスを指定します。ほとんどの場合、送信元アドレスの割り当ては、OS の TCP/IP スタックに任せるのが最善です。これは、アドレスが正しくないと接続が失敗する可能性があるためです。ただし、スタックが複数のアドレスを選択でき、その選択を 1 つのアドレスに制限する必要がある場合は、これらの設定を使用できます。
/ncs-config/southbound-source-address/ipv4 (ipv4-address)	サウスバウンド IPv4 接続に使用する送信元アドレス。設定されていない場合、送信元アドレスは OS によって割り当てられます。
/ncs-config/southbound-source-address/ipv6 (ipv6-address)	サウスバウンド IPv6 接続に使用する送信元アドレス。設定されていない場合、送信元アドレスは OS によって割り当てられます。
/ncs-config/ha	—
/ncs-config/ha/enabled (boolean) [false]	「true」の場合、HA モードが有効になります。
/ncs-config/ha/ip (ipv4-address | ipv6-address) [0.0.0.0]	WAE が他の HA ノードからの着信接続をリッスンする IP アドレス。
/ncs-config/ha/port (port-number) [4570]	WAE が他の HA ノードからの着信接続をリッスンするポート番号。
/ncs-config/ha/tick-timeout (xs:duration) [PT20S]	HA ノード間で送信されるキープアライブティック間のタイムアウトを定義します。値「PT0」は、キープアライブティックが送信されないことを意味します。
/ncs-config/scripts	コミット後のコールバックなど、WAE でさまざまなことを制御するスクリプトを追加できます。新しいCLIコマンドを追加することもできます。スクリプトは /ncs-config/scripts/dir に保存する必要があります。このディレクトリには、スクリプトカテゴリごとにサブディレクトリがあります。一部のスクリプトカテゴリでは、正しいサブディレクトリにスクリプトを追加するだけでスクリプトが有効になります。その他の場合は、いくつかの構成を行う必要があります。
/ncs-config/scripts/dir (string)	このパラメータは複数回指定できます。プラグアンドプレイ スクリプトの場所へのディレクトリパス。scripts ディレクトリには、次のサブディレクトリが必要です。 scripts/command/ post-commit/
/ncs-config/large-scale	—
/ncs-config/large-scale/lsa	—
/ncs-config/large-scale/lsa/enabled (boolean) [false]	個別の Cisco Smart License が必要な Layered Service Architecture（LSA）を有効にします。

その他の WAE CLI コマンド

【注意】シスコ製品をご使用になる前に、安全上の注意（ **www.cisco.com/jp/go/safety_warning/** ）をご確認ください。本書は、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。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

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。