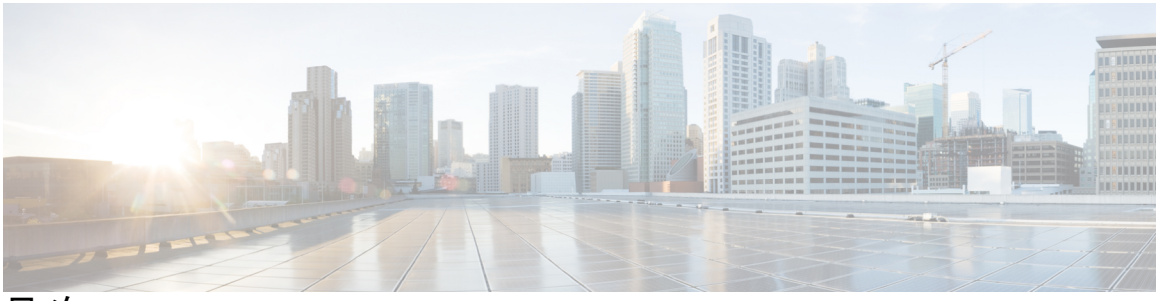
Cisco WAE 7.4.0 ユーザーガイド

初版：2021 年 6 月 15 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 1

Cisco WAE の概要 1

Cisco WAE のアーキテクチャ 2

ネットワーク インターフェイス モジュール 2

ネットワークモデル 3

デルタ集約ルールエンジン 3

単純な集約エンジン 4

Cisco WAE モデリングデーモン（WMD） 5

Cisco WAE アプリケーション 5

オンデマンド帯域幅アプリケーション 5

Bandwidth Optimization アプリケーション 6

Cisco WAE のインターフェイス 6

ネットワークモデル作成ワークフロー 7

第 2 章

ネットワークモデルの構成：Cisco WAE UI 9

Cisco WAE UI の概要 9

Cisco WAE UI を使用したネットワークモデルの構成 12

Cisco WAE UI を使用したネットワークアクセスの設定 13

Cisco WAE UI を使用したエージェントの設定 15

Cisco WAE UI を使用したノードフィルタの設定 16

スタンドアロンネットワークの作成 17

ネットワーク モデル コンポーザ を使用 18

ネットワークの作成とトポロジ収集の構成 18

ネットワーク モデル コンポーザ を使用した追加 NIMO の構成 20
ネットワーク モデル コンポーザ を使用した NIMO 収集の統合 21
ネットワーク モデル コンポーザを使用したトラフィック収集またはカスタムスクリプトの実行 22
ネットワーク モデル コンポーザ を使用してアーカイブを設定します。 23
ネットワーク モデル コンポーザ を使用したジョブのスケジュール 24
プランファイルのダウンロード 25

第 3 章 ネットワークモデルの構成：エキスパートモード 27
エキスパートモードの概要 27
ナビゲーションとコミット 28
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 29
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 30
ネットワーク アクセスの設定 30
エキスパートモードを使用したエージェントの構成 31
エキスパートモードを使用した XTC エージェントの構成 31
エキスパートモードを使用した NetFlow エージェントの構成 32
エキスパートモードを使用した SNMP エージェントの構成 34
エキスパートモードを使用した構成解析エージェントの構成 34
ネットワークモデルの作成 35
追加 NIMO の構成 36
WAE エキスパートモードを使用したアーカイブの構成とプランファイルの表示 37

第 4 章 ネットワークモデルの構成：Cisco WAE CLI 39
WAE CLI の概要 39
動作モード 39
組み込みの動作モードコマンド 40
構成モード 45
組み込みの構成モードコマンド 45
エキスパートモードと WAE CLI の比較 50
WAE CLI を使用したネットワークモデルの構成 52

CLI を使用したデバイスアクセスの構成 52
ネットワーク アクセス プロファイルの構成 53
ネットワークモデルの作成 54
プランファイルのロード 55
追加 NIMO の構成 56
WAE CLI を使用したアーカイブの構成 56
アーカイブ内のプランファイルの管理 57

第 5 章 ネットワーク インターフェイス モジュール（NIMO） 59
NIMO の説明 59
基本的なトポロジ収集 63
IGP データベースを使用したトポロジ収集 63
XTC を使用したトポロジ収集 64
BGP-LS XTC の詳細オプション 66
NIMO 収集の統合 67
アグリゲータとマルチレイヤ収集の構成例 69
セグメント ルーティング トラフィック マトリックス収集 71
VPN 収集 72
LSP 構成の収集 72
XTC を使用した LSP 収集 74
構成解析を使用したポート、LSP、SRLG、および VPN 収集 75
BGP ピア収集 77
BGP トポロジの詳細オプション 78
SNMP を使用した LSP 収集 79
インベントリ収集 80
インベントリ収集の設定 85
auth.enc の作成 87
トラフィック収集 88
トラフィックポーラーの詳細オプション 89
トラフィックポーリングの調整 90
ネットワークモデルのレイアウト（可視化） 93

マルチキャストフローデータの収集 94
トラフィック需要の収集 96
AS プランファイルのマージ 97
ネットワークモデルに対する外部スクリプトの実行 98
外部スクリプトの実行例 100

第 6 章 **Cisco WAE モデリングデーモン（WMD）の構成 101**
WAE モデリングデーモン（WMD）の構成 101

第 7 章 **マルチレイヤ（L3-L1）収集 103**
マルチレイヤ収集の制限事項 104
エキスパートモード：マルチレイヤ収集 104
L3-L1 マッピング情報の構成 105
EPN-M エージェントを使用したマルチレイヤ収集の設定 106
Cisco WAE UI：マルチレイヤ収集 110
Cisco WAE CLI：マルチレイヤ収集 111
L1 回路波長オプション 113
L1 回路波長ガイドライン 115
L1 回路波長の設定例 115

第 8 章 **NetFlow データ収集 117**
NetFlow データ収集 117
NetFlow 収集アーキテクチャ 118
分散 Netflow（DNF）収集 118
集中型 NetFlow（CNF）収集 121
NetFlow 収集の構成 122
集中型 NetFlow 構成ワークフロー 122
CNF NetFlow の要件 123
ライセンシング 123
CNF 用のオペレーティングシステムの準備 123
node-flow-configs-table ファイルの作成 123

CNF 構成ファイルの作成 124
CNF 収集の構成 127
CNF 用の netflow-nimo の構成 127
DNF NetFlow 構成ワークフロー 129
分散 NetFlow の要件 129
ライセンシング 130
DNF クラスタの構成 130
Ansible を使用した DNF クラスタの展開 130
DNF クラスタのシャットダウンまたはアンインストール 132
DNF 収集の構成 132
flow_collector_ias および flow_collector_dmd の構成 132
DNF 用の external-executable-nimo の構成 133
Ansible 構成ファイルの作成 133
group_vars/all 134
DNF クラスタ構成ファイルの作成 135

第 9 章 テレメトリの設定 139
テレメトリの概要 139
WAE でのテレメトリの設定 139

第 10 章 自動化アプリケーション 143
自動化アプリケーション 143
オンデマンド帯域幅の構成ワークフロー 143
オンデマンド帯域幅の設定 145
初期オンデマンド帯域幅の CLI 構成例 146
オンデマンド帯域幅のシャットダウン 148
Bandwidth Optimizationアプリケーション ワークフロー 149
Bandwidth Optimization の設定 150
WAE SR ポリシーの制限事項 151
帯域幅最適化のシャットダウン 151

第 11 章 スケジューラ構成 153
スケジューラの概要 153
スケジューラの構成 153
トポロジ収集を実行するためのトリガーの構成例 155

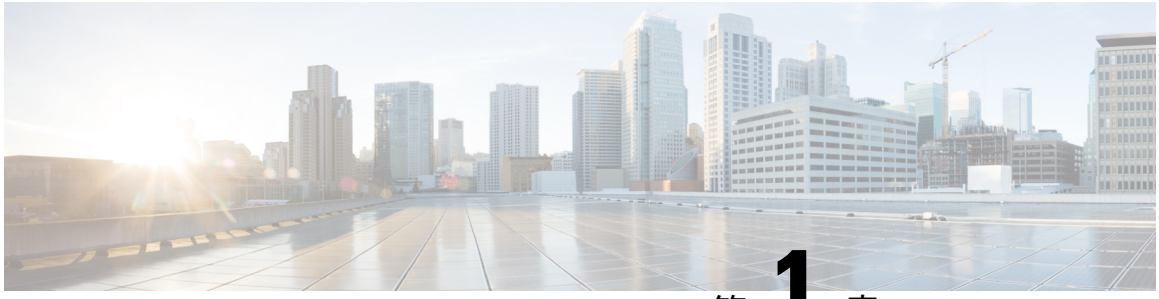
第 12 章 Cisco Smart Licensing 157
シスコ スマートライセンシングの概要 158
スマートライセンシング設定のワークフロー 158
Cisco WAE のスマートライセンシングの有効化 159
Cisco WAE と CSSM 間のトランスポートモードの設定 159
Cisco Smart Software Manager への Cisco WAE の登録 160
Cisco Smart Software Manager へのオフラインモードの Cisco WAE の登録 161
予約の更新 162
予約済みライセンスの返却 163
スマートライセンスの登録と認証ステータス 164

第 13 章 管理 167
ユーザーの管理 167
エージングの構成 168
wae.conf 169
ハイ アベイラビリティの設定 175
ハイ アベイラビリティのトラブルシューティング 178
LDAP の設定 179
CLI を使用した LDAP の設定 179
WAE UI を使用した LDAP の設定 181
LDAP 設定オプション 181
ステータスダッシュボード 183
WAE CLI ロギングについて 184
Syslog 185
Syslog のメッセージと形式 186

データベースのロック 196
グローバルロック 196
トランザクションロック 196
ノースバウンドエージェントとグローバルロック 197
外部データプロバイダーと CDB 197
ユーザーセッションへのロックの影響 198
セキュリティ 198
IPC ポートへのアクセスの制限 200
WAE 運用データのクリア 201
WAE 構成のバックアップと復元 201
WAE 診断 201
WAE 診断ツールの使用 202

第 14 章 セキュリティ 205
主要なセキュリティ概念 205
HTTPS 205
SSL 証明書 205
1 方向 SSL 認証 206

付録 A： その他の WAE CLI コマンド 209
コミットフラグ 209
デバイス アクション 210
サービスアクション 211
wae.conf 構成パラメータ 212



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 （1 ページ）
- Cisco WAE のアーキテクチャ （2 ページ）
- Cisco WAE アプリケーション （5 ページ）
- Cisco WAE のインターフェイス （6 ページ）
- ネットワークモデル作成ワークフロー （7 ページ）

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 プラットフォームの重要な機能の一部は次のようなものです。

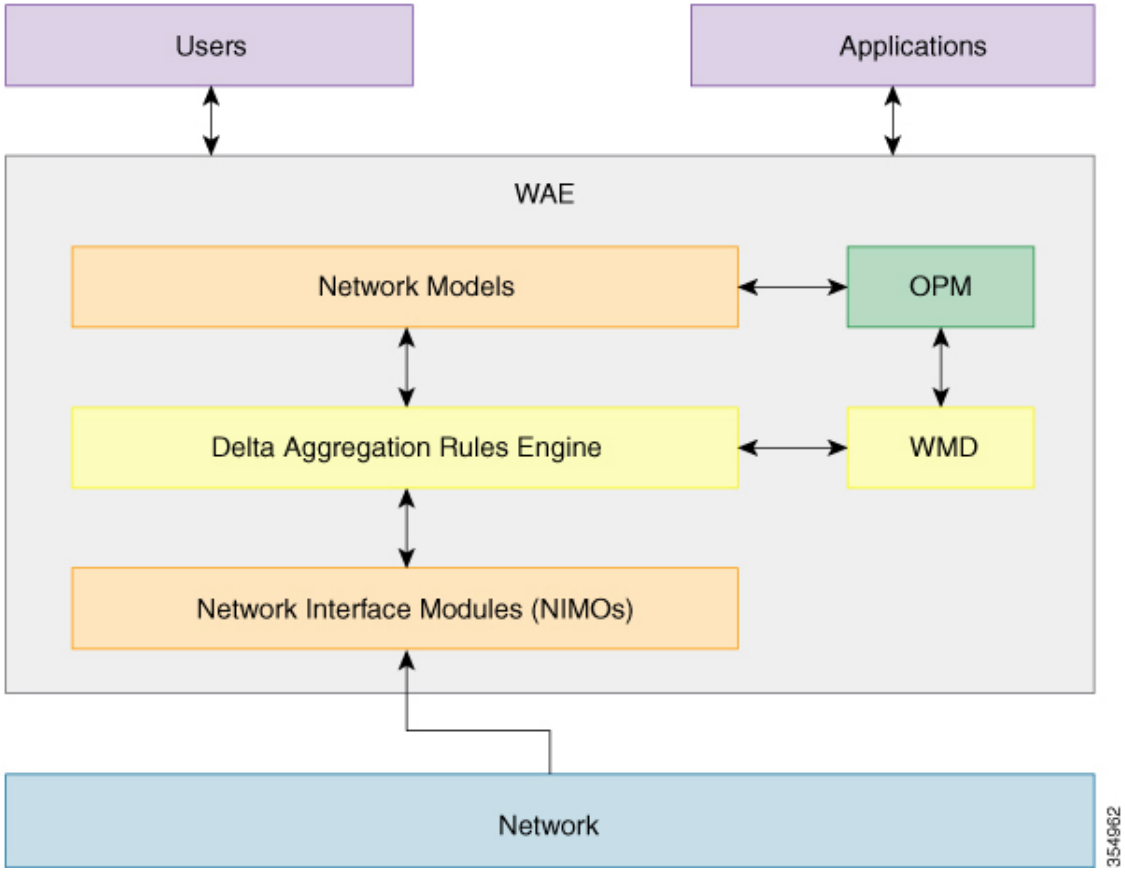
- トラフィック エンジニアリングとネットワークの最適化：TE LSP 構成を計算してネットワークパフォーマンスを改善したり、ローカルまたはグローバルな最適化を実行したりします。
- デマンドエンジニアリング：ネットワーク上のトラフィック需要の追加、削除、または変更がネットワーク トラフィック フローに与える影響を調べます。
- トポロジと予測分析：設計またはネットワーク障害によって引き起こされるネットワークトポロジの変更がネットワークパフォーマンスに与える影響を観察します。
- TE トンネルプログラミング：トンネルパスや予約帯域幅などのトンネルパラメータを変更した場合の影響を調べます。

- 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。

Cisco WAE のアーキテクチャ

本質的に、Cisco 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。

Cisco WAE ネットワークモデルは SQLite で定義され、標準の SQLite 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。



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します。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。

WAE でサポートされるすべての NIMO の包括的なリストについては、ネットワーク インターフェイス モジュール（NIMO） （59 ページ）を参照してください。

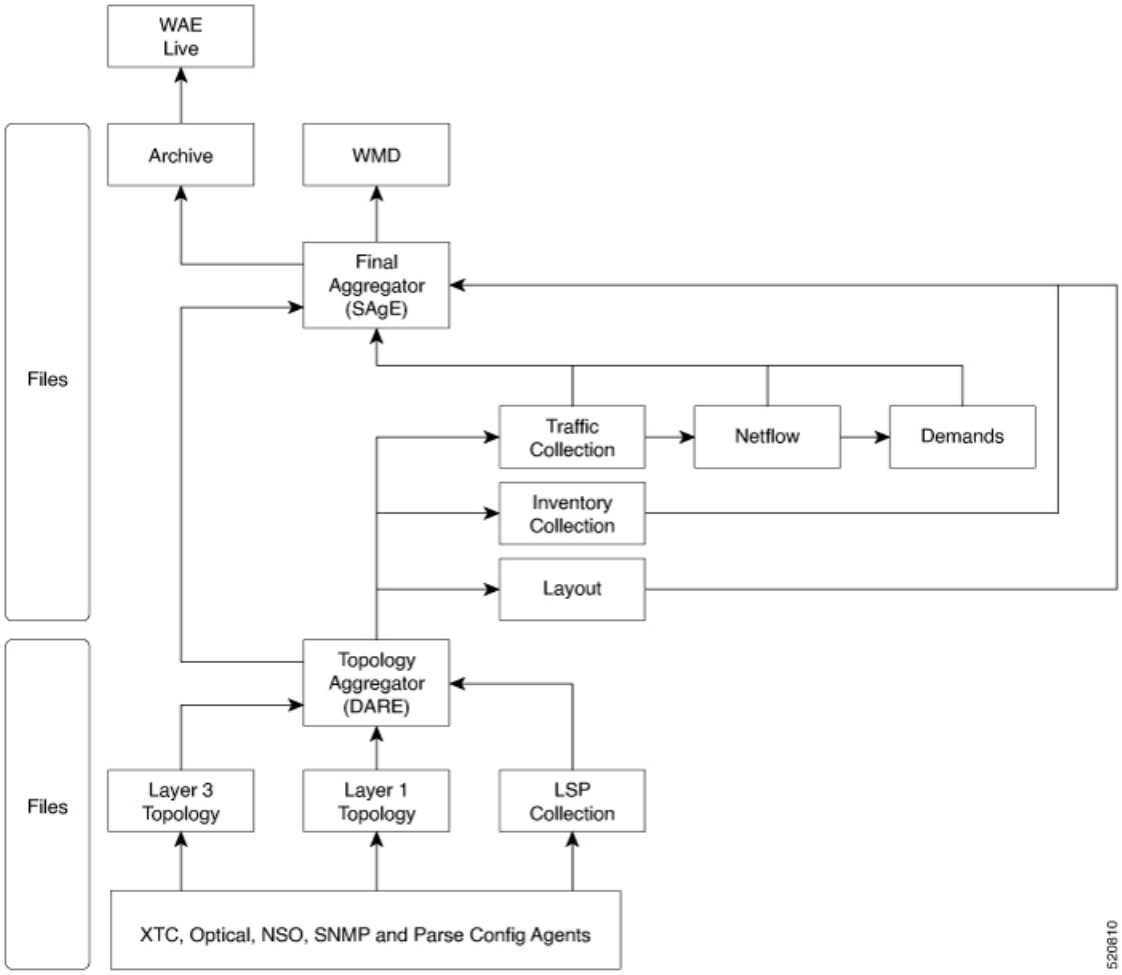
ネットワークモデル

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。

デルタ集約ルールエンジン

DARE アグリゲータは、さまざまな NIMO を 1 つにまとめ、それぞれからモデル情報を選択し、その情報を単一のモデルに統合する WAE コンポーネントです。DARE は、初めに任意の設定済みトポロジ NIMO を統合し、モデルを作成してから、そのモデルに対して他の NIMO を実行します。たとえば、DARE は、LSP 構成 NIMO、L3 トポロジ NIMO、L1 トポロジ NIMO を単一のモデルに統合します。その後、トラフィック収集、インベントリ収集、レイアウト、NetFlow、需要が続きます。

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。



（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 （67 ページ）を参照してください。

単純な集約エンジン

単純な集約エンジン（SAgE）は、トラフィック、インベントリ、レイアウト、NetFlow、需要などのすべてのネットワーク情報を統合し、DARE ネットワークから最終的なネットワークへのトポロジ変更とともにこれらの変更を集約する WAE コンポーネントです。すべての NIMO からのネットワーク情報がプランファイルに書き込まれます。ネットワークの変更は、SAgE からアーカイブできます。

SAgE アグリゲータを使用すると、トラフィック収集、インベントリ収集、レイアウトなどを並行して実行できます。

SAgEアグリゲータを構成する方法については、ネットワーク モデル コンポーザを使用したトラフィック収集またはカスタムスクリプトの実行 （22 ページ）を参照してください。

Cisco WAE モデリングデーモン（WMD）

WMD は、スケジュールされた NIMO の実行を組み込んで、SAgE から変更を受信します。すべての更新は、ネットワークのほぼリアルタイムのプライマリモデルに統合されます。Cisco 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、Cisco WAE OPM API 機能を利用することができます。WMD の設定はオプションであり、 帯域幅アプリケーションを使用する場合にのみ必要です。

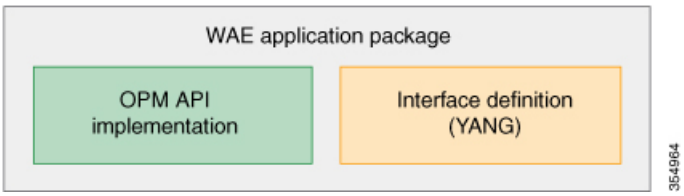
WMD の構成方法については、WAE モデリングデーモン（WMD）の構成 （101 ページ）を参照してください。

Cisco WAE アプリケーション

Cisco 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 Cisco 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、Cisco 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。最適化および予測モジュール（OPM）を使用して実装されています。

  OPM API 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。これにより、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。



Cisco WAE は YANG 定義から API を自動的に生成するため、Cisco 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。Cisco WAE アプリケーションは、ある意味で、Cisco 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 （143 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 （149 ページ）を参照してください。

Cisco WAE のインターフェイス

Cisco WAE には、ネットワークモ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。

Cisco WAE UI

Cisco 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。Cisco WAE UI は、 1 つのネットワーク下の複数のデータ収集の構成をまとめ、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、Cisco WAE UI では実行できない操作がいくつかあります。エキスパートモードまたは Cisco WAE CLI を使用して実行された構成は、Cisco 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：Cisco WAE UI （9 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、Cisco 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 （27 ページ） を参照してください。

Cisco WAE CLI

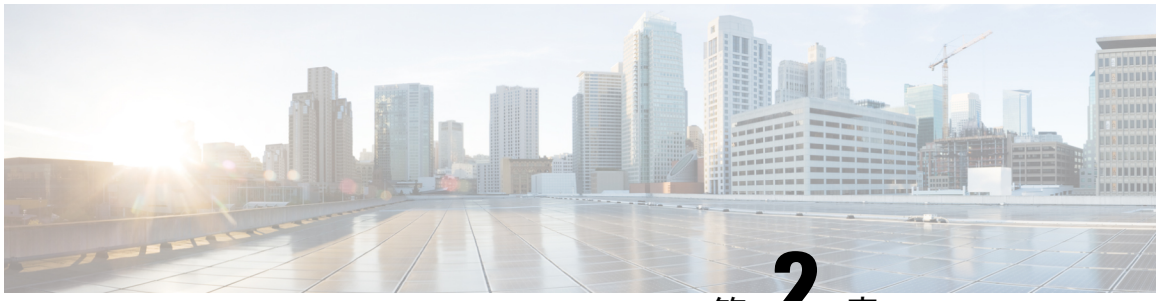
Cisco WAE CLI は、ユーザーがコマンドを入力してビジュアルプロンプトに応答するインターフェイスです。システム応答が返されます。これは、すべての Cisco WAE 構成に必要な最低限のインターフェイスです。エキスパート モードで使用できる操作は、Cisco WAE CLI でも使用できます。ネットワークモデルの構成：Cisco WAE CLI（39 ページ）を参照してください。

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設定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、Cisco WAE UI、または Cisco WAE）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 （67 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）エージェントを設定します。この手順は、XTC、LAG およびポートインターフェイス、マルチレイヤ、Netflow、またはテレメトリ情報を収集する場合にのみ必要です。
3. トポロジ NIMO を使用して、集約されたネットワークと送信元を設定します。
4. デマンド、トラフィック、レイアウト、インベントリなどの追加の収集を設定します。
5. 収集をいつ実行するかをスケジュールします。
6. 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設定します。
7. （オプション）Cisco 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：Cisco WAE UI

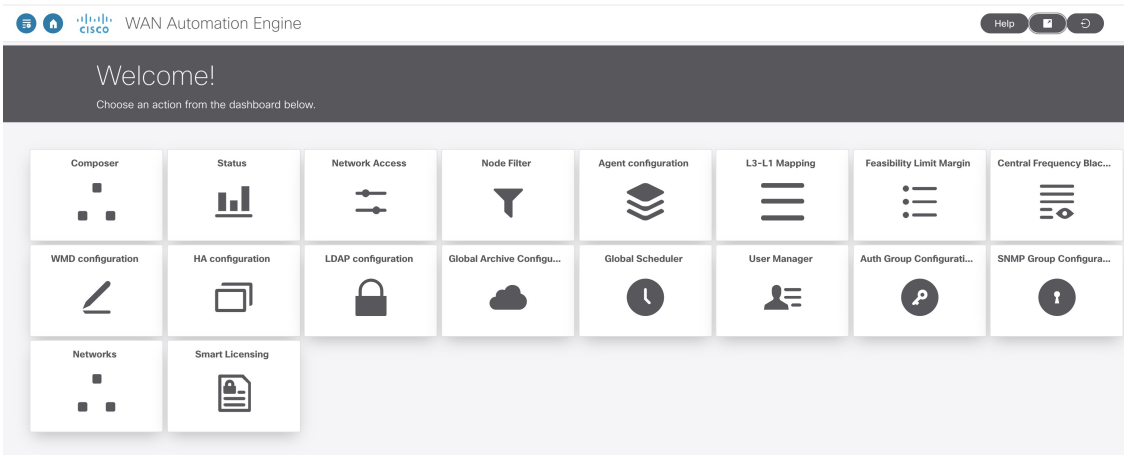
ここでは、次の内容について説明します。

- Cisco WAE UI の概要 （9 ページ）
- Cisco WAE UI を使用したネットワークモデルの構成 （12 ページ）

Cisco WAE UI の概要

Cisco WAE UI には、デバイスとネットワークのアクセス、ネットワークモデルの作成、ユーザー管理、エージェント設定などのための、使いやすい設定ツールが用意されています。

基本的なネットワークモデル設定については、ネットワーク モデル コンポーザで始めることをお勧めします。また、エキスパート モード や Cisco WAE CLI を使用して、特定の操作も実行できます。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設定が保存されます。



（注） ユーザーごとに1つのセッションのみがアクティブであることを確認してください。ユーザーごとに複数のセッションを行うと問題が発生する可能性があるため、お勧めしません。

アイコン	説明	詳細
	メイン Cisco WAE UI ランディングページに戻ります。	—
コンポーザ	ネットワークモデルコンポーザを開き、ネットワークモデルを作成および構築できます。	ネットワークモデルコンポーザ を使用 （18 ページ）
ステータス	ステータスダッシュボードは、システムリークの原因となるプロセスや、リソースを使い切っているプロセスの特定に役立ちます。	ステータスダッシュボード（183 ページ）
Network Access	[ネットワークアクセス設定（network access configuration）]ページを開きます。このページでは、グローバルデバイスとネットワークのログイン情報を設定できます。	Cisco WAE UI を使用したネットワークアクセスの設定 （13 ページ）
ノードフィルタ	[ノードフィルタ（Node Filter）] ページを開きます。このページでは、個々のノードを収集から除外したり含めたりできます。	Cisco WAE UI を使用したノードフィルタの設定 （16 ページ）
エージェントの設定	[エージェントの設定（agent configuration）] ページを開きます。このページでは、エージェントを作成したり変更したりできます。	Cisco WAE UI を使用したエージェントの設定 （15 ページ）
L3-L1 マッピング	[L3-L1マッピング設定（L3-L1 mapping configuration）]ページを開きます。このページでは、マルチレイヤ収集のための L3-L1 ノードおよび回路のマッピングを作成および変更できます。	L3-L1 マッピング情報の構成（105 ページ）
フィージビリティ制限マージン	[フィージビリティ制限マージンの設定（feasibility limit margin configuration）] ページを開きます。このページでは、L1 回路パスの許容可能な品質を設定できます。	「L1 回路波長」トピックは Cisco WAE Design ユーザー ガイド にあります。
中心周波数除外リスト	[中心周波数除外リストの設定（central frequency excludelist configuration）] ページを開きます。このページでは、L1 回路パスの中心周波数 ID として機能しない可能性がある周波数 ID のリストを定義できます。	「L1 回路波長」および「中心周波数 ID の除外リスト」のトピックは Cisco WAE Design ユーザーガイドにあります。

アイコン	説明	詳細
WMD 設定	[WMD設定（WMD configuration）] ページを開きます。このページでは、デバッグ、rpc およびアプリケーション サブスクライバ接続、デマンドなどの WMD オプションを表示できます。これらのオプションを編集するには、エキスパートモードや WAE CLI または WAE UI を使用します。	Cisco WAE モデリングデーモン（WMD）の構成（101 ページ）
HA 設定	[高可用性（HA）設定（high availability (HA) configuration）] ページを開きます。このページでは、HA に使用するノードを指定できます。	ハイアベイラビリティの設定（175 ページ）
LDAP 設定	[LDAP 設定（LDAP configuration）] ページを開きます。このページでは、LDAP の詳細を有効にして設定できます。	• LDAP の設定（179 ページ） • WAE UI を使用した LDAP の設定（181 ページ）
グローバルアーカイブ設定	[グローバルアーカイブ（Global Archive）] ページを開きます。このページでは、プランファイルをダウンロードできます。	プランファイルのダウンロード（25 ページ）
グローバルスケジューラ	[グローバルスケジューラ（Global Scheduler）] ページを開きます。このページでは、スケジュールされたタスクを作成できます。	ネットワーク モデル コンポーザ を使用したジョブのスケジュール（24 ページ）
ユーザ マネージャ	[ユーザー管理（user management）] ページを開きます。このページでは、ユーザーの追加、変更、削除ができます。	ユーザーの管理（167 ページ）
認証グループの設定	[認証グループの設定（Auth Group Configuration）] ページを開きます。このページでは、新しい認証グループを作成できます。	Cisco WAE UI を使用したネットワークアクセスの設定（13 ページ）
SNMP グループの設定	[SNMP グループの設定（SNMP Group Configuration）] ページを開きます。このページでは、新しい SNMP グループを作成できます。	Cisco WAE UI を使用したネットワークアクセスの設定（13 ページ）
ネットワーク	[ネットワーク（Network）] ページを開きます。このページでは、スタンドアロンネットワークを作成できます。	スタンドアロンネットワークの作成（17 ページ）

アイコン	説明	詳細
スマート ライセンス	[スマートライセンス（Smart Licensing）]ページを開きます。このページでは、Cisco WAE のスマートライセンスを有効にして登録できます。	Cisco Smart Licensing （157 ページ）
	左側のメインの Cisco WAE UI ナビゲーションメニューを切り替えます（左側のサイドバーメニューとも呼ばれます）。	—
	新しいタブで Cisco WAE UI のオンラインヘルプを起動します。 （注） 初版発行後、ドキュメントが更新されることがあります。最新の更新については、Cisco.com にあるCisco WAE 7.2.1 ユーザーガイドのドキュメントを参照してください。	—
	エキスパート モード を別のウィンドウで起動します。	ネットワークモデルの構成：エキスパートモード （27 ページ）
	現在のユーザーをログアウトします。	—

Cisco 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、Cisco WAE UI とネットワーク モデル コンポーザ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。

（注） エキスパートモードまたは Cisco WAE CLI を使用して作成されたネットワークモデル設定を Cisco WAE UI に表示するには、集約ネットワークである必要があります。または、集約ネットワークのソースとして追加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。	Cisco WAE UI を使用したネットワークアクセスの設定 （13 ページ）
2.（オプション）特定の情報を収集するエージェントを作成します。エージェントは、XTC を使用した情報収集またはマルチレイヤ収集に必要です。	Cisco WAE UI を使用したエージェントの設定 （15 ページ）

ステップ	詳細
3.（オプション）DARE ネットワークではないネットワークを作成します。	スタンドアロンネットワークの作成 （17 ページ）
ネットワーク モデル コンポーザを使用して次の操作を行います。	
3. ネットワークを作成してトポロジ収集を実行します。	ネットワークの作成とトポロジ収集の構成 （18 ページ）
4. NIMO を使用して追加のデータ収集を構成します。	ネットワーク モデル コンポーザ を使用した追加 NIMO の構成 （20 ページ）
5. NIMO を集約してネットワークを構築します。	ネットワーク モデル コンポーザ を使用した NIMO 収集の統合 （21 ページ）
6.（オプション）ネットワークで実行するトラフィック収集とカスタマースクリプトを設定します。	ネットワーク モデル コンポーザを使用したトラフィック収集またはカスタムスクリプトの実行 （22 ページ）
8.（オプション）アーカイブを構成します。	ネットワーク モデル コンポーザ を使用してアーカイブを設定します。 （23 ページ）
7.（オプション）ネットワーク収集とエージェントを実行するためのスケジューリングジョブを作成します。	ネットワーク モデル コンポーザ を使用したジョブのスケジュール （24 ページ）

Cisco WAE UI を使用した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 UI から、[ネットワークアクセス（Network Access）] をクリックします。

ステップ 2 [+ネットワークアクセスの作成（+ Create 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。
- [認証グループ（Authorization Group）]：デフォルトを選択するか、新しい認証グループを作成します。新しい認証グループを作成する場合は、それに続くフィールドにその名前と該当する情報を入力します。

（注） WAE UI から直接、新しい認証グループを作成することもできます。Cisco WAE UI から、[認証グループの設定（Auth Group Configuration）] を選択し、[認証グループの作成（Create Auth Group）] をクリックします。詳細を入力して **[Save]** をクリックします。

ステップ 4 デフォルトを選択するか、新しい SNMP グループを作成します。新しい SNMP グループを作成する場合は、名前を入力し、SNMPv2c または SNMPv3 のいずれかを選択します。

（注） WAE UI から直接新しい SNMP グループを作成することもできます。Cisco WAE UI から、[SNMP グループの設定（SNMP Group Configuration）] を選択し、[SNMP グループの作成（Create SNMP Group）] をクリックします。詳細を入力して **[Save]** をクリックします。

- [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 5 [保存（Save）] をクリックします。

ステップ 6 （オプション）これらのネットワークアクセスのログイン情報に関連付けられたノードを追加または編集するには、次の手順を実行します。

a) [ノードアクセスの編集（Edit Node Access）] ボタンをクリックし、次のいずれかを実行します。

- ノードリストをエクスポートするには、  をクリックします。

- ノードリストをインポートするには、  をクリックし、[file-path] フィールドに CSV ファイルパスを入力して、[完了（Done）] をクリックします。この操作により、以前に設定されたノードが上書きされます。

  （注） WAE がインストールされているサーバーに CSV ファイルがあることを確認してください。

- ノードを追加するには、  をクリックしてノードの詳細を入力します。
- ノードを編集するには、ノードを選択して  をクリックし、ノードの詳細を入力します。
- ノードを削除するには、ノードを選択して  をクリックします。
- すべてのノードをまとめて削除するには、[すべて削除（Delete All）] ボタンをクリックします。

b) [完了（Done）] をクリックします。

ステップ 7 [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 コンポーザを使用してネットワークモデルを作成します。

Cisco WAE UI を使用したエージェントの設定

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設定する必要があります。このセクションでは、Cisco WAE UI を使用して XTC エージェントを設定する方法について説明します。

ステップ 1 Cisco WAE UI から、[エージェントの設定（Agent Configuration）] をクリックします。

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

（注） テレメトリエージェントと Netflow エージェントは、デフォルトの設定で作成されます。新しいテレメトリエージェントまたはNetflowエージェントを追加することはできません。ただし、カードをクリックすると、設定を変更または削除できます。

他のエージェントを削除するには、ゴミ箱アイコンを使用します。

ステップ 3 エージェントの名前を入力します。

ステップ 4 [コレクタタイプ（Collector Type）] ドロップダウンリストから、[コレクタ（Collector）] を選択します。

ステップ 5 [エージェントの作成（Create Agent）] をクリックします。

ステップ 6 次に表示されるウィンドウで、適切なエージェント設定値を入力します。フィールドの説明を表示するには、マウスポインタをフィールド名の上に置きます。

ステップ 7 [保存（Save）] をクリックします。

ステップ 8 エージェントを実行するには、**[アクション（Actions）]** > **[run-collection]** をクリックします。

次のタスク

ネットワーク モデル コンポーザ を使用して NIMO を設定し、ネットワークモデルを構築します。NIMO タイプの詳細については、NIMO の説明 （59 ページ）を参照してください。

Cisco WAE UI を使用したノードフィルタの設定

Cisco WAE UI を使用すると、個々のノードを収集に含めたり、収集から除外したりできます。

（注）
- ノードフィルタリストには、ノード名およびループバック IP を追加できますが、ノードフィルタ IP には管理 IP を追加しないでください。
- ノード名は ISIS で機能します。
- OSPF データベースにはノード名がないため、フィルタ処理は IP アドレスでのみ機能します。
- ノードフィルタは、セグメントリストのホップをサポートしていません。

ステップ 1 Cisco WAE UI から、[ノードフィルタ（Node Filter）] をクリックします。

ステップ 2 [ノードフィルタの作成（Create Node Filter）] をクリックします。

ステップ 3 [名前（Name）] フィールドにノードフィルタの名前を入力します。

ステップ 4 単一の表現に複数のノードを含めたり除外したりする場合は、[正規表現（Regex）] オプションを使用します。[正規表現（Regex）] と入力し、[正規表現フィルタ（Regex Filter）] ドロップダウンから INCLUDE ONLY または EXCLUDE ONLY を選択します。

（注） [ノードフィルタ（Node Filter）] オプションを使用する場合は、IGNORE FILTER を選択します。

ステップ 5 または、[正規表現（Regex）] の代わりに、各ノードに IP またはノード名を追加できます。追加するには、[ノードフィルタ（Node Filter）] ドロップダウンから、INCLUDE ONLY または EXCLUDE ONLY を選択します。

[新しいノードの追加（Add New Node）] をクリックして、含めるノードまたは除外するノードを一覧表示します。コンマを区切り文字として使用して、複数のノードを選択できます。

（注） [正規表現（Regex）] オプションを使用する場合は、IGNORE FILTER を選択します。

ステップ 6 [保存（Save）] をクリックします。

スタンドアロンネットワークの作成

次の手順を使用して、DARE ネットワークではない別のネットワークを作成します。

ステップ 1 Cisco WAE UI から、[ネットワーク（Networks）] をクリックします。

ステップ 2 [新規ネットワークの作成（Create New Network）] をクリックします。

ステップ 3 ネットワークの名前を入力します。

（注） ネットワークモデル名は、入力後は変更できません。

ステップ 4 [ネットワークの作成（Create Network）] をクリックします。

（注） ネットワーク作成後、ネットワークのタイプは「不明」です。ネットワークを設定する必要があります。

ステップ 5 作成した新しいネットワークをクリックします。

ステップ 6 [NIMOタイプを選択（Choose NIMO Type）] をクリックし、ドロップダウンリストから NIMO を選択します。[次へ（Next）] をクリックします。

ステップ 7 [コレクタ（Collector）] アイコンをクリックして、収集を設定します。

ステップ 8 適切な設定の詳細を入力します。フィールドの説明を表示するには、各フィールドの上にマウスポインタを合わせます。

ステップ 9 [保存（Save）] をクリックします。メインのネットワーク モデル ウィンドウに戻ります。

ステップ 10 設定をアーカイブするには、[設定のアーカイブ（Archive Config）] をクリックします。

a) [アーカイブパス（Archive Path）] を入力します。
b) [NetIntテーブルを含める（Include NetInt Tables）] フィールドで、[true] または [false] を選択します。
c) クリーンアップアクションの値を入力します。
d) [保存（Save）] をクリックします。

（注） 異なるネットワークアーカイブには異なるディレクトリを使用します。複数のアーカイブに同じディレクトリを使用すると、プランファイルが失われたり破損したりする恐れがあります。

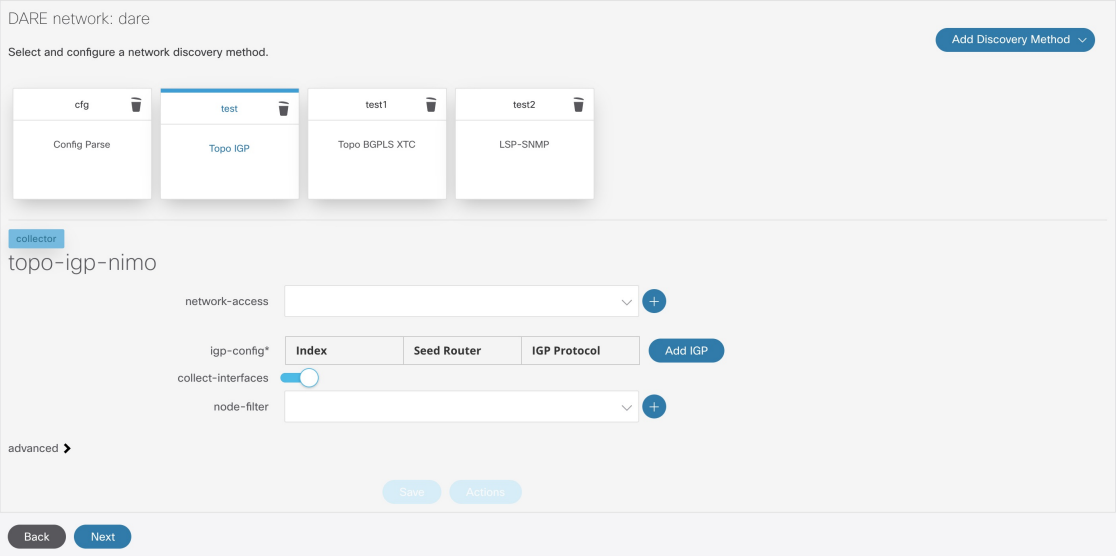
ステップ 11 コレクタアイコンをクリックし、[アクション（Action）] をクリックします。

ステップ 12 NIMO 収集を開始するボタンをクリックします（通常は「run-collection」）。

ネットワーク モデル コンポーザ を使用

ネットワーク モデル コンポーザ では、ネットワークモデル設定の複雑さが見えなくなります。さまざまなNIMOを使用したネットワークモデルの作成から、収集を実行するためのスケジュールの設定、ネットワークモデルのプランファイルを保存するためのアーカイブの設定まで、視覚的なワークフローを提供します。

ネットワーク モデル コンポーザ では、次の一般的な制御が提供されます。



画面の右側にある[追加（Add）]セクションでは、ネットワークモデル、エージェント、NIMO、スケジュールされたタスク、またはアーカイブの作成プロセスが開始されます。

ページの上部にある番号付きのナビゲーションには、ネットワークモデル設定プロセスの現在位置が表示されます。各ステップを完了するたびに、スキップしたり戻ったりしたいステップをクリックできるようになります。

設定中のネットワークモデル：収集（NIMO）、スケジュールされたタスク、またはアーカイブが左側の領域に表示され、その後に選択したネットワークモデル用に作成された設定済みコンポーネント（NIMO、スケジュールされたタスク、またはアーカイブ）が続きます。

選択したコンポーネントの設定オプションが下部に表示されます。

ネットワークの作成とトポロジ収集の構成

完全なネットワークモデルを構成する最初の手順は、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。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 （63 ページ）を参照してください。

（注） ネットワーク モデル コンポーザを使用する前に、Cisco WAE UI を使用したネットワークモデルの構成 （12 ページ）の説明に従って、ネットワークアクセスと必要なエージェントを設定することをお勧めします。ただし、 ネットワーク モデル コンポーザでネットワークアクセスとエージェントを設定する機会があります。

ステップ 1 Cisco WAE UI から、ネットワーク モデル コンポーザをクリックします。

ステップ 2 [+新しいネットワークの作成（+ Create New Network）] をクリックします。

ステップ 3 ネットワークモデル名を入力し、[ネットワークの作成（Create Network）] をクリックします。

（注） ネットワークモデル名は、入力後は変更できません。

ステップ 4 [検出方法の追加（Add Discovery Method）] をクリックします。

ステップ 5 コレクタの [名前（Name）] を入力し、[タイプ（Type）] ドロップダウンから NIMO を選択します。

既存のネットワークから選択するには、[既存のネットワーク（Existing network）] チェックボックスをオンにして、ドロップダウンから [名前（Name）] と [タイプ（Type）] を選択します。

（注） コレクタの名前にスペースを含めることはできません。

ステップ 6 **[+ Add]** をクリックします。

ステップ 7 トポロジアイコン（Topo IGP または Topo BGPLS XTC または SR トラフィックマトリックス）をクリックして、収集を設定します。

ステップ 8 適切な設定の詳細を入力します。フィールドの説明を表示するには、各フィールドの上にマウスポインタを合わせます。

ステップ 9 ドロップダウンから [node-filter] を選択します。node-filter を定義していない場合は、[+] をクリックして詳細を入力します。[includeフィルタ（Include Filter）]、[excludeフィルタ（Exclude Filter）]、[ignoreフィルタ（Ignore Filter）] を選択します。Cisco WAE UI を使用したノードフィルタの設定 （16 ページ）を参照してください

ステップ 10 [保存（Save）] をクリックします。メインのネットワーク モデル ウィンドウに戻ります。

ステップ 11 トポロジアイコン（Topo IGP または Topo BGPLS XTC）を再度クリックします。

ステップ 12 [アクション（Actions）] をクリックし、次のいずれかを選択します。

- [run collection] または [run-xtc-collection]：収集を開始します。
- [ノードリストの更新（Update Node List）]：既存のノードを削除、追加、または編集できます。
- [完了（Done）]：前のウィンドウに戻ります。

ステップ 13 [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、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 コンポーザ を使用した追加 NIMO の構成

このトピックでは、ネットワーク モデル コンポーザ を使用して追加の 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 （59 ページ）を参照してください。

（注）
- NIMO を入力または選択するように求められた場合、「コレクタ」という用語が ネットワーク モデル コンポーザ に表示されます。ネットワーク モデル コンポーザ では、NIMO とコレクタという用語は同じ意味で使用されます。
- この手順では、追加のトポロジNIMO を設定します。非トポロジNIMO（レイアウト、インベントリ、デマンド推論など）は、トポロジ集約後に external-executable-nimo で設定することをお勧めします。

始める前に

作業しているネットワークモデルに基本トポロジNIMO が設定されていることを確認します。

ステップ 1 Cisco WAE UI で ネットワーク モデル コンポーザ をクリックし、NIMO を設定するネットワークモデルをクリックします。

ステップ 2 [検出方法の追加（Add Discovery Method）] をクリックします。

ステップ 3 コレクタの [名前（Name）] を入力し、[タイプ（Type）] ドロップダウンから NIMO を選択します。

既存のネットワークから選択するには、[既存のネットワーク（Existing network）] チェックボックスをオンにして、ドロップダウンから [名前（Name）] と [タイプ（Type）] を選択します。

（注） コレクタの名前にスペースを含めることはできません。

ステップ 4 **[+ Add]** をクリックします。

ステップ 5 [コレクタ（Collector）] アイコンをクリックして、収集を設定します。

ステップ 6 適切な設定の詳細を入力します。フィールドの説明を表示するには、各フィールドの上にマウスポインタを合わせます。

（注） ネットワーク インターフェイス モジュール（NIMO） （59 ページ）トピックを参照することもできます。このトピックは、NIMO および関連する構成オプションにリンクしています。

ステップ 7 [保存（Save）] をクリックします。メインのネットワーク モデル ウィンドウに戻ります。

ステップ 8 コレクタアイコンをクリックし、[アクション（Action）] をクリックします。

ステップ 9 NIMO 収集を開始するボタンをクリックします（通常は「run-collection」）。

次のタスク

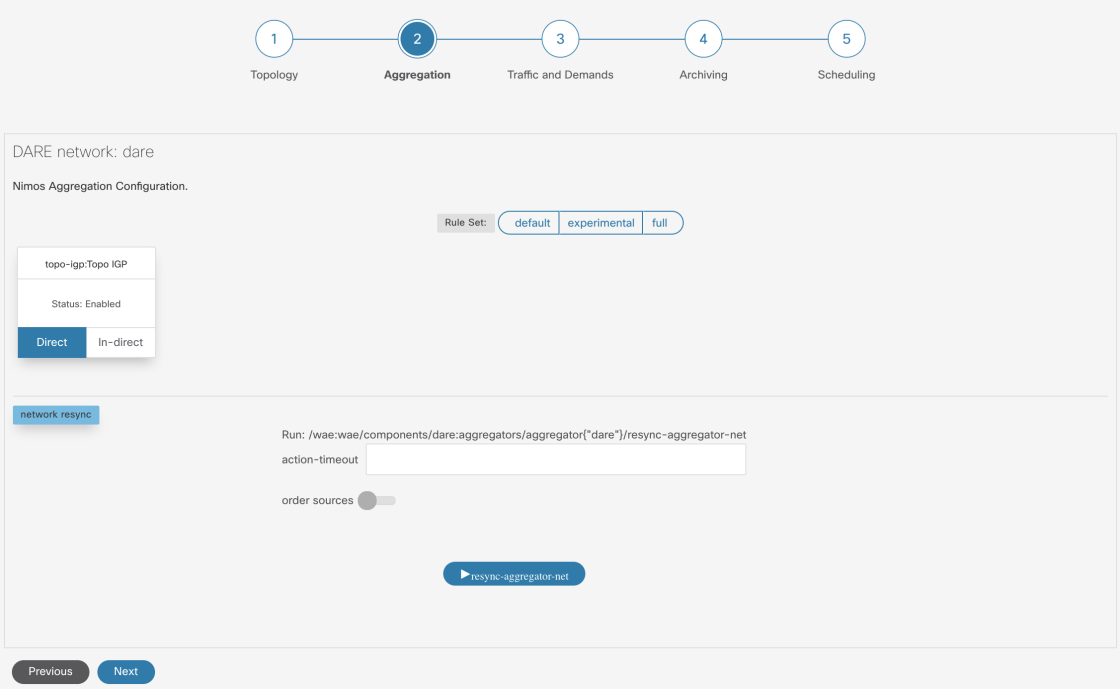
次を実行できます。

- より多くの収集を設定して実行する。
- ネットワーク モデル コンポーザ を使用した NIMO 収集の統合

ネットワーク モデル コンポーザ を使用した NIMO 収集の統合

複数の NIMO の設定後、すべての収集モデルを統合して、完全なネットワークモデルを構築する必要があります。NIMO の集約後、トラフィック統計（traffic-poll-nimo）を収集し、ネットワークモデルに対してカスタムスクリプト（external-executable-nimo）を実行できます。

図 *1*：ネットワークモデルコンポーザ： 集約



ステップ 1 ネットワーク モデル コンポーザ をクリックしてネットワークモデルを選択します。

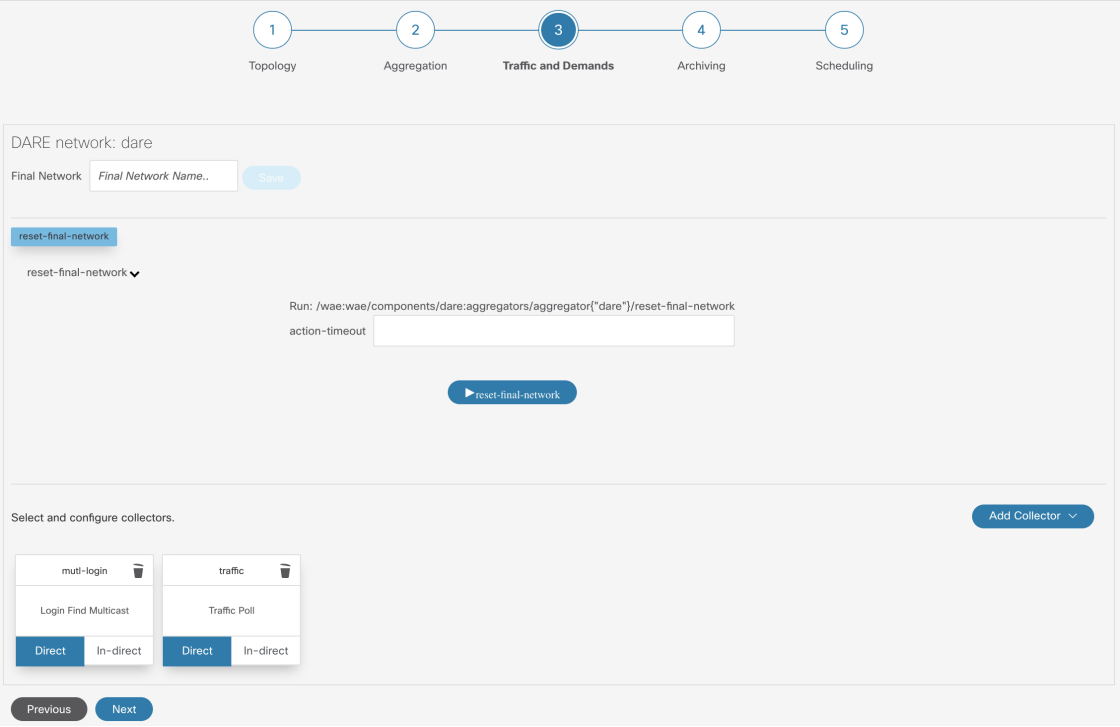
ステップ 2 上部のナビゲーションバーから [集約（Aggregation）] アイコンをクリックします。[集約（Aggregation）] ページは上記のようになります。

ステップ 3 デフォルトでは、すべての NIMO が集約に含まれます。NIMO を集約から除外するには、[間接（Indirect）] をクリックします。その収集モデルに対する変更は、集約中には含まれません。

ステップ 4 [ネットワーク再構築（network rebuild）] セクションでは、送信元の順序の変更を選択できます。[送信元の順序（order source）] を有効にし、矢印を使用して [順序指定済み送信元（Ordered Sources）] リストから順序を変更します。集約の詳細については、NIMO 収集の統合 （67 ページ） トピックを参照してください。

ネットワーク モデル コンポーザを使用したトラフィック収集またはカスタムスクリプトの実行

図 *2:* ネットワーク モデル コンポーザ：トラフィックと需要



始める前に

Cisco WAE UI を使用したネットワークモデルの構成 （12 ページ）で説明されている準備タスクを完了し、収集モデルを集約したことを確認します。

ステップ 1 ネットワーク モデル コンポーザ をクリックしてネットワークモデルを選択します。

ステップ 2 [トラフィックと需要（Traffic and Demands）] をクリックします。

ステップ 3 （オプション）[最終ネットワーク（Final Network）] フィールドに、最終的な SAGE ネットワークを設定するための名前を入力し、[保存（Save）] をクリックします。

SAGE ネットワークを設定しない場合は、[最終ネットワーク（Final Network）] フィールドを空白のままにします。

（注） 最終ネットワークをリセットするには、[reset- final-network] をクリックします。

ステップ 4 [コレクタの追加（Add Collector）] をクリックします。

ステップ 5 コレクタの [名前（Name）] を入力し、[タイプ（Type）] ドロップダウンから NIMO を選択します。

既存のネットワークから選択するには、[既存のネットワーク（Existing network）] チェックボックスをオンにして、ドロップダウンから [名前（Name）] と [タイプ（Type）] を選択します。

（注） コレクタの名前にスペースを含めることはできません。

ステップ 6 [Add] をクリックします。新しいコレクタが表示されます。

ステップ 7 作成したコレクタをクリックします。

ステップ 8 適切な設定の詳細を入力します。フィールドの説明を表示するには、各フィールドの上にマウスポインタを合わせます。

ステップ 9 [保存（Save）] をクリックします。メインのネットワーク モデル ウィンドウに戻ります。

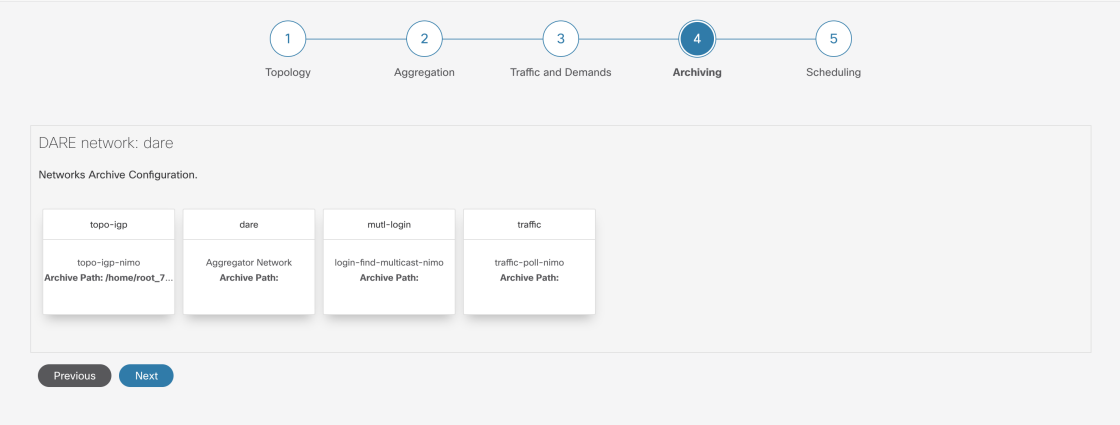
ネットワークモデルコンポーザを使用してアーカイブを設定します。

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。WAE CLI を使用したアーカイブの構成（56 ページ）も参照してください。Cisco WAE CLI を使用してアーカイブを設定する方法について説明しています。

（注） アーカイブをスケジュールするには、ネットワーク モデル コンポーザ を使用したジョブのスケジュール （24 ページ）を参照してください。

図 3：ネットワーク モデル コンポーザ：アーカイブ



ステップ 1 ネットワーク モデル コンポーザ をクリックしてネットワークモデルを選択します。

ステップ 2 [アーカイブ（Archiving）] をクリックします。

ステップ 3 アーカイブを設定する NIMO またはネットワークモデルをクリックします。

ステップ 4 ネットワークが最初に作成されたときに設定した場合は、アーカイブのパス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 5 アーカイブが取得される送信元を選択します。

ステップ 6 [クリーンアップ（Cleanup）]セクションで、[有効化（Enable）]フィールドの値を選択し、[保持日数（Retain Number of Days）] の値を入力します。

ステップ 7 [保存（Save）] をクリックします。

ステップ 8 [有効化（Enable）] をクリックして、アーカイブを有効にします。

（注） アーカイブは、有効にする前に設定する必要があります。

ネットワーク モデル コンポーザ を使用したジョブのスケジュール

この手順では、ネットワーク モデル コンポーザを使用して実行する、さまざまなネットワーク収集とエージェントをスケジュール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て設定可能な追加のスケジュールジョブと設定オプションの詳細については、スケジューラ構成 （153 ページ） を参照してください。

図 *4:* ネットワーク モデル コンポーザ：スケジュール

ステップ 1 ネットワーク モデル コンポーザ をクリックしてネットワークモデルを選択します。

ステップ 2 [スケジュール（Scheduling）] をクリックします。

WAE UI からもスケジュールされたタスクを直接作成できます。Cisco WAE UI から、[グローバルスケジューラ（Global Scheduler）] を選択し、[+スケジュールされたタスクの作成（+Create Scheduled Task）] をクリックします。スケジューラの名前を入力し、[スケジュールされたタスクの作成（Create Scheduled Task）] をクリックします。作成した新しいスケジューラをクリックし、必要な詳細を入力します。

[グローバルスケジューラ（Global Scheduler）] ページには、ネットワークに関係なく、設定されているすべてのスケジューラが一覧表示されます。

ステップ 3 [スケジューラの追加（Add Scheduler）] をクリックします。

ステップ 4 スケジュールするジョブの名前を入力します。

ステップ 5 [Add] をクリックします。

ステップ 6 [スケジュールするジョブ（scheduling job）] アイコンをクリックします。

ステップ 7 アクションを追加するには、  をクリックしてアクション名を入力します。

ステップ 8 [オーダー（order）] フィールドにオーダー番号を入力します。

ステップ 9 アクションを NIMO またはエージェント、または集約ネットワークのどれで実行するかを選選択します。

ステップ 10 ドロップダウンリストから NIMO またはエージェント、または集約ネットワークを選択します。

ステップ 11 action-type および path フィールドに値が入力されていない場合は、適切な値を入力します。

ステップ 12 [保存（Save）] をクリックします。

ステップ 13 （オプション） アクションをさらに追加します。

ステップ 14 トリガーを追加するには、  をクリックしてトリガー名を入力します。

ステップ 15 [すべて（Every）] を選択します。[実行間隔（Run Every）] フィールドに時間間隔を入力し、適切な時間単位を選択します。

cron 設定に精通している場合は、[Cron式（Cron Expression）] を選択し、アクションを実行する時間間隔を設定します。

ステップ 16 [保存（Save）] をクリックします。

ステップ 17 [タスクの実行（Run Task）] をクリックします。

（注） 各アクションは、リストおよび設定された順序で実行されます。

プランファイルのダウンロード

ネットワークモデルは、ダウンロード可能なプランファイル（.pln 形式）で保存されます。

始める前に

アーカイブが設定されていることを確認してください。

ステップ 1 Cisco WAE UI から、[グローバルアーカイブ設定（Global Archive Configuration）] をクリックします。

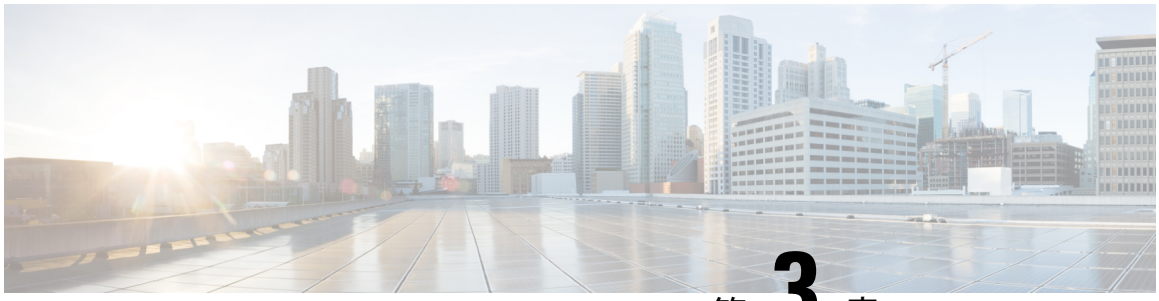
ステップ 2 [タイムゾーン（TimeZone）] オプションを使用して、デフォルト値をローカルタイムゾーンに変更します。+530、-7 などです。[タイムゾーンの追加（Add Time Zone）] をクリックします。

ステップ 3 アーカイブ用に設定したネットワークをクリックします。

ステップ 4 カレンダーの月表示が開きます。プランファイルをダウンロードする日付をクリックします。

ステップ 5 アーカイブされている .pln ファイルのリストとともに日付ビューが開きます。

ステップ 6 .pln ファイルをクリックして、ファイルをダウンロードします。



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 （27 ページ）
- ナビゲーションとコミット （28 ページ）
- 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 （29 ページ）
- WAE エキスパートモードを使用したアーカイブの構成とプランファイルの表示 （37 ページ）

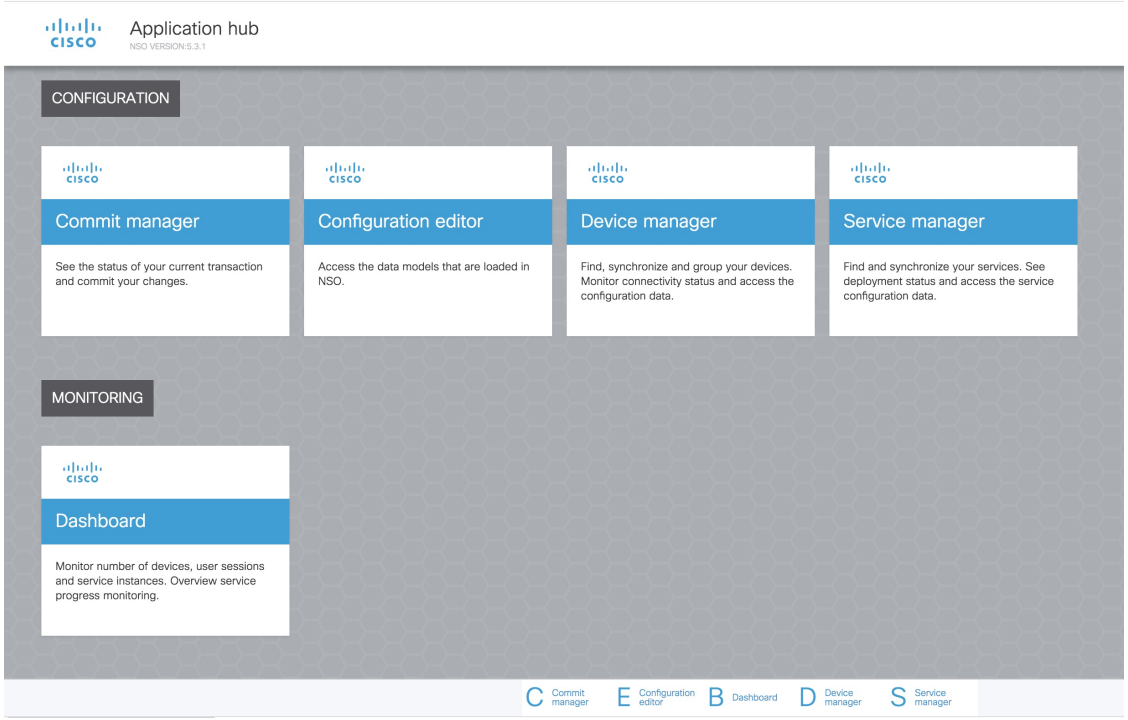
エキスパートモードの概要

エキスパート モードブラウザは、WAE UI では利用できない可能性のある追加のデバイスとサービス機能を備えています。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。

Cisco WAE UI（https://server-ip:8443）の右上隅で、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。



[設定エディタ（Configuration editor）] を使用して、データモデルにアクセスします。[モジュール（Modules）] タブで、次をクリックします。

- wae:wae：グローバル設定とエージェントを構成します。
- wae:networks：ネットワーク設定と NIMO を構成します。

[コミットマネージャ（Commit manager）] を使用して、変更をコミットします。[コミット（Commit）] ボタンをクリックして、構成に加えた変更を保存します。

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、[コミットマネージャ（Commit manager）] に移動し、[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（209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 （30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 （30 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 （31 ページ） • エキスパートモードを使用した構成解析エージェントの構成 （34 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 （67 ページ）」を参照してください。	ネットワークモデルの作成 （35 ページ）
5. 追加のデータ収集を構成します。	追加 NIMO の構成 （36 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 （153 ページ）
7. プランアーカイブを構成して表示します。	WAE エキスパートモードを使用したアーカイブの構成とプランファイルの表示 （37 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設定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) [設定エディタ（Configuration editor）] で、[/ncs:devices] に移動し、[認証グループ（authgroups）] タブをクリックします。
b) [グループ（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 を設定する場合は、[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Cisco WAE UI を使用したネットワークアクセスの設定 （13 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) [コミットマネージャ（Commit manager）] に移動し、[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 （30 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 [設定エディタ（Configuration editor）] で、[/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。

以前に設定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 （30 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、[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 [コミットマネージャ（Commit manager）] に移動し、[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。

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、[設定エディタ（Configuration editor）] で [wae:wae] に移動し、[エージェント（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（30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。

（注） WAE XTC エージェントは、正常に再起動した後でも、バックエンドで更新を続けます。XTC エージェントの更新のステータスは、受信したデータと XTC エージェントのステータスからのデータレポートフラグから確認できます。0 より大きい値は、エージェントの更新が完了したことを意味します。

次のタスク

XTC を使用するネットワークのコレクションを構成します。詳細については、NIMO の説明（59 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 NetFlow エージェントの構成

NetFlow エージェントは、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します。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。

ステップ 1 エキスパートモードから、[設定エディタ（Configuration editor）] で [wae:wae] に移動し、[エージェント（agents）] タブをクリックします。

ステップ 2 [netflow] をクリックします。

ステップ 3 [モード（mode）] ドロップダウンリストから動作モードを選択します。[シングル（single）] または [コントローラとプロセッサ（controller and processor）] のいずれかを選択します。CNF 収集にシングルモードを選択します。

ステップ 4 [設定（config）] タブで、[コントローラ（controller）]、[プロセッサ（processor）]、[jms] および [共通（common）] の詳細をすべて入力します。

選択した動作モードに基づいて、コントローラノード、プロセッサノード、jms および共通フォルダが表示されます。

（注） CNF の場合、モードはシングルですが、コントローラノード、プロセッサノードを設定する必要があります。[コントローラ（Controller）] タブで `cluster-config-file-path` に有効な json ファイルを指定し、[プロセッサ（Processor）] タブで json ファイル内のエントリと一致する有効な `service-instance-id` を指定します。

ステップ 5 [確定する（Commit）] をクリックします。

ステップ 6 [netflow] タブで、**[start]** > **[startの呼び出し（Invoke start）]** をクリックして収集を開始します。

ステップ 7 収集を停止するには、**[stop]** > **[stopの呼び出し（Invoke stop）]** をクリックします。

ステップ 8 **[status]** > **[statusの呼び出し（Invoke status）]** を実行して、CLI を使用してエージェントのステータスを要求します。

```
wae agents netflow status
status true
message
CLUSTER STATUS - BEGIN

    AGENT NODE - BEGIN
        cluster ID:                 wae-netflow-agent
        instance ID:                agent-single
```

（注）
- 標準の OS ターミナルを開きます。
- `waerc` ファイルがソースであることを確認します。
- 次のコマンドを実行します。

  ```
  sudo /home/wae/test/run/packages/cisco-wae-netflow-agent/priv/bin/flow_cluster_manage
  -action prepare-os-for-netflow
  ```
- wae を停止し、システムを再起動します。

これは、NetFlow 収集の最初の実行前に 1 回だけ実行する必要があります。

例

次に、エージェントの設定例を示します。

```
admin@wae# show running-config wae agents netflow
wae agents netflow mode single
wae agents netflow config controller cluster-config-file-path
/opt/wae-netflow/flow-config-single.json
wae agents netflow config processor service-instance-id agent-single
wae agents netflow config common log-level debug
```

次のタスク

NetFlow を使用するネットワークの収集を設定します。詳細については、NIMO の説明 （59 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 SNMP エージェントの構成

継続的なポーラーを使用してデータを収集するように SNMP エージェントを設定し、ネットワークモデルを構築できます。

ステップ 1 エキスパートモードから、[設定エディタ（Configuration editor）] で [wae:wae] に移動し、[エージェント（agents）] タブをクリックします。

ステップ 2 [snmp] 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 名前を入力し、[確認（confirm）] をクリックします。

ステップ 5 作成したポーラーをクリックし、詳細を入力します。

- ドロップダウンを使用して [network-access] を設定します。
- [enabled] を true に設定します。

ステップ 6 [discovery] タブをクリックします。[node-discovery] と [interface-discovery] を有効にしてポーリング期間を設定します。

ステップ 7 [run-snmp-poller] をクリックします。

エキスパートモードを使用した構成解析エージェントの構成

構成解析エージェントは、Cisco、Juniper、Huawei ルータからルータ設定ファイルを収集（run-config-get）できます。エージェントは、ルータのタイプやベンダーを判別することで設定を取得できます。設定ファイルを収集した後、ユーザーは cfg-parse-nimo を設定して、LSP、VPN、ポート、および共有リスクリンクグループ（SRLG）データを収集または解析できます。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 （35 ページ）を参照してください。

次の手順では、エキスパートモードを使用してエージェントを設定する方法について説明します。Cisco WAE UI を使用して、このエージェント（Cisco WAE UI の概要 （9 ページ））の設定もできます。

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 [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 [取得（get）] タブをクリックし、ネットワークアクセスを選択します。

ステップ 5 [確定する（Commit）] をクリックします。

ステップ 6 [cfg-parse] タブに戻り、[run-config-get] > [Invoke run-config-get] をクリックします。

（注） 設定は次の場所に保存されます。

```
<wae-run-dir>/data/agents/cfg-parse/<agent-name>
```

例：

```
admin@wae# show running-config wae agents cfg-parse cfg-parse pc-test
wae agents cfg-parse cfg-parse pc-test
 source-network topo-network
 network-access test-net-access
!
```

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 • インターフェースのキャパシティ（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） • LSP と LSP パス • 名前付きパスと名前付きパスホップ • セグメントリストとセグメントリストホップ

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設定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（30 ページ）およびネットワーク アクセスの設定 （30 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 （31 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

ステップ 5 [選択 - nimo-type（Choice - nimo-type）] ドロップダウンリストから、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。オプションの詳細については、IGP データベースを使用したトポロジ収集 （63 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、設定されたエージェントが必要です。詳細については、XTC を使用したトポロジ収集 （64 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 （31 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 [コミットマネージャ（Commit manager）] に移動し、[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ソースネットワークとして使用して、追加のネットワークコレクションを構成します。詳細については、NIMO の説明 （59 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 （59 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 （35 ページ）」を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks/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 [コミットマネージャ（Commit manager）] に移動し、[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。

WAEエキスパートモードを使用したアーカイブの構成と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。WAE CLI を使用したアーカイブの構成（56 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 エキスパートモードから、[設定エディタ（Configuration editor）] で、[/wae:networks/network/<*network_model_name*>] に移動して、[プランアーカイブ（plan-archive）] タブをクリックします。

ステップ 2 アーカイブディレクトリを入力します。

ステップ 3 [コミットマネージャ（Commit manager）] に移動し、[コミット（Commit）] をクリックします。

ステップ 4 現在のネットワークモデルを、指定したアーカイブディレクトリのプランファイルに保存するには、[実行（run）] をクリックします。

ステップ 5 プランファイルを取得するには：

a) [get] をクリックします。

b) タイムスタンプとプラン形式を入力します。

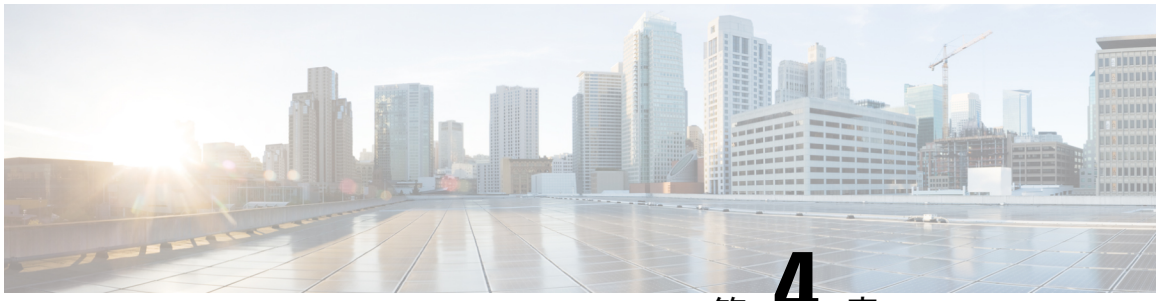
フィールドにカーソルを合わせると、予想される形式が表示されます。

c) [getを呼び出し（Invoke get）] をクリックします。

取得される出力はエンコードされた文字列です。WAE Design GUI からファイルを開きます。

次のタスク

WAE Design GUI から、WAE Archive にあるプランファイルを開くことができます。



第 4 章

ネットワークモデルの構成：Cisco WAE CLI

ここでは、次の内容について説明します。

- WAE CLI の概要 （39 ページ）
- エキスパートモードと WAE CLI の比較 （50 ページ）
- WAE CLI を使用したネットワークモデルの構成 （52 ページ）

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

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)	保留中のコミット確認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、保留中のコミット確認は中止されます。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 （209 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. エキスパートモードから、[設定エディタ（Configuration editor）] で、[/ncs:devices] に移動し、[認証グループ（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**。	`# 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. エキスパートモードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動し、プラス（[+]）記号をクリックして、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># 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. エキスパートモードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動し、プラス（[+]）記号をクリックして、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># networks network as54001 nimo
aggregator sources as54001_topo
networks network as54001 nimo
aggregator sources as54001_xtclsp
networks network as54001 nimo
aggregator sources as54001_conflsp
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 （52 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 （53 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 （31 ページ） • エキスパートモードを使用した構成解析エージェントの構成 （34 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 （54 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 （56 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 （153 ページ）
7. プランアーカイブを構成して表示します。	WAE CLI を使用したアーカイブの構成 （56 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。

```
# 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。

```
# 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。

```
# 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
```

（注） no-auth-no-privを使用して認証と暗号化を選択するオプションがありますが、これらの値はバックエンドでは使用されず、オプションです。

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

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

user@wae(config)# 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)# 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)# 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 （52 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。

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

```
# 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>
```

（注） ノードフィルタは ip-manage では機能しません。

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。

```
# wae nimos network-access network-access as64001 auth-group ABCgroup
# wae nimos network-access network-access as64001 snmp-group snmp_v3_01
# wae nimos network-access network-access as64001 node-access 10.1.1.1 ip-manage
10.18.20.121
# wae nimos network-access network-access as64001 node-access 10.2.2.2 ip-manage
10.18.20.122
# commit

# wae nimos network-access network-access netaccess_01 auth-group ABCgroup
# 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 データベースを使用したトポロジ収集 （63 ページ）およびXTC を使用したトポロジ収集 （64 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 （55 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 （31 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ソースネットワークとして使用して、追加のネットワークコレクションを構成します。詳細については、NIMO の説明 （59 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 （59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>
```

WAE CLI を使用したアーカイブの構成

ネットワーク モデル コンポーザ を使用してアーカイブを設定します。 （23ページ）も可能です。

ステップ 1 WAE CLI を起動し、コンフィギュレーション モードに移行します。

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

ステップ 2 アーカイブディレクトリを設定し、ファイルからアーカイブデータを取得するかどうかを選択します。

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

次に例を示します。

```
(config)# networks network Network_123 plan-archive archive-dir /archive/planfiles/Network_123
(config)# networks network <network_model_name> plan-archive source filename
(config)# commit
```

ステップ 3 アーカイブを実行します。アーカイブを実行すると、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、WAE CLI を使用したアーカイブの構成 （56 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

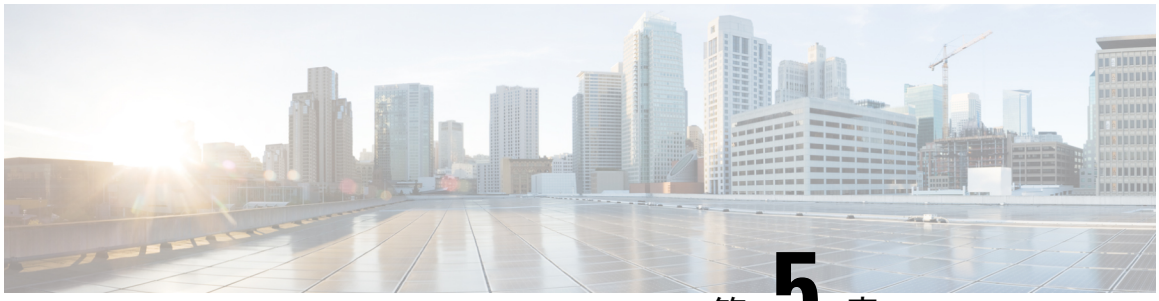
- ネットワークモデルをアーカイブに保存するには：

  (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 （55 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 （59 ページ）
- 基本的なトポロジ収集 （63 ページ）
- NIMO 収集の統合 （67 ページ）
- セグメント ルーティング トラフィック マトリックス収集 （71 ページ）
- VPN 収集 （72 ページ）
- LSP 構成の収集 （72 ページ）
- XTC を使用した LSP 収集 （74 ページ）
- 構成解析を使用したポート、LSP、SRLG、および VPN 収集 （75 ページ）
- BGP ピア収集 （77 ページ）
- SNMP を使用した LSP 収集 （79 ページ）
- インベントリ収集 （80 ページ）
- トラフィック収集 （88 ページ）
- ネットワークモデルのレイアウト（可視化） （93 ページ）
- マルチキャストフローデータの収集 （94 ページ）
- トラフィック需要の収集 （96 ページ）
- AS プランファイルのマージ （97 ページ）
- ネットワークモデルに対する外部スクリプトの実行 （98 ページ）

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 （67 ページ）を参照してください。

収集または機能	NIMO	説明	前提条件/注意事項
ネットワーク収集 NIMO			
IGP データベースを使用したトポロジ収集（63 ページ）	topo-igp-nimo	ログインと SNMP を使用して IGP トポロジを検出します。	これは、基本的なトポロジ収集（トポロジ NIMO）です。結果として得られるネットワークモデルは、他の NIMO の送信元ネットワークとして使用されます。
XTC を使用したトポロジ収集 （64 ページ）	topo-bgpls-xtc-nimo	XTC 経由で BGP-LS を使用してレイヤ 3 トポロジを検出します。トポロジの送信元として生の XTC データを使用します。ノードおよびインターフェース/ポートのプロパティは、SNMP を使用して検出されます。	• 収集を実行する前に、XTC エージェントを設定する必要があります。「エキスパートモードを使用した XTC エージェントの構成（31 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 （72 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 （77 ページ）	topo-bgp-nimo	ログインと SNMP を使用して BGP ピアリングを検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。

収集または機能	NIMO	説明	前提条件/注意事項
構成解析を使用したポート、LSP、SRLG、および VPN 収集 （75 ページ）	cfg-parse-nimo	ネットワーク内のルータ設定から情報を検出して解析します。	• 基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。 • 収集を実行する前に、構成解析エージェントを設定する必要があります。エキスパートモードを使用した構成解析エージェントの構成 （34 ページ） を参照してください。
マルチレイヤ（L3-L1）収集 （103 ページ）	光学ニモ	最終的なネットワーク収集は、他の NIMO と連携してレイヤ 1（光）およびレイヤ 3 トポロジを検出します。	optical-nimo を設定する前に実行する必要がある設定があります。エキスパートモード：マルチレイヤ収集 （104 ページ） を参照してください。
LSP 構成の収集 （72 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 （79 ページ）	lsp-snmp-nimo	SNMP を使用して LSP を検出します。	基本的なトポロジ収集を備えたネットワークモデルが存在する必要があります。
XTC を使用した LSP 収集 （74 ページ）	lsp-pcep-xtc-nimo	XTC を使用して PCEP LSP を検出します。	収集を実行する前に、XTC を使用したトポロジ収集 （64 ページ） を完了する必要があります。
セグメント ルーティング トラフィック マトリックス収集 （71 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 （67 ページ）	—	さまざまな NIMO 情報を単一の統合ネットワークモデルに集約します。	1 つの最終的なネットワークモデルにマージすべき情報が含まれた設定済みのネットワークモデルです。

収集または機能	NIMO	説明	前提条件/注意事項
AS プランファイルのマージ （97 ページ）	inter-as-nimo	さまざまな自律システム（AS）からの計画ファイルは、inter-as-nimo を使用してマージできます。inter-as-nimo は、計画ファイル間の競合を解決します。ネイティブ形式の計画ファイルもサポートされています。	• マージする個々の AS ネットワークモデルで収集が完了していることを確認してください。 • topo-bgpls-xtc NIMO を使用する AS ネットワークモデルには、それぞれ自律システム番号（ASN）が割り当てられている必要があります。
追加の NIMO			
トラフィック収集（88 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（79 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 （72 ページ） を参照してください。
ネットワークモデルのレイアウト（可視化）（93 ページ）	layout-nimo	送信元モデルにレイアウトプロパティを追加して、視覚化を改善します。	• 統合されたネットワークモデルです。 • layout-nimo が設定されたら、レイアウトのプロパティを含む計画ファイルを layout-nimo モデルにインポートして戻す必要があります。
ネットワークモデルに対する外部スクリプトの実行 （98 ページ）	external-executable-nimo	カスタマイズされたスクリプトを実行して、送信元ネットワークモデルに追加データを付加します。	送信元ネットワークモデルとカスタムスクリプトです。

基本的なトポロジ収集

基本的なトポロジ収集（トポロジNIMO）から得られるネットワークモデルは、追加のデータ収集のソースネットワークとして使用されます。トポロジ収集とその他のデータ収集を統合するには、収集を実行する前に、最初にアグリゲータを設定する必要があります。アグリゲータの詳細については、NIMO 収集の統合 （67 ページ）を参照してください。

IGP データベースを使用したトポロジ収集

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

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

（注）

- このトピックで説明されているタスクを実行するときは、ネットワークモデルの作成中であることを前提としています。詳細については、ネットワークモデルの作成 （35 ページ）を参照してください。
- このトピックでは、設定のためにエキスパートモードを使用する方法を示していますが、WAE UI または WAE CLI を使用してオプションを設定する場合にも参照できます。

始める前に

デバイスおよびネットワーク アクセス プロファイルを設定する必要があります。ネットワーク アクセスの設定 （30 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 プロファイルを選択します。

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

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

ステップ 5 プラス（+）記号をクリックして IGP を追加します。

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

a) シードルータの管理 IP アドレスを入力します。

b) ネットワークで実行されている IGP プロトコルを選択します。

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

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

ステップ 8 （オプション）収集から個々のノードを除外したり含めたりするには、[node-filter] タブをクリックし、ノードフィルタを選択します。[node-filter] が定義されていない場合は、Cisco WAE UI を使用したノードフィルタの設定 （16 ページ）を参照してください。

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

ステップ 10 [コミット（Commit）] ボタンをクリックします。

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

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

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

次のタスク

このネットワークモデルを送信元ネットワークとして使用して、追加の収集を構成します。NIMO の説明 （59 ページ） を参照してください。

XTC を使用したトポロジ収集

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

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

始める前に

- デバイスアクセスとネットワークアクセスを構成する必要があります。詳細については、エキスパートモードを使用したデバイスアクセスの構成 （30 ページ）およびネットワーク アクセスの設定 （30 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 （31 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_bgpls_xtc などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

ステップ 6 次の情報を入力します。

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

（注） 詳細オプションについては、BGP-LS XTC の詳細オプション （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 7 [reactive-network] タブをクリックして XTC からの通知をサブスクライブし、ノードまたはリンクの追加または削除を更新します。次の情報を入力します。

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

ステップ 8 [コミット（Commit）] ボタンをクリックします。

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

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

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

例

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

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

次のタスク

このタスクを実行した後、このネットワークモデルをソースネットワークとして使用して、追加の収集を構成できます。詳細については、NIMOの説明 （59ページ）を参照してください。

BGP-LS XTC の詳細オプション

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

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

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

NIMO 収集の統合

アグリゲータは、デルタ集約ルールエンジン（DARE）を使用して、ユーザー指定の NIMO を単一の統合ネットワークモデルに結合します。アグリゲータは、ソース NIMO の機能を読み取ります。アグリゲータ機能の詳細については、ネットワークモデル （3 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 （143 ページ）を参照してください。
- アグリゲータの下に同じ NIMO タイプのネットワークを複数追加することはサポートされていません。外部実行可能な NIMO が設定されている場合、NIMO タイプは **aggregator/sources/source/nimo** で指定する必要があります。

始める前に

- 最終ネットワークモデルに含める NIMO を構成します。
- 最初の設定が完了するまで、収集を実行したり、これらの NIMO を実行したりしないことが重要です。DARE を設定する前に収集を実行する場合は、DARE ネットワークを最初から再構築する必要があります（[/wae:wae/components/aggregators/aggregator] <*network_name*>、次に [resync-aggregator-net] をクリックします）。
- ネットワーク モデル コンポーザ を使用すると、NIMO の集約の設定が簡素化されます。詳細については、ネットワーク モデル コンポーザ を使用 （18 ページ） および ネットワーク モデル コンポーザ を使用した NIMO 収集の統合 （21 ページ） のトピックを参照してください。

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

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

ステップ 3 プラス（[+]）記号をクリックします。

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

ステップ 5 ソースリンクをクリックします。

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

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

（注） オンデマンドの帯域幅と、帯域幅の最適化のために sr-traffic-matrix-nimo を設定します。セグメント ルーティング トラフィック マトリックス収集 （71 ページ） を参照してください。

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

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

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

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

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

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

ステップ 9 [コミット（Commit）] ボタンをクリックします。

ステップ 10 ソース NIMO を実行します。最終ネットワークモデルがソースネットワークモデルからの最新情報で更新されます。アグリゲータとマルチレイヤ収集の構成例 （69 ページ）も参照してください。

例

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

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

アグリゲータが構成されたら、ソース NIMO を実行します。

アグリゲータとマルチレイヤ収集の構成例

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

以下は、L1（optical）および L3（topo-igp-nimo）ネットワークモデルがネットワーク上に構成されていることを示しています。オプティカル NIMO および topo-igp-nimo を設定する方法の詳細については、IGP データベースを使用したトポロジ収集 （63 ページ）および EPN-M エージェントを使用したマルチレイヤ収集の設定 （106 ページ）を参照してください。

レイヤ 1 ネットワークモデル：

```
networks network l1-network
```

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

レイヤ 3 ネットワークモデル：

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

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

アグリゲータを構成します。

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

アグリゲータが構成されたら、L3 および L1 収集を実行します。

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

L1 ネットワーク収集を実行します。

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

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

セグメント ルーティング トラフィック マトリックス収集

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

始める前に

- 基本的なトポロジネットワークモデルが存在する必要があります。IGP データベースを使用したトポロジ収集 （63 ページ）またはXTC を使用したトポロジ収集 （64 ページ）を参照してください。
- テレメトリを設定する必要があります。WAE でのテレメトリの設定 （139ページ） トピックを参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_sr_traffic_matrix などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

ステップ 7 [コミット（Commit）] ボタンをクリックします。

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

例

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

```
# networks network <network-model-name> nimo sr-traffic-matrix-nimo source-network
<source-network>

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

VPN 収集

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

始める前に

ネットワークトポロジ収集が完了している必要があります。詳細については、ネットワークモデルの作成 （35 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_vpn などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

ステップ 7 [vpn-types] タブをクリックします。

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

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

ステップ 9 [コミット（Commit）] ボタンをクリックします。

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

LSP 構成の収集

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

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

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

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

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。詳細については、基本的なトポロジ収集 （63 ページ）を参照してください。

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

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

次のフィールドを設定する必要があります。

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

ステップ 2 NSO インスタンスから LSP 情報を収集します。

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

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

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

ステップ 4 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_lsp_config などです。

ステップ 5 [Add] をクリックします。

ステップ 6 [nimo] タブをクリックします。

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

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

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

ステップ 9 [コミット（Commit）] ボタンをクリックします。

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

XTC を使用した LSP 収集

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

始める前に

XTC（bgpls-xtc-nimo）を使用した BGP-LS トポロジ収集がネットワークで完了していることを確認します。LSP を収集するための送信元ネットワークとしてこのモデルを使用する必要があります。詳細については、XTC を使用したトポロジ収集 （64 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、**`networkABC_lsp_pcep_xtc`** などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

ステップ 7 [xtc-hosts] タブをクリックします。

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

- [name]：XTC ホスト名を入力します。これは任意の名前にできます。
- [xtc-host]：ドロップダウンリストから、以前に設定された XTC ホストの 1 つを選択します。詳細については、エキスパートモードを使用した XTC エージェントの構成 （31 ページ）を参照してください。

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

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

ステップ 10 [コミット（Commit）] ボタンをクリックします。

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

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

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

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

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

構成解析を使用したポート、LSP、SRLG、および VPN 収集

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

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

始める前に

- トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作成（35 ページ） を参照してください。
- 構成解析エージェントが設定され、実行されている必要があります。詳細については、エキスパートモードを使用した構成解析エージェントの構成 （34 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_port-cfg-parse などです。

ステップ 3 [Add] をクリックします。

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

ステップ 5 [NIMO] リンクをクリックして、次の情報を入力します。

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

ステップ 6 [parse] タブをクリックします。

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

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

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

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

ステップ 8 [コミット（Commit）] ボタンをクリックします。

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

次のタスク

このタスクを実行した後、このネットワークモデルを送信元ネットワークとして使用して、追加の収集を設定できます。詳細については、NIMO の説明（59 ページ）を参照してください。

BGP ピア収集

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

始める前に

トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作成（35 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_topo_bgp などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

詳細オプションについては、BGP トポロジの詳細オプション（78 ページ）を参照してください。

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

ステップ 8 [コミット（Commit）] ボタンをクリックします。

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

BGP トポロジの詳細オプション

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

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

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

SNMP を使用した LSP 収集

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

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集 （63 ページ） を参照してください。

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

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_lsp_config などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

ステップ 7 [コミット（Commit）] ボタンをクリックします。

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

インベントリ収集

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

収集されるハードウェア

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

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

ハードウェア階層

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

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

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

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

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

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

インベントリを処理するためのテーブル

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

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

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

ハードウェア テンプレートの設定

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

HWInventoryTemplates テーブル

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

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

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

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

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

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

Type,Identifier,Name

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

例：Module,0,Model

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

複数の名前ソース列をコロンで指定できます。

例：Container,0,Model:Name

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

例

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

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

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

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

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

HWInventoryTemplates エントリの追加

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

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

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

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

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

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

HWNameFormatRules テーブル

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

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

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

例：テーブルのエントリは次のように機能します。

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

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

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

モデルまたは名前によるハードウェアの除外

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

例：テーブルのエントリは次のように機能します。

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

ExcludeHWList			
HWTable	Vendor	モデル	名前

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

HardwareSpec

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

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

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

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

インベントリ収集の設定

始める前に

ネットワークトポロジ収集が完了している必要があります。詳細については、ネットワークモデルの作成 （35 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_inventory などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

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

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

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

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

ステップ 9 [コミット（Commit）] ボタンをクリックします。

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

例

WAE CLI（設定モード）：

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

auth.enc の作成

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

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

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

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

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

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

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

SNMPv2c 情報は、単一の値を使用して定義されます。

- community：デフォルトのコミュニティストリング

SNMPv3 情報は、認証と暗号化の詳細を定義します。

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

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

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

トラフィック収集

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

始める前に

この NIMO には、次のものが必要です。

- 基本的なトポロジネットワークモデル。
- VPN トラフィックを収集する場合、VPN ネットワークモデルが存在する必要があります。VPN 収集 （72 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 （79 ページ） を参照してください。
- 開いているファイルの最大数（ulimit -n）：1,000,000

制限事項

- 外部インターフェイスからのノードトラフィック情報は収集されません。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_traffic_polling などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

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

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

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （90 ページ）を参照してください。
- [qos-enabled]：キュートラフィック収集を有効にする場合は、[true] に設定します。
- [vpn-enabled]：VPN トラフィック収集を有効にする場合は、[true] に設定します。[true] に設定する場合は、ソースネットワークモデルで VPN が有効になっていることを確認します。

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

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （90 ページ）を参照してください。

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

- [enabled]：[true] に設定します。
- [period]：ポーリング期間を秒単位で入力します。60 秒から始めることをお勧めします。ポーリング期間の調整については、トラフィックポーリングの調整 （90 ページ）を参照してください。

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

ステップ 10 [コミット（Commit）] ボタンをクリックします。

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

トラフィックポーラーの詳細オプション

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

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

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

トラフィックポーリングの調整

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

トラフィックポーリングを効率的に実行するには、次の手順を実行します。

1. トラフィックポーラーのロギングレベルを 40 に設定します。

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

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

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

4. 実際の収集時間を検索します。次に例を示します。

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

ネットワークモデルのレイアウト（可視化）

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

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

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

次の 2 つの手順があります。

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

始める前に

- サーバーの Cisco WAE Design バージョンが Cisco WAE バージョンと同じかそれ以上であることを確認してください。
- 基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集（63 ページ） を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。この手順では、例として **networkABC_layout** を使用します。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

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

ステップ 7 [コミット（Commit）] ボタンをクリックします。

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

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

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

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

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

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

例

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

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

2. レイアウトを更新します。

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

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

マルチキャストフローデータの収集

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

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

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

始める前に

トポロジネットワークモデルが存在する必要があります。ネットワークモデルの作成（35ページ） を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークとNIMO名を含む一意の名前をお勧めします。たとえば、networkABC_multicast などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

ステップ 6 [NIMO] リンクをクリックして、次の情報を入力します。

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

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

ステップ 8 [コミット（Commit）] ボタンをクリックします。

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

例

WAE CLI を使用する場合は、次のコマンドを使用します。

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

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

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

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

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

トラフィック需要の収集

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

始める前に

基本的なトポロジネットワークモデルが存在する必要があります。基本的なトポロジ収集（63 ページ） を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。ソースネットワーク名と NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_traffic_demands_config などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

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

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

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

ステップ 10 ツールをクリックし、必要な詳細を入力します。

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

設定にさらに手順を追加するには、手順 9 ～ 11 を繰り返します。

ステップ 12 [コミット（Commit）] ボタンをクリックします。

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

AS プランファイルのマージ

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

各 AS は、異なる WAE サーバーに置くことができます。

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

始める前に

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

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。送信元ネットワークと NIMO 名を含む一意の名前をお勧めします。たとえば、networkABC_merge_as_plan などです。

ステップ 3 [Add] をクリックします。

ステップ 4 [nimo] タブをクリックします。

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

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

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

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

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

ステップ 9 [確定する（Commit）] をクリックします。

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

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

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

ネットワークモデルに対する外部スクリプトの実行

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

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

- インベントリ収集の設定 （85 ページ）
- ネットワークモデルのレイアウト（可視化） （93 ページ）

別の例が外部スクリプトの実行例 （100 ページ）トピックに記載されています。

始める前に

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

ステップ 1 エキスパート モード から、[設定エディタ（Configuration editor）] で、[/wae:networks] に移動します。

ステップ 2 プラス（[+]）記号をクリックして、ネットワークモデル名を入力します。

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

ステップ 3 [nimo] タブをクリックします。

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

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

ステップ 6 [advanced] タブで、以下の情報を入力します。

- [input-file-version]：送信元ネットワークモデルのプランファイルバージョンを入力します（6.3、6.4 など）。デフォルトは現在のバージョンです。
- [input-file-format]：送信元ネットワークモデルのプランファイルフォーマットを指定します。デフォルトは .pln です。
- [argv]：スクリプトの実行に必要な引数を（順番に）入力します。WAE CLI ツールのいずれかを実行している場合は、送信元ネットワークモデルに $$input、結果のネットワークモデル（スクリプトの実行後）に $$output、認証ファイルに $$authfile を入力します（ネットワークアクセスが設定されている必要があります）。$$input、$$output、およびその他の argv 引数は、スクリプトで必要な順序でリストする必要があることに注意することが重要です。例については、外部スクリプトの実行例 （100 ページ）を参照してください。

ステップ 7 [external-executable-nimo] タブで、[実行（run）] をクリックします。

例

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

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network>
advanced argv "<command[s]> <arguments>"
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
```

```
admin@wae(config-network-<network-model-name>)# exit
admin@wae(config)# exit

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

外部スクリプトの実行例

この例では、WAE CLI で external-executable-nimo を使用する方法について説明します。サンプルの Python スクリプト（ext_exe_eg.py）は、ネットワーク内のすべてのインターフェイスに「私のIGPメトリックは<*value*>（My IGP metric is <value>）」という説明を付加します。別の例については、インベントリ収集の設定 （85 ページ）トピックを参照してください。

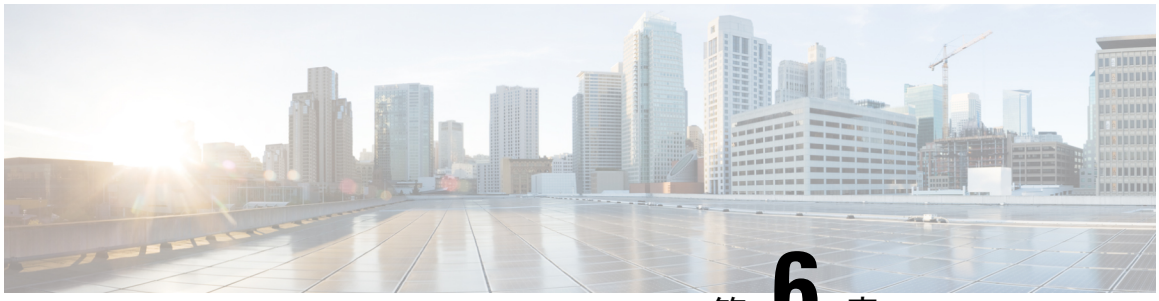
ext_exe_eg.py の内容：

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

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

srcNet =  Network(src)

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

srcNet.write(dest)
```

WAE CLI で、次のように入力します。

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

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

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



第 6 章

Cisco WAE モデリングデーモン（WMD）の構成

Cisco 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 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 Cisco WMD に送信します。Cisco 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。

（注） WMD で使用できないデータおよび過去のデータは、見出し `IgnoredTables` および `IgnoredColumns` の下にある `<run-directory>/packages/cisco-wae-modeling-daemon/priv/etc/sql-patch-config.txt` にリストされています。

DARE および WMD を設定するには、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 （101 ページ）

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。SAgE から変更を取得します。

この手順では、エキスパートモードを使用して WMD を設定する方法について説明します。一方、WAE UI を使用して、WMD を構成することもできます。Cisco WAE UI から、[WMD設定（WMD Configuration）] をクリックし、以下の情報を使用して WMD を設定します。[**保存（Save**）] をクリックして、設定を保存します。

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

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、[設定（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。

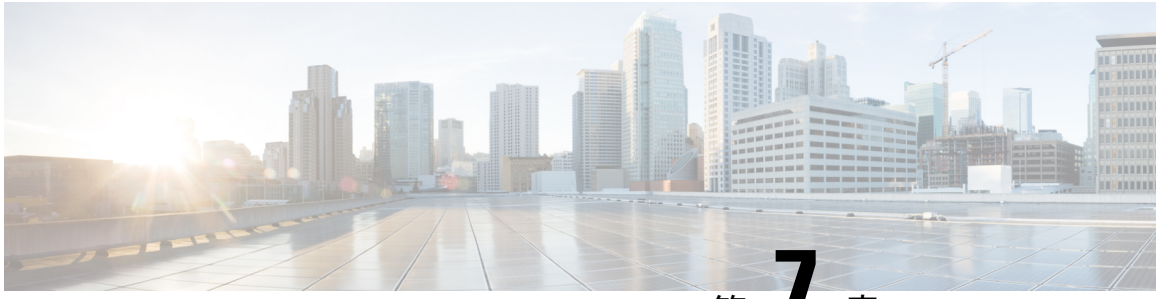
ステップ 6 [dare] をクリックして、次の値を入力します。

- [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。

例

WAE CLI（構成モード）の例：

```
# wae components wmd config network-name <final_model_name> dare dare-destination
<dare_model_name>
```



第 7 章

マルチレイヤ（L3-L1）収集

マルチレイヤ（レイヤ 3 およびレイヤ 1）ネットワーク収集は、高度な収集設定です。このセクションでは、マルチレイヤネットワークから収集を設定する方法について説明します。

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。

- WAE Design で表示およびモデル化できる Spectrum Switched Optical Network（SSON）回路情報（中心周波数 ID、スペクトル幅、sson_enabled、およびprefer_lower_frequency_ids）。L1 リンク属性は、central_frequency_excludelist_id および sson_enabled 列にも関連付けられています。
- L1 ダイバーシティ収集
- 完全な収集を実行せず、ネットワークの変更時にネットワークモデルを更新する EPN-M エージェントの通知サポート
- 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
- L1 ダイバーシティ回路。L1 回路は、他の L1 回路から切り離すように設定可能
- 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（168 ページ）を参照してください。

収集後、モデルを Cisco WAE Design またはエキスパートモードから表示できます（wae:networks/network/<*network_name*>/l1-model）。

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制限事項 （104 ページ）
- エキスパートモード：マルチレイヤ収集 （104 ページ）
- Cisco WAE UI：マルチレイヤ収集 （110 ページ）
- Cisco WAE CLI：マルチレイヤ収集 （111 ページ）
- L1 回路波長オプション （113 ページ）
- L1 回路波長ガイドライン （115 ページ）
- L1 回路波長の設定例 （115 ページ）

マルチレイヤ収集の制限事項

マルチレイヤ（L3-L1）収集には次の制限事項が存在します。

- シスコデバイスのマルチレイヤ収集は、次のプラットフォームでのみサポートされています。
  - Cisco Evolved Programmable Network Manager オプティカルエージェント（EPN-M オプティカルエージェント）（EPN-M エージェントを使用したマルチレイヤ収集の設定（106 ページ））を使用する場合、バージョン 10.9 を実行する NCS 2000 プラットフォームがサポートされます。
  - 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。
- 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ており、パスホップに対してはサポートされていません。
- WAE インスタンスごとに設定できるオプティカル NIMO は 1 つだけです。複数設定した場合、動作は保証されません。

エキスパートモード：マルチレイヤ収集

以下のトピックを使用して、エキスパートモードを使用してマルチレイヤ収集を設定します。また、Cisco WAE UI（Cisco WAE UI：マルチレイヤ収集 （110 ページ））を使用して、設定の詳細に関するこれらのトピックを使用できます。フィールドの説明を表示するには、マウスポ

インタを [エキスパートモード（Expert Mode）] または [Cisco WAE UI] のフィールドの上に置きます。

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 （104 ページ）
2. L3-L1 マッピング情報を取得して設定します。	L3-L1 マッピング情報の構成 （105 ページ）
3. Cisco Evolved Programmable Network Manager（EPNM）オプティカルエージェントを使用して、マルチレイヤ収集を設定して実行します。 EPNM エージェントは、ネットワーク内の Cisco NCS 2000 シリーズ バージョン 10.9 デバイスをサポートします。このエージェントを使用するには、ネットワークでEPNMが実行されている必要があります。	• EPN-M エージェントを使用したマルチレイヤ収集の設定 （106 ページ）

L3-L1 マッピング情報の構成

L3-L1 マッピングは、次のいずれかの方法で収集できます。

- VTXP がネットワークで有効になっている場合、追加の設定は必要ありません。
- LMP がネットワーク上で設定されている場合、LMP がネットワークで有効である場合は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。エキスパートモードを使用した構成解析エージェントの構成 （34 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 で次のように指定する必要があります。
  `networks/<multilayer_network_name>/nimo/optical-nimo/advanced/cfg-parse-agent`.

（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。

- L3-L1 マッピングを手動で設定します。
  - L1 ノードおよびポートへの、L3 ノードおよびインターフェイスのマッピングを入力します。
  - L3 から L1 への回路を指定します。この方法は、トポロジ収集後に L3 から L1 へのすべてのマッピングを検出します。

（注） L3 および L1 インターフェイスとポート、または回路名を把握している必要があります。

次の手順では、エキスパートモードを使用したL3-L1マッピングの手動設定について説明します。Cisco WAE UI も使用できます。詳細については、Cisco WAE UI の概要 （9 ページ）を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で [/wae:wae/nimos] に移動し、[l3-l1-mappings] タブをクリックします。

ステップ 2 [l3-l1-mappings] をクリックします。

ステップ 3 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。

ステップ 4 L3-L1 回路を指定する方法は次のとおりです。

a) [l3-l1-circuit-mapping] タブをクリックし、プラス（[+]）記号をクリックします。
b) L3 および L1 回線名をそれぞれ入力します。
c) [Add] をクリックします。

ステップ 5 L3-L1 インターフェイスおよびポートのマッピングを手動で入力するには、次の手順を実行します。

a) [l3-l1-mapping] タブをクリックし、プラス（[+]）記号をクリックします。
b) [Add] をクリックします。
c) これら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。

例

WAE CLI を（コンフィギュレーションモードで）使用している場合は、次のとおりです。

```
# wae nimos l3-l1-mappings l3-l1-mappings <mapping_name>
l3-l1-circuit-mapping <l3_circuit> <l1_circuit>

# commit
```

EPN-M エージェントを使用したマルチレイヤ収集の設定

Cisco Evolved Programmable Network Manager（Cisco EPN Manager）エージェントは、L1 デバイス用の Cisco Network Convergence System（NCS）2000 プラットフォームリリース 10.9 までをサポートします。このエージェントを使用するには、ネットワークで Cisco EPN Manager バージョン 2.2.2.1 が実行されている必要があります。エージェントは、リンクおよび回路のステータスが変化したときに Cisco EPN Manager からの通知も受け取り、それに応じてネットワーク

モデルを更新します。エキスパートモードを使用している場合、ノード、回路などへの変更は、パス（wae:wae/agents/optical-agent:optical-agents/optical-agent/<epnm_agent_name>）から、[エージェントモデル（agent-model）] タブで確認できます。

（注） EPNM によって報告されるさまざまなノードタイプは、デフォルトで ROADM ノードタイプに設定されます。

始める前に

。

- エキスパートモード：マルチレイヤ収集 （104 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。
- EPN-M サーバー証明書がインストールされていることを確認します。詳細については、Cisco WAE インストール ガイド の「セキュリティ」の章（「EPNM サーバーの証明書のインストール」トピック）を参照してください。
- （オプション）Cisco WAE UI またはエキスパートモードで、フィージビリティ制限マージンを設定します（**/wae:wae/components/nimos/feasibility-limit-margins/feasibility-limit-margin**）。この設定では、指定された帯域幅に一致する回路の収集されたフィージビリティ制限に、指定されたマージンが追加されます。詳細については、「L1 回路の波長」Cisco WAE Design ユーザー ガイドを参照してください。
- （オプション）Cisco WAE UI またはエキスパートモードで、中心周波数のブロックリストを設定します（**/wae:wae/components/nimos/central-frequency-excludelists/central-frequency-excludelists**）。この設定では、L1 回路パスの中心周波数 ID として機能しない可能性がある周波数 ID のリストを定義します。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。高度なオプションと設定のガイドラインについては、次のトピックを参照してください。
  - L1 回路波長オプション （113 ページ）
  - L1 回路波長ガイドライン （115 ページ）
  - L1 回路波長の設定例 （115 ページ）

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 トポロジ収集ネットワークモデルを設定し、実行します。

- lag
- get-physical-ports

（注） 詳細については、IGP データベースを使用したトポロジ収集 （63 ページ）を参照してください。

ステップ 2 [設定エディタ（Configuration editor）] で [/wae:wae/agents/optical-agents] に移動し、[オプティカルエージェント（optical-agent）] タブをクリックします。

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-WAE-Optical-EPNM-Agent] を選択します。

ステップ 5 [cisco-wae-optical-epnm-agent] リンクをクリックします。

ステップ 6 [epnm-server-conf] タブを選択し、Cisco EPN Manager サーバーの検証済みドメインとアクセスグループ（NCS デバイスで設定された認証グループ）を入力します。

（注） [詳細（advanced）] タブをクリックして、L3-L1 マッピンググループ、データ記録オプション（[net-recorder] が「記録」に設定されている場合、ファイルは [net-record-dir] 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、および [pool-size-per-query]（L1 要素ごとに EPNM に送信可能な並列クエリの数）を入力します。

WAE CLI（コンフィギュレーション モード）の例：

```
admin@wae(config)# wae agents optical-agents optical-agent
<agent-name> cisco-wae-optical-epnm-agent epnm-server-conf epnm-server-fqdn <fqdn>
epnm-server-access <authgroup>
cisco-wae-optical-epnm-agent advanced net-recorder <net-recorder-option>
cisco-wae-optical-epnm-agent advanced net-record-dir <net-recorder-storage-directory>
cisco-wae-optical-epnm-agent advanced pool-size-per-query <number-of-queries>
cisco-wae-optical-epnm-agent advanced notification subscribe-to-notifications <true or false>
```

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 L1 収集を実行するには、[cisco-wae-optical-epnm-agent] タブに戻り、**[run-collection]** > **[Invoke run-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（[+]）記号をクリックして、オプティカル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_optical などです。
c) [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 L3 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 [詳細（advanced）] タブをクリックして次の設定を行います。

- [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。

- [enable-delta-modeler]：差分モデラーを有効にするには true を選択します。true に設定すると、エージェントが受信した変更により、光トポロジが自動的に更新されます。
- [feasibility-limit-margin-list]：（オプション）フィージビリティ マージン リスト名を選択します。
- [central-frequency-excludelists]：（オプション）周波数除外リストを選択します。

ステップ 12 [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。

a) [詳細（advanced）] タブをクリックして、次を含む高度な機能を設定します。

- [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、[map-lambda-id-to] フィールドで選択された Lambda マッピング値（Lambda ID、ITU チャネル番号、G.694.1、中心主は、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。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。
- [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 （105 ページ）を参照）。
- [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。
- collect-user-properties — エージェントからユーザー プロパティを収集しない場合は false に設定します。デフォルト値は true です。

ステップ 13 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。適切なソース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構成例 （69 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。

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 14 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 15 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 ソース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 16 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）。

次のタスク

アーカイブが設定されている場合は、プランファイルを開き、WAE Design を使用して L1 および L3 トポロジも表示できます。その後、最適化ツールを実行したり、ネットワークに変更を加えたり、パッチを作成したりできます。WAE 設計の詳細については、Cisco WAE Design ユーザー ガイド を参照してください。アーカイブおよびプランファイルの詳細については、次のトピックを参照してください。

- ネットワーク モデル コンポーザ を使用してアーカイブを設定します。 （23 ページ）
- アーカイブ内のプランファイルの管理 （57 ページ）

Cisco WAE UI：マルチレイヤ収集

このワークフローでは、Cisco WAE UI を使用してマルチレイヤ収集を設定する手順の概要について説明します。設定オプションの詳細については、エキスパートモードのマルチレイヤトピックを参照するか、マウスポインタをフィールドの上に置いてください。

始める前に

マルチレイヤ収集の制限事項を確認します。

ステップ 1 ネットワークで VTXP が有効になっていない場合は、次のいずれかを実行して L3-L1 マッピングを設定します。

- LMP ネットワークの場合、構成解析エージェントを作成して設定します（**[Cisco WAE UI]** > **[エージェントの設定（Agent Configuration）]** および [cfg-parse-agent] を選択）。
- L3-L1 マッピングを情報を手動で入力します（**[Cisco WAE UI]** > **[L3-L1マッピング（L3-L1 Mapping）]**）。

ステップ 2 オプティカルエージェントを設定します（**[Cisco WAE UI]** > **[エージェントの設定（Agent Configuration）]** および [cisco-wae-optical-epnm-agent] を選択）。

ステップ 3 （オプション）L1 フィージビリティマージンを設定します（**[Cisco WAE UI]** > **[フィージビリティ制限マージン（Feasibility Limit Margins）]**）。

ステップ 4 （オプション）中心周波数のブロックリストを設定します（**[Cisco WAE UI]** > **[中心周波数除外リスト（Central Frequency Excludelists）]**）。

ステップ 5 IGP データベースを使用して L3 トポロジ収集を作成します（**[Cisco WAE UI]** > **[コンポーザ（Composer）]** > **[トポロジ（Topology）]** および [Topo IGP] を選択）。

ステップ 6 L1 トポロジ収集を作成します（**[Cisco WAE UI]** > **[コンポーザ（Composer）]** > **[トポロジ（Topology）]** および [オプティカル（Optical）] を選択）。

ステップ 7 トポロジ収集を集約します（**[Cisco WAE UI]** > **[コンポーザ（Composer）]** > **[集約（Aggregation）]**、L1 および L3 収集が [直接（Direct）] に設定されていることを確認し、[ネットワークの再構築（Rebuild Network）] をクリック）。

ステップ 8 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）。

次のタスク

アーカイブが設定されている場合は、プランファイルを開き、WAE Design を使用して L1 および L3 トポロジも表示できます。その後、最適化ツールを実行したり、ネットワークに変更を加えたり、パッチを作成したりできます。WAE 設計の詳細については、Cisco WAE Design ユーザー ガイド を参照してください。アーカイブおよびプランファイルの詳細については、次のトピックを参照してください。

Cisco WAE CLI：マルチレイヤ収集

Cisco WAE CLI を使用して（設定モードで）マルチレイヤ収集を設定する例を示します。この例には、EPNM オプティカルエージェントの設定と手動 L1-L3 マッピングが含まれています。

EPNM オプティカルエージェントの設定

```
wae@wae(config-optical-agent-<optical_agent_name>)#cisco-wae-optical-epnm-agent
epnm-server-conf epnm-server-fqdn <FQDN> epnm-server-access <epnm_auth_group>
```

ネットワークの作成

```
wae@wae(config)# networks network <l3_network_name>
wae@wae(config-network-<l3_network_name>)# nimo topo-igp-nimo network-access <access_group>
wae@wae(config-network-<l3_network_name>)# nimo topo-igp-nimo igp-config 1
igp-protocol <ospf/isis> seed-router <seed-ip>
wae@wae(config-network-<l3_network_name>)# commit
wae@wae(config-network-<l3_network_name>)# exit

wae@wae(config)# networks network <ml_network_name>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo network-access <access_group>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo source-network
<l3_network_name>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo optical-agents
<optical_agent_name>
wae@wae(config-network-<ml_network_name>)# commit
wae@wae(config-network-<ml_network_name>)# exit

wae@wae(config)# networks network <aggregator_network_name>
wae@wae(config-network-<aggregator_network_name>)# commit
```

L3 トポロジの設定

```
wae@wae(config)# networks network <network_name>
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo collect-interfaces true
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo advanced interfaces lag true
 get-physical-ports true
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo advanced igp isis-level both
 login-record-mode playback login-record-dir /home/wae/records/
wae@wae(config-network-<network_name>)# commit
```

L3-L1 マッピングの設定（マッピングごとに繰り返します）

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

注：optical-nimo の送信元ネットワークとして topo-igp-nimo ネットワークを使用します。

（オプション）Lambda マッピングの設定

次のいずれかを実行します。

- ファイルからロード：

  オプティカルエージェントが作成されたら、lambda マッピング設定 XML ファイル（lambda-id から channel-id/wavelength/central-frequency へのマッピング）を /wae/agents/optical-agents/optical-agent[agent-name]/lambda-mappings にロード（ncs_load または netconf-console --edit-config..）します。

  ```
  ncs_load -lmj <file_name>

  netconf-console --edit-config <file_name>
  ```

（注） netconf-console コマンドには、「python-paramiko」システムパッケージが必要です。

- オプティカル nimo で Lambda マッピングを有効にし、lambda マッピングのベースを選択します。

  ```
  wae@wae(config)# networks network <ml_network_name> nimo optical-nimo optical-agents

  <optical_agent_name> map-lambdas true map-lambda-id-to
  <channel-id/wavelength/central-frequency>
  wae@wae(config)# commit
  ```

（オプション）フィージビリティ制限マージンの設定

```
admin@wae(config)# networks network <ml_network_name> nimo optical-nimo optical-agents
<optical_agent_name>
admin@wae(config-optical-agents-<optical_agent_name>)# advanced
feasibility-limit-margin-list <L1_circuit_bandwidth>
feasibility-limit-margin <margin_value>
```

異なる帯域幅に複数のマージン値を設定するには、2 番目のコマンドを繰り返します。

DARE（アグリゲータ）の設定

```
admin@wae(config)# wae components aggregators aggregator <aggregator_network_name>
admin@wae(config-aggregator-<aggregator_network_name>)# sources source <l3_network_name>
admin@wae(config-source-<l3_network_name>)# exit
admin@wae(config-aggregator-<l3_network_name>)# sources source <ml_network_name>
admin@wae(config-source-<ml_network_name>)# commit
```

L3 トポロジ収集の実行

```
wae@wae# networks network <l3_network_name> nimo topo-igp-nimo run-collection
```

L1 トポロジ収集の実行

オプティカルプラグインを開始します。

```
wae@wae# wae agents optical-agents optical-agent <optical_agent_name>
cisco-wae-optical-epnm-agent run-collection
```

（注） 集約は、ネットワーク収集プロセスとしてバックグラウンドで実行されます。

プランファイルの生成

```
wae@wae# wae components getplan run network <network_name> | exclude planfile-content |
 save <path/for/plan/file.txt>
```

L1 回路波長オプション

次の表では、中心周波数に使用可能な詳細オプションについて説明します。

表 *1 : L1* 回路波長オプション

フィールド	説明
次のオプションは、**/wae:networks/network /<*network-name*>/nimo/optical-nimo:optical-nimo/advanced/network-options** で使用できます。	
anchor-frequency	THz 単位のアンカー周波数。デフォルトは 193.1 THz です
central-frequency-granularity	GHz 単位の中心周波数の粒度。デフォルトは 25 GHz です。
central-frequency-excludelists-name	central-frequency-id-excludelist テーブルで言及されている、ブロックされたリストの中心周波数 ID を設定するために与えられた名前のリスト。
frequency-id-lower-bound	周波数 ID の下限。
frequency-id-upper-bound	周波数 ID の上限。
use-pre-configured-excludelist-per-link-type	L1 リンクタイプ（80 チャネル、96 チャネル、80 + 96 チャネル、およびナイキスト）に基づいて事前定義された、周波数ブロックリスト ID のセットを使用します。デフォルトは true です。

フィールド	説明
次のオプションは、**/wae:wae/nimos /optical-nimo:central-frequency-excludelists** で使用できます。	
name	中心周波数 ID ブロックリストの名前。
type	excludelist-80-channel、96-channel、Nyquist またはOtherに関連付けられたチャネルタイプ。
frequency-id-lower-bound	チャネルタイプに関連付けられた周波数IDの下限。
frequency-id-upper-bound	チャネルタイプに関連付けられた周波数IDの上限。
central-frequency-excludelist-ids	ブロックされたリストの中心周波数IDのリスト。

周波数の下限と上限、およびブロックされたリストの値

- 80 チャネルをサポートする L1 リンクには、次のものが効果的に含まれている必要があります。
  - 下限および上限 = [-47, 113]
  - ブロックリスト ID = {-47, -45, -43, …, 113}（奇数の ID）
- 96 チャネルをサポートする L1 リンクには、次のものが効果的に含まれている必要があります。
  - 下限および上限 = [-71, 121]
  - ブロックリスト ID = {-71, -69, …, 3, …, 121}（奇数の ID）
- 96 チャネルをサポートする L1 リンクには、次のものが効果的に含まれている必要があります。
  - 下限および上限 = [-71, 121]
  - ブロックリスト ID = {-71, -69, …, 3, …, 121}（奇数の ID）
- ナイキストリンクには、次のものが効果的に含まれている必要があります。
  - 下限および上限 = [-71, 121]
  - ブロックリスト ID なし

L1 回路波長ガイドライン

次のリストでは、L1 周波数オプションを設定するときに役立つ情報を提供します。

1. アンカー周波数と中心周波数の粒度は、指定されたエージェントネットワークで一定です。
2. ネットワークごとに、グローバルアンカー周波数、中心周波数の粒度、上限と下限、およびブロックリストオプションを設定できます。デフォルト値は、アンカー周波数、中心周波数の粒度、および全体的な上限と下限にのみ使用する必要があります。
3. 事前設定されたさまざまなブロックリストが提供され、さまざまな L1 リンクタイプ（80 チャネルシステム、96 チャネルシステム、ナイキスト 96 チャネルシステム）に対応します。

（注） ブロックリストには、上限と下限が含まれます。

4. 事前設定されたブロックリストを L1 リンクに自動的に関連付けるかどうかを選択できます。これは [use-pre-configured-excludelist-per-link-type] 設定オプションで指定できます。デフォルトでは、このオプションは [true] に設定されています。
5. 事前設定されたブロックリストを編集するには、[central-frequency-excludelists] 設定オプションを使用して新しいブロックリストエントリを作成し、[boolean use-pre-configured-excludelist-per-link-type] を [false] に設定します。
6. [build-Optical-topology] アクションを実行して、ネットワークオプションとブロックリストの周波数 ID に加えられた変更をプランファイルに組み込みます。

L1 回路波長の設定例

次に、L1 周波数の設定例をいくつか示します。

1. ネットワークオプションまたは中心周波数ブロックリスト ID に固有のものは何も設定しないでください。デフォルト値を使用します。

```
(config)#  networks network <network-name> nimo optical-nimo
optical-agents <agent-name>
```

2. ネットワークで使用されるすべてのリンクタイプのカスタム中心周波数ブロックリスト名を指定します。

```
(config)# networks network <network-name> nimo optical-nimo
advanced network-options use-preconfigured-excludelist-per-link-type false
central-frequency-excludelists-name [ 80-excludelist 96-excludelist nyquist ]

wae nimos central-frequency-excludelists central-frequency-excludelist 80-excludelist

    channel-type 80-channel-system
```

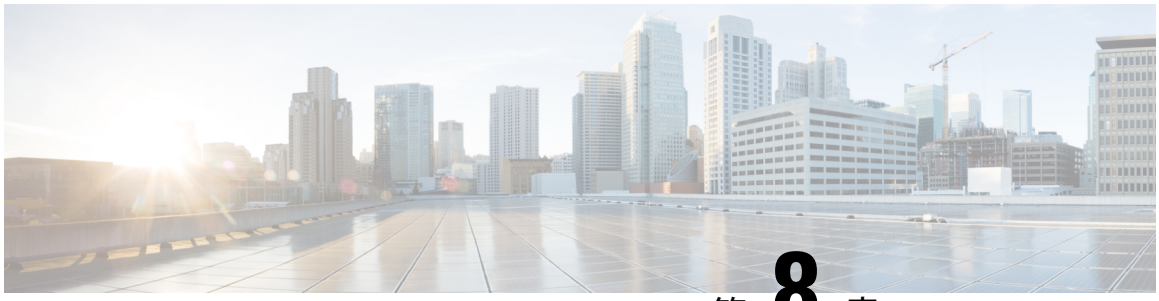
```
    id-list 1,2,3,4,5,6,7,8,9,10
    !
wae nimos central-frequency-excludelists central-frequency-excludelist 96-excludelist

    channel-type 96-channel-system
    id-list 9,11,13,45,80
!
wae nimos central-frequency-excludelists central-frequency-excludelist nyquist
    channel-type nyquist-channel-system
    id-list 5,19,76
!
```

3. ネットワークに異なるアンカー周波数と中心周波数の粒度を設定します。

```
(config)# networks network <network-name> nimo optical-nimo
advanced network-options anchor-frequency <anchor-frequency-value>
central-frequency-granularity <central-frequency-granularity-value>
```

4. デフォルトの中心周波数を指定しますが、リンクレベルのブロックリストは指定しません。

```
(config)# wae nimos central-frequency-excludelists central-frequency-excludelist
my-other
    channel-type other
    id-list 5,19,76
!networks network <nimo-name> nimo optical- nimo advanced network-options
use-preconfigured-excludelist-per-link-type false central-frequency-excludelists-name
 [ my-other ]
```



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 （117 ページ）
- NetFlow 収集アーキテクチャ （118 ページ）
- NetFlow 収集の構成 （122 ページ）
- 集中型 NetFlow 構成ワークフロー （122 ページ）
- DNF NetFlow 構成ワークフロー （129 ページ）

NetFlow データ収集

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

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

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

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

NetFlow 収集アーキテクチャ

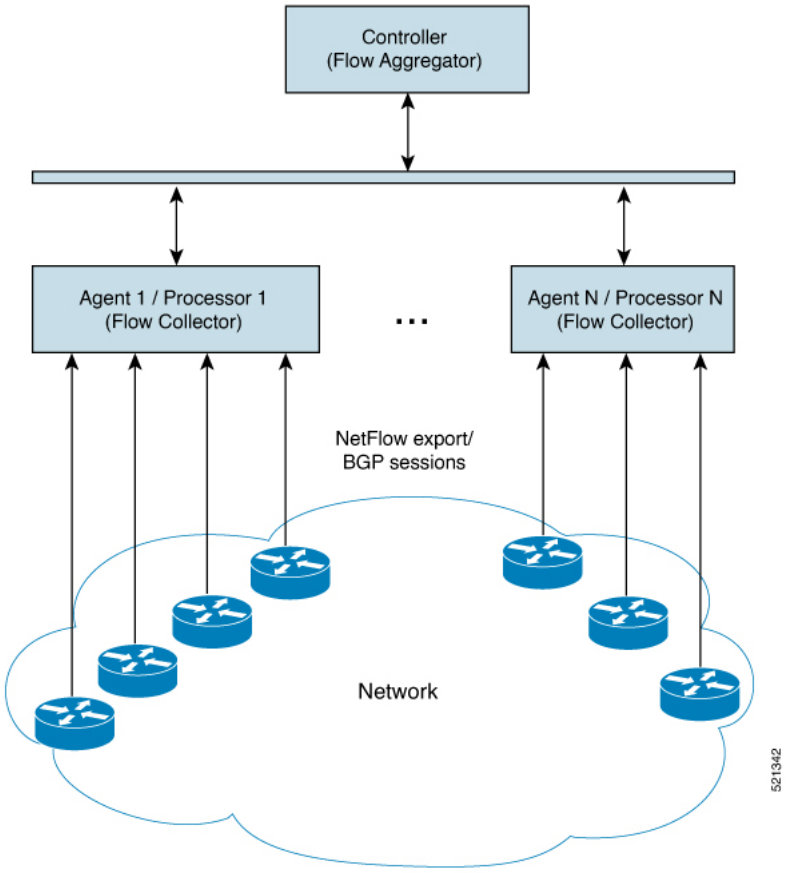
分散型 NetFlow（DNF）収集アーキテクチャは通常、大規模なネットワークに使用されます。このアーキテクチャは、JMS ブローカ、コントローラノード、および 1 つ以上のエージェントで構成されています。ブローカとコントローラノードは、WAE 収集サーバーがインストールされているマシンと同じマシンで実行します。各エージェントは異なるマシンで実行します。DNF 収集では、Ansible を使用してブローカ、コントローラノード、エージェントなどのコンポーネントを個別にインストールする必要があります。

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

分散 Netflow（DNF）収集

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

図 *5 : DNF* アーキテクチャ



Java メッセージサーバー（JMS）ブローカ

クラスタ内のコントローラノード、エージェント、およびクライアントが情報を交換できるように、分散フロー収集のセットアップごとに1つのJMSブローカインスタンスが存在します。すべての情報はブローカを介して交換され、すべてのコンポーネントが相互に通信できます。DNF は、専用の JMS ブローカをサポートします。

すべてのJMSクライアント（コントローラノード、エージェント、および`flow_collector_ias`インスタンス）が機能できるように、ブローカで次の機能が有効になっています。

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

コントローラノードとエージェント

- コントローラ ノード

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

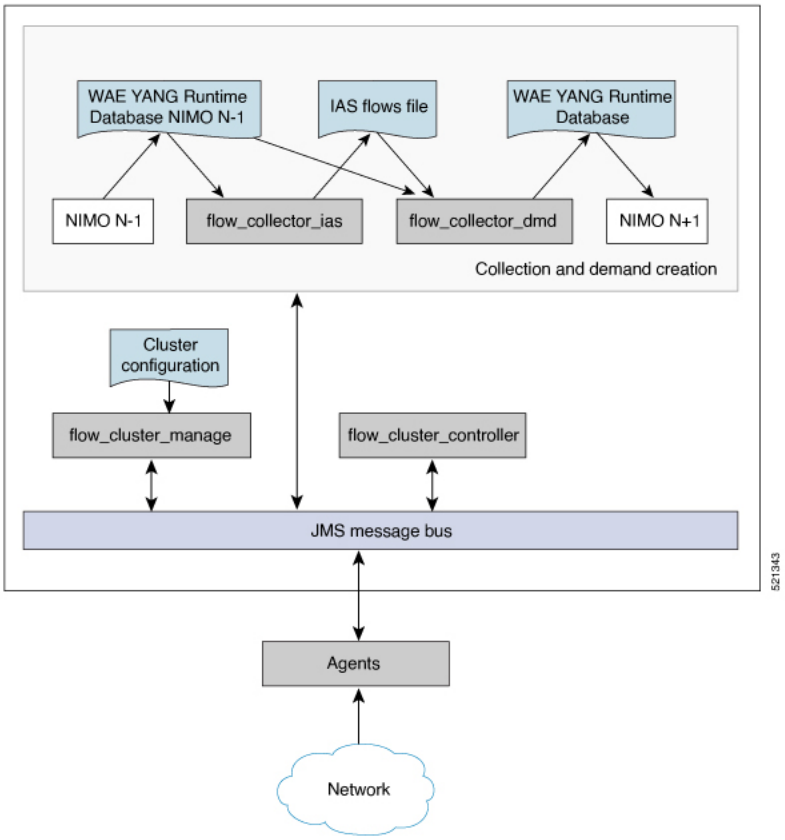
  - すべてのエージェントのステータスを監視および追跡します。

- 最後に完了した IAS 計算のステータスを監視および追跡します。
- すべてのエージェントからクライアントに返される IAS フローデータを集約します（`flow_collector_ias`）。
- クラスタからの構成およびステータスリクエストを処理します（`flow_cluster_manage`）。

- [エージェント（Agents）]

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

図 *6 : DNF* 収集ワークフロー



- **`flow_cluster_manage`**：この CLI ツールは、クラスタの構成とステータスの取得に使用されます。クラスタ構成ファイルを受け取り、構成をクラスタに送信します。

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

  - `<wae-installation-directory>docs/api/netflow/distributed-netflow-rest-api.html`

- http://<*controller-IP-address*>:9090/api-doc たとえば、クラスタ構成を取得するには：

  たとえば、クラスタ構成を取得するには：

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

  たとえば、クラスタ構成を設定するには：

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

  たとえば、クラスタのステータスを取得するには：

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

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

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

集中型 NetFlow（CNF）収集

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

NetFlow 収集の構成

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

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

- NetFlow v5、v9、および IPFIX データグラムは、フロー収集サーバーの UDP ポート番号にエクスポートされます。デフォルト設定は 2100 です。IPv6 フローのエクスポートには、NetFlow v9 または IPFIX が必要です。
- フローコレクタサーバーの iBGP ルートリフレクタ クライアントとして設定されたルータで BGP セッションを定義します。ルータ自体でこれを設定できない場合は、関連するすべてのルーティングテーブルの完全なビューを備えた BGP ルートリフレクタ サーバーを代わりに使用できます。
- フロー エクスポート データグラムの送信元 IPv4 アドレスが iBGP メッセージの送信元 IPv4 アドレスと同じネットワークアドレス空間にある場合は、同じアドレスになるように構成します。
- BGP ルータ ID を明示的に構成します。
- BGP ルートを受信する場合、BGP **AS_path** 属性の最大長は 3 ホップに制限されます。その理由は、単一の IP プレフィックスに付加された BGP 属性（**AS_path** を含む）の合計長が非常に大きくなる（最大 64 KB）可能性があることを考慮して、過度のサーバーメモリ消費を防ぐためです。

集中型 NetFlow 構成ワークフロー

CNF を構成して収集を開始するには、次の手順を実行します。

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

ステップ 1 CNF NetFlow の要件 （123 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 （123 ページ）
ステップ 3 node-flow-configs-table ファイルの作成 （123 ページ）
ステップ 4 CNF 構成ファイルの作成 （124 ページ）
ステップ 5 CNF 収集の構成 （127 ページ）

a) CNF 用の netflow-nimo の構成 （127 ページ）

CNF NetFlow の要件

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

ライセンシング

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

CNF 用のオペレーティングシステムの準備

OS を CNF 用に準備するには、OS ターミナルから次の `flow_manage` コマンドを実行します。

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

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

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

（注） このコマンドの実行後、サーバーを再起動する必要があります。

node-flow-configs-table ファイルの作成

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

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

表 *2 : <NodeFlowConfigs>* テーブルの列

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

以下は、<NodeFlowConfigs> テーブルの例です。

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

CNF 構成ファイルの作成

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

次に例を示します。

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

`waerc` ファイルをソースに設定します。

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

ここで、*<input-path>* は、node-flow-configs-table ファイルの作成 （123 ページ）で作成された `node-flow-configs-table` のパス、*<output-path>*は、生成されるシードクラスタ構成ファイルが配置されるパスです。

<input-path> ファイルの例は次のようになります。

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

シードクラスタ構成ファイルの出力が次のようになっていることを確認します。

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

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

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

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

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


CNF 収集の構成

CNF 用の netflow-nimo の構成

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他のNIMO収集を含む最終ネットワークモデルです。
- シングルモードで動作するように WAE NetFlow エージェントを設定します。エキスパートモードを使用した NetFlow エージェントの構成 （32 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。簡単に識別できる一意の名前をお勧めします。たとえば、networkABC_CNF_flow_get などです。

ステップ 3 [nimo] タブをクリックします。

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

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

ステップ 6 [設定（config）] タブをクリックします。

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

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

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

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

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

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

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

DNF NetFlow 構成ワークフロー

DNF を構成して収集を開始するには、次の手順を実行します。

ステップ 1 分散 NetFlow の要件 （129 ページ）が満たされていることを確認します。

ステップ 2 DNF クラスタの構成 （130 ページ）

a) Ansible を使用した DNF クラスタの展開 （130 ページ）

ステップ 3 DNF 収集の構成 （132 ページ）

a) flow_collector_ias および flow_collector_dmd の構成 （132 ページ）

b) DNF 用の external-executable-nimo の構成 （133 ページ）

分散 NetFlow の要件

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

さらに、すべてのクラスタ要素（コントローラノード、エージェント、JMS ブローカ）に以下が必要です。

- エージェントのシステム要件が、WAE のインストールに必要な要件と同じ要件を満たしています。
- WAE Planning ソフトウェアは、適切なライセンスファイルを使用してサーバー（インストールサーバー）にインストールされる必要があります。
- ルータは、フローをエクスポートし、フロー収集サーバーとの BGP ピアリングを確立するように構成する必要があります。フロー収集プロセスは、入力方向のルータによってキャプチャおよびエクスポートされる IPv4 および IPv6 フローをサポートしています。
- Python3 ベースの Ansible 2.9.18 以降。
- Java 仮想マシン（JVM）ですべての要素（コントローラノード、エージェント、JMS ブローカ）に対して同じインストールパスを使用している。Java 実行可能ファイルは、すべてのユーザーが読み取り可能なパスにある必要があります。
- クラスタ（ブローカ、コントローラノード、およびすべてのエージェント）専用の各サーバーに同じ名前の sudo 対応、SSH 対応ユーザーが存在する。このユーザー名は group_vars/all Ansible ファイル（このセクションで後述）で使用されるため、書き留めておきます。

ライセンシング

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

DNF クラスタの構成

Ansible を使用した DNF クラスタの展開

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

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

```
sudo yum install ansible
```

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

次の例を参考にしてください。

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

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

設定例：

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

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

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

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

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

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

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

ステップ 5 `package/linux/wae/etc/netflow/ansible/group_vars/all` ファイルを変更します。参照および使用されているとおりに、すべての関連変数を変更します。group_vars/all （134 ページ）を参照してください

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

```
SSH_USER_PASS: "ciscowae"
```

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

使用目的

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

または

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

SSH_USER_PASS が設定されているかどうかによって異なります。

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

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

ステップ 8 クラスタを起動します。次のコマンドを使用します。

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

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

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

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

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

（注） JAVA_HOME を設定する必要があります。

例：

export JAVA_HOME=/usr/java_latest

`<path-of-config>/flow-config-cluster.json` の作成については、DNF クラスタ構成ファイルの作成 （135 ページ）を参照してください。

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

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

DNF クラスタのシャットダウンまたはアンインストール

クラスタをシャットダウンするには、次のコマンドを使用します。

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

アンインストールするには、次のコマンドを実行します。

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

DNF 収集の構成

flow_collector_ias および flow_collector_dmd の構成

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

編集する前に、このファイルの権限を変更します。

chmod +x nimo_flow_collector_ias_dmd.sh

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

nimo_flow_collector_ias_dmd.sh の例：

```
#!/bin/bash

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

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

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

DNF 用の external-executable-nimo の構成

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

始める前に

- 送信元ネットワークモデルが必要です。これは、トポロジ収集と、含めたいその他の NIMO 収集を含む最終ネットワークモデルです。
- DNF NetFlow 構成ワークフロー （129 ページ）の準備作業が完了したことを確認します。

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

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

ステップ 3 [nimo] タブをクリックします。

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

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

ステップ 6 [advanced] タブで、以下の情報を入力します。

- [argv]：<directory_path>/nimo_flow_collector_ias_dmd.sh $$input $$output を入力します。

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

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

Ansible 構成ファイルの作成

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

- コントローラノードサーバー（インストールサーバー）には WAE プランニングソフトウェアがインストールされていて、デフォルトのディレクトリが使用されている。特に、イン

ストールサーバーで DNF に使用される構成ファイルが *<wae_installation_directory>*/etc/netflow/ansible にある。

- DNF 構成で専用の JMS ブローカが使用される。
- 構成例では、次の値が使用されている。
  - コントローラノードおよび JMS ブローカの IP アドレス：198.51.100.10
  - エージェント 1 の IP アドレス：198.51.100.1
  - エージェント 2 の IP アドレス：198.51.100.2
  - エージェント 3 の IP アドレス：198.51.100.3

group_vars/all

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

次のオプションを編集します。

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

例（コメントは削除）：

```
LOCAL_WAE_INSTALLATION_DIR_NAME: "/wae/wae-installation"
WAE_INSTALLATION_FILE_NAME: "wae-linux-v7.4.0.bin"
TARGET_JDK_OR_JRE_HOME: "/usr/lib/jvm/jre-11-openjdk-11.0.7"
LOCAL_LICENSE_FILE_PATH: "/home/user1/.cariden/etc/MATE_Floating.lic"
TARGET_SSH_USER: ssh_user
```

DNF クラスタ構成ファイルの作成

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

次に例を示します。

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

`waerc` ファイルをソースに設定します。

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

ここで、*<input-path>* は、node-flow-configs-table ファイルの作成 （123 ページ）で作成された `node-flow-configs-table` のパス、*<output-path>*は、生成されるシードクラスタ構成ファイルが配置されるパスです。

<input-path> ファイルの例は次のようになります。

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

シードクラスタ構成ファイルの出力が次のようになっていることを確認します。

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
```

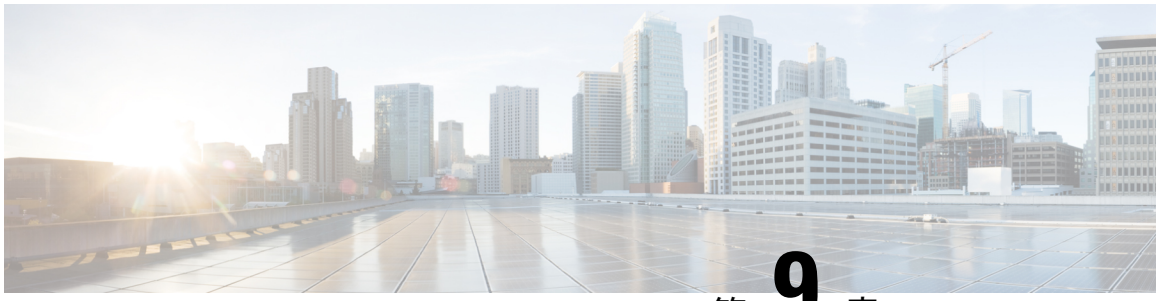
```
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
      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

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

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

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

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



第 9 章

テレメトリの設定

ここでは、次の内容について説明します。

- テレメトリの概要 （139 ページ）
- WAE でのテレメトリの設定 （139 ページ）

テレメトリの概要

モデル駆動型テレメトリ（MDT）のストリーミングにより、IOS XR ルータから対象のデータを選択し、WAE などのコレクタに構造化された形式で送信するメカニズムが提供され ます。コレクタは、ほぼリアルタイムのモニタリングやネットワークの最適化にそのデータを使用できます。MDT の詳細については、『Cisco IOS XR Telemetry Configuration Guide』の「Configure Model-driven Telemetry」の章を参照してください。

WAE は、IOS XR の運用 YANG モデルを理解し、IOS XR ルータからストリーミングされたテレメトリを受信し、データを解析して保存できます。データが WAE に保存された後、sr-traffic-matrix NIMO はデータを読み取り、それを使用して需要に応じたネットワークモデルを作成できます。

WAE でのテレメトリの設定

始める前に

ルータのセグメントルーティングを有効にし、デバイスでトラフィックコレクタを設定して、システムにトラフィックがあることを確認します。次のコマンドを使用して、ルータのプレフィックスとトンネルトラフィックを確認します。

プレフィックストラフィック：

```
sh traffic-collector ipv4 counters prefix <prefix-name>
```

トンネルトラフィック：

```
show traffic-collector ipv4 counters tunnels <tunnel-name>
```

ステップ 1 WAE テレメトリエージェントを設定します。

```
admin@wae(config)# wae agents telemetry-agent ports <port-number1> <port-number2> <port-numberxx>
admin@wae(config)# commit
```

（注） エージェントがテレメトリ情報を受信するために使用するポートは、WAE マシンで使用できる必要があります。

ステップ 2 WAE に使用される TCP を使用して、ポートで Key-Value Google Protocol Buffers（KV-GPB）でエンコードされたテレメトリを送信するようにデバイスでテレメトリを設定します。ルータには、センサーグループ、接続先グループ、およびサブスクリプションの 3 つの属性を定義する必要があります。この定義を行う方法については、『Cisco IOS XR Telemetry コンフィギュレーション ガイド』の「モデル駆動型テレメトリの設定」の章を参照してください。この手順の最後に例が示されています。

ステップ 3 sr-traffic-matrix-nimo を設定します。詳細については、セグメント ルーティング トラフィック マトリックス収集 （71 ページ）を参照してください。

```
admin@wae(config)# networks network <network-model-name> nimo sr-traffic-matrix-nimo source-network
 <source-network> collection-period <collection-period-in-seconds>
```

`collection-period` 設定はデフォルトで有効になっており、60 秒に設定されています。

次のオプションを使用できます。

```
admin@wae(config)# networks network srt nimo sr-traffic-matrix-nimo ?
Possible completions:
  advanced
  collection-period   Frequency in seconds for automatic periodic generation of demands ('0' value
 disables periodic demand generation).
  source-network      Source network for this network to use.
  <cr>

admin@wae(config)# networks network srt nimo sr-traffic-matrix-nimo advanced ?
Possible completions:
  action-timeout             Specifies the timeout value (in minutes) for running actions - default
 of '0' specifies the system default.
  copy-network               When set to 'true', copies the source network into this NIMO network
and create demands in the new model.
  telemetry-agent-callback   Callback for telemetry-agent to inform sr-traffic-matrix-nimo about
new telemetry data.
```

`telemetry-agent-callback` は設定オプションではなく、テレメトリエージェントによって内部的に使用されるアクションです。

（注） SR ポリシー（XTC-nimo）の場合、収集の実行中に、送信元ネットワークの `use-signaled-name` プロパティを true（デフォルトでは true）に設定する必要があります。

RSVP LSP トンネル（lsp-snmp-nimo）の場合、LSP 収集の実行中に、送信元ネットワークの `use-signaled-name` プロパティを false（デフォルトでは false）に設定する必要があります。

ステップ 4 sr-traffic-matrix-nimo 収集を実行してデマンドを生成します。

```
admin@wae# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection
```

デフォルトでは、デマンドはローカルにキャッシュされた情報を使用して生成されます。ただし、WAEテレメトリエージェントからの未処理のテレメトリデータを使用してデマンドを生成する場合は、**use-cache** オプションを false に設定する必要があります。次に例を示します。

```
admin@wae# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection use-cache
 false
```

例

1. WAE テレメトリエージェントを設定します

```
admin@wae# config terminal
Entering configuration mode terminal
admin@wae(config)# wae agents telemetry-agent ports 1624
admin@wae(config)# commit
```

2. テレメトリデータを WAE に送信するようにルータを設定します。

a. sensor-group を定義します

```
telemetry model-driven
 sensor-group SRTM
  sensor-path Cisco-IOS-XR-infra-tc-oper:traffic-collector/afs/af/counters/tunnels
  sensor-path
Cisco-IOS-XR-infra-tc-oper:traffic-collector/vrf-table/default-vrf/afs/af/counters
 !
!
```

b. destination-group を定義します

```
telemetry model-driven
 destination-group my_workstation
  address-family ipv4 10.152.130.41 port 1624
   encoding self-describing-gpb
   protocol tcp
 !
```

（注） 上記の例の IP アドレスとポートは、WAE テレメトリエージェントで以前に設定されたものと同じである必要があります。

c. サブスクリプションを定義します

```
telemetry model-driven
 subscription ABC
  sensor-group-id SRTM sample-interval 5000
  destination-id my_workstation
 !
!
```

3. SR LSP トラフィック マトリックス NIMO を設定します（sr-traffic-matrix-nimo）

```
admin@wae# config terminal
```

```
Entering configuration mode terminal
admin@wae(config)# networks network srtm nimo sr-traffic-matrix-nimo source-network igp
 collection-period 50s
admin@wae(config)# commit
```

WAE とルータ間の接続を表示するには、shell CLI netstat コマンドを使用します。次に例を示します。

```
# netstat -an | grep :1624 | grep ESTABLISHED
tcp        0     28 10.10.10.10:1624        10.152.130.41:61092      ESTABLISHED
```

ここで、10.10.10.10 は WAE マシンのアドレスで、10.152.130.41 は接続されたルータのアドレスです。

4. sr-traffic-matrix-nimo 収集を実行してデマンドを生成します。

```
admin@wae# networks network srtm nimo sr-traffic-matrix-nimo run-collection
status true
message Succeeded: Retrieved 12 SR demands from network srtm
admin@wae# show running-config networks network srtm model demands | nomore
networks network igp
model demands demand "PE1|PE2|default"
source node node-name PE1
destination node node-name PE2
service-class-name default
traffic 22.203833
!
.......
!
model demands demand "PE4|PE3|default"
source node node-name PE4
destination node node-name PE3
service-class-name default
traffic 22.202989
!
!
admin@wae#
```

第 10 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 （143 ページ）
- オンデマンド帯域幅の構成ワークフロー （143 ページ）
- オンデマンド帯域幅のシャットダウン （148 ページ）
- Bandwidth Optimizationアプリケーション ワークフロー （149 ページ）
- 帯域幅最適化のシャットダウン （151 ページ）

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（プライマリモデル）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 （5 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。
- アグリゲータ（ステップ4）の一部としてsr-traffic-matrix-nimoが有効になっていることを確認してください。セグメント ルーティング トラフィック マトリックス収集 （71 ページ）を参照してください

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 （30 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 （30 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 （31 ページ）
4. アグリゲータの構成	NIMO 収集の統合 （67 ページ） （注） sr-traffic-matrix-nimoが設定の一部であることを確認してください。
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（101ページ）
6. 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） （59 ページ）
7.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 （145 ページ）
8. 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（143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（146ページ）を参照してください。

ステップ1 エキスパートモードから、[設定エディタ（Configuration editor）] で [/wae:wae/components/bw-on-demand:bw-on-demand] に移動し、[設定（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-agents]：XTC エージェントを選択します。
- [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。
- [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。
- [reopt-interval]：LSP を再最適化するまでの期間を入力します。
- [metric-reopt-interval]：LSP がメトリック用に再最適化されるまでの期間を入力します。
- [priority-mode]：優先モードを有効にするには、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
```

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 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 3 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。

（注）
- アグリゲータ（ステップ4）の一部としてsr-traffic-matrix-nimoが有効になっていることを確認してください。セグメント ルーティング トラフィック マトリックス収集 （71 ページ）を参照してください

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 （30 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 （30 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 （31 ページ）
4. アグリゲータの構成	NIMO 収集の統合 （67 ページ） （注） sr-traffic-matrix-nimoが設定の一部であることを確認してください。
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（101ページ）
6. 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） （59 ページ）
7.Bandwidth Optimizationアプリケーションの構成	Bandwidth Optimization の設定 （150 ページ）

手順	詳細
8. 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 （149 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、[設定（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-agents]：XTC エージェントを選択します。
- [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。
- [util-hold-margin]：util-threshold インターフェイス使用率の下限マージンを入力します。このマージンを下回ると、既存の戦術的 SR ポリシーが削除されます。
- [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。
- [del-lsps]：アプリケーションが無効化されたときに、アプリケーションが作成したすべての戦術的 SR ポリシーを削除するには、true を選択します。
- [max-global-reopt-interval]：既存の戦術的 SR ポリシーをグローバルに再最適化する時間間隔を入力します。

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 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 3 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 11 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 （153 ページ）
- スケジューラの構成 （153 ページ）
- トポロジ収集を実行するためのトリガーの構成例 （155 ページ）

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） エキスパートモードまたは Cisco WAE CLI を使用したスケジューラ構成は、Cisco WAE UI に表示されません。Cisco WAE UI を使用してネットワーク収集とエージェントの実行をスケジュールするには、ネットワーク モデル コンポーザ を使用したジョブのスケジュール （24 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。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で [wae:wae] > [コンポーネント（components）] タブ > [スケジューラ（scheduler）] > [タスク（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。

列挙型の RPC パラメータの値は、完全修飾パスで指定する必要があります。

例：

```
admin@wae# show running-config wae components scheduler tasks task filter
wae components scheduler tasks task filter
action filter
  order 1
  rpc path /wae:networks/network{node-filter}/opm/node-filtering:node-filtering/run
  rpc params 1
   key   delimiter
   value /wae:networks/network{node-filter}/opm/node-filtering:node-filtering/run/delimiter/COMMA

   type  ENUMERATION
  !
```

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 を使用したトポロジ収集 （64 ページ）」を参照してください。

ステップ 1 エキスパート モードから、[設定エディタ（Configuration editor）] で [wae:wae] > [コンポーネント（components）] タブ > [スケジューラ（scheduler）] > [タスク（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。

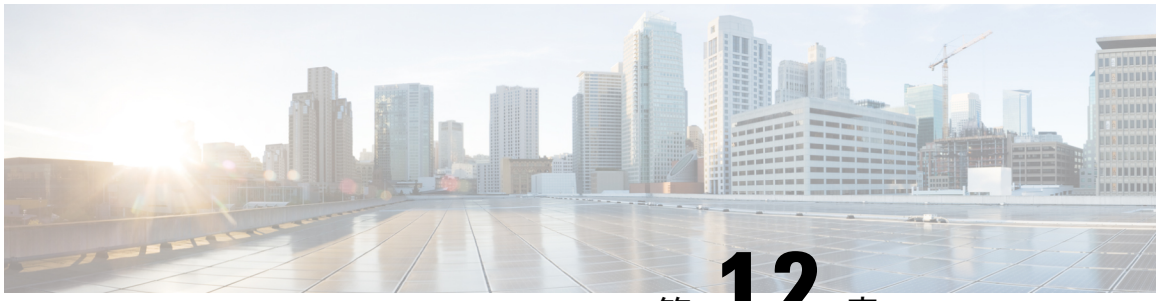
a) 送信元パスを入力します（この例では、XTC リンクステータスが変更される場所です）：**/wae/agents/xtc/xtc{TTE-xtc11}/pce/xtc-topology-objects/xtc-links**。
b) [subscription-type] ドロップダウンリストから、[operational] を選択します。

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 12 章

Cisco Smart Licensing

Cisco WAE は、Cisco Smart Licensing と従来のライセンスをサポートしています。従来のライセンスから Cisco Smart Licensing への切り替えを希望する場合は、Cisco WAE アカウント担当者にお問い合わせください。2 種類のライセンスの違いについては、Cisco.com で紹介している Cisco Smart Licensing の概要を参照してください。

（注） さまざまな機能パックのライセンスは個別に提供されます。Cisco WAE スマートライセンスとは統合されていません。

Cisco WAE のすべての機能を使用するには、ライセンスが必要です。ライセンスの取得について質問がある場合は、シスコのサポート担当者またはシステム管理者にお問い合わせください。

（注） スマートライセンスから従来のライセンスに戻す場合は、Cisco WAE UI からスマートライセンスを無効にして、`~/.cariden/etc` フォルダにある `MATE_Smart.lic` ファイルを手動で削除します。

この章で説明するタスクは、管理機能を持つユーザーが実行できます。

高可用性（HA）を設定する前に、セットアップの両方でスマートライセンスを設定します。以下の手順に従って、システムの両方でスマートライセンスを設定します。

ここでは、次の内容について説明します。

- シスコ スマートライセンシングの概要 （158 ページ）
- スマートライセンシング設定のワークフロー （158 ページ）
- Cisco WAE のスマートライセンシングの有効化 （159 ページ）
- Cisco WAE と CSSM 間のトランスポートモードの設定 （159 ページ）
- Cisco Smart Software Manager への Cisco WAE の登録 （160 ページ）
- Cisco Smart Software Manager へのオフラインモードの Cisco WAE の登録 （161 ページ）
- スマートライセンスの登録と認証ステータス （164 ページ）

シスコ スマートライセンシングの概要

シスコは、ノードロックライセンスをインストールするのではなく、シンプルな登録とライセンス消費レポートプロセスで WAE ソフトウェアライセンスとエンドポイントライセンスの消費を簡単かつ効率的にモニターできるようにする、スマートライセンスを提供しています。購入したシスコ製品およびライセンスの詳細は、Cisco Smart Software Manager（CSSM）と呼ばれる集中型データベースに保持されます。

スマートライセンシング設定のワークフロー

ステップ 1 Cisco Systems でスマートアカウントを作成します。これを実行するには、Smart Account Request に移動し、Web サイトの指示に従います。

ステップ 2 次のいずれかを実行します。

- WAE Design を使用している場合は、ローカルの WAE Design セットアップ（Windows、Mac、または Linux）で WAE Design GUI または CLI を使用してスマートライセンスを有効にします。FlexLM ライセンス：詳細については、*WAE Design GUI* のインストールガイドを参照してください。
- WAE Collector Server を使用している場合は、waerc をソースに設定した後に、次のコマンドを実行します。

```
license_install -smart-lic-host <hostname or IP> -smart-lic-port 2022 -smart-lic-username admin
 -smart-lic-password Admin@123
```

where

`-smart-lic-host` は、WAE サーバーが実行されているマシンのホスト名または IP です（WAE がインストールされ、すべての nimo が実行されている場所）。

`-smart-lic-port` は、WAE サーバーが netconf メッセージをリッスンするポートです。変更していない限り、これは 2022 です。

`-smart-lic-username` は、WAE サーバーにログインするためのユーザー名です。変更していない限り、これは admin です。

`-smart-lic-password` は、WAE サーバーにログインするためのパスワードです。変更していない限り、これは Admin@123 です。

コマンドを実行すると、`MATE_Smart.lic` ファイルが `~/.cariden/etc` フォルダに作成されます。

ステップ 3 Cisco Smart Software Manager への Cisco WAE の登録 （160 ページ）

ステップ 4 （オプション）デフォルトでは、スマートライセンス情報がクラウドに直接送信されます。これを変更するには、Cisco WAE と CSSM 間のトランスポートモードの設定 （159 ページ） で説明されている手順に従ってください。

Cisco WAE のスマートライセンシングの有効化

始める前に

スマートアカウントがあることを確認します。ない場合、「スマートアカウントのリクエスト」に移動し、Web サイトの指示に従います。

（注） WAE UI の [スマートライセンシング（Smart Licensing）] から [スマート ソフトウェア ライセンシングの無効化（Disable Smart Software Licensing）] をクリックすることで、いつでもスマートライセンシングを無効にできます。

ステップ 1 WAE UI から、[スマートライセンシング（Smart Licensing）] をクリックします。

ステップ 2 [スマート ソフトウェア ライセンシングの有効化（Enable Smart Software Licensing）] をクリックします。

ステップ 3 [OK] をクリックして、スマートライセンシングを有効にすることを確認します。[Smart Licensing] ページが表示されます。

ステップ 4 次のいずれかを実行します。

a) Cisco.com で CSSM に Cisco WAE を登録していない場合、Cisco WAE は評価モードになります（利用可能な期間は 90 日間）。Cisco Smart Software Manager への Cisco WAE の登録 （160 ページ）の説明に従って、Cisco WAE を登録します。

b) CSSM に Cisco WAE をすでに登録している場合は、使用するライセンスを選択します。

（注） WAE Design のスマートライセンシングを有効にするには、Cisco WAE Design GUI のインストールガイドを参照してください。

Cisco WAE と CSSM 間のトランスポートモードの設定

デフォルトでは、スマートライセンス情報がクラウドに直接送信されます。この値を変更するには、次の手順を実行します。

ステップ 1 [スマート ソフトウェア ライセンシング（Smart Software Licensing）] ページで、[トランスポート設定（Transport Settings）] フィールドの [表示/編集（View/Edit）] をクリックします。[トランスポート設定（Transport Settings）] ウィンドウが表示されます。

ステップ 2 通信モードを選択します。

- [ダイレクト モード（Direct mode）]：ライセンス情報を直接クラウドに送信します。これがデフォルトです。この URL は編集できません。

- [トランスポート ゲートウェイ（Transport Gateway）]：Cisco Call Home トランスポート ゲートウェイまたはシスコ スマート ソフトウェア ライセンシング サテライトを使用します。（サテライトは顧客の構内に設置され、CCSM 機能の一部を提供します。詳細については、スマートライセンシングの概要を参照してください。

  登録にサテライトを使用している場合は、サテライトの 7-202001 バージョンをインストールしてください。

- [HTTP プロキシ（HTTP Proxy）]：クラウドとの間での通信に HTTP/HTTPS プロキシを使用します。プロキシ IP アドレス、ポート、ユーザー名、およびパスワードを入力します。

ステップ 3 [保存（Save）] をクリックします。

Cisco Smart Software Manager への Cisco WAE の登録

Cisco Smart Software Manager（CSSM）に Cisco WAE を登録するには、CSSM からトークンを取得し、WAE UI に入力する必要があります。この作業が必要になるのは 1 回限りです。Cisco WAE がすでに登録されている場合は、製品の登録を解除してから、再度登録する必要があります。ライセンス登録の更新や承認など、CSSM の使用方法の詳細については、『Cisco Smart Software Manager User Guide』を参照してください。[スマートライセンス（Smart Licensing）] ページを使用して、WAE UI 内でこれらの操作にアクセスすることもできます。

始める前に

- スマートアカウントがあることを確認します。ない場合、「スマートアカウントのリクエスト」に移動し、Web サイトの指示に従います。
- Cisco WAE でスマートライセンスが有効になっていることを確認します。Cisco WAE のスマートライセンシングの有効化 （159 ページ） を参照してください。

ステップ 1 Cisco Software Central の Web サイトに移動します。

ステップ 2 トークンを取得します。トークンがすでにある場合は、次の手順に進みます。

ステップ 3 トークンが無効になっている場合は、次の手順に従って新しいトークンを取得できます。

a) Cisco Software Central で、[ライセンス（License）] > [スマート ソフトェア ライセンシング（Smart Software Licensing）] を選択します。
b) 該当するバーチャル アカウントを選択します。
c) [全般（General）] タブをクリックし、[新規トークン（New Token）] をクリックします。
d) 指示に従って、名前と期間を入力します。輸出規制機能を有効にするために制限付きトークンを作成する必要がある場合は、[輸出規制機能を許可（Allow Export-Controlled Functionality）] ボタンをクリックします。
e) [トークンの作成（Create Token）] をクリックします。
f) トークン ID をクリップボードにコピーし、次のステップに進みます。

ステップ 4 WAE UI から、[登録（Register）] をクリックします。[スマートソフトウェアライセンシング製品の登録（Smart Software Licensing Product Registration）] ウィンドウが表示されます。

ステップ 5 ステップ 3 でコピーしたトークン ID を入力し、製品インスタンスを登録します。

（注） [すでに登録されている場合は、この製品インスタンスを再登録します（Reregister this product instance if it is already registered）] オプションを選択して、ライセンスを再登録します。

ステップ 6 [スマート ソフトウェア ライセンシングのステータス（Smart Software Licensing Status）] ページが表示されます。

ステップ 7 [ライセンスの選択...（Choose Licenses...）] をクリックします。 .

ステップ 8 該当するすべてのライセンスをライセンステーブルでクリックし、対応するライセンスインスタンスの数を入力します。

ステップ 9 [OK] をクリックして変更を保存します。[スマートライセンスの使用状況（Smart License Usage）] テーブルが更新され、変更が反映されます。データを手動で更新するには、ページの上部にある [更新（Refresh）] アイコンをクリックします。

ステップ 10 ライセンスに関連する詳細がすべてこの画面に表示されます。[アクション（Actions）] ドロップダウンには、次の追加オプションがあります。

- 認証を今すぐ更新
- 登録を今すぐ更新
- 再登録
- 登録解除
- スマート ソフトウェア ライセンシングの無効化

Cisco Smart Software Manager へのオフラインモードの Cisco WAE の登録

始める前に

- スマートアカウントがあることを確認します。ない場合、「スマートアカウントのリクエスト」に移動し、Web サイトの指示に従います。
- Cisco WAE でスマートライセンスが有効になっていることを確認します。Cisco WAE のスマートライセンシングの有効化 （159 ページ） を参照してください。
- オフラインモードで使用するスマートアカウントライセンスの予約には、シスコからの特定の権限が必要です。詳細については、スマートライセンシングの概要を参照してください。

（注） オフラインモードでは、特定のライセンスの予約（SLR）のみがサポートされ、永久ライセンス予約（PLR）はサポートされません。

ステップ 1 WAE UI から、[スマートライセンシング（Smart Licensing）] をクリックします。

ステップ 2 [登録（Register）] をクリックします。[スマートソフトウェアライセンシング製品の登録（Smart Software Licensing Product Registration）] ウィンドウが表示されます。

ステップ 3 オフラインモードで使用する場合は、[ここから開始（Start Here）] をクリックします。

ステップ 4 [はい、自分のスマートアカウントはライセンス予約が有効になっています（Yes, My Smart Account License Reservation Enabled）] をクリックします。

ステップ 5 [スマートライセンスの予約（Smart License Reservation）] 画面で、[予約要求コードを生成（Generate Reservation Request Code）] をクリックします。ライセンス予約要求コードが、次に表示される画面に表示されます。[クリップボードにコピー（Copy to Clipboard）] ボタンを使用して、このコードをコピーします。

ステップ 6 Cisco Software Central の Web サイトに移動し、適切な仮想アカウントを選択します。

ステップ 7 [ライセンス（Licenses）] タブをクリックし、[ライセンス予約（License Reservation）] をクリックします。ここにライセンス予約要求コードを貼り付け、[次へ（Next）] をクリックします。

ステップ 8 [ライセンスの選択（Select Licenses）] 画面で、[特定のライセンスを予約する（Reserve a Specific License）] オプションボタンを選択し、リストから必要なライセンスを予約して、[次へ（Next）] をクリックします。

ステップ 9 [レビューと確認（Review and Confirm）] 画面で [認証コードの生成（Generate Authorization Code）] をクリックします。[クリップボードにコピー（Copy to Clipboard）] ボタンを使用してコードをコピーします。

ステップ 10 WAE UI から [スマートライセンシング（Smart Licensing）] に戻ります。[予約承認コードを入力（Enter Reservation Authorization Code）] をクリックします。予約承認コードを貼り付けて、[承認コード/ファイルのインストール（Install Authorization Code/File）] をクリックします。

ステップ 11 WAE UI の [スマートソフトウェアライセンシング（Smart Software Licensing）] 画面で、[ライセンスを選択...（Choose Licenses...）] オプションを使用して、用途に応じた必要な数のライセンスをチェックアウトします。

予約の更新

追加のライセンスを予約するには、[予約の更新（Update Reservation）] オプションを使用します。

ステップ 1 WAE UI の [スマート ソフトウェア ライセンシング（Smart Software Licensing）] 画面で、[製品インスタンス名（Product Instance Name）] をメモします。

ステップ 2 Cisco Software Central の Web サイトに移動し、適切な仮想アカウントを選択します。

ステップ 3 [製品インスタンス（Product Instances）] タブをクリックし、WAE UI 画面の [製品インスタンス名（Product Instance Name）] に一致する製品インスタンスの名前を検索します。

ステップ 4 この製品インスタンスに対して、[アクション（Actions）] ドロップダウンを選択し、[予約済みライセンスの更新（Update Reserved Licenses）] を選択します。

ステップ 5 [ライセンスの選択（Select Licenses）] 画面で、[特定のライセンスを予約する（Reserve a Specific License）] オプションボタンを選択し、リストから必要なライセンスを予約して、[次へ（Next）] をクリックします。

ステップ 6 [レビューと確認（Review and Confirm）] 画面で [認証コードの生成（Generate Authorization Code）] をクリックします。[クリップボードにコピー（Copy to Clipboard）] ボタンを使用してコードをコピーします。

ステップ 7 WAE UI から [スマートライセンシング（Smart Licensing）] に戻ります。[予約の更新...（Update Reservation...）] をクリックします。. 予約承認コードを貼り付けて、[承認コード/ファイルのインストール（Install Authorization Code/File）] をクリックします。

ステップ 8 ライセンス予約確認コードが生成されます。このコードをコピーします。

ステップ 9 Cisco Software Central の Web サイトに戻ります。[ライセンス予約の更新（Update License Reservation）] 画面の最後の手順は、確認コードの入力です。**[確認コードを入力（Enter Confirmation Code）]** をクリックします。

ステップ 10 予約確認コードを入力して、[OK] をクリックします。

ステップ 11 WAE UI の [スマート ソフトウェア ライセンシング（Smart Software Licensing）] 画面で、[ライセンスを選択...（Choose Licenses...）] オプションを使用して、用途に応じた必要な数のライセンスをチェックアウトします。

予約済みライセンスの返却

[予約済みライセンスの返却（Return Reserved Licenses）] オプションを使用して、予約したライセンスを返却できます。

ステップ 1 WAE UI から [スマートライセンシング（Smart Licensing）] を選択します。[予約済みライセンスの返却...（Return Reserved Licenses...）] をクリックします。.

ステップ 2 [返却ライセンスの確認（Confirm Return Licenses）] 画面で、[予約返却コードの生成（Generate Reservation Return Code）] をクリックします。

ステップ 3 [クリップボードにコピー（Copy to Clipboard）] ボタンを使用して、ライセンス予約返却コードをコピーします。

ステップ 4 Cisco Software Central Web サイトに移動し、適切な仮想アカウントを選択します。

ステップ 5 [製品インスタンス（Product Instances）] タブをクリックし、WAE UI 画面の [製品インスタンス名（Product Instance Name）] に一致する製品インスタンスの名前を検索します。

ステップ 6 この製品インスタンスに対して、[アクション（Actions）] ドロップダウンを選択し、[削除（Remove）] を選択します。

ステップ 7 [製品インスタンスの削除（Remove Product Instance）] ポップアップで、予約返却コードを貼り付け、[製品インスタンスの削除（Remove Product Instance）] をクリックします。

ステップ 8 [製品インスタンスの削除（Remove Product Instance）] ポップアップを閉じます。[登録ステータス（Registration Status）] が [未登録（Unregistered）] 状態に戻っていることが確認できます。

[登録ステータス（Registration Status）] が [未登録（Unregistered）] 状態に戻ると、[ライセンス予約の無効化（Disable License Reservation）] および [予約要求コードを生成（Generate Reservation Request Code）] オプションが使用可能になります。

ステップ 9 WAE UI の [スマートライセンシング（Smart Licensing）] でページが更新されると、[登録ステータス（Registration Status）] が [未登録（Unregistered）] 状態に戻っていることが確認できます。

スマートライセンスの登録と認証ステータス

登録ステータス

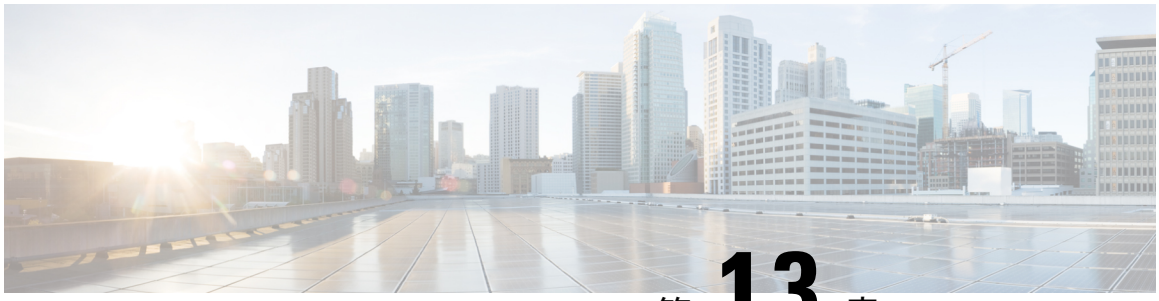
ライセンス登録ステータスは、Cisco WAE が Cisco.com のシスコ スマート ソフトウェア ライセンシングに正常に登録されているかどうかを表します。

ライセンス登録ステータス	説明
未登録	スマート ソフトウェア ライセンシングは Cisco WAE で有効になっていますが、Cisco WAE は CSSM に登録されていません。
登録済み	Cisco WAE は、CSSM に登録されています。Cisco WAE は ID 証明書を受信しています。この ID 証明書は、将来シスコのライセンス担当者との通信に使用されます。 デフォルトでは、登録の更新は 30 日ごとに自動的に行われます。手動で更新する場合は、[スマート ソフトウェア ライセンシング（Smart Software Licensing）] ページの右上にあるドロップダウンリストから [登録を今すぐ更新（Renew Registration Now）] をクリックします。 登録を解除するには、[スマート ソフトウェア ライセンシング（Smart Software Licensing）] ページの右上にあるドロップダウン リストから [登録解除（Deregister）] をクリックします。

ライセンス認証ステータス

ライセンス認証ステータスは、購入したライセンスに対するライセンスの使用状況、および Cisco Smart Licensing に準拠しているかどうかを表しています。購入したライセンス数を超えると、その製品はコンプライアンス違反となります。

ライセンス認証ステータス	説明
評価モード	Cisco WAE は、評価モードで実行されています（90 日で期限切れになります）。
承認済み（Authorized）	Cisco WAE に有効なスマートアカウントがあり、登録されています。Cisco WAE が要求するすべてのライセンスの使用が承認されています。
コンプライアンス違反（Out of Compliance）	Cisco WAE は、購入されたライセンス数を超過しました。 （特に、製品インスタンスの仮想アカウントに、1 つ以上のライセンス タイプが不足しています）。
評価期限切れ	評価期間が終了し、Cisco WAE はライセンスなしの状態になります。
認証が期限切れ（Authorization Expired）	Cisco WAE は、認証の有効期限前に、ライセンス認証を正常に更新できませんでした。 認証を更新するには、[スマート ソフトウェア ライセンシング（Smart Software Licensing）] ページの右上にあるドロップダウンリストから [認証を今すぐ更新（Renew Registration Now）] をクリックします。
未使用（Not in Use）	ライセンスは使用されていません。



第 13 章

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 （167 ページ）
- エージングの構成 （168 ページ）
- wae.conf （169 ページ）
- ハイ アベイラビリティの設定 （175 ページ）
- LDAP の設定 （179 ページ）
- ステータスダッシュボード （183 ページ）
- WAE CLI ロギングについて （184 ページ）
- データベースのロック （196 ページ）
- セキュリティ （198 ページ）
- WAE 運用データのクリア （201 ページ）
- WAE 構成のバックアップと復元 （201 ページ）
- WAE 診断 （201 ページ）

ユーザーの管理

すべてのユーザーが管理者ロールを持ちます。次の手順では、ユーザーを作成、変更、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、[ユーザー管理（User Manager）] アイコン（  ）をクリックします。

ステップ 2 ユーザーを追加するには、  をクリックして、該当するすべてのフィールドを入力します。

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) ユーザーの行を選択して  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、ユーザーの行をクリックしてから  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/components/dare:aggregators に移動し、[エージング（aging）] タブを選択します。

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

  （注） l3-node-aging-duration の値は l3-port-aging-duration より大きくする必要があり、l3-port-aging-duration の値は l3-circuit-aging-duration より大きくする必要があります。

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

  （注） l1-node-aging-duration の値は l1-port-aging-duration より大きくする必要があり、l1-port-aging-duration の値は l1-link-aging-duration より大きくする必要があります。

ステップ 2 [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。

（注） wae.conf ファイルに変更を加えた後は、必ず WAE を再起動してください。

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 for kicker event stream. -->
<!-- See tailf-kicker.yang -->
<!-- Required for cluster mode. -->
<stream>
  <name>kicker-events</name>
  <description>NCS event according to tailf-kicker.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, tailf:aes-cfb-128-encrypted-string&#xa;      and
tailf:aes-256-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>

    <AES256CFB128>
      <key>0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef</key>
    </AES256CFB128>
  </encrypted-strings>


  <logs>
    <syslog-config>
      <facility>daemon</facility>
    </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>error</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-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>

  <progress-trace>
    <enabled>true</enabled>
    <dir>./logs</dir>
  </progress-trace>
</logs>
```

```
<ssh>
  <algorithms>
    <kex>diffie-hellman-group14-sha1</kex>
    <mac>hmac-sha2-512,hmac-sha2-256,hmac-sha1</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>$WAE_ROOT/lib/exec/wae-ldap-auth</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>P1Y</absolute-timeout>
    <custom-headers>
      <header>
        <name>X-Content-Type-Options</name>
        <value>nosniff</value>
      </header>
      <header>
        <name>Content-Security-Policy</name>
        <value>default-src 'self'; script-src 'self'; img-src 'self' data:;
block-all-mixed-content; base-uri 'self'; frame-ancestors 'none'; style-src 'self'
'unsafe-inline'</value>
      </header>
    </custom-headers>
    <idle-timeout>PT30M</idle-timeout>
    <allow-symlinks>true</allow-symlinks>
    <enabled>true</enabled>
    <transport>
      <tcp>
        <enabled>true</enabled>
        <ip>0.0.0.0</ip>
        <port>8080</port>
        <redirect>https://@HOST@:8443</redirect>
        <!-- Uncomment this to enable support for IPv6&#xa;        <extra-listen>&#xa;
         <ip>::</ip>&#xa;          <port>8080</port>&#xa;        </extra-listen>&#xa;
      -->
      </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>
        <!-- Uncomment this to enable support for IPv6&#xa;        <extra-listen>&#xa;
         <ip>::</ip>&#xa;          <port>8443</port>&#xa;        </extra-listen>&#xa;
      -->
      </ssl>
    </transport>

    <cgi>
      <enabled>true</enabled>
      <php>
        <enabled>false</enabled>
      </php>
    </cgi>
  </webui>

  <rest>
    <enabled>true</enabled>
    <enable-legacy>true</enable-legacy>
  </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>
        <!-- Uncomment this to enable support for IPv6&#xa;        <extra-listen>&#xa;
         <ip>::</ip>&#xa;          <port>2022</port>&#xa;        </extra-listen>&#xa;
      -->
      </ssh>
      <tcp>
        <enabled>false</enabled>
        <ip>127.0.0.1</ip>
        <port>2023</port>
      </tcp>
    </transport>
  </netconf-north-bound>

  <netconf-call-home>
    <enabled>false</enabled>

    <transport>
      <tcp>
        <ip>0.0.0.0</ip>
        <port>4334</port>
      </tcp>
    </transport>
  </netconf-call-home>

  <!-- <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 （212 ページ） を参照してください。

ハイ アベイラビリティの設定

Cisco WAE は、高可用性（HA）と自動フェールオーバーをサポートしています。WAE ノードの 2 つのインスタンスは、並行して実行するように設定されています。プライマリノードはプライマリモードで設定され、セカンダリノードはスタンバイモードで設定されます。プライマリノードは、ポート 4570（または wae.conf ファイルで設定されたポート）でセカンダリノードからの接続をリッスンします。コミットされた CDB データは、定期的にセカンダリノードにミラーリングされます。スタンバイモードのノードで実行される CDB への書き込み操作（NIMO 操作、エージェントプロセス、またはスケジューラアクションなど）は失敗することに注意してください。

プライマリノードに障害が発生すると、セカンダリノードがプライマリノードとして引き継ぎます。セカンダリノードでプライマリモードが有効になると、書き込み操作が許可され、CDB

が再構築され、スケジュールされたジョブが実行されます。セカンダリノードでプライマリノードが以前に実行した操作が再開されます。

ステップ 1 プライマリノードとセカンダリノードの両方で、wae.conf ファイルを編集して HA を有効にします。

```
<ha>
  <enabled>true</enabled>
  <ip>0.0.0.0</ip>
  <!-- The following port configuration is optional.
       Default port is 4570. This option can be used
       to override the default port -->
  <!-- <port>4570</port> -->
</ha>
```

（注） `/etc/hosts` ファイルがホスト名から IP アドレスへのマッピングで更新されていることを確認してください。

ステップ 2 スーパーバイザを使用して両方のノードで Cisco WAE を再起動する

```
sudo supervisorctl restart wae:*
```

ステップ 3 両方のノードで、次のいずれかを実行します。

- Cisco WAE CLI から：

```
# wae ha-config nodes n1-name <hostname1>
# wae ha-config nodes n1-address <server-ip1>
# wae ha-config nodes n1-wae-uname <user1>
# wae ha-config nodes n2-name <hostname2>
# wae ha-config nodes n2-address <server-ip2>
# wae ha-config nodes n2-wae-uname <user2>
# wae ha-config cluster-id <cluster-id>
# wae ha-config temp-dir-location <temp-location>
```

プライマリノードで be-primary を開始し、ステータスを確認します。

```
# wae ha-config be-primary
# wae ha-config status
```

セカンダリノードで be-secondary を開始し、ステータスを確認します。

```
# wae ha-config be-secondary
# wae ha-config status
```

（注） `temp-dir-location` は、アーカイブプランファイルがプライマリからセカンダリの場所にコピーされる場所へのパスです。そのため、複製中にすべてのアーカイブファイルを保持するのに十分なスペースがあるディレクトリへのパスを追加することをお勧めします。

- WAE UI から：

1. [HA設定（HA configuration）] アイコンをクリックします。

   （注） `/etc/hosts` ファイルがホスト名から IP アドレスへのマッピングで更新されていることを確認してください

   。

2. N1 ノードと N2 ノードの詳細を入力します。

（注）　両方のノードのノード名に完全修飾ドメイン名を指定します。

3. [クラスタID（Cluster-ID）] を入力します。
4. [プライマリ指定（be-primary）]、[セカンダリ指定（be-secondary）]、または[指定なし（be-none）] を選択します。
5. [セカンダリ（Secondary）] をクリックします。

（注）
- ノードがセカンダリノードとして選択されると、[WAE UI] → [HA設定（HA Configuration）] および [WAE UI] → [ステータス（Status）] ページのみがそのノードに対して有効になります。
- ノードを [指定なし（be-none）] から [セカンダリ指定（be-secondary）] に移動するときは、ノードで収集が実行されていないこと、およびエージェント/スケジューラタスクが設定されていないことを確認してください。

ステップ 4 ノードのステータスは、[HA設定（HA Configuration）] ページに表示されます。

（注）　両方のノードで HA を設定した後は、プライマリノードでのみ設定を変更できます。

ステップ 5 データ同期のために、両方のノード間でパスワードなしの ssh を設定します。

SSH 認証キーの生成

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
1x:x2:x4:22:5a:7x:2x:a5:a5:4x:6x:88:2c:33:x8:77 root@remote-host
```

公開キーをリモートホストにコピーする

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub user@remote-host
user@remote-hosts's password:
```

または、サーバーに openssh-clients（ssh-copy-id コマンドユーティリティを提供するパッケージ）がインストールされていない場合は、次のコマンドで認証キーをコピーできます。

```
# cat ~/.ssh/id_rsa.pub | ssh user@remote-host "cat >> ~/.ssh/authorized_keys"
```

（注）　フェイルオーバーの場合、データ同期は反対方向に行われるため、両方のノードにパスワードなしの ssh を設定する必要があります。

ステップ 6 グローバルスケジューラを使用してデータ複製をスケジュールする： `<wae-run-directory>/networks`、`<wae-run-directory>/agents` および任意のアーカイブの下で、プライマリノードでの HA データ同期を使用してファイルを複製します。

データ複製には `rsync` ユーティリティが必要です。

システムで `rsync` ユーティリティが使用できない場合は、次のコマンドを使用してインストールします。

```
sudo yum install rsync
```

（注）
- プライマリノードがダウンすると、セカンダリノードがプライマリとして起動します。プライマリノードは、復元されると [なし（None）] として起動します。このノードを、セカンダリとして再度手動で設定する必要があります。
- フェイルオーバー後に完全なネットワークモデルを取得するには、XTC ベースの NIMO で収集の実行をスケジュールする必要があります。収集の実行がスケジュールされていない場合、収集が手動で実行されるまで、XTC ベースのリアクティブな変更は機能しません。
- ノードが [なし（None）] 状態から [セカンダリ（Secondary）] 状態に移行するときは、そのノードで収集またはエージェントが実行されていないことを確認してください。
- Netflow ワークフローと layout-nimo は、HA ではサポートされていません。

ハイ アベイラビリティのトラブルシューティング

HA 構成のトラブルシューティング時に、次の 2 つのログを確認する必要があります。

- <*wae_run_directory*>/logs/wae-java-vm.log
- <*wae_run_directory*>/logs/devel.log

次の表に、HA エラーとその意味を示します。

エラー コード	ENUM	説明
25	CONFD_ERR_HA_CONNECT	リモート HA ノードへの接続に失敗しました。
26	CONFD_ERR_HA_CLOSED	リモート HA ノードが WAE への接続を閉じたか、confd_ha_beseecondary() コール中にプライマリからの同期応答の待機にタイムアウトがありました。
27	CONFD_ERR_HA_BADFXS	リモート HA ノードには、WAE とは異なる FXS ポートのセットまたは異なるバージョンの FXS ポートがあります。
28	CONFD_ERR_HA_BADTOKEN	リモート HA ノードに WAE とは異なるトークンがあります。
29	CONFD_ERR_HA_BADNAME	リモート HA ノードの名前が、WAE でキャプチャされた名前とは異なります。

エラー コード	ENUM	説明
30	CONFD_ERR_HA_BIND	着信 HA 接続用の HA ソケットのバインドに失敗しました。
31	CONFD_ERR_HA_NOTICK	リモート HA ノードがインターバルライブティックの生成に失敗しました。

LDAP の設定

Cisco WAE は、Lightweight Directory Access Protocol（LDAP）を使用して外部ユーザーの認証をサポートしています。

WAE で LDAP を設定する前に、次のことを確認してください。

- LDAP ディレクトリツリーとその内容に精通している必要があります。
- LDAP サーバーと収集の詳細をインストールして設定します。
- LDAPS プロトコルを使用するには、SSL 証明書を取得してキーストアに追加します。

  SSL 証明書を取得してインポートするコマンド

  次のコマンドを使用して、自己署名証明書を cert.pem ファイルに保存します。

  ```
  # openssl s_client -connect <ldap-host>:<ldap-ssl-port> </dev/null 2>/dev/null | sed
   -n '/^-----BEGIN/,/^-----END/ { p }' > cert.pem
  ```

  次のコマンドを使用して、デフォルトのキーストアパスを取得します。通常、デフォルトのキーストアパスは /etc/pki/java/cacerts for CentOS 7 with open-jdk です。

  ```
  # $WAE_ROOT/lib/exec/test-java-ssl-conn <ldap-host> <ldap-ssl-port> 2>1 | grep
  "trustStore is:"
  ```

  次のコマンドを使用して、証明書をデフォルトのキーストアにインポートします

  ```
  # sudo keytool -import -keystore <default-key-store-path> -storepass changeit -noprompt
   -file cert.pem
  ```

LDAP 設定のトラブルシューティングを行うには、次のログを表示します。

- LDAP 設定ログ：<*wae_run_directory*>/logs/wae-javavm.log
- LDAP 認証ランタイム ログ：<wae_run_directory> /logs/wae-ldap-auth.log

CLI を使用した LDAP の設定

始める前に

LDAP の設定 （179 ページ） の説明に従って、前提条件を満たしていることを確認します。

ステップ 1 wae.conf ファイルを編集して、外部認証を有効にします。

```
<<external-authentication>
  <enabled>true</enabled>
  <executable>$WAE_ROOT/lib/exec/wae-ldap-auth.sh</executable>
</external-authentication>
```

ステップ 2 WAE を再起動します。

```
# wae --start
```

ステップ 3 WAE CLI を使用して LDAP サーバーの詳細を設定します。

例：LDAP 設定

```
# wae_cli -u admin
# conf

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldap
(config)# wae ldap-config server 10.220.121.47
(config)# wae ldap-config port 389
(config)# wae ldap-config search-base ou=people,dc=planetexpress,dc=com
(config)# wae ldap-config principal-expression "(uid={0})"
(config)# commit
```

例：SSL および admin ユーザーを使用した LDAP 設定

```
# wae_cli -u admin
# conf

(config)# devices authgroups group ldap-search default-map
(config)# devices authgroups group ldap-search default-map remote-name cn=admin,dc=company,dc=com
(config)# devices authgroups group ldap-search default-map remote-password HelloDolly
(config)# commit

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldaps
(config)# wae ldap-config server 10.222.121.48
(config)# wae ldap-config port 636
(config)# wae ldap-config search-base ou=people,dc=company,dc=com
(config)# wae ldap-config principal-expression "(uid={0})"
(config)# wae ldap-config ldap-auth-group ldap-search
(config)# wae ldap-config keystore-path /home/centos/apps/wae712/wae/etc/wae-ldap-keystore
(config)# wae ldap-config keystore-pass wae-ldap-ks#
(config)# commit
(config)# exit
```

例：MS Active Directory サーバーの LDAP 設定

```
# wae_cli -u admin
# conf

(config)# devices authgroups group  ad-user ldap-search default-map
(config)# devices authgroups group  ad-user ldap-search default-map remote-name
CN=waeuser1,CN=Users,DC=woadtest,DC=local
(config)# devices authgroups group  ad-user ldap-search default-map remote-password HelloWAE
(config)# commit

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldap
(config)# wae ldap-config server waelab.cisco.com
```

```
(config)# wae ldap-config port 389
(config)# wae ldap-config search-base cn=users,dc=woadtest,dc=local
(config)# wae ldap-config principal-expression "(sAMAccountName={0})"
(config)# wae ldap-config ldap-auth-group ad-user
(
(config)# commit
(config)# exit
```

WAE UI を使用した LDAP の設定

始める前に

LDAP の設定 （179 ページ） の説明に従って、前提条件を満たしていることを確認します。

ステップ 1 wae.conf ファイルを編集して、外部認証を有効にします。

```
<<external-authentication>
  <enabled>true</enabled>
  <executable>$WAE_ROOT/lib/exec/wae-ldap-auth.sh</executable>
</external-authentication>
```

ステップ 2 WAE を再起動します。

ステップ 3 WAE UI から、LDAP 設定アイコン（  ）をクリックします。

ステップ 4 デフォルトでは、有効なトグルスイッチはオンになっています。オンでない場合は、ユーザー認証にLDAPサーバーの使用を有効にして、スイッチをオンに切り替えます。

ステップ 5 LDAPオプションを入力します。詳細については、LDAP設定オプション （181ページ）を参照してください。

ステップ 6 [保存（Save）] をクリックします。

LDAP 設定オプション

表 *3 : LDAP* フィールドの説明

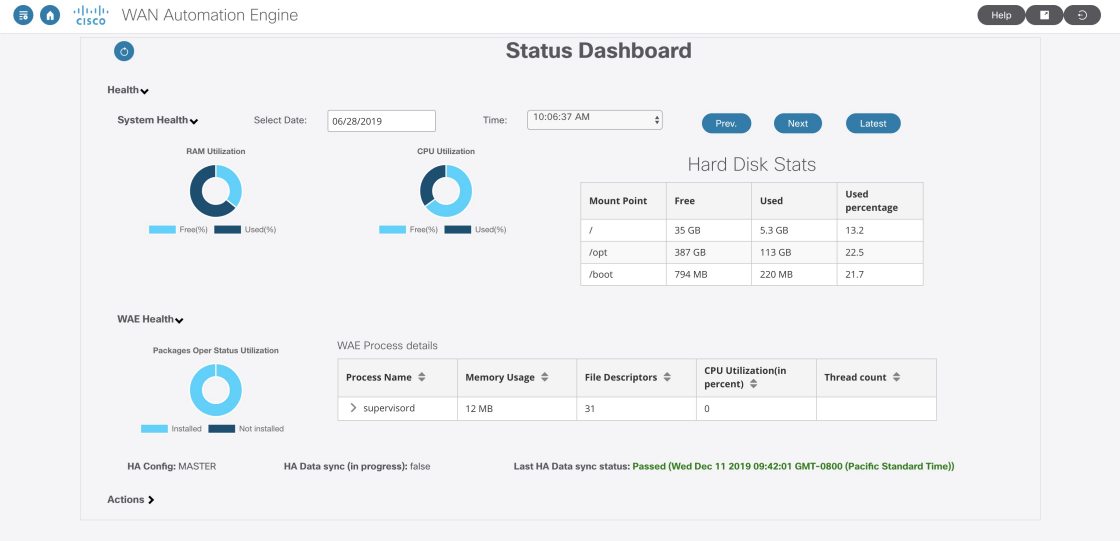
フィールド	説明
プロトコル	LDAP サーバーに到達するために使用するプロトコルです。 • LDAP：通信をクリアテキストで送信します。 • LDAPS：暗号化された安全な通信を送信します。 デフォルト値は LDAPS です。

フィールド	説明
サーバ <ldap-server>	LDAP サーバーの IP アドレスまたは FQDN。これはサーバーのホスト名の末尾に DNS ドメイン名を付加したものです。 FQDN 形式：<LDAP_hostname> <domain>.com
ポート	LDAP サーバーに到達するために使用するポートです。暗号化されていない認証の場合、デフォルトは TCP 389 です。暗号化されている認証の場合、デフォルトは TCP 636 です。 デフォルト値は 389 です。
検索ベース（Search Base） <ldap-base-ou>	WAE サーバーへのログイン許可を持っている必要があるすべてのユーザーアカウントの、基本検索 OU の識別名です。
主表現	デフォルト： `LDAP.Principal.Expr: (userPrincipalName={0})`, `{0}` トークンは、ログインページでユーザーが入力したユーザー名に置き換えられます。 `userPrincipalName=` は、LDAP 検索ベースでユーザーを識別するユーザーオブジェクトの LDAP 属性と一致する必要があります。 LDAP スキーマから、ユーザー固有の属性 `uid` を使用します。 WAE サーバーは、LDAP 検索ベースツリーの下のすべてのオブジェクトを検索します。 `uid=cisco-mate-user1` 一般的な選択肢には、`userPrincipalName` または `userName` などがあります。
LDAP 認証グループ（LDAP Auth Group）	LDAP 検索の実行に使用するユーザー名およびパスワード。 enable-password フィールドは LDAP には使用されません。UI での設定時に任意のダミー値を入力します。
キーストアパス（Keystore Path）	SSL が有効な場合のキーストアパス。
キーストアパスワード（Keystore Pass）	キーストアのパスワード。

ステータスダッシュボード

Cisco WAE が突然動作を停止したり、クラッシュしたりする場合があります。トラフィックポーラーが機能しなくなったり、アーカイブ中に問題が発生したりすることがあります。これらの問題は、WAE システムリソースが使い果たされていることが原因である場合があります。WAE のステータスダッシュボードは、システムリークの原因となるプロセスや、リソースを使い切っているプロセスを特定することにより、このような状況の対処に役立ちます。

ステータスダッシュボードにアクセスするには、Cisco WAE UI に移動し、[ステータスダッシュボード（Status Dashboard）] をクリックします。



ステータスダッシュボードは、主に 2 つのセクションに分かれています。

- ヘルス（Health）
- アクション（Actions）

[ヘルス（Health）] は、さらに [システムヘルス（System Health）] と [WAEヘルス（WAE Health）] に分けられます。

[システムヘルス（System Health）] は、RAM 使用率、CPU 使用率、およびシステムレベルのハードディスク統計をキャプチャします。ツールは 10 分ごとに実行され、統計が生成されます。[日付の選択（Select Date）] および [時刻（Time）] フィールドを使用して、必要なレポートにアクセスします。

デフォルトでは、最新のレポートが表示されます。レポート間を移動するには、[前へ（prev）]、[次へ（next）] ボタンを使用します。

RAM 使用率と CPU 使用率のグラフには、使用済みスペースと空きスペースが表示されます。使用済みまたは空き領域にカーソルを合わせると、実際の使用率が表示されます。

[WAEヘルス（WAE Health）] は、メモリ使用量、ファイル記述子、CPU 使用率など、プロセスレベルの使用状況の詳細をキャプチャします。プロセスで問題が発生した場合は、日付と時刻のフィールドを使用して、関連するすべてのプロセスレベルの詳細にアクセスします。

Cisco WAE の展開後、一部のパッケージがアンインストールされたままになる場合があります。[パッケージ運用使用率ステータス（Packages Oper Status Utilization）] チャートは、インストール済みおよびアンインストール済みパッケージの割合を示します。インストール済みまたは未インストールの領域をクリックすると、インストール済みまたは未インストールのパッケージのリストが表示されます。

HA 設定が有効になっている場合、システムがプライマリかセカンダリかを示します。

[アクション（Actions）] セクションには、NIMO アクションとエージェントアクションのステータスが表示されます。

[NIMOアクション（NIMO Actions）] セクションで、NIMO カードをクリックして NIMO ステータスの詳細を取得します。

（注） [NIMOアクション（NIMO Actions）] は、過去のアクションデータではなく、現在のアクションのみを表示します。

[エージェントアクション（Agent Actions）] セクションには、特定のエージェントのアクションステータスが表示されます。

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。
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。

記号	フォーマット文字列	備考
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。
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が中止されました。
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）。

記号	フォーマット文字列	備考
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ています。構成を元に戻しています。
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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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"	セカンダリは、異なる fxs ファイルを持つプライマリに接続されています。
TOKEN_MISMATCH	"Token mismatch, slave is not allowed"	セカンダリは、不正な認証トークンでプライマリに接続されています。
HA_SLAVE_KILLED	"Slave ~s killed due to no ticks"	セカンダリノードは、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 を持つセカンダリが到着しました。
HA_FAILED_CONNECT	"Failed to connect to master: ~s"	セカンダリがプライマリに接続できなかったため、ライブラリをセカンダリにする呼び出しの試行に失敗しました。
HA_BAD_VSN	"Incompatible HA version (~s, expected ~s), slave is not allowed"	セカンダリは、互換性のない HA プロトコルバージョンでプライマリ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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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。
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。

記号	フォーマット文字列	備考
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 （200 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 （201 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
  ```

WAE 診断

Cisco WAE には、次の操作が可能な診断ユーティリティが含まれています。

- システムの正常性に関する診断チェックを実行し、推奨事項を作成する。
- エンジニアが問題をトラブルシューティングするために必要となる、必須のデータや正常性チェックレポートをすべて収集する。

このツールには、正常性チェックスクリプトを追加できる拡張可能なフレームワークが備わっています。追加の正常性チェックスクリプトは、Python またはシェルのいずれかに含めることができ、<wae-install-directory> /bin/diagnostics/ ディレクトリに配置する必要があります。

WAE 診断ツールの使用

WAE 診断は、wae-diagnostics コマンドを使用して実行できます。

（注） コマンドを実行する前に waerc をソースに設定します。

```
wae-diagnostics [-h] [-c] [-D] [-j] [-e] [-d] [--run-diagnostics] [-o OUT_DIR] -r RUN_DIR
 -i INSTALL_DIR
```

where

必須の引数	
-r RUN_DIR	run-dir：RUN_DIR 実行ディレクトリパス
INSTALLDIR	install-dir：INSTALL_DIR インストールディレクトリ パス
オプションの引数	
-h	help-- このヘルプメッセージを表示して終了
-c	collect-logs-- すべてのログとトラブルシューティング データを収集して照合。 --collect-db-files を使用して指定されていない限り、DB ファイルを除外します。
-D	collect-db-files-- db ファイルを収集
-j	java-stats-- Java スレッド統計情報を収集
-e	enable-debug-- ログレベルをデバッグに設定
-d	disable-debug-- デバッグログレベルを無効化
--run-diagnostics	WAE で診断を実行
-o OUT_DIR	out-dir--OUT_DIR 出力ディレクトリ パス。データアーカイブはこのディレクトリに作成されます。指定しない場合、データアーカイブは現在のディレクトリに作成されます。

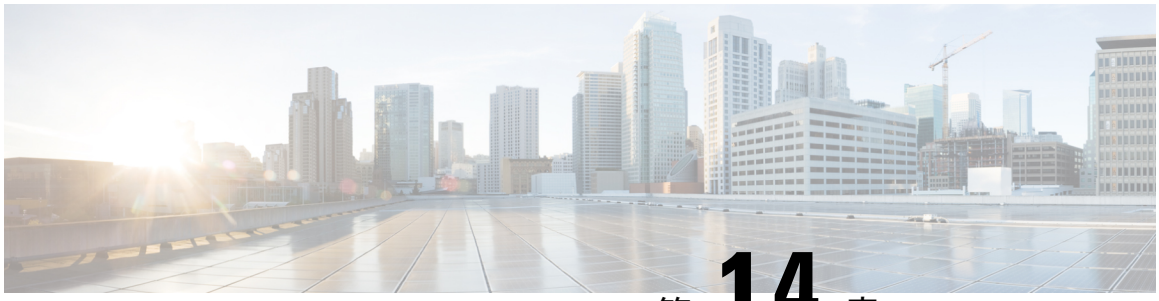
例：

- 次のコマンドは、WAEインストールで診断チェックを実行し、ログを含む診断情報を収集します。

```
wae-diagnostics -r <run-directory-path> -i <install-dir-path>
```

- 次のコマンドは、WAEインストールで診断チェックを実行し、ログ、ネットワークデータ、およびJVMデータを含む診断情報を収集します。

```
wae-diagnostics -cDj -r <run-directory-path> -i <install-dir-path>
```



第 14 章

セキュリティ

• 主要なセキュリティ概念 （205 ページ）

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

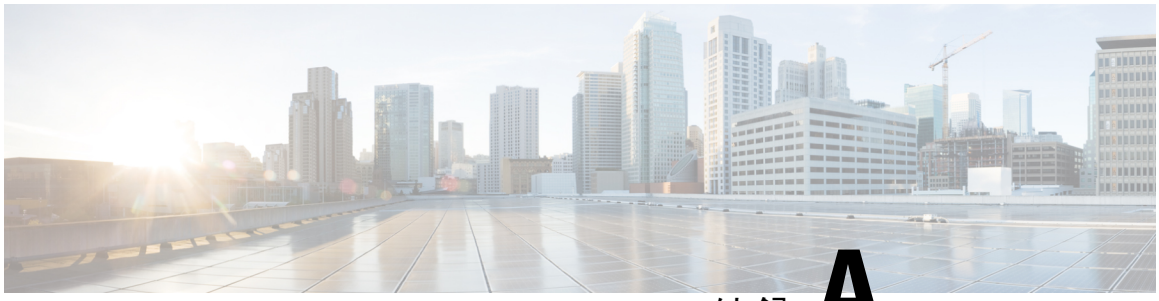
これは、クライアントが適切なサーバー（中間サーバーではなく）に接続していることを保証する必要がある場合に使用される認証方法で、オンライン バンキングの Web サイトなどのパブリック リソースに適しています。認証は、クライアントがサーバー上のリソースへのアクセスを要求したときに開始されます。リソースが存在するサーバーは、その ID を証明するために、サーバー証明書（別名 SSL 証明書）をクライアントに送信します。クライアントは受信したサーバー証明書を、クライアントまたはブラウザにインストールする必要がある別の信頼できるオブジェクト（サーバー ルート証明書）と照合して検証します。サーバーの検証後、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、 サーバーは HTML フォームへの有効なユーザー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できます。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、接続するサーバーの 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 （209 ページ）
- デバイス アクション （210 ページ）
- サービスアクション （211 ページ）
- wae.conf 構成パラメータ （212 ページ）

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、接続されたセカンダリノード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、更新がセカンダリノードに完全に反映され、セカンダリノードのサブスクライバ（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。

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、キー交換は終了します。
/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。

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。