ゲスト シェル

ゲスト シェルは仮想化された Linux ベースの環境で、Python などの カスタム Linux アプリケーションを実行して Cisco デバイスを自動で制御および管理するために設計されています。システムの自動プロビジョニング（デイ ゼロ）も含まれます。このコンテナ シェルは、ホスト デバイスから分離された安全な環境を提供します。ユーザはそこで、スクリプトまたはソフトウェア パッケージをインストールし、実行することができます。

このモジュールでは、ゲスト シェルとそれを有効にする方法について説明します。

ゲストシェルの制約事項

- ゲストシェルは、Cisco Catalyst 9200L SKU ではサポートされません。
- スタンバイルートプロセッサ（RP） では、NETCONF セッションを確立できません。
- スケールが 2000 Aps に設定され、クライアントが 10000 に設定されている場合、**show tech-support wireless** コマンドなどを実行すると、Python スクリプトでエラーが発生します。

  show tech-support wireless などのコマンドでは、大量のデータが出力されるため、ゲストシェル内のメモリが枯渇する可能性があります。大量のデータが出力されるコマンドを使用する場合は、出力結果をファイルにリダイレクトします。IOS CLI は、*/bootflash/guest-share* ディレクトリ内のファイルに出力結果を書き込むことができ、ゲストシェルからアクセスできます。
- Cisco Catalyst 9200CX シリーズ スイッチは、管理インターフェイス、AppGigabitEthernet インターフェイス、VirtualPortGroup インターフェイスをサポートしていません。ゲスト

シェルで実行されているアプリケーションやスクリプトは、外部ネットワークと通信できません。

ゲスト シェルについて

ゲスト シェルの概要

ゲストシェルは仮想化されたLinuxベースの環境であり、Pythonなどの カスタムLinuxアプリケーションを実行してシスコのデバイスを自動で制御および管理するために設計されています。ゲストシェルを使用して、サードパーティ製Linuxアプリケーションをインストール、更新、および操作することもできます。ゲストシェルはシステムイメージとともにバンドルされ、Cisco IOS コマンド **guestshell enable** を使用してインストールできます。

ゲスト シェル環境は、ネットワーキングではなく、ツール、Linuxユーティリティ、および管理性を意図したものです。

ゲスト シェルは、ホスト（Cisco スイッチおよびルータ）システムとカーネルを共有します。ユーザーはゲストシェルのLinuxシェルにアクセスし、コンテナのルートファイルシステムにあるスクリプトおよびソフトウェアパッケージを更新できます。ただし、ゲスト シェル内のユーザーは、ホストのファイル システムおよびプロセスを変更することはできません。

ゲスト シェル コンテナは、IOxを使用して管理されます。IOxは、Cisco IOS XEデバイスのためのシスコのアプリケーション ホスティング インフラストラクチャです。IOxは、シスコ、パートナー、およびサードパーティの開発者によって開発されたアプリケーションおよびサービスをネットワーク エッジ デバイスでシームレスにホスティングすることを、各種の多様なハードウェア プラットフォームにおいて可能にします。

ゲストシェルのソフトウェア要件

ゲストシェルコンテナを使用すると、ユーザは、システム上で自分のスクリプトやアプリケーションを実行できるようになります。Intel x86プラットフォーム上のゲストシェルコンテナは、CentOS 8.0の最小限のrootfsを持つLinuxコンテナ（LXC）になります。ランタイム中に、CentOS 8.0 でYumユーティリティを使用して、Python バージョン3.0などの他のPythonライブラリをインストールすることができます。また、PIPを使用してPythonパッケージをインストールまたは更新することもできます。

表 *1:* ゲストシェルのソフトウェア要件

	ゲスト シェル（LXC コンテナ）
オペレーティング システム	Cisco IOS XE
プラットフォーム	サポートされているすべての Cisco IOS XE プラットフォーム

	ゲスト シェル（**LXC** コンテナ）
ゲスト シェル環境	• CentOS 7 は Cisco IOS XE Amsterdam 17.2.1 以前のリリースでサポートされています。 • CentOS 8 は Cisco IOS XE Amsterdam 17.3.1 以降のリリースでサポートされています。 （注） CentOS は Python 3.6 のみをサポートします。
Python 2.7	Cisco IOS XE Amsterdam 17.3.1 までサポート
Python 3.6	Cisco IOS XE Amsterdam 17.1.1 以降のリリースでサポートされています。 Cisco IOS XE Amsterdam 17.1.1 および Cisco IOS XE Amsterdam 17.2.1 では、Python V2 がデフォルトです。ただし、Cisco IOS XE Amsterdam 17.3.1 以降のリリースでは、Python V3 がデフォルトです。 （注） Cisco Catalyst 9200 シリーズ スイッチは、Cisco IOS XE Amsterdam 17.3.1 以降のリリースで Python バージョン 3 をサポートしています。
事前にインストールされたカスタムの Python ライブラリ	• Cisco 組込イベント マネージャ • Cisco IOS XE CLI • NETCONF API の NCCLIENT ライブラリ
サポートされる rootfs	SSH、Yum のインストール、および Python PIP のインストール
GNU C コンパイラ	サポート対象外
RPM のインストール	対応
アーキテクチャ	x86 および ARM

ゲスト シェルのセキュリティ

シスコは、ゲスト シェル内のユーザまたはアプリケーションによってホスト システムが攻撃されることがないよう、セキュリティを提供しています。ゲスト シェルは、ホスト カーネルから分離され、非特権コンテナとして動作します。

ゲスト シェルのハードウェア要件

この項では、可変メモリ構成を持つ、サポート対象のプラットフォームにおけるハードウェア要件に関する情報を提供します。

表 *2:* ゲストシェルのリソース要件

プラットフォーム	最小メモリ
Cisco 1000 シリーズ サービス統合型ルータ	4 GB
Cisco Cloud Services Router 1000V シリーズ	4 GB
Cisco ISR 4000 シリーズ サービス統合型ルータ	8 GB DRAM（Cisco IOS XE Fuji 16.8.1 以前のリリース。） 4 GB DRAM（Cisco IOS XE Fuji 16.8.1 以降のリリース。）

他のすべてのプラットフォームは、ゲストシェルをサポートするのに十分なリソースを備えた状態で出荷されます。

（注） 仮想サービスがインストールされているアプリケーションとゲストシェルコンテナを同時に使用することはできません。

ゲスト シェルのストレージ要件

Cisco Catalyst 9300 シリーズ スイッチおよび Cisco Catalyst 9500 シリーズ スイッチでは、ゲストシェルを正常にインストールするには 1100 MB のハードディスク空き容量が必要です。

Cisco 4000 シリーズ サービス統合型ルータでは、ゲストシェルは、ネットワーク インターフェイス モジュール（NIM）の SSD（ハードディスク）がある場合、そこにインストールされます。ハードディスク ドライブが使用可能な場合、ゲスト シェルのインストールにブートフラッシュを選択することはできません。Cisco 4000 シリーズ サービス統合型ルータでは、ゲストシェルを正常にインストールするには 1100 MB のハードディスク（NIM-SSD）空き容量が必要です。

Cisco 4000 シリーズ サービス統合型ルータおよび Cisco ASR 1000 シリーズ アグリゲーション サービス ルータ（オプションのハードディスクがそのルータに追加されている場合）では、ゲストシェルをハードディスクにインストールしており、そのハードディスクがルータに挿入されている場合にのみリソースのサイズ変更を実行できます。

（注） ブートフラッシュを介してインストールしたゲストシェルでは、アプリケーションホスティング設定コマンドを使用したリソースのサイズ変更はできません。

ゲスト シェルのインストール中にハード ディスク容量が不足した場合、エラー メッセージが表示されます。

次に、Cisco ISR 4000 シリーズ サービス統合型ルータでのエラーメッセージの例を示します

```
% Error:guestshell_setup.sh returned error:255, message:
Not enough storage for installing guestshell. Need 1100 MB free space.
```

ブートフラッシュまたはハード ディスクの空き領域は、ゲスト シェルが追加データを格納するために使用されることがあります。Cisco 4000 シリーズ サービス統合型ルータでは、ゲスト シェルに 800 MB のストレージ空き領域があります。ゲスト シェルはブートフラッシュにアクセスするため、その空き領域の全体を使用できます。

表 *3:* ゲスト シェルおよびゲスト シェル *Lite* が使用できるリソース

リソース	デフォルト	最小/最大
CPU	1 % （注） 1 % は非標準。800 CPU ユニット/システム CPU ユニットの全体	1/100 %
メモリ	256 MB 512 MB（Cisco Cloud Services Router 1000V シリーズ）	256/256 MB 512/512 MB（Cisco Cloud Services Router 1000V シリーズ）

ゲスト シェルの有効化と実行

guestshell enable コマンドは、ゲスト シェルをインストールします。このコマンドは、無効化されているゲスト シェルを再アクティブ化する際にも使用されます。

ゲスト シェルが有効化された状態でシステムをリロードすると、ゲスト シェルは有効化されたままになります。

（注） **guestshell enable** コマンドを使用する前に、IOx を設定しておく必要があります。

guestshell run bash コマンドは、ゲスト シェルの bash プロンプトを開きます。このコマンドを動作させるには、ゲスト シェルが事前に有効化されていることが必要です。

（注） 次のメッセージがコンソールに表示される場合、IOx が有効化されていません。**show iox-service** コマンドの出力をチェックして、IOx の状態を確認してください。

```
The process for the command is not responding or is otherwise unavailable
```

ゲストシェルを有効にする方法の詳細については、「Configuring the AppGigabitEthernet Interface for Guest Shell」および「Enabling Guest Shell on the Management Interface」のセクションを参照してください。

ゲスト シェルの無効化と破棄

guestshell disable コマンドを使用することで、ゲスト シェルを終了して無効化できます。ゲスト シェルが無効化された状態でシステムをリロードすると、ゲスト シェルは無効化されたままになります。

guestshell destroy コマンドは、フラッシュのファイル システムから rootfs を削除します。すべてのファイル、データ、インストールされている Linux アプリケーション、およびカスタムの Python ツールとユーティリティが削除され、回復できなくなります。

デバイスでのゲスト シェルへのアクセス

ネットワーク管理者は、Cisco IOS コマンドを使用して、ゲストシェル内のファイルおよびユーティリティを管理することができます。

ゲスト シェルのインストール中に、SSH アクセスがキー ベースの認証でセットアップされます。ゲストシェルへのアクセスは、Cisco IOS の最も高い特権（15）を持つユーザに制限されます。このユーザは、sudo の実行者である guestshell Linux ユーザとして Linux コンテナへのアクセスを許可され、すべてのルート操作を実行できます。ゲストシェルから実行されるコマンドは、ユーザが Cisco IOS 端末にログインしたときと同じ特権で実行されます。

ゲスト シェル プロンプトでは、標準的な Linux コマンドを実行できます。

管理ポートを介してのゲスト シェルへのアクセス

ゲストシェルは、デフォルトで、アプリケーションによる管理ネットワークへのアクセスを許可します。ユーザは、ゲスト シェル内から管理 VRF のネットワーキング設定を変更することはできません。

（注） 管理ポートがないプラットフォームの場合、VirtualPortGroup を Cisco IOS 設定内のゲストシェルに関連付けることができます。詳細については、「VirtualPortGroup の設定例」の項を参照してください。

Cisco Catalyst 9200 シリーズ スイッチ、Cisco Catalyst 9300 シリーズ スイッチ、および Cisco Catalyst 9400 シリーズ スイッチは、ゲストシェルにアクセスするために AppGigabitEthernet インターフェイスおよび管理インターフェイス（mgmt-if）をサポートします。

Catalyst 9500 シリーズ スイッチ、Catalyst 9500 ハイ パフォーマンス シリーズ スイッチ、および Catalyst 9600 シリーズ スイッチでは、AppGigabitEthernet インターフェイスはサポートされません。

（注） Cisco Catalyst 9200L SKU はゲストシェルをサポートしていません。

前面パネルポートまたは光ファイバアップリンクを使用した デイ ゼロ ゲスト シェル プロビジョニング

デイゼロでは、デバイスに管理接続がなく、唯一の接続が前面パネルポートまたはファイバアップリンクポートのいずれかを介して行われる場合、ゲストシェルは使用可能なポートを使用するように内部的に設定されます。AppGigabitEthernet インターフェイスは、ゲストシェルをサーバに接続します。

ゲストシェルがサーバに接続されると、デバイスは構成スクリプトをダウンロードし、デバイスを設定します。この設定には、仮想マシン（VM）のダウンロード、設定、起動も含まれます。デイゼロ設定が完了すると、設定に基づいてシステムがリブートする場合があります。システムがユーザ固有の設定のみで起動することを確認します。

USB ポートを使用したゲストシェル接続

デバイスは、シリアルアダプタを使用して複数の他のデバイスに接続します。このシリアルアダプタは、デバイスの前面パネルにある USB ポートを介して接続されます。

VM はシリアルアダプタを制御し、VM の実行中に USB インターフェイスにアタッチされている接続済みデバイスに変更があると、VM に通知されます。

ゲスト シェルでのスタッキング

ゲストシェルは、1+1 高可用性をサポートします。1+1 高可用性とは、一方のデバイスがアクティブと指定され、もう一方がスタンバイと指定されている場合を意味します。N+1 高可用性はサポートされていません。

ゲストシェルがインストールされている場合、フラッシュのファイルシステムには *guest-share* ディレクトリが自動的に作成されます。このディレクトリは、スタックメンバー間で同期されます。*guest-share* フォルダに保存されているファイルは、アクティブデバイスがダウンしてスタンバイデバイスが引き継いだ場合でも保持されます。高可用性スイッチオーバーの際に最大 50 MB のデータを保持するには、このディレクトリにデータを格納します。*guest-share* フォルダのサイズが 50 MB を超える場合は、スタックメンバーに同期されません。

高可用性スイッチオーバーの際は、新しいアクティブデバイスがそれぞれ独自のゲストシェルインストールを作成し、ゲストシェルを同期状態に復元します。古いファイルシステムは保持されません。ゲストシェルの状態は、すべてのスタックメンバー間で内部的に同期されます。

Cisco IOx の概要

Cisco IOx（IOs＋linuX）はエンドツーエンドアプリケーション フレームワークであり、Cisco ネットワーク プラットフォーム上のさまざまなタイプのアプリケーションに対し、アプリケーション ホスティング機能を提供します。Cisco ゲストシェルは特殊なコンテナ展開であり、システムの開発に役立つアプリケーションの 1 つです。

Cisco IOx は、構築済みアプリケーションをパッケージ化し、それらをターゲットデバイス上にホストする開発者の作業を支援する一連のサービスを提供することにより、アプリケーションのライフサイクル管理とデータ交換を容易にします。IOxのライフサイクル管理には、アプリケーションおよびデータの配布、展開、ホスティング、開始、停止（管理）、およびモニタが含まれます。IOx サービスにはアプリケーションの配布および管理ツールも含まれており、ユーザがアプリケーションを発見して IOx フレームワークに展開するのに役立ちます。

Cisco IOx アプリケーション ホスティングは、次の機能を提供します。

- ネットワークの不均質性の遮蔽。
- デバイス上にホストされているアプリケーションのライフサイクルをリモートで管理する Cisco IOx アプリケーション プログラミング インターフェイス（API）。
- 一元化されたアプリケーションのライフ サイクル管理。
- クラウド ベースの開発。

IOx のトレースとロギングの概要

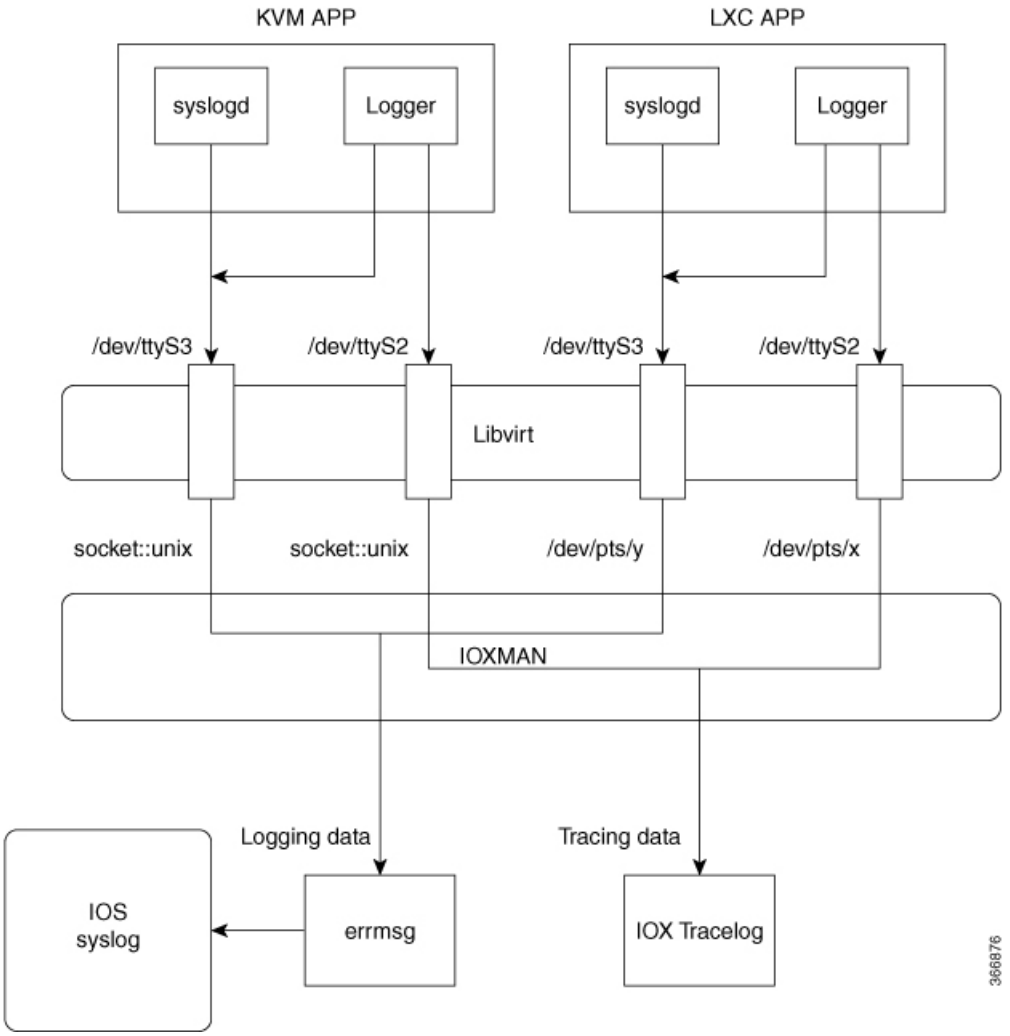
IOx のトレースとロギングの機能を使用すると、ホスト デバイスでゲスト アプリケーションを個別に実行できます。これにより、ホストへのデータのロギングとトレースをレポートするのに役立ちます。トレースデータはIOxトレースログに保存され、ロギングデータはホストデバイスの Cisco IOS syslog に保存されます。

トレース データをホスト デバイス上の適切なストレージ デバイスにリダイレクトすると、ゲスト アプリケーションのデバッグに役立ちます。

IOXMAN 構造体

ゲスト アプリケーション、システム LXC、または KVM インスタンスはそれぞれ独自の syslog およびログファイルを使用して設定されます。これらのファイルは表示可能なファイル システム内に保存され、ホスト デバイスからはアクセスできません。Cisco IOS syslog へのデータのロギングとホスト上のIOxトレースログへのデータのトレースをサポートするため、次の図に示すように、ホストにデータを配信するための 2 つのシリアルデバイス（**/dev/ttyS2** と **/dev/ttyS3**）がゲストアプリケーションで指定されています。

図 *1 : IOXMAN* 構造体



IOXMANは、トレースインフラストラクチャを確立してロギングサービスまたはトレースサービス（シリアルデバイスをエミュレートするLibvirtを除く）を提供するプロセスです。IOXMANは、ゲストアプリケーションのライフサイクルに基づいて、トレースサービスを有効または無効にし、ロギングデータを Cisco IOS syslog に送信し、トレースデータを IOx トレースログに保存し、各ゲストアプリケーションの IOx トレースログを維持します。

ゲストシェルからの NETCONF アクセス

NETCONF-YANG にはゲストシェル内からアクセスできるため、ユーザーは Python スクリプトを実行し、NETCONF プロトコルを使用してシスコカスタムパッケージ CLI を呼び出すことができます。

ゲスト シェル アプリケーションは、ユーザー名として guestshell を使用することで、ローカルホストおよび NETCONF ポートへのパスワードレス SSH 接続を行わずに SSH 接続を確立します。このユーザー名は、デバイスに設定されている実際のユーザーに対応していません。デバ

イスに guestshell ユーザーが設定されている場合でも、このパスワードレスアクセスへの接続はありません。PRIV15 権限レベルを持つユーザーのみがゲストシェル内から NETCONF にアクセスできます。

認証と認可はバイパスされません。代わりに、ゲストシェルにアクセスを許可するときに認証と許可が行われます。最大の権限を持つユーザーのみがこのアクセスを許可されます。

ユーザーは外部ポートを開かずにゲストシェルから NETCONF サービスにアクセスできます。デバイスの NETCONF-YANG サーバーに接続する前に、ゲストシェルで初期化コマンドを実行する必要があります。これらのコマンドは次のとおりです。

```
iosp_client -f netconf_enable guestshell <port-number> and
iosp_client -f netconf_enable_passwordless guestshell <username>
```

iosp_client -f netconf_enable guestshell *port-number* コマンドは、**netconf-yang ssh local-vrf guestshell** コマンドを設定し、NETCONF-YANG が稼働するまで接続をブロックします。

iosp_client -f netconf_enable_passwordless guestshell<username> コマンドは、ゲストシェルアクセスに必要な SSH キーを作成します。

ゲストシェルからの NETCONF-YANG アクセスを削除するには、次のコマンドを使用します。

```
iosp_client -f netconf_disable guestshell and
iosp_client -f netconf_disable_passwordless guestshell  <username>
```

iosp_client -f netconf_disable guestshell コマンドは、ゲストシェル内から NETCONF へのアクセスを無効にします。ただし、NETCONF-YANG の設定は引き続き存在します。NETCONF-YANG をシャットダウンするには、**no netconf-yang** コマンドを使用します。

iosp_client -f netconf_disable_passwordless guestshell *username* コマンドは、指定されたユーザーの SSH キーを削除します。ユーザーはパスワードなしで NETCONF にアクセスすることはできません。ただし、パスワードを使用すれば接続できます。

netconf_enable_guestshell python API は、iosp_client 関数、iosp_client -f netconf_enable guestshell 830 および iosp_client -f netconf_enable_passwordless guestshell guesthell の組み合わせを実行します。この API は、unfamiliar-to-user iosp_client 関数を隠蔽します。この関数が呼び出されると、すべてのコマンドが完了するまで応答を返しません。関数がエラーを返さない限り、NETCONF が確実に実行されて、パスワードレスのセットアップが完了しており、接続の作成を開始できます。

ロギングとトレースのシステム フロー

ここでは、IOx のロギングとトレースの仕組みについて説明します。

LXC のロギング

1. ゲスト OS が、ゲストアプリケーションで **/dev/ttyS2** を有効にします。
2. ゲスト アプリケーションが、**/dev/ttyS2** にデータを書き込みます。
3. Libvirt が、ホストで **/dev/pts/x** への **/dev/ttyS2** をエミュレートします。

4. IOXMAN が、エミュレートされたシリアル デバイス **/dev/pts/x** を XML ファイルから取得します。
5. IOXMAN が、使用可能なデータを **/dev/pts/x** からリッスンして読み取り、メッセージの重大度を設定して、メッセージをフィルタ処理し、解析してキューに格納します。
6. **errmsg** を使用してホストの **/dev/log** デバイスにメッセージを送信するタイマーが開始されます。
7. データが Cisco IOS syslog に保存されます。

KVM のロギング

1. ゲスト OS が、ゲストアプリケーションで **/dev/ttyS2** を有効にします。
2. ゲスト アプリケーションが、**/dev/ttyS2** にデータを書き込みます。
3. Libvirt が、ホストで **/dev/pts/x** への **/dev/ttyS2** をエミュレートします。
4. IOXMAN が、エミュレートされた TCP パスを XML ファイルから取得します。
5. IOXMAN が、UNIX ソケットを開き、リモートソケットに接続します。
6. IOXMANが、使用可能なデータをソケットから読み取り、メッセージのシビラティ（重大度）を設定して、メッセージをフィルタ処理し、解析して、キューに格納します。
7. **errmsg** を使用してホストの **/dev/log** デバイスにメッセージを送信するタイマーが開始されます。
8. データが Cisco IOS syslog に保存されます。

LXC のトレース

1. ゲスト OS が、ゲストアプリケーションで **/dev/ttyS3** を有効にします。
2. メッセージを **/dev/ttyS3** にコピーするように **syslogd** を設定します。
3. ゲスト アプリケーションが、**/dev/ttyS3** にデータを書き込みます。
4. Libvirt が、ホストで **/dev/pts/y** への **/dev/ttyS3** をエミュレートします。
5. IOXMAN が、エミュレートされたシリアル デバイス **/dev/pts/y** を XML ファイルから取得します。
6. IOXMAN が、使用可能なデータを **/dev/pts/y** からリッスンして読み取り、フィルタ処理し、解析して、メッセージを IOx トレースログに保存します。
7. IOx トレースログが満杯の場合は、IOXMANがトレースログ ファイルを **/bootflash/tracelogs** にローテーションします。

KVM のトレース

1. ゲスト OS が、ゲストアプリケーションで **/dev/ttyS3** を有効にします。
2. メッセージを **/dev/ttyS3** にコピーするように syslog を設定します。
3. ゲスト アプリケーションが、**/dev/ttyS3** にデータを書き込みます。
4. Libvirt が、ホストで TCP パスへの **/dev/ttyS3** をエミュレートします。
5. IOXMAN が、エミュレートされた TCP パスを XML ファイルから取得します。
6. IOXMAN が、UNIX ソケットを開き、リモートソケットに接続します。
7. IOXMANが、使用可能なデータをソケットから読み取り、メッセージのシビラティ（重大度）レベルを設定して、メッセージをフィルタ処理し、解析して、IOx トレースログに格納します。
8. IOx トレースログが満杯の場合は、IOXMAN がトレースログ ファイルを **/bootflash/tracelogs** にローテーションします。

メッセージのロギングとトレース

ここでは、Cisco IOS syslog へのメッセージのロギングとトレースについて説明します。

Cisco IOS Syslog でのメッセージのロギング

ゲストアプリケーションから受信したどのロギングメッセージでも、IOXMAN はメッセージのシビラティ（重大度）をデフォルトで NOTICE に設定してから Cisco IOS syslog に送信します。IOSd で受信されたメッセージはコンソールに表示され、次のメッセージ形式で syslog に保存されます。

***Apr 7 00:48:21.911: %IM-5-IOX_INST_NOTICE:ioxman: IOX SERVICE guestshell LOG: Guestshell test**

Cisco IOS syslog に準拠するために、IOXMAN はロギングメッセージのシビラティ（重大度）レベルをサポートしています。シビラティ（重大度）のあるロギングメッセージを報告するには、ゲストアプリケーションでメッセージの先頭にヘッダーを追加する必要があります。

```
[a123b234,version,severity]

a123b234 is magic number.
Version:        severity support version.  Current version is 1.
Severity:       CRIT is 2
                ERR is 3
                WARN is 4
                NOTICE is 5
                INFO is 6
                DEBUG is 7
```

次に、メッセージログの例を示します。

```
echo “[a123b234,1,2]Guestshell failed” > /dev/ttyS2
```

ゲストアプリケーションから Cisco IOS syslog にロギングデータを報告するには、次の手順を実行します。

1. C プログラミングを使用している場合は、**write()** を使用してロギングデータをホストに送信します。

```
#define SYSLOG_TEST       “syslog test”
     int fd;
     fd = open("/dev/ttyS2", O_WRONLY);
     write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
     close(fd);
```

2. シェルコンソールを使用している場合は、**echo** を使用してロギングデータをホストに送信します。

```
echo “syslog test” > /dev/ttyS2
```

IOx トレースログへのメッセージのトレース

ゲストアプリケーションから IOx トレースログにトレースメッセージを報告するには、次の手順を実行します。

1. C プログラミングを使用している場合は、**write()** を使用してトレースメッセージをホストに送信します。

```
#define SYSLOG_TEST       “tracelog test”
     int fd;
     fd = open("/dev/ttyS3", O_WRONLY);
     write(fd, SYSLOG_TEST, strlen(SYSLOG_TEST));
     close(fd);
```

2. C プログラミングを使用している場合は、**syslog()** を使用してトレースメッセージをホストに送信します。

```
#define SYSLOG_TEST       “tracelog test”

syslog(LOG_INFO, “%s\n”, SYSLOG_TEST);
```

3. シェルコンソールを使用している場合は、**echo** を使用してトレースデータをホストに送信します。

```
echo “tracelog test” > /dev/ttyS3
   or
logger “tracelog test”
```

ゲスト シェルを有効にする方法

IOx の管理

始める前に

IOxは開始まで最長で2分かかります。ゲストシェルを正常に有効にするには、CAF、IOXman、および Libvirtd サービスが実行している必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **iox**
4. **exit**
5. **show iox-service**
6. **show app-hosting list**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**iox** 例： `Device(config)# iox`	IOx サービスを設定します。
ステップ 4	**exit** 例： `Device(config)# exit`	グローバル コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 5	**show iox-service** 例： `Device# show iox-service`	IOx サービスのステータスを表示します。
ステップ 6	**show app-hosting list** 例：	デバイスに対して有効になっている app-hosting サービスのリストを表示します。

	コマンドまたはアクション	目的
	`Device# show app-hosting list`	

例

次に、**show iox-service** コマンドの出力例を示します。

```
Device# show iox-service

IOx Infrastructure Summary:
---------------------------
IOx service (CAF) 1.10.0.0 : Running
IOx service (HA)           : Running
IOx service (IOxman)       : Running
IOx service (Sec storage) : Not Running
Libvirtd   1.3.4           : Running
Dockerd    18.03.0         : Running
Application DB Sync Info  : Available
Sync Status              : Disabled
```

次に、**show app-hosting list** コマンドの出力例を示します。

```
Device# show app-hosting list

App id                           State
---------------------------------------------------------
guestshell                       RUNNING
```

ゲスト シェルの管理

（注） VirtualPortGroups はルーティング プラットフォームでのみサポートされています。

始める前に

ゲスト シェルのアクセスが機能するには、IOx が構成されて実行している必要があります。IOx が構成されていない場合は、IOx の構成を求めるメッセージが表示されます。IOx を削除すると、ゲスト シェルにもアクセスできなくなります。ただし rootfs は影響を受けません。

ゲストシェルを有効にして操作するように、アプリケーションまたは管理インターフェイスも設定する必要があります。ゲストシェルのインターフェイスを有効にする方法の詳細については、「Configuring the AppGigabitEthernet Interface for Guest Shell」および「Enabling Guest Shell on the Management Interface」のセクションを参照してください。

手順の概要

1. **enable**

2. **guestshell enable**
3. **guestshell run** *linux-executable*
4. **guestshell run bash**
5. **guestshell disable**
6. **guestshell destroy**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**guestshell enable** 例： `Device# guestshell enable`	ゲスト シェル サービスの有効化。 （注） • **guestshell enable** コマンドは、ネットワーキングに管理 Virtual Routing and Forwarding（VRF）インスタンスを使用します。 • フロント パネル ネットワー キングに VirtualPortGroups（VPG）を使用している場合は、まず VPG を構成する必要があります。 • ゲスト IP アドレスとゲートウェイ IP アドレスは同じサブネット内にある必要があります。

ステップ 3	**guestshell run** *linux-executable*
	例：
	`Device# guestshell run python`
	または
	`Device# guestshell run python3`
		ゲスト シェルで Linux プログラムを実行します。
		（注）
		Cisco IOS XE Amsterdam 17.3.1 以降のリリースでは、Python バージョン 3 のみがサポートされます。

ステップ 4	**guestshell run bash** 例： `Device# guestshell run bash`	Bash シェルを開始して、ゲスト シェルにアクセスします。
ステップ 5	**guestshell disable** 例： `Device# guestshell disable`	ゲスト シェル サービスを無効化します。
ステップ 6	**guestshell destroy** 例：	ゲスト シェル サービスを非アクティブ化して、アンインストールします。

	コマンドまたはアクション	目的
	`Device# guestshell destroy`	

アプリケーション ホスティングを使用したゲストシェルの管理

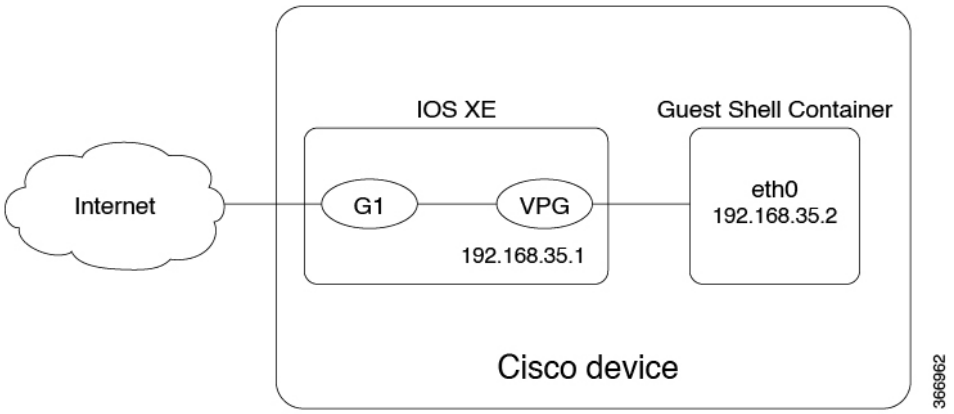
（注） この項は、シスコ ルーティング プラットフォームに適用されます。VirtualPortGroups は、Cisco Catalyst スイッチング プラットフォームではサポートされていません。

ゲスト シェルのアクセスが機能するには、IOx が構成されて実行している必要があります。IOx が構成されていない場合は、IOx の構成を求めるメッセージが表示されます。IOx を削除すると、ゲスト シェルにもアクセスできなくなります。ただし rootfs は影響を受けません。

（注） この手順（アプリケーションホスティングを使用したゲストシェルの管理）を使用して、Cisco IOS XE Fuji 16.7.1 以降のリリースのゲストシェルを有効にします。Cisco IOS XE Everest 16.6.x 以前では、ゲスト シェルの管理 （15 ページ） の手順を使用します。

```
Device(config)# interface GigabitEthernet1
Device(config-if)# ip address dhcp
Device(config-if)# ip nat outside
Device(config-if)# exit

Device(config-if)# interface VirtualPortGroup0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
Device(config-if)# ip nat inside
Device(config-if)# exit

Device(config)# ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 overload
Device(config)# ip access-list standard GS_NAT_ACL
Device(config)# permit 192.168.0.0 0.0.255.255

Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-default-gateway 192.168.35.1 guest-interface 0
Device(config-app-hosting)# end

Device# guestshell enable
Device# guestshell run python
```

図 *2:* アプリケーション ホスティングを使用したゲスト シェルの管理



前面パネルのネットワーキングでは、GigabitEthernet インターフェイスと VirtualPortGroup インターフェイスを上の図に示すように設定する必要があります。ゲストシェルは Virtualportgroup を送信元インターフェイスとして使用し、NAT を通じて外部ネットワークに接続します。

内部 NAT の設定には、次のコマンドを使用します。これにより、ゲスト シェルがインターネットに到達し、たとえば、Linux ソフトウェア更新プログラムを取得できるようになります。

```
ip nat inside source list
ip access-list standard
permit
```

上の例の **guestshell run** コマンドは Python 実行可能ファイルを実行します。また、**guestshell run** コマンドを使用して他の Linux 実行可能ファイルを実行することもできます。たとえば、**guestshell run bash** コマンドは bash シェルを起動し、**guestshell disable** コマンドはゲストシェルをシャットダウンして無効にします。後でシステムをリロードしても、ゲストシェルは無効のままになります。

ゲストシェルの AppGigabitEthernet インターフェイスの設定

（注） 次のタスクは、AppGigabitEthernet インターフェイスを持つ Catalyst スイッチにのみ適用されます。他のすべての Catalyst スイッチは、管理ポートを使用します。

手順の概要

1. **enable**
2. **configure terminal**
3. **interface AppGigabitEthernet** *interface-number*
4. **switchport mode trunk**
5. **exit**
6. **app-hosting appid** *name*
7. **app-vnic AppGigabitEthernet trunk**

8. **vlan** *vlan-ID* **guest-interface** *guest-interface-number*
9. **guest-ipaddress** *ip-address* **netmask** *netmask*
10. **exit**
11. **exit**
12. **app-default-gateway** *ip-address* **guest-interface** *network-interface*
13. **nameserver#** *ip-address*
14. **end**
15. **guestshell enable**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**interface AppGigabitEthernet** *interface-number* 例： `Device(config)# interface AppGigabitEthernet 1/0/1`	AppGigabitEthernet インターフェイスを設定し、インターフェイス コンフィギュレーション モードを開始します。
ステップ 4	**switchport mode trunk** 例： `Device(config-if)# switchport mode trunk`	インターフェイスを永続的なトランキングモードに設定して、ネイバーリンクのトランクリンクへの変換をネゴシエートします。
ステップ 5	**exit** 例： `Device(config-if)# exit`	インターフェイス コンフィギュレーション モードを終了し、グローバルコンフィギュレーションモードに戻ります。
ステップ 6	**app-hosting appid** *name* 例： `Device(config)# app-hosting appid guestshell`	アプリケーションを設定し、アプリケーション ホスティング コンフィギュレーション モードを開始します。
ステップ 7	**app-vnic AppGigabitEthernet trunk** 例： `Device(config-app-hosting)# app-vnic AppGigabitEthernet trunk`	トランクポートをアプリケーション ホスティングの前面パネルポートとして設定し、アプリケーション ホスティング トランク コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 8	**vlan** *vlan-ID* **guest-interface** *guest-interface-number* 例： `Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0`	VLAN ゲストインターフェイスを設定し、アプリケーション ホスティング VLAN アクセス IP コンフィギュレーション モードを開始します。
ステップ 9	**guest-ipaddress** *ip-address* **netmask** *netmask* 例： `Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask 255.255.255.0`	（オプション）静的 IP を設定します。
ステップ 10	**exit** 例： `Device(config-config-app-hosting-vlan-access-ip)# exit`	アプリケーション ホスティング VLAN アクセス IP コンフィギュレーション モードを終了し、アプリケーション ホスティング トランク コンフィギュレーション モードに戻ります。
ステップ 11	**exit** 例： `Device(config-config-app-hosting-trunk)# exit`	アプリケーション ホスティング トランク コンフィギュレーション モードを終了し、アプリケーション ホスティング コンフィギュレーション モードに戻ります。
ステップ 12	**app-default-gateway** *ip-address* **guest-interface** *network-interface* 例： `Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0`	デフォルトの管理ゲートウェイを設定します。
ステップ 13	**nameserver#** *ip-address* 例： `Device(config-app-hosting)# name-server0 172.16.0.1`	ドメインネームシステム（DNS）サーバを設定します。
ステップ 14	**end** 例： `Device(config-app-hosting)# end`	アプリケーション ホスティング コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 15	**guestshell enable** 例： `Device# guestshell enable`	ゲスト シェル サービスの有効化。

管理インターフェイスでのゲストシェルの有効化

（注） このタスクは、Cisco Catalyst 9200 シリーズ スイッチ、Cisco Catalyst 9300 シリーズ スイッチ、Cisco Catalyst 9400 シリーズ スイッチ、Cisco Catalyst 9500 シリーズ スイッチ、Cisco Catalyst 9600 シリーズ スイッチに適用できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **app-hosting appid** *name*
4. **app-vnic management guest-interface** *interface-number*
5. **end**
6. **show app-hosting list**
7. **guestshell enable**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例： `Device> enable`	特権 EXEC モードを有効にします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configure terminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**app-hosting appid** *name* 例： `Device(config)# app-hosting appid guestshell`	アプリケーションを設定し、アプリケーション ホスティング コンフィギュレーション モードを開始します。
ステップ 4	**app-vnic management guest-interface** *interface-number* 例： `Device(config-app-hosting)# app-vnic management guest-interface 0`	仮想ネットワーク インターフェイスおよびゲスト インターフェイスの管理ゲートウェイを設定し、アプリケーションホスティング ゲートウェイ コンフィギュレーション モードを開始します。
ステップ 5	**end** 例： `Device(config-app-hosting-mgmt-gateway)# end`	アプリケーションホスティング管理ゲートウェイ コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 6	**show app-hosting list** 例：	インストールされているアプリケーションの現在のステータスを表示します。

	コマンドまたはアクション	目的
	`Device# show app-hosting list`	（注） ゲストシェルは、インストールされている場合にのみ、アプリケーションのリストに表示されます。
ステップ 7	**guestshell enable** 例： `Device# guestshell enable`	ゲスト シェル サービスの有効化。

ゲストシェルからの NETCONF アクセスの有効化と無効化

始める前に

ゲストシェル内から次のコマンドを初期化して、NETCONF-YANGアクセスを初期化します。

手順の概要

1. **iosp_client -f netconf_enable guestshell** *port-number*
2. **iosp_client -f netconf_enable_passwordless guestshell** *username*
3. **iosp_client -f netconf_disable guestshell**
4. **iosp_client -f netconf_disable_passwordless guestshell** *username*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**iosp_client -f netconf_enable guestshell** *port-number* 例： `Guest Shell: iosp_client -f netconf_enable guestshell 3`	**netconf-yang ssh local-vrf guestshell** コマンドを設定し、NETCONF-YANG が稼働するまで接続をブロックします。
ステップ 2	**iosp_client -f netconf_enable_passwordless guestshell** *username* 例： `Guest Shell: iosp_client -f netconf_enable guestshell guestshell`	ゲストシェルアクセスに必要なSSHキーを作成します。
ステップ 3	**iosp_client -f netconf_disable guestshell** 例： `GuestShell: iosp_client -f netconf_disable guestshell`	ゲストシェル内からNETCONFへのアクセスを削除します。 • NETCONF-YANG 設定は引き続き存在します。NETCONF-YANG をシャットダウンするには、**no netconf-yang** コマンドを使用します。
ステップ 4	**iosp_client -f netconf_disable_passwordless guestshell** *username*	指定したユーザーのアクセスキーを削除します。

	コマンドまたはアクション	目的
	例： `Guest Shell: iosp_client -f netconf_disable_passwordless guestshell guestshell`	• NETCONF アクセスはユーザーに対して引き続き有効です。ただし、ユーザーは NETCONF に接続するためにパスワードを使用する必要があります。

例

Python インタープリタのアクセス

Python はインタラクティブに使用できますが、Python スクリプトをゲスト シェルで実行することもできます。**guestshell run python** コマンドを使用してゲスト シェルで Python インタープリタを起動し、Python 端末を開きます。

（注） Cisco IOS XE Amsterdam 17.3.1 より前のリリースでは、Python V2 がデフォルトです。Cisco IOS XE Amsterdam 17.1.1 および Cisco IOS XE Amsterdam 17.2.1 では、Python V3 がサポートされています。Cisco IOS XE Amsterdam 17.3.1 以降のリリースでは、Python V3 がデフォルトです。

Cisco IOS XE Amsterdam 17.3.1 より前のリリース

guestshell run コマンドは、Linux 実行可能ファイルの実行に相当する Cisco IOS であり、Cisco IOS からの Python スクリプトの実行時に絶対パスを指定します。次の例は、コマンドの絶対パスを指定する方法を示しています。

```
Guestshell run python /flash/guest-share/sample_script.py parameter1 parameter2
```

次に、Cisco Catalyst 3650 シリーズ スイッチまたは Cisco Catalyst 3850 シリーズ スイッチで Python を有効にする例を示します。

```
Device# guestshell run python

Python 2.7.11 (default, March 16 2017, 16:50:55)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>
```

次の例は、Cisco ISR 4000 シリーズ サービス統合型ルータで Python を有効にする方法を示しています。

```
Device# guestshell run python

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>>>
```

Cisco IOS XE Amsterdam 17.3.1 以降のリリース

次の例は、Cisco Catalyst 9000 シリーズ スイッチ上で Python を有効にする方法を示しています。

```
Device# guestshell run python3

Python 3.6.8 (default, Nov 21 2019, 22:10:21)
[GCC 8.3.1 20190507 (Red Hat 8.3.1-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.>>>>>
```

ゲスト シェルの設定例

例：ゲスト シェルの管理

Cisco IOS XE Amsterdam 17.1.x から Cisco IOS XE Amsterdam 17.2.x

次の例は、ゲストシェルを有効にする方法を示しています。Cisco IOS XE Amsterdam 17.1.x および Cisco IOS XE Amsterdam 17.2.x では、Python V2.7 および Python V3.6 がサポートされています。ただし、これらのリリースでは Python V2.7 がデフォルトです。

```
Device> enable
Device# guestshell enable

Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python
or
Device# guestshell run python3

Python 2.7.5 (default, Jun 17 2014, 18:11:42)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>>>


Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

Cisco IOS XE Amsterdam 17.3.1 以降のリリース

次の例は、ゲストシェルを有効にする方法を示しています。Cisco IOS XE Amsterdam 17.3.1 以降のリリースでは、Python V3.6 のみがサポートされます。

```
Device> enable
Device# guestshell enable

Management Interface will be selected if configured
Please wait for completion
Guestshell enabled successfully

Device# guestshell run python3

Python 3.6.8 (default, Nov 21 2019, 22:10:21)
[GCC 8.3.1 20190507 (Red Hat 8.3.1-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.>>>>>

>>>>>

Device# guestshell run bash

[guestshell@guestshell ~]$

Device# guestshell disable

Guestshell disabled successfully

Device# guestshell destroy

Guestshell destroyed successfully
```

VirtualPortGroup 設定の例

（注） VirtualPortGroups は Cisco ルーティング プラットフォームでのみサポートされています。

ゲストシェルネットワーキングにVirtualPortGroupインターフェイスを使用する場合、VirtualPortGroup インターフェイスには設定済みの静的 IP アドレスが必要です。フロントポートインターフェイスはインターネットに接続されている必要があり、ネットワーク アドレス変換（NAT）は VirtualPortGroup とフロント パネル ポートの間で設定されている必要があります。

次に示すのは、VirtualPortGroup の設定例です。

```
Device> enable
Device# configure terminal
Device(config)# interface VirtualPortGroup 0
Device(config-if)# ip address 192.168.35.1 255.255.255.0
```

```
Device(config-if)# ip nat inside
Device(config-if)# no mop enabled
Device(config-if)# no mop sysid
Device(config-if)# exit
Device(config)# interface GigabitEthernet 0/0/3
Device(config-if)# ip address 10.0.12.19 255.255.0.0
Device(config-if)# ip nat outside
Device(config-if)# negotiation auto
Device(config-if)# exit
Device(config)# ip route 0.0.0.0 0.0.0.0 10.0.0.1
Device(config)# ip route 10.0.0.0 255.0.0.0 10.0.0.1
!Port forwarding to use ports for SSH and so on.
Device(config)# ip nat inside source static tcp 192.168.35.2 7023 10.0.12.19 7023
extendable
Device(config)# ip nat outside source list NAT_ACL interface GigabitEthernet 0/0/3
overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit

! App-hosting configuration
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic gateway1 virtualportgroup 0 guest-interface 0
Device(config-app-hosting-gateway)# guest-ipaddress 192.168.35.2 netmask 255.255.255.0
Device(config-app-hosting-gateway)# exit
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 1500
Device(config-app-resource-profile-custom)# memory 512
Device(config-app-resource-profile-custom)# end

Device# guestshell enable
Device# guestshell run python
```

例：ゲストシェルの AppGigabitEthernet インターフェイスの設定

（注） 次のタスクは、AppGigabitEthernet インターフェイスを持つ Catalyst スイッチにのみ適用されます。他のすべての Catalyst スイッチは、管理ポートを使用します。

次の例は、ゲストシェルの AppGigabitEthernet インターフェイスを設定する方法を示しています。ここでは、VLAN 4094 がネットワークアドレス変換（NAT）を作成します。これはゲストシェルに使用されます。VLAN 1 は外部インターフェイスです。

```
Device> enable
Device# configure terminal
Device(config)# ip nat inside source list NAT_ACL interface vlan 1 overload
Device(config)# ip access-list standard NAT_ACL
Device(config-std-nacl)# permit 192.168.0.0 0.0.255.255
Device(config-std-nacl)# exit
Device(config)# vlan 4094
Device(config-vlan)# exit
Device(config)# interface vlan 4094
Device(config-if)# ip address 192.168.2.1 255.255.255.0
```

```
Device(config-if)# ip nat inside
Device(config-if)# exit
Device(config)# interface vlan 1
Device(config-if)# ip nat outside
Device(config-if)# exit
Device(config)# ip routing
Device(config)# ip route 0.0.0.0 0.0.0.0 209.165.201.1
Device(config)# interface AppGigabitEthernet 1/0/1
Device(config-if)# switchport mode trunk
Device(config-if)# exit
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic AppGigEthernet trunk
Device(config-config-app-hosting-trunk)# vlan 4094 guest-interface 0
Device(config-config-app-hosting-vlan-access-ip)# guest-ipaddress 192.168.2.2 netmask
255.255.255.0
Device(config-config-app-hosting-vlan-access-ip)# exit
Device(config-config-app-hosting-trunk)# exit
Device(config-app-hosting)# app-default-gateway 192.168.2.1 guest-interface 0
Device(config-app-hosting)# name-server0 172.16.0.1
Device(config-app-hosting)# name-server1 198.51.100.1
Device(config-app-hosting)# end
Device# guestshell enable
```

例：管理インターフェイスでのゲストシェルの有効化

この例は、Cisco Catalyst 9200 シリーズ スイッチ、Cisco Catalyst 9300 シリーズ スイッチ、Cisco Catalyst 9400 シリーズ スイッチ、Cisco Catalyst 9500 シリーズ スイッチ、Cisco Catalyst 9600 シリーズ スイッチに適用できます。

```
Device> enable
Device# configure terminal
Device(config)# app-hosting appid guestshell
Device(config-app-hosting)# app-vnic management guest-interface 0
Device(config-app-hosting-mgmt-gateway)# end
Device# guestshell enable
```

例：ゲスト シェルの使用

ゲスト シェル プロンプトから Linux のコマンドを実行できます。次の例は、一部の Linux コマンドの使用法を示しています。

```
[guestshell@guestshell~]$ pwd
/home/guestshell

[guestshell@guestshell~]$ whoami
guestshell

[guestshell@guestshell~]$ uname -a
Linux guestshell 5.4.85 #1 SMP Tue Dec 22 10:50:44 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

Cisco 4000 シリーズ サービス統合型ルータは、CentOS Linux リリース 7.1.1503 で提供される **dohost** を使用します。

（注） **dohost** コマンドには、**ip http server** コマンドがデバイス上で設定されていることが必要です。

例：ゲスト シェルのネットワーキング設定

下記は、ゲストシェルのネットワーキング設定例です。

- ドメイン ネーム システム（DNS）の設定
- プロキシの設定
- プロキシの設定を使用するための YUM または PIP の設定

ゲスト シェルの DNS 設定の例

ゲスト シェルのサンプル DNS 構成は次のとおりです。

```
[guestshell@guestshell ~]$ cat/etc/resolv.conf
nameserver 192.0.2.1

Other Options:
[guestshell@guestshell ~]$ cat/etc/resolv.conf
domain cisco.com
search cisco.com
nameserver 192.0.2.1
search cisco.com
nameserver 198.51.100.1
nameserver 172.16.0.6
domain cisco.com
nameserver 192.0.2.1
nameserver 172.16.0.6
nameserver 192.168.255.254
```

例：プロキシ環境変数の設定

ネットワークがプロキシの背後にある場合は、Linux でプロキシ変数を設定します。必要な場合は、環境にこれらの変数を追加します。

次の例は、プロキシ変数を設定する方法を示しています。

```
[guestshell@guestshell ~]$cat /bootflash/proxy_vars.sh
export http_proxy=http://proxy.example.com:80/
export https_proxy=http://proxy.example.com:80/
```

```
export ftp_proxy=http://proxy.example.com:80/
export no_proxy=example.com
export HTTP_PROXY=http://proxy.example.com:80/
export HTTPS_PROXY=http://proxy.example.com:80/
export FTP_PROXY=http://proxy.example.com:80/
guestshell ~] source /bootflash/proxy_vars.sh
```

例：プロキシ設定用の Yum および PIP の構成

次の例は、プロキシ環境変数の設定に Yum を使用する方法を示しています。

```
cat /etc/yum.conf | grep proxy
[guestshell@guestshell~]$ cat/bootflash/yum.conf | grep proxy
proxy=http://proxy.example.com:80/
```

PIP のインストールでは、プロキシ設定に使用される環境変数が選択されます。PIP インストールには -E オプションを指定した sudo を使用します。環境変数が設定されていない場合は、次の例に示すように PIP コマンドでそれらを明示的に定義します。

```
sudo pip --proxy http://proxy.example.com:80/install requests
sudo pip install --trusted-bost pypi.example.com --index-url
http://pypi.example.com/simple requests
```

次の例では、Python の PIP インストールを使用する方法を示します。

```
Sudo -E pip install requests
[guestshell@guestshell ~]$ python
Python 2.17.11 (default, Feb 3 2017, 19:43:44)
[GCC 4.7.0] on linux2
Type "help", "copyright", "credits" or "license" for more information
>>>import requests
```

ゲスト シェルに関するその他の参考資料

関連資料

関連項目	マニュアル タイトル
Python モジュール	『CLI Python モジュール』
ゼロ タッチ プロビジョニング	『ゼロ タッチ プロビジョニング』

シスコのテクニカル サポート

説明	リンク
シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンライン リソースを提供しています。 お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service（Field Notice からアクセス）、Cisco Technical Services Newsletter、Really Simple Syndication（RSS）フィードなどの各種サービスに加入できます。 シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。	http://www.cisco.com/support

ゲスト シェルの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコ ソフトウェアイメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、www.cisco.com/go/cfn に移動します。Cisco.com のアカウントは必要ありません。

表 *4:* ゲスト シェルの機能情報

機能名	リリース	機能情報
ゲスト シェル	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	ゲスト シェルは、お客様がシスコ スイッチの自動制御および管理のためのカスタム Python アプリケーションを実行できる、埋め込み Linux 環境であるセキュア コンテナです。システムの自動化されたプロビジョニングも含まれます。このコンテナ シェルは、ホスト デバイスから分離された安全な環境を提供します。ユーザはそこで、スクリプトまたはソフトウェア パッケージをインストールし、実行することができます。 Cisco IOS XE Everest 16.5.1a では、この機能は次のプラットフォームに実装されていました。 • Cisco Catalyst 3650 シリーズ スイッチ • Cisco Catalyst 3850 シリーズ スイッチ • Cisco Catalyst 9300 シリーズ スイッチ • Cisco Catalyst 9500 シリーズ スイッチ Cisco IOS Everest 16.5.1b では、この機能は次のプラットフォームに実装されていました。 • Cisco 4000 シリーズ サービス統合型ルータ
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco IOS XE Everest 16.6.2 で、Cisco Catalyst 9400 シリーズ スイッチに実装されました。

機能名	リリース	機能情報
	Cisco IOS XE Fuji 16.7.1	Cisco IOS XE Fuji 16.7.1 では、この機能は次のプラットフォームに実装されていました。 • Cisco ASR 1000 シリーズ アグリゲーション サービス ルータ • Cisco Cloud Services Router 1000V シリーズ Cisco IOS XE Fuji 16.7.1 では、ゲスト シェル機能の場合、ロギングとトレーシング サポートが Cisco ASR 1000 アグリゲーション サービス ルータに実装されました。
	Cisco IOS XE Fuji 16.8.1	Cisco IOS XE Fuji 16.8.1 では、この機能は Cisco Catalyst 9500 ハイ パフォーマンス シリーズ スイッチに実装されていました。
	Cisco IOS XE Fuji 16.9.1	Cisco IOS XE Fuji 16.9.1 では、この機能は Cisco 1000 シリーズ サービス統合型ルータに実装されていました。
	Cisco IOS XE Gibraltar 16.11.1b	Cisco IOS XE Gibraltar 16.11.1b では、この機能は次のプラットフォームに実装されていました。 • Cisco Catalyst 9800-40 ワイヤレスコントローラ • Cisco Catalyst 9800-80 ワイヤレスコントローラ
	Cisco IOS XE Gibraltar 16.12.1	

機能名	リリース	機能情報
		この機能は、Cisco IOS XE Gibraltar 16.12.1 で次のプラットフォームに実装されました。 • Cisco Catalyst 9200 シリーズ スイッチ （注） この機能は C9200L SKU ではサポートされていません。 • Cisco Catalyst 9300L SKU • Cisco Catalyst 9600 シリーズ スイッチ
	Cisco IOS XE Amsterdam 17.3.1	この機能は、Cisco IOS XE Amsterdam 17.3.1 で次のプラットフォームに実装されました。 • Cisco Catalyst 8200 シリーズエッジプラットフォーム • Cisco Catalyst 8300 シリーズエッジプラットフォーム • Cisco Catalyst 8500 および 8500L シリーズ エッジ プラットフォーム

機能名	リリース	機能情報
ゲストシェルからのNETCONFアクセス	Cisco IOS XE Bengaluru 17.6.1	NETCONF にはゲストシェル内からアクセスできるため、ユーザーは Python スクリプトを実行し、NETCONF プロトコルを使用してシスコカスタムパッケージ CLI を呼び出すことができます。 この機能は、17.6.1 で次のプラットフォームに実装されました。 • Cisco Catalyst 9200 シリーズ スイッチ • Cisco Catalyst 9300 および 9300L シリーズ スイッチ • Cisco Catalyst 9400 シリーズ スイッチ • Cisco Catalyst 9500 および 9500 ハイ パフォーマンス シリーズ スイッチ • Cisco Catalyst 9600 シリーズ スイッチ
ゲストシェルでの Python 3 のサポート	Cisco IOS XE Amsterdam 17.1.1	Python バージョン 3.6 は、ゲストシェルでサポートされています。Python バージョン 3.6 は、すべてのサポート対象プラットフォームで使用できます。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。