IOx アプリケーション ホスティング

この章は、次の項で構成されています。

アプリケーション ホスティングに関する情報

ホスト アプリケーションは、サービス ソリューションとしてのソフトウェアであり、コマンドを使用してリモートで実行できます。アプリケーションのホスティングによって、管理者には独自のツールやユーティリティを利用するためのプラットフォームが与えられます。

このモジュールでは、アプリケーションホスティング機能とその有効化の方法について説明します。

アプリケーション ホスティングの必要性

仮想環境への移行により、再利用可能なポータブルかつスケーラブルなアプリケーションを構築する必要性が高まりました。アプリケーションのホスティングによって、管理者には独自のツールやユーティリティを利用するためのプラットフォームが与えられます。ネットワークデバイスでホスティングされているアプリケーションは、さまざまな用途に利用できます。これは、既存のツールのチェーンによる自動化から、設定管理のモニタリング、統合に及びます。

Cisco のデバイスは Linux ツール チェーンを使用して構築されたサードパーティ製の市販アプリケーションをサポートしています。ユーザは、シスコが提供するソフトウェア開発キットと相互にコンパイルされたカスタム アプリケーションを実行できます。

IOx の概要

IOx は Cisco が開発したエンド 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。

IR1101 向けの IOx アーキテクチャは、ハイパーバイザ アプローチを使用する他のシスコ プラットフォームとは異なります。他のプラットフォームでは、IOx は仮想マシンとして動作します。一方 IR1101 では、IOx はプロセスとして動作しています。

シスコ アプリケーションのホスティングの概要

IR1101 では、ユーザは、アプリケーション ホスティング CLI を使用してアプリケーションを展開できます。アプリケーション ホスティング CLI は、他の古いプラットフォームでは利用できません。アプリケーションを展開する方法は他に Local Manager または Fog Director を使用する方法があります。

アプリケーション ホスティングは、次のサービスを提供します。

- コンテナ内の指定されたアプリケーションを起動する。
- 使用可能なリソース（メモリ、CPU、およびストレージ）を確認し、それらを割り当て、管理する。
- コンソール ロギングのサポートを提供する。
- REST API を介してサービスへのアクセスを提供する。
- CLI エンドポイントを提供する。
- Cisco Application Framework（CAF）と呼ばれるアプリケーション ホスティング インフラストラクチャを提供する。
- VirtualPortGroup および管理インターフェイスを介したプラットフォーム固有のネットワーキング（パケットパス）のセットアップを支援する。

コンテナは、ホスト オペレーティング システムでゲスト アプリケーションを実行するために提供される仮想環境と呼ばれています。Cisco IOS XE 仮想化サービスは、ゲスト アプリケーションを実行するための管理性とネットワーキング モデルを提供します。仮想化インフラストラクチャにより、管理者はホストとゲスト間の接続を指定する論理インターフェイスを定義できます。IOx は、論理インターフェイスをゲスト アプリケーションが使用する仮想ネットワーク インターフェイス カード（vNIC）にマッピングします。

コンテナに展開されるアプリケーションは、TAR ファイルとしてパッケージ化されます。これらのアプリケーションに固有の設定は、TAR ファイルの一部としてもパッケージ化されています。

デバイス上の管理インターフェイスは、アプリケーション ホスティング ネットワークを IOS 管理インターフェイスに接続します。アプリケーションのレイヤ 3 インターフェイスは、IOS 管理インターフェイスからレイヤ 2 ブリッジ トラフィックを受信します。管理インターフェイ

スは、管理ブリッジを使用してコンテナ/アプリケーション インターフェイスに接続します。IPアドレスは、管理インターフェイスIPアドレスと同じサブネット上にある必要があります。

IOXMAN

IOXMANは、シリアルデバイスをエミュレートするLibvirtを除く、ゲストアプリケーションのロギングまたはトレース サービスを提供するトレース インフラストラクチャを確立するプロセスです。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。

IOx アプリケーションへの GPS アクセス

以前は、モデムでGPSが有効になっていると、NMEAストリームがIOxに転送されませんでした。このリリースでは、NMEAストリームをngioliteモジュールからIOxに転送できます。これを有効にするには2つの手順があります。

- Linux と IOx の間にトンネルを作成する。
- すべての NMEA メッセージをトンネル経由で IOx に転送する。

システムコードはトンネルの存在を確認し、存在しない場合はデータをIOxに送信できません。

この機能をサポートするために、IR1101 と IR1800 の 2 つのセルラーモデム用に 2 つの新しいトンネルが作成されます。デフォルトでは2つのトンネルが作成され、どちらのモデムでもGPS/NMEAが有効になっていれば、次のようにNMEAストリームが対応するトンネルを介して送信されます。

Modem0：

[Linux] /dev/ttyTun5 および /dev/ttyTun6 [IOx]。/dev/ttyTun5 へのソフトリンクは /dev/ttyTunNMEA0 という名前で作成され、/dev/ttyTun6 へのソフトリンクは /dev/ttyNMEA0 という名前で作成されます。これらは、IOx からアクセスできます。

Modem1：

[Linux] /dev/ttyTun7 and /dev/ttyTun8 [IOx]。/dev/ttyTun7 へのソフトリンクは /dev/ttyTunNMEA1 という名前で作成され、/dev/ttyTun8 へのソフトリンクは /dev/ttyNMEA1 という名前で作成されます。これらは、IOx からアクセスできます。

次のコマンドは、GPS の状態を表示します。

```
IR1101#show app-hosting list
App id State
------------------------------------------------------------
gps RUNNING
```

IOx コンテナアプリケーションとしてのゲストシェル

ゲスト シェルは、仮想化された Linux ベースの環境であり、Cisco デバイスの自動制御と管理のための Python アプリケーションを含む、カスタム Linux アプリケーションを実行するように設計されています。ゲストシェルを使用すると、ユーザーはサードパーティ製 Linux アプリケーションのインストール、更新、操作、および IOS CLI へのアクセスを行うこともできます。

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（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 rootfs 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することができます。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。

ゲストシェルは通常、システムイメージとともにバンドルされており、Cisco IOS コマンド **guestshell enable** を使用してインストールできます。ただし、この方法では、イメージのサイズが約 75 MB 増加します。これは、帯域幅が限られているか、LTE を介してイメージをダウンロードする一部のユーザーにとっては問題です。

これらのユーザーを考慮して、ゲストシェルは単一の tar ファイルとして使用できるようになり、他の IOX アプリケーションと同様にダウンロードしてシステムにインストールできます。その結果、ユニバーサル リリース イメージのサイズは増加しません。

（注） 第 0 日のゲストシェル プロビジョニングは、このアプローチでは機能しません。

ゲストシェルは、デフォルトで、管理インターフェイスを介して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。IR1101 のように専用管理ポートを持たないプラットフォームの場合、VirtualPortGroup を IOS 設定内のゲストシェルに関連付けることができます。

ゲストシェルの設定例については、こちらを参照してください。

ゲストシェルをデバイスにインストールするには、tar ファイルをルータにコピーし、次のコマンドを実行します。

```
app-hosting install appid guestshell package <path to tar file>
```

ステータスを確認するには、次のコマンドを使用します。

```
show app-hosting list
```

ゲストシェルが正常に展開されると、**guestshell enable**、**guestshell run bash**、**guestshell run python3** などの標準のゲストシェルコマンドが機能します。

次のリソースでは、guesthell を使用した Python スクリプトの実行について説明しています。

CLI Python モジュール

（注） 17.5.1 では python3 のみがサポートされています。

重要：インストールする前に

デバイスにゲストシェルをインストールする前に、次のコマンドを実行して、デバイスに IOx コンテナキーがプログラムされていることを確認してください。

```
Router#show software authenticity keys | i Name
Product Name : SFP-VADSL2-I
Product Name : SFP-VADSL2-I
Product Name : IR1101
Product Name : IR1101
Product Name : Cisco Services Containers
Product Name : Cisco Services Containers
```

出力には、製品名が「Cisco Services Containers」の行が 1 つ以上含まれている必要があります。コンテナキーがデバイスにプログラムされていない場合は、ゲストシェルをインストールできません。

次のようなエラーが表示されます。

```
*Aug 26 15:47:21.484: %IOSXE-3-PLATFORM: R0/0: IOx: App signature verification failed
with non-zero exit code
*Aug 26 15:47:21.588: %IM-6-INSTALL_MSG: R0/0: ioxman: app-hosting: Install failed: App
 package signature (package.sign)
verification failed for package manifest file package.mf. Re-sign the application and
then deploy again.
```

コンテナキーをデバイスにインストールするためのソフトウェアベースのメカニズムはありません。キーは製造施設でプログラムする必要があります。2020 年 1 月 1 日以降に出荷された IR1100 デバイスでは、コンテナキーがプログラムされています。

ゲストシェルの tar ファイルは、特定のリリースの IOS-XE イメージとともに発行されます。詳細については、https://developer.cisco.com/docs/iox/#!iox-resource-downloads/downloadsを参照してください。

IR1101 ルータでのアプリケーション ホスティング

ここでは、IR1101 産業向けルータに固有のアプリケーションホスティングの特性について説明します。

（注） IR1101 CPU は、他のルータのように x86 アーキテクチャに基づいていません。したがって、アプリケーションが ARM 64 ビット アーキテクチャに準拠している必要があります。

アプリケーション ホスティングは、アプリケーション ホスティング CLI、Local Manager および Fog Director を使用して実現できます。

IOx URL アクセス方法

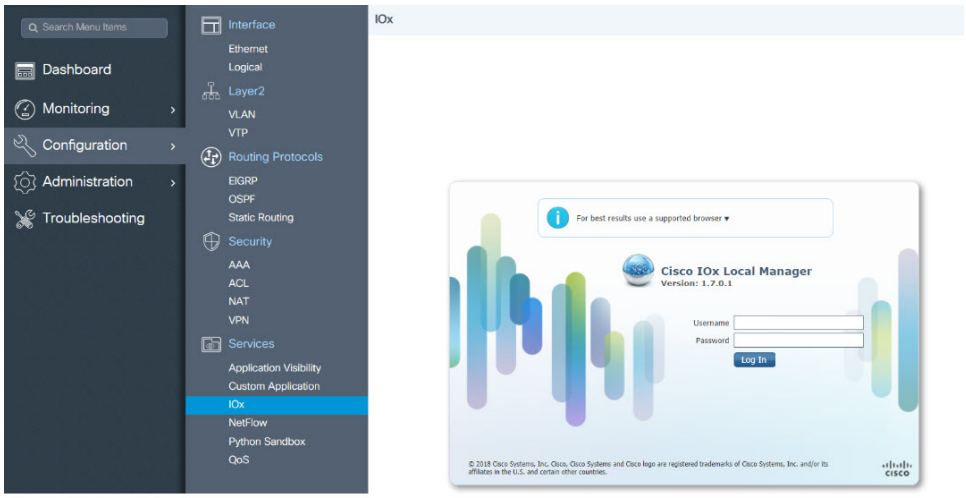
IOx URL には、2 つの異なる方法でアクセスできます。

1. IOx ログインへの直接 URL を使用します。
2. Web ユーザーインターフェイス（WebUI）を介して IOx ログインに移動します。

1 番目の方法の構文は、**https://***IR1101-IP-ADDRESS***/iox/login** です。

2 番目の方法の構文は、**https://***IR1101-IP-ADDRESS* で、その後、次に示すように IOx に移動します。

図 1 : Local Manager



1. WebUI から、**[Configuration] > [Services] > [IOx]** をクリックします。
2. 設定されたユーザ名とパスワードを使用してログインします。
3. 『Cisco IOx Local Manager Reference Guide』のアプリケーション ライフサイクルの手順を実行します。

IOX URL ユーザー制限

2 番目の方法では、IOx ユーザーがルータ全体の設定を使用できるようにします。一部の組織では、IOx ユーザーはルータを管理するユーザーとは異なります。この場合、IOx ユーザーのアクセスをルータの WebUI 全体ではなく、IOx ローカルマネージャ WebUI のみに制限する必要があります。

現在、IOx ユーザーは権限 15 ユーザーとして設定されています。IOx ユーザーをローカルマネージャのみに制限するために、次のコマンドを使用できます。

```
Router(conf)# no ip http server
Router(conf)# ip http secure-server
```

```
Router(conf)# ip http session-module-list list2 OPENRESTY_PKI,NG_WEBUI
Router(conf)# ip http secure-active-session-modules list2
```

コマンド **no ip http server** は、https のない Web サーバーをオフにします。次のコマンド **ip http secure-server** は https モードをオンにします。

OPENRESTY_PKI と NG_WEBUI のみを含めると、IOX ローカル マネージャ モジュールのみが有効になるため、すべてのユーザーは、権限 15、https://IR1101-IP-ADDRESS/iox/login がある場合にのみ IOX ローカルマネージャにアクセスできます。

さらに、すべてのユーザーに対して、WebUI アクセス、**https://***IR1101-IP-ADDRESS* が無効になります。

（注） この方法は、すべてのユーザーに対してメイン Web ページ **https://***IR1101-IP-ADDRESS* を無効にし、すべてのユーザーに対して **https://***IR1101-IP-ADDRESS***/iox/login** のみを有効にします。一般的な管理と設定に IR1101 メインルータ WebUI を使用しない場合は、この方法を使用します。

VirtualPortGroup

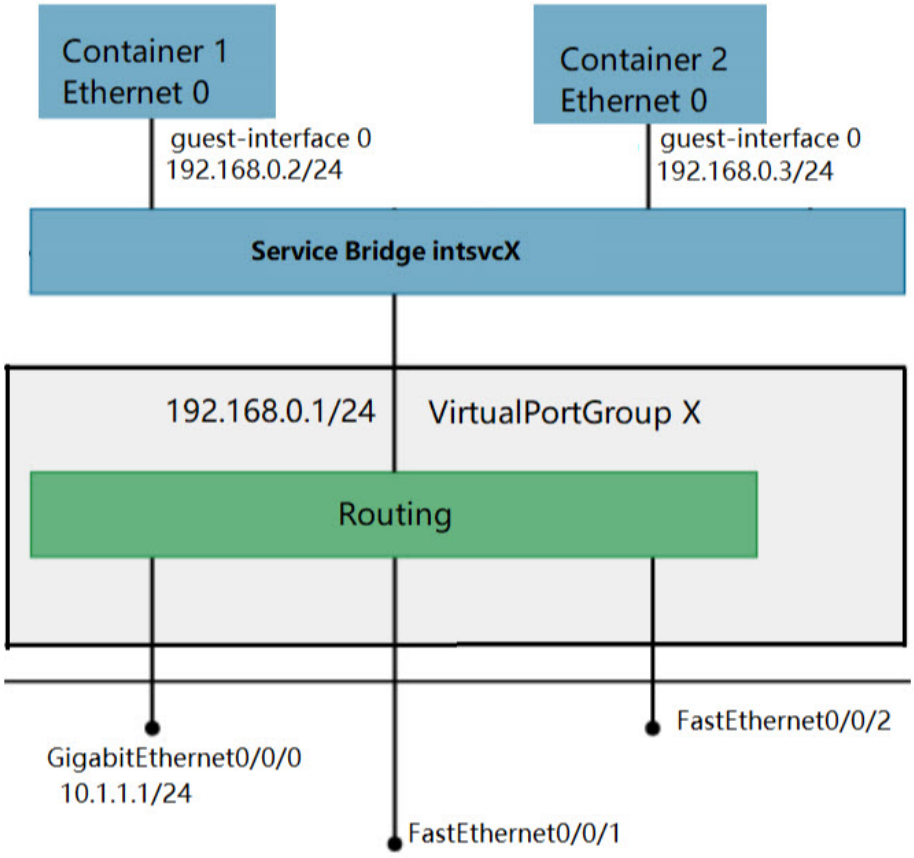
VirtualPortGroup は、Linux ブリッジ IP アドレスにマッピングする Cisco IOS 上のソフトウェア構成要素です。そのため、VirtualPortGroup は、Linux コンテナのスイッチ仮想インターフェイス（SVI）を表します。各ブリッジには、複数のインターフェイスを含めることができ、それぞれ異なるコンテナにマッピングされます。各コンテナには、複数のインターフェイスを含めることもできます。

VirtualPortGroup インターフェイスは、interface virtualportgroup コマンドを使用して設定します。これらのインターフェイスが作成されると、IP アドレスとその他のリソースが割り当てられます。

VirtualPortGroup インターフェイスは、アプリケーション ホスティング ネットワークを IOS ルーティング ドメインに接続します。アプリケーションのレイヤ 3 インターフェイスは、IOS からルーティングされたトラフィックを受信します。VirtualPortGroup インターフェイスは、SVC ブリッジを介してコンテナ/アプリケーション インターフェイスに接続します。

IR8x9 ルータとは異なるため、次の図は VirtualPortGroup とその他のインターフェイス間の関係を理解する上で役に立ちます。

図 *2:* 仮想ポート グループ マッピング



vNIC

コンテナのライフサイクル管理には、内部論理インターフェイスごとに 1 つのコンテナをサポートするレイヤ 3 ルーティング モデルが使用されます。これは、各アプリケーションに対して仮想イーサネットペアが作成されることを意味します。このペアのうち vNIC と呼ばれるインターフェイスは、アプリケーション コンテナの一部です。vpgX と呼ばれるもう 1 つのインターフェイスは、ホスト システムの一部です。

NIC は、コンテナ内の標準イーサネット インターフェイスで、プラットフォーム データプレーンに接続してパケットを送受信します。IOx は、コンテナ内の各 vNIC について、ゲートウェイ（VirtualPortGroup インターフェイス）、IP アドレス、および一意の MAC アドレス割り当てを行います。

コンテナ/アプリケーション内の vNIC は、標準のイーサネット インターフェイスと見なされています。

アプリケーション ホスティングの設定方法

IOx の有効化

IOx Local Manager へのアクセスを有効にするには、次の作業を実行します。IOx Local Manager を使用することで、ホスト システム上のアプリケーションの管理、制御、モニタ、トラブルシューティング、および関連するさまざまなアクティビティを実行できます。

（注） 次の手順では、IP HTTP コマンドは IOX を有効にしませんが、ユーザは WebUI にアクセスして IOX Local Manager に接続できるようになります。

手順の詳細

手順	コマンド	目的
1.	**enable** 例： **Device>enable**	特権 E ドを有 す。 パスワ 力しま された
2.	**configure terminal** 例： Device#**configure terminal**	グロー フィキ ション 開始し
3.	**iox** 例： Device(config)#**iox**	IOx を ます
4.	**ip http server** 例： Device(config)#**ip http server**	IP また ステム HTTP 有効化
5.	**ip http secure-server** 例： Device(config)#**ip http secure-server**	セキュ （HTT バを有 す。

手順	コマンド	目的
6.	**username** *name* **privilege** *level* **password** {**0** | **7** | *user-password* }*encrypted-password* 例： `Device(config)#username cisco privilege 15 password 0 cisco`	ユーザー スの認証 ムとユー 権限レベ 立します。 ユーザー 権レベル 設定する あります。
7.	**end** 例： `Device(config-if)#end`	インター ス コンフ レーショ ドを終了 権 EXEC に戻りま

レイヤ 3 データ ポートへの VirtualPortGroup の設定

複数のレイヤ 3 データポートを 1 つ以上の VirtualPortGroup またはコンテナにルーティングできます。VirutalPortGroups とレイヤ 3 のデータポートは、異なるサブネット上にある必要があります。

レイヤ 3 データポートで外部ルーティングを許可するには、**ip routing** コマンドを有効にします。

手順の詳細

ステップ	コマンド
1.	**enable** 例： `Device>enable`
2.	**configure terminal** 例： `Device#configure terminal`

ステップ	コマンド
3.	**ip routing** 例： `Device(config)#ip routing`
4.	**interface type number** 例： `Device(config)#interface gigabitethernet 0/0/0`
5.	**no switchport** 例： `Device(config-if)#no switchport`
6.	**ip address ip-address mask** 例： `Device(config-if)#ip address 10.1.1.1 255.255.255.0`
7.	**exit** 例： `Device(config-if)#exit`
8.	**interface type number** 例： `Device(config)#interface virtualportgroup 0`
9.	**ip address ip-address mask** 例： `Device(config-if)#ip address 192.168.0.1 255.255.255.0`

ステップ	コマンド
10.	**end** 例： `Device(config-if)#end`
11.	**configure terminal** Enter configuration commands, one per line. CNTL/Z で終了します。 例： `Device#configure terminal`
12.	**app-hosting appid app1** 例： `Device(config)#app-hosting appid app1`
13.	**app-vnic gateway0 virtualportgroup 0 guest-interface 0** 例： `Device(config-app-hosting)#app-vnic gateway0 virtualportgroup 0 guest-interface 0`
14.	**guest-ipaddress 192.168.0.2 netmask 255.255.255.0** 例： `Device(config-app-hosting-gateway0)#guest-ipaddress 192.168.0.2 netmask 255.255.255.0`
15.	**app-default-gateway 192.168.0.1 guest-interface 0** 例： `Device(config-app-hosting-gateway0)#app-default-gateway 192.168.0.1 guest-interface 0`
16.	**end** 例： `Device#end`

ネイティブ Docker のサポート

ネイティブ Docker のサポートが 17.2.1 リリースに追加されました。この機能により、ユーザは Docker アプリケーションを IR1101 に展開できます。アプリケーションのライフサイクルプロセスは、「アプリケーションのインストールとアンインストール」の項の手順と同様です。

Dockerアプリケーションの場合、アプリケーション設定の一部としてエントリポイント設定が必要です。エントリポイントの設定については、次の例を参照してください。

```
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#app-hosting appid app3
Router(config-app-hosting)#app-vnic gateway0 virtualportgroup 0 guest-interface 0
Router(config-app-hosting-gateway0)#guest-ipaddress 192.168.0.7 netmask 255.255.255.0
Router(config-app-hosting-gateway0)#app-default-gateway 192.168.0.1 guest-interface 0
Router(config-app-hosting)#app-resource docker
Router(config-app-hosting-docker)#run-opts 1 "--entrypoint '/bin/sleep 10000'"
Router(config-app-hosting-docker)#end
Router#
```

Docker アプリケーションの出力を次の例に示します。

```
Router#show app-hosting detail
App id : app1
Owner : iox
State : RUNNING
Application
Type : docker
Name : aarch64/busybox
Version : latest
Description :
Path : bootflash:busybox.tar
Activated profile name : custom
Resource reservation
Memory : 431 MB
Disk : 10 MB
CPU : 577 units
VCPU : 1
Attached devices
Type Name Alias
---------------------------------------------
serial/shell iox_console_shell serial0
serial/aux iox_console_aux serial1
serial/syslog iox_syslog serial2
serial/trace iox_trace serial3
Network interfaces
---------------------------------------
eth0:
MAC address : 52:54:dd:e9:ab:7a
IPv4 address : 192.168.0.7
Network name : VPG0
Docker
------
Run-time information
Command :
Entry-point : /bin/sleep 10000
Run options in use : --entrypoint '/bin/sleep 10000'
Application health information
Status : 0
Last probe error :
Last probe output :
Router#
```

IOx コンテナアプリケーションのデジタル IO

リリース 17.2.1 では、IOx コンテナアプリケーションがデジタル IO にアクセスできるようになりました。alarm contact コマンドに新しい CLI が追加されました。

```
Router(config)# alarm contact ?
  <0-4>          Alarm contact number (0: Alarm port, 1-4: Digital I/O)
  attach-to-iox  Enable Digital IO Ports access from IOX
Router (config)# alarm contact attach-to-iox
```

attach-to-iox コマンドを有効にすると、IOx へのすべてのデジタル IO ポートを完全に制御できます。ポートは、4 文字のデバイス /dev/dio-[1-4] として IOX アプリケーションに公開されます。読み取りまたは書き込みの機能を使用して、デジタル IO ポートの値を取得または設定できます。

モードを更新する場合は、モード値を文字型デバイスファイルに書き込むことができます。これは、状態の読み取り/書き込み、モードの変更、およびポートの真のアナログ電圧の読み取りを行う IOCTL コールによって実行されます。この方法に従って、アナログセンサーを IR1101 に接続できます。すべてのポートが最初に入力モードに設定され、電圧は 3.3v にプルアップされます。

次に、IOCTL コールの例を示します。

デジタル IO ポートの読み取り

```
cat /dev/dio-1
```

デジタル IO ポートへの書き込み

```
echo 0 > /dev/dio-1
echo 1 > /dev/dio-1
```

モード変更

```
echo out > /dev/dio-1
echo in > /dev/dio-1
```

サポートされている IOCTL のリスト

```
DIO_GET_STATE = 0x1001
DIO_SET_STATE = 0x1002
DIO_GET_MODE = 0x1003
DIO_SET_MODE_OUTPUT = 0x1004
DIO_SET_MODE_INPUT = 0x1005
DIO_GET_THRESHOLD 0x1006
DIO_SET_THRESHOLD = 0x1007
DIO_GET_VOLTAGE = 0x1009
```

IOCTL を使用した状態の読み取り

```
import fcntl, array
file = open("/dev/dio-1","rw")
state = array.array('L',[0])
fcntl.ioctl(file, DIO_GET_STATE, state)
print(state[0])
```

IOCTL を使用したモードの変更

```
import fcntl
file = open("/dev/dio-1","rw")
fcntl.ioctl(file, DIO_SET_MODE_OUTPUT, 0)
```

署名付きアプリケーションのサポート

シスコの署名付きアプリケーションが IR1101 でサポートされるようになりました。署名付きアプリケーションをインストールするには、デバイスで署名付き検証を有効にする必要があります。署名付き検証を有効にするには、次の手順を実行します。

```
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#
Router(config)#app-hosting signed-verification
Router(config)#
Router(config)#exit
```

署名付き検証を有効にした後、「IOx アプリケーションホスティング」の「アプリケーションのインストールとアンインストール」の項の手順に従ってアプリケーションをインストールします。

アプリケーションのインストールとアンインストール

手順の詳細

ステップ	コマンド
1.	**enable** 例： `Device>enable`

ステップ	コマンド
2.	**app-hosting install appid** *application-name* **package** *package-path* 例： `Device#app-hosting install appid lxc_app package flash:my_iox_app.tar`
3.	**app-hosting activate appid** *application-name* 例： `Device#app-hosting activate appid app1`

ステップ	コマンド
4.	**app-hosting start appid** *application-name* 例： `Device#app-hosting start appid app1`
5.	**app-hosting stop appid** *application-name* 例： `Device#app-hosting stop appid app1`
6.	**app-hosting deactivate appid** *application-name* 例： `Device#app-hosting deactivate appid app1`
7.	**app-hosting uninstall appid** *application-name* 例： `Device#app-hosting uninstall appid app1`

アプリケーションのリソース設定の上書き

リソースの変更は、app-hosting activate コマンドが設定された後にのみ有効になります。

手順の詳細

ステップ	コマンド
1.	**enable** 例： `Device>enable`
2.	**configure terminal** 例： `Device#configure terminal`
3.	**app-hosting appid** *name* 例： `Device(config)#app-hosting appid app1`
4.	**app-resource profile** *name* 例： `Device(config-app-hosting)#app-resource profile custom`
5.	**cpu** *unit* 例： `Device(config-app-resource-profile-custom)# cpu 800`

ステップ	コマンド
6.	**memory** *memory* 例： `Device(config-app-resource-profile-custom)# memory 512`
7.	**vcpu** *number* 例： `Device(config-app-resource-profile-custom)# vcpu 2`
8.	**end** 例： `Device(config-app-resource-profile-custom)# 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の確認

手順の詳細

1. **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）。

Example:

```
Device>enable
```

2. show iox-service

すべての IOx サービスのステータスを表示します。

Example:

```
Device# show iox-service
IOx Infrastructure Summary:
---------------------------------------
IOx service (CAF) 1.8.0.2 : Running
IOx service (HA) : Not Supported
IOx service (IOxman) : Running
Libvirtd 1.3.4 : Running
Device#
```

3. show app-hosting detail

アプリケーションに関する詳細情報を表示します。

Example:

```
Device#show app-hosting detail
App id                 : app1
Owner                  : iox
State                  : RUNNING
Application
  Type                 : lxc
  Name                 : nt08-stress
  Version              : 0.1
  Description          : Stress Testing Application
  Path                 : usbflash0: my_iox_app.tar
Activated profile name : custom
Resource reservation
  Memory               : 64 MB
  Disk                 : 2 MB
  CPU                  : 500 units
Attached devices
  Type              Name               Alias
  ---------------------------------------------
  serial/shell     iox_console_shell   serial0
  serial/aux       iox_console_aux     serial1
  serial/syslog    iox_syslog          serial2
  serial/trace     iox_trace           serial3

Network interfaces
   ---------------------------------------
eth0:
   MAC address         : 52:54:dd:fa:25:ee
```

4. show app-hosting list

アプリケーションとそれらのステータスの一覧を表示します。

Example:

```
Device#show app-hosting list
App id                           State
------------------------------------------------------
app1                             RUNNING
```

デジタル IO 拡張機能

一部のデジタル I/O ポートを IOSd で管理し、他のデジタル IO ポートを IOx コンテナアプリで管理できるようにサポートが追加されました。更新された CLI が追加され、デジタル IO 拡張機能の YANG モデルが更新されました。

CLI の 17.5.1 バージョンは次のとおりです。

```
Router(config)# alarm contact attach-to-iox
```

（注） リリース 17.5.1 では、**alarm contact attach-to-iox**によりすべてのデジタル IO ポート（1 ー 4）に対し IOX で制御できました。

CLI の 17.6.1 バージョンは次のとおりです。

```
Router(config)#alarm contact 1 ?
application Set the alarm application
attach-port-to-iox Enable selected Digital IO Ports access from IOX
description Set alarm description
enable Enable the alarm/digital IO port
output Set mode as output
severity Set the severity level reported
threshold Set the digital IO threshold
trigger Set the alarm trigger

Router(config)#alarm contact 1 attach-port-to-iox

Router#show alarm
Alarm contact 0:
Not enabled.
Digital I/O 1:
Attached to IOX.
Digital I/O 2:
Not enabled.
Digital I/O 3:
Not enabled.
Digital I/O 4:
Not enabled.
```

更新されたCLIでは、1ー4はコンテナアプリケーションのIOxに割り当てるデジタルI/Oポートの数です。

（注） リリース 17.6.1 では、各デジタル IO ポートを IOX に個別に割り当てることができます。

アプリケーション ホスティングの設定例

次の例を参照してください。

例：IOx の有効化

```
Device> enable
Device# configure terminal
Device(config)# iox
Device(config)# ip http server
Device(config)# ip http secure-server
Device(config)# username cisco privilege 15 password 0 cisco
Device(config)# end
```

例：レイヤ 3 データ ポートへの VirtualPortGroup の設定

```
Device> enable
```

```
Device# configure terminal
Device(config)# ip routing
Device(config)# interface gigabitethernet 0/0/0
Device(config-if)# no switchport
Device(config-if)# ip address 10.1.1.1 255.255.255.0
Device(config-if)# exit
Device(config)# interface virtualportgroup 0
Device(config-if)# ip address 192.168.0.1 255.255.255.0
Device(config-if)# end
```

例：アプリケーションのインストールとアンインストール

```
Device> enable
Device# app-hosting install appid app1 package flash:my_iox_app.tar
Device# app-hosting activate appid app1
Device# app-hosting start appid app1
Device# app-hosting stop appid app1
Device# app-hosting deactivate appid app1
Device# app-hosting uninstall appid app1
```

例：アプリケーションのリソース設定の上書き

```
Device# configure terminal
Device(config)# app-hosting appid app1
Device(config-app-hosting)# app-resource profile custom
Device(config-app-resource-profile-custom)# cpu 800
Device(config-app-resource-profile-custom)# memory 512
Device(config-app-resource-profile-custom)# vcpu 2
Device(config-app-resource-profile-custom)# end
```

USB ストレージへの IOx アクセス

お客様から、IOx で実行されている Docker コンテナ内に USB メモリーをマウントする機能が必要との要望がありました。ブートフラッシュでは読み取り/書き込みサイクル数が制限されており、コンテナが eMMC に継続的に書き込むと、ユニットが早期に耐用期限切れになる場合があります。USB メモリーを使用すると、Docker コンテナはブートフラッシュの整合性を損なうことなく、継続的に書き込みを行うことができます。

機能の要件および制限事項

この機能には、次のことが適用されます。

- IR1101 の USB メモリーでサポートされるファイルシステムのタイプは、VFAT、EXT2、および EXT3 です。ただし、IOx は、EXT2 および EXT3 ファイルシステムを使用した USB メモリーのマウントのみをサポートします。シスコでは、次の理由から EXT3 を推奨しています。
  - EXT3 はジャーナリング ファイルシステムです。つまり、フラグメンテーションの問題はありません。

- EXT3 ファイルシステムでの読み取り/書き込みの大幅な高速化
- VFAT には 4 GB の最大ファイルサイズ制限があります。これは、コンテナが大きなファイルを継続的に書き込む際に問題になります。

- IOxアプリケーションによる書き込み操作の進行中にUSBメモリーが取り外されると、コピー操作に含まれるすべてのファイルが失われます。
- IOX とアプリケーションが USB メモリーを使用しているときに取り外すと、IOX は実行状態のままになります。USBメモリーをストレージとして使用するアプリケーションの機能は、USB メモリーでの読み取りや書き込みができないため、深刻な影響を受けます。

IOx アプリケーションで USB メモリーを使用できるようにする

IOxアプリケーションでUSBメモリーを使用できるようにするには、実行オプションを発行する必要があります。次の例を参照してください。

```
Router(config-app-hosting-docker)#run-opts 1 "-v /mnt/usb0:/usbflash0"
```

このコマンドは、IOx アプリケーション ファイルシステム内に USB メモリーファイルシステムをマウントするため、IOx アプリケーションの次のログに示すように、USB メモリーは /usbflash0 フォルダで使用できます。

```
/ # ls -al usbflash0/
total 705424
drwxrwxrwx    4 root     root          4096 Nov 10 22:42 .
drwxr-xr-x    1 root     root          4096 Nov 15 17:22 ..
-rw-r--r--    1 65534    65534    720025859 Nov 10 22:46 ir1101-universalk9.SSA.bin
drwx------    2 65534    65534        16384 Nov  8 16:32 lost+found
#
```

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。